Revisiting Lower and Upper Bounds for Selective Decommitments

Rafail Ostrovsky
UCLA
USA
rafail@cs.ucla.edu

Vanishree Rao
UCLA
USA
vanishri@cs.ucla.edu

Alessandra Scafuro
University of Salerno
ITALY
scafuro@dia.unisa.it

Ivan Visconti
University of Salerno
ITALY
visconti@dia.unisa.it

Abstract

In [DNRS99, DNRS03], Dwork et al. opened the question of the existence of commitment schemes that are secure against selective opening attacks (SOA, for short). In [BHY09] Bellare, Hofheinz, and Yilek, and Hofheinz in [Hof11] solved this open problem by presenting a scheme based on non-black-box use of a one-way permutation and a super-constant number of rounds. The achieved solution however opened other challenging questions about the possibility of improving the round complexity and of obtaining fully black-box schemes where access to an underlying primitive or to an adversary are black-box only. The recent work of Xiao ([Xia11]) investigates on how to achieve nearly optimal SOA-secure commitment schemes where optimality concerns both the round complexity and the black-box use of cryptographic primitives. The work of Xiao focuses on a simulation-based security notion of SOA. Moreover in [Xia11] different results are achieved for security against parallel SOA and concurrent SOA.

In this work we first point out issues on the claims of [Xia11] that actually re-open several of the questions left open in [BHY09, Hof11]. Then we provide different schemes and lower bounds that produce a very different state-of-the-art compared to the one given in [Xia11].

More specifically, by specifying as (x, y) the round complexity of a scheme that requires x rounds in the commitment phase and y rounds in the opening phase, and by considering only (as in [Xia11]) the definition of black-box simulation for SOA-security, we show that:

1. There is an issue in the result of [Xia11] on the existence of $(3, 1)$-round schemes for parallel SOA; indeed we are also able to contradict the above impossibility by presenting a $(3, 1)$-round scheme based on black-box use of trapdoor commitments. Moreover, we show a $(3, 3)$-round scheme based on black-box use of any one-way permutation.
2. There is an issue in the parallel SOA security proof of the black-box $(4, 1)$-round scheme given in [Xia11], thus that scheme might not work. We show instead a $(4, 1)$-round scheme based on black-box use of any weak trapdoor commitment scheme, and a $(5, 1)$-round scheme based on black-box use of any one-way permutation.
3. There is an issue in the proof of security of the concurrent SOA-secure scheme of [Xia11] when the simulator cannot sample from the distribution of committed messages by itself (but needs to query an external party for the same). Furthermore, we contradict the security of the above scheme by showing that there can not exist such a scheme, regardless of the round complexity and of the black-box use of cryptographic primitives.
4. There exists a $(3, 1)$-round scheme based on any (non-black-box) one-way permutation. This improves the result of [BHY09, Hof11] from logarithmic round complexity to 3 (optimal).

All our schemes are secure for parallel SOA and also secure for concurrent SOA under the restriction that all commitment phases are played before any opening phase. Moreover, in all our constructions the simulator does not need to know the distribution of the messages committed to by the sender. In light of our result on the impossibility of a scheme that is SOA-secure under full-fledged concurrent composition (see Item 3 above), the concurrency achieved by our schemes is essentially optimal.

1 Introduction

Commitment schemes are a fundamental building block in cryptographic protocols. The binding property guarantees that a committed message can not be opened in two different ways. The hiding property guarantees that before the opening phase begins, no information about the committed message is revealed. Binding and hiding are preserved under concurrent composition, in the sense that even a concurrent malicious sender is not able to open a committed message in two ways, and even a concurrent malicious receiver is not able to detect any relevant information from committed messages as long as only commitment phases have been played so far.

In [DNRS99, DNRS03], Dwork et al. pointed out a more subtle hiding definition where the malicious receiver is allowed to ask for the opening of some of the committed messages, with the goal of breaking the hiding of the remaining committed messages. Dwork et al. therefore opened the question of the existence of commitment schemes that are secure against a selective opening attack (SOA, for short). The question is particularly interesting since commitment schemes are often used in larger protocols, where often it is the case that only some commitments are opened but the security of the whole scheme still relies on the hiding of the unopened commitments. For instance, the importance of SOA-secure commitments for constructing zero-knowledge sets is discussed in [GM06][1].

The above challenging open question was solved affirmatively in [BHY09] by Bellare, Hofheinz, and Yilek (see also the extended version of Hofheinz [Hof11]) that presented a scheme based on non-black-box (NBB, for short) use of any one-way permutation (OWP, for short) and super-constant number of rounds. However, the above result left open several other questions about the round complexity and the black-box use of the underlying cryptographic primitives.

In a recently work [Xia11], Xiao addressed the above open questions and investigated on how to achieve nearly optimal schemes where optimality concerns both the round complexity and black-box (BB, for short) use of cryptographic primitives. In particular, Xiao addressed SOA-security of commitment schemes for both parallel composition and concurrent composition.

1.1 Our Contribution

In this work we first point out issues on the claims of [Xia11]. Our issues essentially re-open all the major open questions that were supposed to be closed in [Xia11]. We next show how to solve (in some cases in a nearly optimal way) all of them. Interestingly our final claims are very different (and in some cases also in contrast) w.r.t. the ones claimed in [Xia11].

More in details, by specifying as (x, y) the round complexity of a scheme when the commitment phase takes x rounds and the opening phase takes y rounds, and by considering only the definition of BB simulation for SOA security, we revisit [Xia11] claims and re-open some challenging open questions as follow:

1. There is an issue in the impossibility result of [Xia11] on the existence of $(3, 1)$-round schemes secure under parallel composition. Therefore, this re-opens the question of the achievability of $(3, 1)$-round

[1] In [GM06] some forms of zero-knowledge sets where proposed, and their strongest definition required SOA-secure commitments that were not available at that time. They then achieved only a weaker definition where there is a restriction on the distribution of the committed databases.

SOA secure schemes.

2. There is an issue in the proof of the BB $(4,1)$-round scheme of [Xia11] for parallel composition, and thus their scheme might not work. Although this issue does not rule out the existence of an alternative successful security proof, it does re-open the question of obtaining a constant-round scheme that is provably SOA-secure.
3. There is an issue in the security proof of the construction of [Xia11] that is claimed to be secure under concurrent composition in the strong sense; i.e., concurrent composition when the simulator by itself cannot sample from the distribution of messages committed to by the honest sender (but needs to query an external party for it).[2] Therefore, this re-opens the possibility of achieving schemes that are strongly SOA-secure concurrent composition for any round complexity.

With this, the state-of-the-art almost rolls back to the works of [BHY09] and [Hof11]. In this work we solve the above open problems (still sticking with black-box simulation as formalized in [Xia11]) as follows.

1. We present a $(3,1)$-round scheme based on BB use of any trapdoor commitments (TCom, for short), a $(3,3)$-round scheme based on BB use of any OWP, a $(4,1)$-round scheme based on BB use of any weak[3] trapdoor commitment (wTCom, for short), and a $(5,1)$-round scheme based on BB use of any OWP.
2. We show that when the simulator does not know the distribution of the messages committed by the honest sender, then there exists no scheme that achieves strong concurrent SOA security, regardless of the round complexity and of the BB use of cryptographic primitives. Notice that this lower bound *contradicts* the claimed security of the construction given in [Xia11].
3. There exists a $(3,1)$-round scheme based on NBB use of any OWP, and moreover there does not exist a $(2,1)$-round scheme. This improves the round complexity in [BHY09] from logarithmic in the security parameter to 3 rounds only, which is indeed optimal.

Notice that both our $(3,1)$-round protocols *contradict* the impossibility given in [Xia11]. Moreover, note that even ignoring the BB use of a cryptographic primitive, our $(3,1)$ and $(4,1)$-round protocols (based on TCom) do not require **NP** reductions, in contrast to our NBB $(3,1)$-round protocol based on OWPs.

All our constructions are in fact secure under concurrent composition as long as all commitment phases are played before any opening phase (we will refer to this form of composition as "concurrency with barrier", and it obviously implies parallel composition), and moreover our simulators do not require to know the distribution of the messages committed by the honest sender. In light of our impossibility for full-fledged concurrent composition (see item 2 of the above list), the concurrency achieved by our schemes seems to be optimal. Therefore we achieve the strongest form of security against SOA attacks, as specified in [Xia11] (see the paragraph "Stronger definitions of hiding" in [Xia11]) and in [Hof11] (see Item 3 in paragraph "Discussion of the Definitional Choices" in [Hof11]). For the sake of clarifying our contribution, we compare our results with the previous state-of-the-art in the table below.

Here, for instance, $(3,1)$ PAR in the "Imposs." row under [Xia11] means that [Xia11] claims impossibility for a $(3,1)$-round scheme that is SOA-secure under parallel composition; NBB $(\log n, 1)$ CwB OWP in the "Upper Bounds" row under [BHY09] means that [BHY09] shows a $(\log n, 1)$-round scheme based on NBB use of OWPs that is SOA-secure under concurrent-with-barrier composition; CC is shorthand for concurrent composition (as per definition of [Xia11]), t-SH refers to a $(t,1)$-round statistically-hiding commitment scheme. In the last column, we list the results of [Xia11] that we contradict.

[2] For simplicity, we shall hereafter refer to this case as the simulator not knowing the distribution.

[3] This result indeed requires a relaxed definition of trapdoor commitment where the trapdoor is required to be known already during the commitment phase in order to later equivocate.

	[BHY09, Hof11]	[Xia11]	This Paper	This Paper on [Xia11]
Impossible	BB $(1,1)$	$(3,1)$ PAR $(o(\log n/\log\log n),1)$ CC	CC (unknown distribution)	~~(3,1) PAR~~
Achieved	NBB $(\log n,1)$ CwB OWP	BB $(4,1)$ PAR OWP BB $(\omega(t\log n),1)$ CC t-SH	BB $(3,1)$ CwB TCom BB $(4,1)$ CwB wTCom BB $(5,1)$ CwB OWP BB $(3,3)$ CwB OWP NBB $(3,1)$ CwB OWP	~~BB (4,1) PAR OWP~~ ~~BB $(\omega(t\log n),1)$ CC t-SH~~ (unknown distribution)

2 Preliminaries

Notation. We denote by $n \in \mathbb{N}$ the security parameter and by PPT the property of an algorithm of running in probabilistic polynomial-time. A function ϵ is negligible in n (or just negligible) if for every polynomial $p(\cdot)$ there exists a value $n_0 \in \mathbb{N}$ such that for all $n > n_0$ it holds that $\epsilon(n) < 1/p(n)$. We denote by $[k]$ the set $\{1,\ldots,k\}$. Sometimes in the paper we abuse the notation by $i \in [k]$ to mean $\forall i \in [k]$; $\mathtt{poly}(n)$ stands for polynomial in n. We denote by $x \leftarrow \mathcal{D}$ the sampling of an element x from the distribution $\mathcal{D}$. We also use $x \xleftarrow{\$} \mathsf{A}$ to indicate that the element x is uniformly sampled from set A. We denote by $(v_A, v_B) \leftarrow \langle A, B\rangle$ the pair of outputs of parties A and B, respectively, after the completion of their interaction. We use $v \xleftarrow{\$} \langle A, B\rangle$ or $v \xleftarrow{\$} \mathsf{Alg}()$ when the algorithms $A, B, \mathsf{Alg}()$ are randomized. The function $\mathtt{out}$ indicates the outcome of an experiment. Finally, let P_1 and P_2 be two parties running a protocol that uses protocol $\langle A, B\rangle$ as a sub-protocol. When we say that party "P_1 runs $\langle A(\cdot), B(\cdot)\rangle$ with P_2" we always mean that P_1 executes the procedure of party A and P_2 executes the procedure of party B. In the paper we use the words decommitment and opening interchangeably. The same holds for the words statement and theorem.

2.1 Commitment Schemes

In the following definitions we assume that parties are stateful and that malicious parties obtain auxiliary inputs, although for better readability we omit them.

Definition 1 (Bit Commitment Scheme [BHY09], [Hof11])**.** *A commitment scheme is a tuple of PPT algorithms* $\mathsf{Com} = (\mathsf{Gen}, \mathsf{S}, \mathsf{R})$ *implementing the following two-phase functionality.* $\mathsf{Gen}(1^n)$ *outputs a public key* pk *that is given to* S *and* R*. Given to* S *an input* $b \in \{0,1\}$*, in the first phase (*$\mathtt{commitment\ phase}$*)* S *interacts with* R *to commit to the bit* b*, we denote this interaction as* $\langle \mathsf{S}(pk, \mathtt{com}, b), \mathsf{R}(pk, \mathtt{recv})\rangle$*. In the second phase (*$\mathtt{opening\ phase}$*)* S *interacts with* R *again to reveal the bit* b*, we denote this interaction as* $\langle \mathsf{S}(\mathtt{open}), \mathsf{R}(\mathtt{open})\rangle$ *and* R *finally outputs the bit* b' *or* $\perp$*. Then, for all* $pk \xleftarrow{\$} \mathsf{Gen}(1^n)$*, consider the following two experiments:*

Experiment $\mathbf{Exp}^{\text{binding}}_{\mathsf{Com},\mathsf{S}^*}(n)$*:*	**Experiment** $\mathbf{Exp}^{\text{hiding-}b}_{\mathsf{Com},\mathsf{R}^*}(n)$*:*
run $\langle \mathsf{S}^*(pk, \mathtt{com}), \mathsf{R}(pk, \mathtt{recv})\rangle$*;*	
$b_0 \xleftarrow{\$} \langle \mathsf{S}^*(\mathtt{open}, 0), \mathsf{R}(\mathtt{open})\rangle$*;*	$b' \xleftarrow{\$} \langle \mathsf{S}(pk, \mathtt{com}, b), \mathsf{R}^*(pk, \mathtt{recv})\rangle$*;*
rewind S^* *and* R *back after the second step;*	*output* b'*.*
$b_1 \xleftarrow{\$} \langle \mathsf{S}^*(\mathtt{open}, 1), \mathsf{R}(\mathtt{open})\rangle$*;*	
output 1 *iff* $\perp \neq b_0 \neq b_1 \neq \perp$ *.*	

$\mathsf{Com} = (\mathsf{Gen}, \mathsf{S}, \mathsf{R})$ *is a commitment scheme if the following conditions hold:*

Completeness. *If* S *and* R *are honest, for any* S*'s input* $b \in \{0,1\}$ *the output of* R *in the opening phase is* $b' = b$*.*

Hiding. *For any PPT malicious receiver* R^**, there exists a negligible function* ϵ *such that the following holds:*

$$\mathbf{Adv}^{\text{hiding}}_{\mathsf{Com},\mathsf{R}^*} = |\mathsf{Pr}[\mathtt{out}(\mathbf{Exp}^{\text{hiding-0}}_{\mathsf{Com},\mathsf{R}^*}(n) = 1)] - \mathsf{Pr}[\mathtt{out}(\mathbf{Exp}^{\text{hiding-1}}_{\mathsf{Com},\mathsf{R}^*}(n) = 1)]| \leq \epsilon(n)$$

Binding. *For any PPT malicious sender* S^* *there exists a negligible function* ϵ *such that:*

$$\mathsf{Pr}[\mathtt{out}(\mathbf{Exp}^{\text{binding}}_{\mathsf{Com},\mathcal{A}}) = 1] \leq \epsilon(n)$$

A commitment scheme is statistically hiding (resp. binding) if the hiding (resp. binding) condition holds even against an unbounded malicious Receiver (resp. Sender).

The above definition is a slight modification of the one provided in [BHY09, Hof11] and is more general in the fact the it includes the algorithm Gen that provides public parameters to both parties. However, note that for non-interactive commitment schemes (e.g., the one presented in [GL89] constructed on OWP) the algorithm Gen is not used. Under OWF there exists a two-round commitment scheme [Nao91] where Gen is run only by R and the public parameter is sent by R to S in the first round. For better readability, when we construct or use as sub-protocol a commitment scheme that does not use secret/public parameters we refer to it only as $\mathsf{Com} = (\mathsf{S}, \mathsf{R})$ omitting the algorithm Gen.

Remark 1 (Binding definition)**.** *The binding property states that there exists no efficient* S^* *that can produce two distinct accepting openings for the same commitment phase with non-negligible probability. In order to translate this definition into a game one can define the game as follows:* S^* *is run once for the commitment phase and then, keeping the same commitment phase, the game runs* S^* *twice to obtain openings to two distinct values. The output of the game is* 1 *if both are accepting. We follow the definition of [BHY09, Hof11] in which to formalize the fact that* S^* *is run twice to obtain distinct openings they propose the following: run* S^* *once; rewind it up to after the commitment phase and then run it again. Since* S^* *is run with the same randomness then they need to provide an additional input to obtain distinct openings, this is the reason why in the opening phase the malicious server is invoked as* $\mathsf{S}^*(\mathtt{open}, b)$ *with the bit* b *as additional input. One can think about other ways to formalize the binding game, however we follow [BHY09, Hof11].*

For the definitions of trapdoor commitments we use the notation of [MY04].

Definition 2 (Trapdoor Commitment)**.** *A tuple of PPT algorithms* $\mathsf{TC} = (\mathsf{TCgen}, \mathsf{S}, \mathsf{R}, \mathsf{TCfakeDec})$ *is a trapdoor commitment scheme if* $\mathsf{TCgen}(1^n)$ *outputs a public key/secret key pair (pk,sk),* TCgen_{pk} *is the related functionality that restricts the output of* TCgen *to the public key,* $(\mathsf{TCgen}_{pk}, \mathsf{S}, \mathsf{R})$ *is a commitment scheme, and* $\mathsf{TCfakeDec}$ *is such that:*

Trapdoor Property. *For any* $b \in \{0,1\}$*, for all* $r \in \{0,1\}^n$*,* $(pk, sk) \leftarrow \mathsf{TCgen}(r)$*, and for any PPT malicious receiver* R^* *there exists a negligible function* ϵ *such that the following holds:*

$$\mathbf{Adv}^{\text{trapdoor}}_{\mathsf{TC},\mathsf{R}^*} = \mathsf{Pr}[\mathtt{out}(\mathbf{Exp}^{\mathsf{Trap}}_{\mathsf{TC}}(n)) = 1] - \mathsf{Pr}[\mathtt{out}(\mathbf{Exp}^{\mathsf{Com}}_{\mathsf{TC}}(n)) = 1] \leq \epsilon(n)$$

Experiment $\mathbf{Exp}^{\mathsf{Com}}_{\mathsf{TC}}(n)$ *:*	*Experiment* $\mathbf{Exp}^{\mathsf{Trap}}_{\mathsf{TC}}(n)$*:*
run $\langle \mathsf{S}(pk, \mathtt{com}, b), \mathsf{R}^*(pk, sk, b, \mathtt{recv})\rangle$*;*	*run* $(\xi, \cdot) \xleftarrow{\$} \langle \mathsf{S}(pk, \mathtt{com}, \cdot), \mathsf{R}^*(pk, sk, b, \mathtt{recv})\rangle$*;*
$b' \xleftarrow{\$} \langle \mathsf{S}(\mathtt{open}), \mathsf{R}^*(\mathtt{open})\rangle$*;*	$b' \xleftarrow{\$} \langle \mathsf{TCfakeDec}(sk, \mathtt{open}, b, \xi), \mathsf{R}^*(\mathtt{open})\rangle$*;*
output b'*;*	*output* b'*;*

The variable ξ denotes the state shared between algorithms S and $\mathsf{TCfakeDec}$. Note that the above definition allows a sender to perform the commitment phase without knowing the trapdoor and still be able in the decommitment phase to equivocate to any bit. The trapdoor information, i.e., the secret parameter, is needed only in the decommitment phase. This is the standard definition of trapdoor commitment in which once the trapdoor has been obtained one can open any previously computed commitment to any value.

It is possible to consider a weaker definition of trapdoor commitment in which in order to be able to to equivocate the trapdoor is needed in the commitment phase too. Appendix A.1.1 gives the definition of *weak* trapdoor commitments together with a discuss about possible constructions.

Definition 3 (Hiding in presence of Selective Opening Attacks (slight variation of [BHY09, Hof11]))**.** *Let $k = \mathtt{poly}(n)$, let $\mathcal{B}$ be a k-bit message distribution and $\mathbf{b} \xleftarrow{\$} \mathcal{B}$ be a k-bit vector, let $\mathcal{I} = \{\mathcal{I}_k\}_{k\in\mathbb{N}}$ be a family of sets, where each $\mathcal{I}_k$ is a set of subsets of $[k]$ denoting the set of legal subsets of (indexes of) commitments that the receiver (honest or malicious) is allowed to ask for the opening. A commitment scheme $\mathsf{Com} = (\mathsf{Gen}, \mathsf{S}, \mathsf{R})$ is secure against selective opening attacks if for all k, all sets $I \in \mathcal{I}$, all k-bit message distributions $\mathcal{B}$, all PPT relations $\mathcal{R}$, there exists an expected PPT machine Sim such that for any PPT malicious receiver R^* there exists a negligible function ϵ such that:*

$$\mathbf{Adv}^{\mathsf{soa}}_{\mathsf{Com}} = \mathsf{Pr}[(\mathtt{out}(\mathbf{Exp}^{\mathrm{real}}_{\mathsf{Com},\mathsf{S},\mathsf{R}^*}(n)) = 1] - \mathsf{Pr}[\mathtt{out}(\mathbf{Exp}^{\mathrm{ideal}}_{\mathsf{Com},\mathsf{Sim},\mathsf{R}^*}(n)) = 1] \leq \epsilon(n)$$

Experiment $\mathbf{Exp}^{\mathrm{real}}_{\mathsf{Com},\mathsf{S},\mathsf{R}^*}(n)$*:*	*Experiment* $\mathbf{Exp}^{\mathrm{ideal}}_{\mathsf{Com},\mathsf{Sim},\mathsf{R}^*}(n)$*:*
$pk \xleftarrow{\$} \mathsf{Gen}(1^n)$*;*	$pk \xleftarrow{\$} \mathsf{Gen}(1^n)$*;*
$\mathbf{b} \xleftarrow{\$} \mathcal{B}$*;*	$\mathbf{b} \xleftarrow{\$} \mathcal{B}$*;*
$I \xleftarrow{\$} \langle \mathsf{S}_i(pk, \mathtt{com}, \mathbf{b}[i])_{i\in[k]}, \mathsf{R}^*(pk, \mathtt{recv})\rangle$*;*	$I \xleftarrow{\$} \mathsf{Sim}(pk)$*;*
$ext \xleftarrow{\$} \langle \mathsf{S}_i(\mathtt{open})_{i\in I}, \mathsf{R}^*(\mathtt{open})\rangle$*;*	$ext \xleftarrow{\$} \mathsf{Sim}(\mathbf{b}[i])_{i\in I}$*;*
output $\mathcal{R}(I, \mathbf{b}, ext)$*.*	*output* $\mathcal{R}(I, \mathbf{b}, ext)$*.*

We denote by $ext \xleftarrow{\$} \langle \mathsf{S}_i(\cdot), \mathsf{R}^*(\cdot)\rangle$ the output of R^* after having interacted concurrently with k instances of S each one denoted as S_i. In the paper we refer to an instance a session. A malicious receiver R^* can run many sessions in concurrency with the following limitation. R^* runs concurrently commitment phases for polynomially many sessions, but it can initiate the first decommitment phase only after the commitment phase of all the sessions has been completed (and therefore the set of indexes has been requested). This means that the set of indexes I (i.e., the commitments asked to be opened), depends only of the transcript of the commitment phase. We call this definition **concurrent-with-barrier** *(CwB, for short), meaning that many commitment phases (decommitment phases) can be run concurrently but the commitment phase of any session cannot be interleaved with the decommitment of any other session. Notice that as in [Xia11], our definition assumes that the honest receiver chooses to open only a subset of the commitments, but this is done independently of the transcript (i.e., $I \xleftarrow{\$} \mathcal{I}$). This means that in the "SOA-commitment" functionality (differently from traditional commitment functionality) the receiver also has an input that corresponds to opening/not opening.*

Remark 2. *In this paper, unless otherwise mentioned, a SOA-secure commitment scheme is a commitment scheme that is SOA-secure under CwB composition. In fact, all our positive results are for the setting of CwB composition.*

2.1.1 Motivations of Our Definitions

We now discuss the main choices that we made to obtain the above definitions.

Concurrency-with-barrier vs parallel and fully concurrent composition. Xiao provides two main definitions: SOA-security under parallel (PAR) composition and SOA-security under "fully" concurrent composition (CC). In the fully concurrent definition there is no barrier between commitment and decommitment phase (i.e., a malicious receiver is allowed to interleave the commitment phase of one session with the decommitment phase of another, basically having the power of deciding which decommitment/commitment to execute, after having seen some other commitment *and* decommitment transcripts). This definition is pretty general, but unfortunately, as we will show, achieving this result is impossible (under the assumption that the simulator does not know the distribution of the messages committed by the honest sender); this is in contrast to [Xia11] where it is claimed that this result is achievable. The concurrent-with-barrier composition that we adopted (following [Hof11]) implies security under parallel repetition while due to the barrier between commitment and decommitment phase, it is weaker than the fully concurrent definition of [Xia11].

Decommitment phase can be interactive. Following [Hof11] our definition is more general than the one of [Xia11] since it allows also the decommitment phase to be interactive.

Honest party behaviour. We follow the definition of [Xia11] in defining the behaviour of the honest receiver i.e, R chooses the subset of commitments to be opened according to some distribution $\mathcal{I}$. To see why this definition makes sense think about extractable commitments where the sender and receiver engage in many commitments but finally only half of them are required to be opened. However, concerning the honest sender, we assume that R^* interacts with k independent senders, that are oblivious to each another, and play with input $\mathbf{b}[j]$, while [Xia11] consider a single sender S^k who gets as input the complete k-bit string and plays k independent sessions with R^*.

Comparison with the definitions of [BHY09, Hof11]. In [BHY09, Hof11] the behaviour of the honest receiver is not explicitly defined, implying that the honest receiver always obtains all the openings. In order to be more general and to make SOA-secure commitments useful in more general scenarios (i.e., in the classical implementation of extractable commitments [PRS02] the *honest* receiver asks for the opening of a random subset of the commitments), we deviate from this definition allowing the honest receiver to ask for the opening of a subset of the commitments. Moreover, the set of indexes I chosen by (possibly malicious) receiver is explicitly given as input to the relation $\mathcal{R}$.

Summing up, the definition that we adopt mainly follows the one of [BHY09, Hof11] and is more general than the one of [Xia11] in the fact that it allows interaction also during the decommitment phase, and provides concurrency-with-barrier that implies the definition of security under parallel composition. Moreover, our definition is more general than the one of [Hof11] since it allows also the the honest receiver to choose the commitments to be opened. However, our definition is weaker than the fully concurrent definition of [Xia11] that however we show to be impossible to achieve in the strong sense.

3 Upper Bounds

3.1 SOA-secure Commitment Scheme on Trapdoor Commitments

In this section we show how to use trapdoor commitment schemes in a black-box way to obtain a round-optimal construction for SOA-secure commitment schemes. In particular we show that if two-round trapdoor commitment schemes exist then a 3-round commitment scheme that is secure under selective opening

attack exists. Under a weaker assumption that *weak* trapdoor commitment schemes exist we present a 4-round construction.

The main idea behind both the protocols, roughly speaking, is to require the sender to commit to its private input using a trapdoor commitment scheme such that: 1) the public parameters of the scheme are chosen by the receiver, 2) the trapdoor information (needed for equivocation) is extractable by rewinding the receiver. This allows the black-box simulator to extract the trapdoor and in turn to commit to a random value in the commitment phase, and to open to the appropriate bit in the opening phase. More in details the protocol's structure is the following: R runs the generation algorithm of the trapdoor commitment scheme (TCgen) to generate the public parameters used by S to commit to its private bit. To guarantee that R computes the public parameters by honestly running TCgen (therefore knowing the corresponding trapdoor), we use the cut-and-choose technique; i.e., we require that R provides $2n$ public parameters to S and S verifies the correctness of a random n-size subset of them. For this subset R is required to send the randomness used to run TCgen, whic allows S to check the public parameters but it also reveals the trapdoor parameters. To allow simulation, we require that S uses the remaining n public keys (public parameters for which it has not received the trapdoor) to commit to the n shares of the secret bit b (we call these commitments sub-commitments). With this, in order to equivocate, it is sufficient for the simulator to extract only one among the n trapdoors it has not already received trapdoors for from R. In the protocol that uses regular trapdoor commitments (namely Prot. 1) Sim can commit without knowing the trapdoor, thus it can complete the commitment phase, before starting the extraction phase and the whole commitment phase requires 3 rounds only. In the protocol that uses weak trapdoor commitments (namely Prot. 5), Sim needs to extract the trapdoor before committing, and therefore, for each session, the extraction must be successfully completed before Sim is required to commit to the (shares of the) bit. This thus incurs one more round in the commitment phase of Prot. 5.

3.2 (3,1)-round SOA-Secure Scheme Based On Trapdoor Commitments

Let us denote as $\mathsf{TC} = (\mathsf{TCgen}, \mathsf{S}_{\mathsf{TC}}, \mathsf{R}_{\mathsf{TC}}, \mathsf{TCfakeDec})$ a trapdoor commitment scheme and let n be the security parameter. In the following we show a protocol $\mathsf{SOACom} = (\mathsf{S}_{\mathsf{soa}}, \mathsf{R}_{\mathsf{soa}})$ that uses TC as sub-protocol in a black-box fashion. If the sub-protocol TC is a (2,1)-round [4] commitment scheme (where receiver speaks first) then the following construction is a (3,1)-round commitment, non-interactive decommitment scheme. We indicate with $\mathsf{S}_{\mathsf{TC}}^{\bar{d}_i}, \mathsf{R}_{\mathsf{TC}}^{\bar{d}_i}$ the i-th invocation of the commitment/decommitment algorithm of the protocol TC using the public key $\mathsf{pk}^{\bar{d}_i}$, where d_i is the challenge of the cut-and-choose i.e, S did not ask to see the correspondent secret key.

Protocol 1. *[(3,1)-Round SOA-Secure Commitment Scheme]*

Commitment phase.

$\mathsf{R}_{\mathsf{soa}}$**:** *for* $i = 1, \ldots, n$*:*

1. $r_i^0, r_i^1 \xleftarrow{\$} \{0,1\}^n$; $(\mathsf{pk}_i^0, \mathsf{sk}_i^0) \leftarrow \mathsf{TCgen}(r_i^0)$; $(\mathsf{pk}_i^1, \mathsf{sk}_i^1) \leftarrow \mathsf{TCgen}(r_i^1)$;
2. *send* $\{\mathsf{pk}_i^0, \mathsf{pk}_i^1\}_{i\in[n]}$ *to* $\mathsf{S}_{\mathsf{soa}}$;

$\mathsf{S}_{\mathsf{soa}}$**:** *On input a bit* b*. Upon receiving* $\{\mathsf{pk}_i^0, \mathsf{pk}_i^1\}_{i\in[n]}$*:*

1. *secret share the bit* b*: for* $i = 1, \ldots, n-1$*:* $b_i \xleftarrow{\$} \{0,1\}$*, such that* $b = \leftarrow (\bigoplus_{i=1}^{n} b_i)$;
2. *for* $i = 1, \ldots, n$ *do in parallel:*

[4] [Ped92] is an example of a two-round trapdoor commitment scheme in which the first round is played by the receiver and it serves to send the public parameter to the sender.

- *send* $d_i \xleftarrow{\$} \{0,1\}$ *to* $\mathsf{R}_{\mathsf{soa}}$*;*
- *run* $\langle \mathsf{S}_{\mathsf{TC}}^{\bar{d}_i}(\mathsf{pk}_i^{\bar{d}_i}, \mathtt{com}, b_i), \mathsf{R}_{\mathsf{TC}}^{\bar{d}_i}(\mathsf{pk}_i^{\bar{d}_i}, \mathtt{recv})\rangle$ *with* $\mathsf{R}_{\mathsf{soa}}$*;*

$\mathsf{R}_{\mathsf{soa}}$**:** *Upon receiving* $d_1, \ldots, d_n$*: if all commitment-phases of* TC *protocol were successfully completed, send* $\{r_i^{d_i}\}_{i\in[n]}$ *to* $\mathsf{S}_{\mathsf{soa}}$*;*

$\mathsf{S}_{\mathsf{soa}}$**:** *Upon receiving* $\{r_i^{d_i}\}_{i\in[n]}$ **check consistency***: for* $i = 1, \ldots, n$*:* $(\mathsf{pk}_i'^{d_i}, \mathsf{sk}_i'^{d_i}) \leftarrow \mathsf{TCgen}(r_i^{d_i})$*; if* $\mathsf{pk}_i'^{d_i} \neq \mathsf{pk}_i^{d_i}$ $\mathtt{ABORT}$.

Decommitment phase.

$\mathsf{S}_{\mathsf{soa}}$**:** *for* $i = 1, \ldots, n$*: run* $b_i' \leftarrow \langle \mathsf{S}_{\mathsf{TC}}^{\bar{d}_i}(\mathtt{open}), \mathsf{R}_{\mathsf{TC}}^{\bar{d}_i}(\mathtt{open})\rangle$ *with* $\mathsf{R}_{\mathsf{soa}}$*;*

$\mathsf{R}_{\mathsf{soa}}$**:** *If all opening phases were successful completed output* $b' \leftarrow \bigoplus_{i=1}^{n} b_i'$*. Otherwise output* $\perp$*.*

Theorem 1 (Protocol 1 is secure under selective opening attacks)**.** *If* $\mathsf{TC} = (\mathsf{TCgen}, \mathsf{S}_{\mathsf{TC}}, \mathsf{R}_{\mathsf{TC}}, \mathsf{TCfakeDec})$ *is a trapdoor commitment scheme, then Protocol 1 is a commitment scheme secure against selective opening attacks.*

For lack of space, the proof appears in Appendix D.1.

The case of a $(4,1)$-round construction is very similar, and only deviates in the commitment of the bit that is sent in the 4-th round instead of the 2nd round. Since now the commitment to be opened is played after the receiver has already sent half of the trapdoors, the new scheme can be based on weaker form of trapdoor commitment schemes where in order to equivocate, it is required to use the trapdoor already in the commitment phase.

For lack of space, further details appear in Appendix C.1.

3.3 (3,3)-Round SOA-Secure Commitment Scheme on Any BB OWP

In this section we present a $(3,3)$-round SOA-secure commitment scheme based on the BB use of any OWP. As a main ingredient, we use the extractable commitment scheme ExtCom shown in Prot.4 (App. A.3) that is based on the BB use of statistically-binding commitments that in turn are constructed using OWPs in a BB way.

The idea behind the protocols is as follows. The sender and the receiver first engage in a coin-flipping protocol where the receiver commits to its random-string, then the sender sends its random string in the clear, and finally the receiver opens its commitments. Simultaneously, the sender commits to its input bit b, n pairs of times (with the two commitments in each pair indexed by 0 and 1). At the completion of the coin-flipping protocol, in the decommitment phase, the sender opens only one of the commitments in each pair according to the outcome of the coin-flipping.

To allow simulation, the commitment of the receiver in the coin-flipping protocol is implemented via extractable commitment scheme, so that the simulator can extract such string already in the commitment phase. Furthermore, we require that sender sends its random string for the coin-flipping only in the decommitment phase (when the simulator have already received the bit b to open). This gives the simulator an opportunity to craft its random string as to point to the extractable-commitments to b. To see why, first note that if a malicious sender somehow knows the receiver's random string before it sends its own, then it can easily open the commitment to either 0 or 1. In each pair, it just commits to 0 in one of the commitments and 1 in the other. Then once it knows the receiver's random string it just crafts its random string to point to the commitments of 0 if it needs to open to 0 and similarly for 1. The same idea is used

for simulation too; the simulator is able to extract the receiver's random string (due to the extractability of the receiver's commitments) and hence is able to equivocate in the opening phase. Furthermore, as it will appear more clearly in the protocol, since the sender would send its commitments/decommitments always *after* it receives commitments/decommitments from the receiver, we require that the sender's $2n$ commitments to its input bit are implemented via extractable commitment scheme so that we avoid malleability issues that may compromise the binding property.

Details follow in Prot. 2. In the following we denote by $\langle \mathsf{S_{ext}}^i(\mathtt{com}, a_i), \mathsf{R_{ext}}^i(\mathtt{recv})\rangle$ the i-th of the n parallel executions of the extractable commitment scheme run by $\mathsf{R_{soa}}$ to perform the coin flipping, while by $\langle \mathsf{S_{ext}}^{i,\sigma}(\mathtt{com}, b), \mathsf{R_{ext}}^{i,\sigma}(\mathtt{recv})\rangle$ we denote the commitment in position σ of the i-th pair (among the n pairs) of parallel executions run by $\mathsf{S_{soa}}$ to commit its input b.

Protocol 2. $/\mathsf{SOACom} = (\mathsf{S_{soa}}, \mathsf{R_{soa}})/$

Commitment phase.

$\mathsf{R_{soa}}$: *For $i = 1, \ldots, n$ do in parallel:*

1. $a_i \xleftarrow{\$} \{0,1\}$;
2. *run* $\langle \mathsf{S_{ext}}^i(\mathtt{com}, a_i), \mathsf{R_{ext}}^i(\mathtt{recv})\rangle$ *with* $\mathsf{S_{soa}}$;

$\mathsf{S_{soa}}$: *on input a bit b. For $i = 1, \ldots, n$ do in parallel:*

1. *run* $\langle \mathsf{S_{ext}}^{i,0}(\mathtt{com}, b), \mathsf{R_{ext}}^{i,0}(\mathtt{recv})\rangle$ *with* $\mathsf{R_{soa}}$;
2. *run* $\langle \mathsf{S_{ext}}^{i,1}(\mathtt{com}, b), \mathsf{R_{ext}}^{i,1}(\mathtt{recv})\rangle$ *with* $\mathsf{R_{soa}}$;

Decommitment Phase.

$\mathsf{S_{soa}}$: *If all commitment phases played with* $\mathsf{R_{soa}}$ *are successfully completed, send* $d \xleftarrow{\$} \{0,1\}^n$ *to* $\mathsf{R_{soa}}$;

$\mathsf{R_{soa}}$: *Open all commitments:*

for $i = 1 \ldots, n$: run $\langle \mathsf{S_{ext}}^i(\mathtt{open}), \mathsf{R_{ext}}^i(\mathtt{open})\rangle$ *with* $\mathsf{S_{soa}}$;

$\mathsf{S_{soa}}$: *If all openings provided by* $\mathsf{R_{soa}}$ *are valid, for $i = 1, \ldots, n$:*

1. $\sigma_i \leftarrow d_i \oplus a_i$;
2. *run* $\langle \mathsf{S_{ext}}^{i,\sigma_i}(\mathtt{open}), \mathsf{R_{ext}}^{i,\sigma_i}(\mathtt{open})\rangle$ *with* $\mathsf{R_{soa}}$;

$\mathsf{R_{soa}}$: *If all the corresponding openings provided by* $\mathsf{S_{soa}}$ *open to the same bit b, and if for every i, $\sigma_i = d_i \oplus a_i$, then output b. Otherwise, output $\perp$.*

Note that the commitment phase of the protocol above (Prot. 2) basically consists of running the commitment phase of the $(3,1)$-round extractable commitment scheme ExtCom in both directions (i.e. from $\mathsf{S_{soa}}$ to $\mathsf{R_{soa}}$ and vice versa). Therefore, it seems that the commitment phase requires 4-rounds. However, by merging the third round of the extractable commitment played by $\mathsf{S_{soa}}$ with the first round of the opening phase (played by $\mathsf{S_{soa}}$ as well), we obtain a 3-round commitment phase.

Theorem 2 (Protocol 2 is secure under selective opening attacks)**.** *If* ExtCom *is an extractable commitment scheme, then Protocol 2 is a commitment scheme secure against selective opening attacks.*

For lack of space, the proof can be found in Appendix D.3.

Our $(5,1)$-round SOA-secure commitment scheme on BB use of any OWP is very similar to Prot. 2 and is essentially based on shifting the first two rounds of the opening phase to the commitment phase although it requires a new opening strategy to allow simulation. The opening phase indeed is such that sender can open either the commitments that are always in the positions defined by the coin flipping or always in the positions defined by the binary negation of it. For lack of space further details appear in Appendix C.2.

3.4 Non-Black-Box (3,1)-Round Scheme on Any OWP

In this section we present a round optimal non-black-box construction based on any OWP. We stress that we only use the primitive in a non-black-box way while our simulator is still black-box. The previous non-black-box construction provided in [BHY09, Hof11] requires a poly-logarithmic number of rounds due to the use of a concurrent ZK proof system in the commitment phase. We circumvent the use of the heavy and round expensive machinery by combining the use of extractable commitments and a special 3-round $\mathcal{WIPOK}$, namely FLS, and obtain a (3,1)-round commitment scheme. The main idea is to require the receiver to commit, via the extractable commitment ExtCom (Prot. 4), to a secret α in the commitment phase, while the sender independently commits to its secret bit by using a regular statistically-binding commitment. Then, in the decommitment phase, the sender sends the bit (without sending any decommitment information) and it proves using FLS that there exists an opening to that bit or it knows the secret α hidden in the extractable commitment given by the receiver. By the extractability [5] of the commitment scheme used by R, the simulator can commit to random value in the commitment phase, then it extracts the secret of R and thus it can open to any bit in the decommitment phase by proving the theorem using the extracted secret as witness.

Note that here we are using statistically binding commitment scheme (and thus the underlying one-way permutation) in a non-black-box way since we use the code of commitment procedure in the statement to be proved. The simulator instead only makes a black-box use of the code of the adversary.

There is an issue concerning the extractable commitment sent by the receiver. Note that we need that the commitment sent by the receiver must never be opened in the protocol, since otherwise a malicious sender can easily equivocate by proving the statement using the secret of the receiver. However, on the other hand, if the extractable commitment is never opened by the receiver, a malicious receiver can cheat in the commitment phase without being detected (i.e., recall that in Prot. 4, in order to commit to a string s, the sender commits using $\mathsf{Com}_{\mathsf{NI}}$ to n pairs of strings such that the XOR of each pair gives the *same* string s. Now in this scenario, with the extractable commitment never being opened, a malicious receiver could commit to pairs of strings XORing to distinct strings).

We address the above issue by relying on the fact that the extractor E of ExtCom outputs openings to at least one pair of strings with overwhelming probability. With a careful choice of the statement to be proved, this will be sufficient for the simulator to equivocate. In what follows we provide a formal description of the protocol.

$(3,1)$-round Scheme on any (Non-Black-Box) OWP. A main ingredient of the protocol is $(3,1)$-round extractable commitment scheme $\mathsf{ExtCom} = (\mathsf{S}_{\mathsf{ext}}, \mathsf{R}_{\mathsf{ext}})$ shown in Prot. 4. Recall that the latter is based on BB use of any non-interactive statistically-binding commitment scheme $\mathsf{Com}_{\mathsf{NI}} = (\mathsf{S}_{\mathsf{NI}}, \mathsf{R}_{\mathsf{NI}})$ (and

[5] Note that in this construction we use the extractability property in a loose sense, indeed it suffices for us that the extractor is able to obtain at least the opening of a pair of *minor* commitments, regardless of whether such opening coincides with the major decommitment. This is due to the fact that in the protocol we do not run the full ExtCom protocol but only its commitment phase.

hence of OWP). For notation, we refer to the single message of the commitment phase of $\mathsf{Com_{NI}}$ by C. Also, since we are using the commitment scheme in a non-black-box way we need to deal with some details of the protocol. We use the variable τ_{ext} to refer to the n pairs of (minor) commitments of the first round of ExtCom, and we use the notation $(\tau_{\mathsf{ext}_i}^0, \tau_{\mathsf{ext}_i}^1)$ to explicitly refer to the i-th pair of strings of τ_{ext}. Let $\mathsf{FLS} = (\mathsf{FLS1}, \mathsf{FLS2}, \mathsf{FLS3})$ be the special $\mathcal{WIPOK}$ protocol described in Section A.2.1 for the following language $\Lambda_{\mathtt{com}}$: $(b, \mathsf{C}, \tau_{\mathsf{ext}}, \mathsf{c_{wi}}) \in \Lambda_{\mathtt{com}}$ if there exists an opening of C to b OR there exists i such that the string committed in $\mathsf{c_{wi}}$ is equal to the XOR of the strings committed to in the pair $(\tau_{\mathsf{ext}_i}^0, \tau_{\mathsf{ext}_i}^1)$.

Protocol 3. *[(3,1)-Round SOA-Secure Scheme on Any (Non-Black-Box) OWP]*

Commitment phase.

$\mathsf{R_{soa}}$**:** $\alpha \xleftarrow{\$} \{0,1\}^n$*; run* $\langle \mathsf{S_{ext}}(\mathtt{com}, \alpha), \mathsf{R_{ext}}(\mathtt{recv}) \rangle$ *with* $\mathsf{S_{soa}}$*; let* τ_{ext} *be as above;*

$\mathsf{S_{soa}}$**:** *On input a bit b:*

1. *run* $\langle \mathsf{S_{NI}}(\mathtt{com}, b), \mathsf{R_{NI}}(\mathtt{recv}) \rangle$ *with* $\mathsf{R_{soa}}$*; let* C *be the output;*
2. *run* $\mathsf{FLS1}$*;*

$\mathsf{R_{soa}}$**:** *Run* $\mathsf{FLS2}$*;*

Decommitment phase.

$\mathsf{S_{soa}}$**:** *run* $\langle \mathsf{S_{NI}}(\mathtt{com}, 0^n), \mathsf{R_{NI}}(\mathtt{recv}) \rangle$ *with* $\mathsf{R_{soa}}$ *and let* $\mathsf{c_{wi}}$ *be the resulting message sent to* $\mathsf{R_{soa}}$*; run* $\mathsf{FLS3}$ *for the theorem* $(b, \mathsf{C}, \tau_{\mathsf{ext}}, \mathsf{c_{wi}}) \in \Lambda_{\mathtt{com}}$*. Send b to* $\mathsf{R_{soa}}$*;*

$\mathsf{R_{soa}}$**:** *If* $\mathsf{FLS3}$ *is accepting, then output b; otherwise, output* $\perp$*.*

Theorem 3 (Protocol 3 is secure under selective opening attacks)**.** *If* $\mathsf{Com_{NI}} = (\mathsf{S_{NI}}, \mathsf{R_{NI}})$ *is a statistically binding commitment scheme and* FLS *is a special* $\mathcal{WIPOK}$ *for* NP*, then Protocol 3 is a commitment scheme secure against selective opening attacks.*

For lack of space, the proof appears in Appendix D.5.

4 Issues in Some of the Claims of [Xia11]

In this section we point out some issues regarding some of the main results in [Xia11].

Revisiting proof of Theorem 3.3 in [Xia11]. Theorem 3.3 in [Xia11] claims that their $(4,1)$-round protocol is SOA-secure under parallel composition with BB use of OWPs. The protocol essentially is just the equivocal commitment scheme of Di Crescenzo and Ostrovsky [CO99], which in turn begins with a 3-round preamble for coin-flipping followed by Naor's commitment. In the preamble, firstly, the receiver commits to a random string; secondly, the sender sends a random string in the clear to the receiver; finally, the receiver opens its commitment. The theorem claims that if this preamble is constructed using a non-interactive commitment scheme, then the resulting protocol is SOA-secure under parallel composition with BB use of a OWP. Furthermore, the proof claims that one can build an expected polynomial-time simulator by borrowing the simulation strategy of Goldreich and Kahan [GK96]; i.e., the simulator learns the random strings committed to by the receiver in each of the parallel session and then rewinds the receiver to send new challenges in every session that would enable the simulator to equivocate in the opening phase.

Unfortunately, as we argue below, this simulation strategy of [GK96] cannot be directly applied in the SOA setting. Firstly, note that the simulator of [GK96] was built for a stand-alone zero-knowledge protocol where an atomic sub-protocol is repeated several times. Then if the verifier aborts in any execution of the

atomic sub-protocol, it automatically does so in the whole stand-alone protocol (i.e., it cannot selectively abort in a few sub-protocols proceeding ahead in the rest of the sub-protocols). This marks a crucial difference between the stand-alone zero-knowledge setting and selective-opening attacks. This is because in the SOA-setting the adversarial receiver interacts with multiple senders and can decide to abort only a subset of the sessions of its choice.

In fact, similar issues have been discussed in [CO99] and overcame by assuming that the malicious concurrent verifier is non-aborting, which is precisely the opposite case with SOA, since aborts are a major mechanism to attack the simulator.

The above observation clarifies that in contrast to what is claimed in the proof of [Xia11], the simulator does not necessarily learn the random strings committed to by the receiver in each of the sessions, since there could exist receivers that always abort some specific sessions. Moreover, even if we try to implement the same simulation strategy of [GK96] after taking into account the fact that an adversarial receiver can selectively abort, the problem still persists and the resulting simulator does not run in expected polynomial-time. Specifically, as in [GK96], suppose that the simulator first, in every session, honestly plays the sender in the coin-flipping preamble and continues to rewind the receiver until the latter aborts exactly those sessions that it aborted in the main-thread. Such a simulator obviously does not run in expected polynomial-time; to see why, observe that the simulator may need to handle the receiver distinctly for every distinct subset of sessions that the receiver may abort; also one needs to take into account the fact that the receiver may abort distinct subsets of sessions with distinct probabilities.

To conclude, the $(4, 1)$-round protocol of [Xia11] might still work but it requires a new simulation strategy. Whether such a strategy exists is an interesting open question.

Revisiting proof of Theorem 3.5 in [Xia11]. Theorem 3.5 in [Xia11] claims that if the preamble of [CO99] is implemented through the $\omega(\log(n))$-round preamble of Prabhakaran, Rosen and Sahai given in [PRS02], then the resulting protocol is an $\omega(\log(n))$-round scheme that is SOA-secure under concurrent composition with BB use of OWPs. Moreover, Theorem 3.5 also applies to the strong definition where the same simulator must work with respect to all distribution of messages, including the ones selected by the adversary and unknown to the simulator.

According to their proof, the simulatability of the protocol follows from the simulation strategy of [PRS02]. Specifically, like in the claims for their other positive results, the proof here also claims that the simulator can obtain the random strings committed to by the receiver in every session. Now, to begin with, by the same reason as mentioned for the previous issue, namely, there could exist adversarial receivers who always abort some specific sessions thus rendering the very claim to be immediately untrue. This would not be a problem because a session that is always aborted does not require to be "solved" by the simulator. However, a direct use of the simulator of [PRS02] would lead to other problems. To see why, we first observe that, in the fully concurrent setting, a receiver may adaptively select which sessions it would query to receive decommitments for, as long as, by the end, the set of indices I that it would query to open belongs to $\mathcal{I}$. On the other hand, the proof of concurrent zero knowledge of [PRS02] (used by [Xia11]) critically relies on the fact that the simulator aborts (i.e., reaches the end of a preamble without solving the session) with negligible probability only. On the setting of SOA-security, a malicious verifier who can adaptively decide I may query in the rewinding threads for openings of sessions that were not queried to be opened in the main-thread. The simulator could handle such sessions in two possible ways. For one, it can query the external oracle for the bit corresponding to such a session[6]. This would lead to

[6]Note that here we are critically considering the case in which the distribution is not known to the simulator, and therefore the only way to answer consistently for it is to query the oracle. If instead the distribution is known, the simulator could

a deviation in the distribution of the resulting I queried to the external party, since the number of queries performed in the simulation will be larger with respect to the real game. On the other hand, it can simply abort the rewinding threads containing new sessions that require new queries. This would immediately counteract the necessary condition (i.e., the simulator should abort with negligible probability only) for the results of [PRS02] to be usable. Note that these two observations crucially rely on the fact that the protocol is claimed to be SOA-secure in the strong sense, namely, the simulator does not know the distribution of the messages committed to by the honest sender, and it is supposed to work for *all* message distributions. Similar to the previous issues, we do not fix the protocol itself. However, no fix is possible at all in this case - irrespective of the round complexity. Indeed, we present a negative result (in Theorem 4) that establishes the impossibility of schemes (with any round complexity) that satisfy SOA-security under concurrent composition and unknown message distributions.

Revisiting proof of Theorem 4.4 in [Xia11]. Theorem 4.4 in [Xia11] states that, there exists no $(3,1)$-round commitment scheme that is SOA-secure even under parallel composition, when security is proved using a black-box simulator. The proof essentially assumes that the structure of such a protocol is the following: the sender sends a message in the first round, followed by the receiver sending a message in the second round, and then the sender sends a message in the third round; i.e., the proof only considers protocols that have such a structure in the commitment phase and proves that such protocols do not exist relative to any oracle. However, we argue that this assumption loses generality. In fact, we present two $(3,1)$-round schemes (Protocol 1 and Protocol 3) in which the receiver speaks first, such that security in the concurrent-with-barrier setting (that is strictly stronger that the parallel composition setting [Xia11]) is proved using a black-box simulator. Furthermore, Protocol 1 only requires BB use of trapdoor commitments). As we explain in Appendix B, the proof of Theorem 4.4. of [Xia11] implies the impossibility of a 2-round protocol.

5 Impossibility of Fully Concurrent Security

The last question to answer is whether there exist protocols that actually achieve the definition of security under strong fully concurrent composition (as defined in [Xia11]), or if the concurrent-with-barrier definition is the best one can hope to achieve. Recall that the strong fully concurrent composition assumes that the simulator does not know the distribution of the messages committed by the sender. In this section we show that in contrast to the claim of Theorem 5.1 of [Xia11], the strong fully concurrent security definition of [Xia11] is impossible to achieve. This holds regardless of the round complexity of the protocol and of the black-box use of cryptographic primitives. The only requirement that we use for the impossibility is that the simulator is black-box and does not know the distribution of the messages committed by the sender. In the following we first recall the definition provided in [Xia11], then we show a family of adversary $\{\mathsf{R}^*_p\}_{p\in\{0,1\}}$ such that for any string commitment protocol there exists no black-box simulator, unless the binding of the protocol is compromised. Finally, as a corollary, we show that this result holds also for bit commitment schemes.

Definition of hiding under SOA - concurrent composition [Xia11]. Let $\mathcal{B}$, $\mathcal{I}$, k be as defined in Def. 3 and $\mathbf{b} \stackrel{\$}{\leftarrow} \mathcal{B}$ the input given to the honest sender S.

Security is defined as the comparison of two experiments. In the real world experiment R^* interacts with S in k concurrent sessions and is allowed to pick the set I incrementally. Namely, R^* can generate one commitment phase transcript, ask the sender to decommit that instance, then use this transcript to generate other commitment phases or to select other commitment phases to be opened. The output of this

sample from the distribution and therefore manage in some way the opening of new sessions started during rewinding thread.

experiment is defined as $\langle \mathsf{S}^k(\mathbf{b}), \mathsf{R}^*_{\mathsf{soa}} \rangle = (\tau^k, I, \{b_i, w_i\}_{i \in I})$, where τ^k is the transcript of the commitment phases of the k concurrent sessions, I is the final subset of positions asked incrementally by R^* during the execution, $\{b_i, w_i\}_{i \in I}$ are pairs such that b_i is the bit committed to and w_i is the opening data (recall that this definition assumes that the decommitment is non-interactive, however our impossibility result holds even for protocol with interactive decommitment phase). In the ideal world experiment, an expected PPT simulator Sim without the knowledge of the vector $\mathbf{b}$ interacts with R^* while incrementally giving as output a set I for which it receives the bits $\{b_i\}_{i \in I}$. Finally, Sim outputs τ^k and $\{b_i, w_i\}_{i \in I}$. This can be seen as if Sim has access to an oracle $\mathcal{O}$ that knows the vector $\mathbf{b}$, and can be queried with the index of a session j and answers with the bit $\mathbf{b}[j]$. The output $(\mathsf{Sim}_k^{\mathsf{R}^*_{\mathsf{soa}}} | \mathbf{b})$ of this experiment is $(\tau^k, I, \{b_i, w_i\}_{i \in I})$ where τ^k,$\{b_i, w_i\}_{i \in I}$ are outputs of Sim while I is the set containing the indexes queried by Sim to the oracle $\mathcal{O}$.

A bit commitment scheme Π is SOA-secure under concurrent composition if, for fixed $\mathcal{I}, \mathcal{B}$ and k, there exists Sim such that for all R^* it holds that $\langle \mathsf{S}^k(\mathbf{b}), \mathsf{R}^*_{\mathsf{soa}} \rangle$ and $(\mathsf{Sim}_k^{\mathsf{R}^*_{\mathsf{soa}}} | \mathbf{b})$ are computationally indistinguishable. If Sim exists for all $\mathcal{B}, \mathcal{I}$ and $\mathsf{R}^*_{\mathsf{soa}}$ then the protocol is said SOA-secure under strong concurrent composition. The same definition can be extended to concurrent SOA-secure string commitment scheme. From the above definition we note the following:

1. As stated in [Xia11], the above definition is the weakest one since the order of the quantifier is such that the simulator *knows* the message distribution $\mathcal{B}$. Such a definition is motivated by the fact the it makes the lower bounds proved in [Xia11] stronger. However, the constructions shown in [Xia11] are claimed to achieve the strong(er) definition in which the message distribution $\mathcal{B}$ is not known by Sim.
2. The set I given as output in the ideal world experiment is not controlled by Sim but corresponds to the set of queries made by Sim to the oracle. If this was not the case then a simulator can just ask for all the openings at the beginning, perfectly simulate the sender and give as output the set asked by R^* instead of the queries actually made to the oracle. This restriction essentially means that Sim should be very careful in querying the oracle since each query will appear in the final output and there is no possibility to abort or rewind the simulation, as instead it is possible with the transcript of the conversation with R^*.

Theorem 4. *There exists no string commitment protocol that is SOA-secure under strong concurrent composition.*

Proof Idea. To prove this theorem we adapt the proof provided by Lindell in [Lin03] to the case of SOA-commitment scheme. [Lin03] shows that there exist functionalities for which proving that a protocol is secure under m-concurrent composition using a black-box simulator requires that the protocol has at least m rounds. As corollary it holds that for such functionalities unbounded concurrency proved using a black-box simulator is impossible to achieve. Such a theorem cannot be directly applied to the case of SOA-secure commitments since it is provided only for two functionalities in which both parties have private inputs (blind signatures and OT functionalities). In the setting of SOA-secure commitments the receiver has no private input and there is no ideal functionality involved. Therefore in our proof, we convert the role of the functionality into the role of the oracle $\mathcal{O}$, and in deriving the contradiction we do not break the privacy of the receiver but the correctness of the protocol (i.e. the binding).

The proof is based on the following two observations. First of all, since the simulator is black-box the only advantage that it can exploit to carry out a successful simulation is to rewind the adversary. Moreover, rewinds must be effective, in the sense that upon each rewind the simulator should change the transcript in order to "extract" information from the adversary (obviously if the transcript is not changed then the

rewind is useless). The second crucial observation is that in SOA commitments, the commitments that must be opened are decided by R^* and in order to obtain the string to be produced an opening to, Sim must query an external oracle (recall that we are considering the strong definition in which the simulator does not know the message distribution). Moreover, the distribution of I due to Sim in the ideal world should not be distinguishable from that due to the malicious receiver interacting with the sender in the real world. Thus the idea of the proof is to show that there exists an adversarial strategy that makes the rewinding attempts of any black-box Sim ineffective, unless Sim queries the oracle a number of time that is distinguishable from the number of openings asked by the adversary in the real experiment. Then the next step is to show that if nevertheless there exists a simulator that is able to deal with such an adversary (without rewinding), then such a simulator can be used by a malicious sender to break the binding of the protocol. For lack of space, the formal proof can be found in Appendix D.6.

Theorem 5. *There exists no SOA-secure commitment scheme in the strong fully concurrent setting.*

References

[BHY09] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *EUROCRYPT*, pages 1–35, 2009.

[CO99] Giovanni Di Crescenzo and Rafail Ostrovsky. On concurrent zero-knowledge with preprocessing. In *CRYPTO*, pages 485–502, 1999.

[DNRS99] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry Stockmeyer. Magic functions. In *Foundations of Computer Science (FOCS'99)*, pages 523–534, 1999.

[DNRS03] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry Stockmeyer. Magic functions. *J. ACM*, 50(6):852–921, 2003.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *in 22nd STOC*, pages 416–426. ACM Press, 1990.

[GK96] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for np. *J. Cryptology*, 9(3):167–190, 1996.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *STOC*, pages 25–32, 1989.

[GM06] Rosario Gennaro and Silvio Micali. Independent zero-knowledge sets. In *ICALP*, volume 4052 of *Lecture Notes in Computer Science*, pages 181–234. Springer, 2006.

[Hof11] Dennis Hofheinz. Possibility and impossibility results for selective decommitments. *J. Cryptology*, 24(3):470–516, 2011.

[Lin03] Yehuda Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *STOC*, pages 683–692, 2003.

[LS90] Dror Lapidot and Adi Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In *CRYPTO*, pages 353–365, 1990.

[MOSV06] Daniele Micciancio, Shien Jin Ong, Amit Sahai, and Salil Vadhan. Concurrent zero knowledge without complexity assumptions. In Shai Halevi and Tal Rabin, editors, *Theory of cryptography conference - Proceedings of TCC 2006*, volume 3876 of *Lecture Notes in Computer Science*, pages 1–20, New York, NY, USA, March 2006. Springer.

[MY04] Philip D. MacKenzie and Ke Yang. On simulation-sound trapdoor commitments. In *EUROCRYPT'04*, pages 382–400, 2004.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '91, pages 129–140, London, UK, 1992. Springer-Verlag.

[PRS02] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *In 43rd FOCS*, pages 366–375, 2002.

[PW09] Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In *TCC*, pages 403–418, 2009.

[Xia11] David Xiao. (Nearly) round-optimal black-box constructions of commitments secure against selective opening attacks. In *TCC*, pages 541–558, 2011.

A Other Definitions and Tools

A.1 Black-Box Reductions

Generally, a reduction from a primitive $\mathcal{X}$ to a primitive $\mathcal{Y}$ (also referred to as a construction of primitive $\mathcal{X}$ using primitive $\mathcal{Y}$) involves showing that if there exists an implementation A of $\mathcal{Y}$, then there exists an implementation B^A of $\mathcal{X}$. This is equivalent to saying that for every adversary that breaks B^A, there exists an adversary that breaks A. Such a reduction is fully-black-box if it ignores the internal structure of $\mathcal{Y}$'s implementation and if the proof of correctness is black-box as well (i.e., the adversary for breaking $\mathcal{Y}$ ignores the internal structure of both $\mathcal{Y}$'s implementation and of the adversary breaking $\mathcal{X}$).

A.1.1 Weak Trapdoor Commitment Schemes

Definition 4 (Weak Trapdoor Commitment)**.** *A tuple of PPT algorithms* $\mathsf{wTCom}=(\mathsf{wTCgen}, \mathsf{S}, \mathsf{R}, \mathsf{TCfakeCom}, \mathsf{TCfakeDec})$ *is a weak trapdoor commitment scheme if* $\mathsf{TCgen}(1^n)$ *outputs a public key/secret key pair (pk,sk),* wTCgen_{pk} *is the related functionality that restricts the output of* wTCgen *to the public key,* $(\mathsf{wTCgen}_{pk}, \mathsf{S}, \mathsf{R})$ *is a commitment scheme and* $\mathsf{TCfakeCom}, \mathsf{TCfakeDec}$ *are such that:*

Weak Trapdoor Property. *For any* $b \in \{0,1\}$*, for all* $r \in \{0,1\}^n$ *and* $(pk, sk) \leftarrow \mathsf{TCgen}(r)$*; for any PPT malicious receiver* R^* *there exists a negligible function* ϵ *such that the following holds:*

$$\mathbf{Adv}^{\mathsf{wtrap}}_{\mathsf{wTCom},\mathsf{R}^*} = \Pr[\mathtt{out}(\mathbf{Exp}^{\mathsf{wTrap}}_{\mathsf{wTCom}}(n)) = 1] - \Pr[\mathtt{out}(\mathbf{Exp}^{\mathsf{Com}}_{\mathsf{wTCom}}(n)) = 1] \leq \epsilon(n)$$

Experiment $\mathbf{Exp}^{\mathsf{Com}}_{\mathsf{wTCom}}(n)$*:*	*Experiment* $\mathbf{Exp}^{\mathsf{wTrap}}_{\mathsf{wTCom}}(n)$*:*
run $\langle \mathsf{S}(pk, \mathtt{com}, b), \mathsf{R}^*(pk, sk, b, \mathtt{recv})\rangle$*;*	*run* $(\xi, \cdot) \xleftarrow{\$} \langle \mathsf{TCfakeCom}(pk, sk, \mathtt{com}), \mathsf{R}^*(pk, sk, b, \mathtt{recv})\rangle$*;*
$b' \xleftarrow{\$} \langle \mathsf{S}(\mathtt{open}), \mathsf{R}^*(\mathtt{open})\rangle$*;*	$b' \xleftarrow{\$} \langle \mathsf{TCfakeDec}(\mathtt{open}, b, \xi), \mathsf{R}^*(\mathtt{open})\rangle$*;*
output b'*;*	*output* b'*;*

As before, the variable ξ denotes the state shared by algorithms $\mathsf{TCfakeCom}$ and $\mathsf{TCfakeDec}$.

It is possible to show that there exist weak trapdoor commitment schemes that are not also "regular" trapdoor commitment schemes. Take any "regular" trapdoor commitment scheme in which the decommitment phase is non-interactive. A weak commitment scheme can be constructed adding as a final round of the commitment phase the (statistically binding) commitment of the opening. The idea is that here the sender is bounded to the opening already in the commitment phase, therefore obtaining the trapdoor only in the opening phase does not allow to equivocate.

A.2 Witness Indistinguishable Proof of Knowledge

For a NP-language L let be $\mathcal{R}_L$ the witness-relation that contains all the pairs (x, w) such that $x \in L$ and w is a witness for that. We indicate with $w \in \mathcal{R}_L(x)$ that $\mathcal{R}_L(x, w) = 1$.

Definition 5 (Interactive Proof of Knowledge System [FS90])**.** *A tuple of interactive algorithms $(\mathcal{P}, \mathcal{V}, E)$ is an interactive proof of knowledge system for a* **NP** *language L, with witness-relation $\mathcal{R}_L$, if the following conditions hold:*

- *Completeness. $\forall x \in L$, $\forall w \in \mathcal{R}_L(x)$ it holds that $\mathsf{Pr}[\langle \mathcal{P}(x, w), \mathcal{V}(x) \rangle = 1] = 1$.*
- *Soundness (Proof of Knowledge). There exists an expected PPT extractor E such that $\forall$ malicious provers P^*, there exists a negligible function ϵ such that for all auxiliary inputs $z \in \{0,1\}^*$ it holds that:*

$$\mathsf{Pr}[\langle \mathsf{P}^*(x, z), \mathcal{V}(x) \rangle = 1] - \mathsf{Pr}[E(x, \mathsf{P}^*(x, z)) \in \mathcal{R}_L] \geq 1 - \epsilon(|x|).$$

The probability is taken over the coin tosses of $\mathcal{V}$, P^*, E. The extractor has black-box access to the malicious prover P^*.

Definition 6 (Witness Indistinguishable Proof of Knowledge System $\mathcal{WIPOK}$ [FS90])**.** *A proof of knowledge system $(\mathcal{P}, \mathcal{V}, E)$ for a* **NP** *language L and with witness relation $\mathcal{R}_L$, is witness-indistinguishable if for every PPT malicious verifier V^*, there exists a negligible function ϵ such that, $\forall x, \forall w_0, w_1 \in \mathcal{R}_L(x)$ and $z \in \{0,1\}^*$:*

$$\mathsf{Pr}[\langle \mathcal{P}(x, w_0), \mathsf{V}^*(x, z) \rangle = 1] - \mathsf{Pr}[\langle \mathcal{P}(x, w_1), \mathsf{V}^*(x, z) \rangle = 1] < \epsilon(|x|)$$

The probability is taken over the coin tossed by V^* and $\mathcal{P}$ (the auxiliary input z given to V^* could contain x, w_0, w_1).

A.2.1 Special 3-Round $\mathcal{WIPOK}$ [LS90]

In the following we describe the 3-round $\mathcal{WIPOK}$ protocol for the NP-complete language graph Hamiltonicity (HC), provided by Lapidot and Shamir in [LS90], and we will refer to this construction as FLS protocol. The reason why this construction is special, is that only the size of the statement need to be known before the last round. The actual statement can therefore be decided during the execution of a larger protocol, and this is very important when one aims at optimizing the overall round complexity.

We now show the protocol assuming that the instance G is known from the beginning, and we discuss later why its knowledge can be postponed to the very last round.

FLS protocol consists of k **parallel executions** (with the same input G) of the following protocol:

Inputs: $\mathcal{V}$, $\mathcal{P}$ have as input a graph G, $\mathcal{P}$ has as auxiliary input a witness $w \in \mathcal{R}_{\mathsf{HC}}(G)$. Let n be the number of vertexes of G. G is represented by a $n \times n$ adjacency matrix M where $\mathsf{M}[i][j] = 1$ if there exists an edge between vertexes i and j in G. A non-edge position i, j is a pair of vertexes that are not connected in G and for which $\mathsf{M}[i][j] = 0$.

FLS1 ($\mathcal{P} \to \mathcal{V}$): $\mathcal{P}$ picks a random n-vertex cycle graph H and commits bit-by-bit to the corresponding adjacency matrix using a statistically binding commitment scheme.

FLS2 ($\mathcal{V} \to \mathcal{P}$): $\mathcal{V}$ responds with a randomly chosen bit b.

FLS3 ($\mathcal{P} \to \mathcal{V}$):

- if $b = 0$, $\mathcal{P}$ opens all the commitments, showing that the matrix committed in step FLS1 is actually an n-vertex cycle.
- if $b = 1$, $\mathcal{P}$ sends a permutation π mapping the vertex of H in G. Then it opens the commitment of the adjacency matrix of H corresponding to the non-edges of the graph G.
- $\mathcal{V}$ accepts if and only if all k sessions are accepting.

FLS protocol has the following properties:

Concurrent WI: The protocol enjoys concurrent witness indistinguishability. Indeed, the single execution is zero-knowledge which implies WI and is preserved under parallel and concurrent composition.

Proof of knowledge: Getting the answer for both $b = 0$ and $b = 1$ allows the extraction of the cycle. The reason is the following. For $b = 0$ one gets the random cycle H. Then for $b = 1$ one gets the permutation mapping the random cycle in the actual cycle w that is given to $\mathcal{P}$ at the beginning (or before the last message of) the protocol.

Knowledge of witness and theorem is required only in step FLS3: The crucial property is that the first step is independent of the witness and the theorem, since it only requires the sampling of a random n-cycle (n is the size of the theorem and must be known in advance). The witness is used *only* in the last step FLS3. Looking ahead this allows the simulator to equivocate in the decommitment phase in which it will be required to perform FLS3, without changing the transcript of the commitment phase. This property turns out to be very important to achieve SOA-secure protocols.

A.3 Constant-Round Extractable Commitment Schemes

A key component of three of our protocols - Prot. 2, Prot. 6 and Prot. 3 - is a constant-round statistically-binding extractable commitment scheme. Roughly speaking, extractability means that given a black-box access to an adversarial sender, with a restriction that we can execute only commitment phases, we can extract the bit committed by the sender. In the sequel we provide the definition of extractable commitment schemes and we show a constant-round construction that satisfies it. Since we would only need the scheme to be statistically-binding, we shall focus only on the statistically-binding variant.

Definition 7 (Extractable Commitment Scheme [PW09])**.** $\mathsf{ExtCom} = (\mathsf{Gen}_{\mathsf{ext}}, \mathsf{S}_{\mathsf{ext}}, \mathsf{R}_{\mathsf{ext}})$ *is said to be an extractable commitment scheme if* $(\mathsf{Gen}_{\mathsf{ext}}, \mathsf{S}_{\mathsf{ext}}, \mathsf{R}_{\mathsf{ext}})$ *is a statistically-binding commitment scheme that satisfies the following property:*

Extractability: *there exists an expected polynomial-time extractor E that has oracle access to an adversarial sender $\mathsf{S}_{\mathsf{ext}}{}^*$ only for the commitment-phase and outputs a commitment-phase transcript τ together with a bit b' such that τ is identically distributed to the view of the interaction $\langle \mathsf{S}_{\mathsf{ext}}{}^*(\mathtt{com}, \cdot), \mathsf{R}_{\mathsf{ext}}(\mathtt{recv}) \rangle$ and if τ is accepting then the probability that $b' = \perp$ is negligible. Finally if $b' \neq \perp$ then it is statistically impossible to open τ to any value other then b'.*

We consider the interactive extractable commitment scheme that was used in [PW09]. We remark here that this is simpler than the concurrently-extractable commitments introduced by [MOSV06], where the sender may adversarially interleave multiple sessions with a receiver. Indeed, in [MOSV06], upon rewinding, the sender may initiate new sessions, and it is needed to extract in these new sessions too. [MOSV06] gave a construction that required a super-logarithmic number of rounds. However, in our setting, as we shall explain later in detail, we require only a commitment scheme that is extractable in the stand-alone setting, which can be constructed in a constant number of rounds. The crucial difference consists in the fact that even in case there are several sessions, and it is needed to extract from all of them, there is no need to extract from the ones that start during rewinding threads. The simplification comes from the setting of concurrency with barrier (while [MOSV06] works in the full-fledged concurrent setting).

Protocol 4. *[Extractable Commitment* ExtCom*]*
Let $\mathsf{Com}_{\mathsf{NI}} = (\mathsf{S}_{\mathsf{NI}}, \mathsf{R}_{\mathsf{NI}})$ *be any statistically-binding commitment scheme with non-interactive commitment and opening phases. Such schemes exist based on one-way permutations.*

$\mathsf{S}_{\mathsf{soa}}$**'s input:** $b \in \{0,1\}$.

Commitment Phase.

$\mathsf{S}_{\mathsf{ext}}$: *For* $i = 1, \ldots, n$*:*

1. *sample* $b_i^0, b_i^1 \xleftarrow{\$} \{0,1\}$ *such that* $b_i^0 \oplus b_i^1 = b$*;*
2. *run* $c_i^0 \xleftarrow{\$} \langle \mathsf{S}_{\mathsf{NI}}(\mathtt{com}, b_i^0), \mathsf{R}_{\mathsf{NI}}(\mathtt{recv}) \rangle$ *and* $c_i^1 \xleftarrow{\$} \langle \mathsf{S}_{\mathsf{NI}}(\mathtt{com}, b_i^1), \mathsf{R}_{\mathsf{NI}}(\mathtt{recv}) \rangle$ *with* $\mathsf{R}_{\mathsf{ext}}$*;*

$\mathsf{R}_{\mathsf{ext}}$: $e \xleftarrow{\$} \{0,1\}^n$ *and send it to* $\mathsf{S}_{\mathsf{ext}}$*;*

$\mathsf{S}_{\mathsf{ext}}$: *For* $i = 1, \ldots, n$*: run* $\beta(e_i) \leftarrow \langle \mathsf{S}_{\mathsf{NI}}(\mathtt{open}, b_i^{e_i}), \mathsf{R}_{\mathsf{NI}}(\mathtt{open}) \rangle$ *with* $\mathsf{R}_{\mathsf{ext}}$*;*

$\mathsf{R}_{\mathsf{ext}}$: *If any of the openings is invalid then* $\mathtt{ABORT}$*;*

Decommitment Phase.

$\mathsf{S}_{\mathsf{ext}}$: *For* $i = 1, \ldots, n$*: run* $\gamma(e_i) \leftarrow \langle \mathsf{S}_{\mathsf{NI}}(\mathtt{open}, b_i^{1-e_i}), \mathsf{R}_{\mathsf{NI}}(\mathtt{open}) \rangle$ *with* $\mathsf{R}_{\mathsf{ext}}$*;*

$\mathsf{R}_{\mathsf{ext}}$: *If any of the openings is invalid, then output* $\perp$*. Otherwise, if for all* $i \in [n]$ *it holds that* $\beta(e_i) \oplus \gamma(e_i) = \tilde{b}$*; output* $\tilde{b}$*. Otherwise, output* $\perp$*.*

Remark 3. *Following the terminology of [MOSV06] we call the decommitments to* $\mathsf{Com}_{\mathsf{NI}}$ *commitments,* minor decommitments*, and the decommitments to all the* $2n$ $\mathsf{Com}_{\mathsf{NI}}$ *commitments in* ExtCom*,* major decommitments*.*

In the rest of the paper we use the above protocol ExtCom as sub-protocol. We use the notation $\langle \mathsf{S}_{\mathsf{ext}}(\mathtt{com}, b), \mathsf{R}_{\mathsf{ext}}(\mathtt{recv}) \rangle$ to denote the 3-round commitment phase of the protocol shown above, and $\langle \mathsf{S}_{\mathsf{ext}}(\mathtt{open}), \mathsf{R}_{\mathsf{ext}}(\mathtt{open}) \rangle$ to denote the non-interactive opening phase.

Theorem 6 (ExtCom is a statistically-binding extractable commitment scheme)**.** *If* $\mathsf{Com}_{\mathsf{NI}} = (\mathsf{S}_{\mathsf{NI}}, \mathsf{R}_{\mathsf{NI}})$ *is a statistically-binding commitment scheme, then* ExtCom *is a statistically-binding extractable commitment scheme.*

Proof. We prove hiding, binding, and extractability of ExtCom in the following. In the proof we use the following notation. We denote by $\alpha = ((c_1^0, c_1^1), \ldots, (c_n^0, c_n^1))$, the vector of minor commitments executed in the first round of the protocol; by $\beta(e) := (\beta(e_1), \ldots, \beta(e_n))$ the openings (minor decommitments) received in the commitment phase (third round); and by $\gamma(e) = (\gamma(e_1), \ldots, \gamma(e_n))$ the openings (minor decommitments) received in the decommitment phase.

Hiding. We show that if $\mathsf{Com_{NI}}$ satisfies hiding, then ExtCom also satisfies hiding.

Suppose there exists a PPT adversary $\mathsf{R_{ext}}^*$ that breaks hiding of ExtCom with probability δ (i.e., $\mathbf{Adv}^{\text{hiding}}_{\mathsf{ExtCom},\mathsf{R_{ext}}^*} = \delta$). Then we construct an efficient adversary $\mathsf{R_{NI}}^*$ that breaks hiding of $\mathsf{Com_{NI}}$ with probability $\delta/2n$.

The proof proceeds by a standard hybrid argument. Consider the following series of hybrids, H_i, for $i \in [0, n]$:

H_i**:** The sender in this experiment, referred to as $\mathsf{S}^{\mathsf{i}}_{\mathsf{ext}}$, behaves as $\mathsf{S_{ext}}$ with the only exception that during the first round message of the commit phase ($\mathsf{S_{ext}}1$), it chooses b_j^0, $b_j^1 \xleftarrow{\$} \{0,1\}$, such that of $b_j^0 \oplus b_j^1 = 0$ for $j > i$, and b_j^0, $b_j^1 \xleftarrow{\$} \{0,1\}$, such that of $b_j^0 \oplus b_j^1 = 1$ for $j \leq i$. Finally, when $\mathsf{R_{ext}}^*$ outputs a bit b', the same is set to be the output of the experiment. We denote by $\mathsf{Hyb}^i_{\mathsf{ExtCom},\mathsf{R_{ext}}^*} \to 1$ the event that the output of this hybrid experiment is 1.

Observe that H_0 corresponds to $\langle \mathsf{S_{ext}}(\mathtt{com}, 0), \mathsf{R_{ext}}(\mathtt{recv}) \rangle$ and H_n corresponds to $\langle \mathsf{S_{ext}}(\mathtt{com}, 1), \mathsf{R_{ext}}(\mathtt{recv}) \rangle$.

In the hiding experiment of $\mathsf{Com_{NI}}$, $\mathsf{R_{NI}}^*$ is given a commitment $c^* \leftarrow \langle \mathsf{R_{NI}}(\mathtt{recv}), \mathsf{S_{NI}}(\mathtt{com}, b) \rangle$ for a random bit b, and its objective is to guess b. It does so by interacting with $\mathsf{R_{ext}}^*$ as follows:

1. Sample $\mu \xleftarrow{\$} [n]$.
2. Compute the first round message $\mathsf{S_{ext}}1$ as follows:
   - For $i > \mu$, sample b_i^0, $b_i^1 \xleftarrow{\$} \{0,1\}$ such that $b_i^0 \oplus b_i^1 = 0$.
   - For $i < \mu$, sample b_i^0, $b_i^1 \xleftarrow{\$} \{0,1\}$ such that $b_i^0 \oplus b_i^1 = 1$.
   - For $i = \mu$, $\forall j \in \{0,1\}$, and for $(i,j) = (\mu, 1-\theta)$, sample $\theta \xleftarrow{\$} \{0,1\}$ and set $c_i^\theta \leftarrow c^*$. Also, sample $b_i^{1-\theta} \xleftarrow{\$} \{0,1\}$.
3. For $\forall i \in [n] - \{\mu\}$, $\forall j \in \{0,1\}$, and for $(i,j) = (\mu, 1-\theta)$, run $c_i^j \xleftarrow{\$} \langle \mathsf{R_{NI}}(\mathtt{recv}), \mathsf{S_{NI}}(\mathtt{com}, b_i^j) \rangle$ with $\mathsf{R_{ext}}^*$. Also, send c_i^θ to $\mathsf{R_{ext}}^*$.
4. Upon receiving a challenge e from $\mathsf{R_{ext}}^*$, check whether $e_\mu = \theta$. If so, then output a random bit and halt; otherwise, once $\mathsf{R_{ext}}^*$ outputs a bit b', output $b' \oplus b_\mu^{1-\theta}$.

Denote the event that $\mathsf{R_{NI}}^*$ outputs a bit $\tilde{b}$ in the above interaction by $\mathbf{Exp}^{\text{hiding-}b}_{\mathsf{Com_{NI}},\mathsf{R_{NI}}^*(\mathsf{R_{ext}}^*)} \to \tilde{b}$.

Note that if $b \oplus b_\mu^\theta = 1$, then $\mathsf{R_{NI}}^*$ has played H_μ; otherwise, it has played $H_{\mu-1}$.

Now we analyze the success probability of $\mathsf{R_{NI}}^*$, $\mathbf{Adv}^{\text{hiding}}_{\mathsf{Com_{NI}},\mathsf{R_{NI}}^*}$.

$$
\begin{aligned}
&\mathbf{Adv}^{\text{hiding}}_{\mathsf{Com_{NI}},\mathsf{R_{NI}}^*}\\
=&\mathsf{Pr}[\mathbf{Exp}^{\text{hiding-0}}_{\mathsf{Com_{NI}},\mathsf{R_{NI}}^*} \to 1] - \mathsf{Pr}[\mathbf{Exp}^{\text{hiding-1}}_{\mathsf{Com_{NI}},\mathsf{R_{NI}}^*} \to 1]\\
=&\frac{1}{n}\sum_{i=1}^{n}(\mathsf{Pr}[\mathbf{Exp}^{\text{hiding-0}}_{\mathsf{Com_{NI}},\mathsf{R_{NI}}^*(\mathsf{R_{ext}}^*)} \to 1|\mu = i, b_\mu^{1-\theta} = 1, \theta \neq e_\mu].\mathsf{Pr}[b_\mu^{1-\theta} = 1].\mathsf{Pr}[\theta \neq e_\mu]\\
&+ \mathsf{Pr}[\mathbf{Exp}^{\text{hiding-0}}_{\mathsf{Com_{NI}},\mathsf{R_{NI}}^*(\mathsf{R_{ext}}^*)} \to 1|\mu = i, b_\mu^{1-\theta} = 0, \theta \neq e_\mu].\mathsf{Pr}[b_\mu^{1-\theta} = 0].\mathsf{Pr}[\theta \neq e_\mu])\\
-&\frac{1}{n}\sum_{i=1}^{n}(\mathsf{Pr}[\mathbf{Exp}^{\text{hiding-1}}_{\mathsf{Com_{NI}},\mathsf{R_{NI}}^*(\mathsf{R_{ext}}^*)} \to 1|\mu = i, b_\mu^{1-\theta} = 1, \theta \neq e_\mu].\mathsf{Pr}[b_\mu^{1-\theta} = 1].\mathsf{Pr}[\theta \neq e_\mu]\\
&+ \mathsf{Pr}[\mathbf{Exp}^{\text{hiding-1}}_{\mathsf{Com_{NI}},\mathsf{R_{NI}}^*(\mathsf{R_{ext}}^*)} \to 1|\mu = i, b_\mu^{1-\theta} = 0, \theta \neq e_\mu].\mathsf{Pr}[b_\mu^{1-\theta} = 0].\mathsf{Pr}[\theta \neq e_\mu])\\
=&\frac{1}{4n}\sum_{i=1}^{n}(\mathsf{Pr}[\mathbf{Exp}^{\text{hiding-0}}_{\mathsf{Com_{NI}},\mathsf{R_{NI}}^*(\mathsf{R_{ext}}^*)} \to 1|\mu = i, b_\mu^{1-\theta} = 1, \theta \neq e_\mu]\\
&+ \mathsf{Pr}[\mathbf{Exp}^{\text{hiding-0}}_{\mathsf{Com_{NI}},\mathsf{R_{NI}}^*(\mathsf{R_{ext}}^*)} \to 1|\mu = i, b_\mu^{1-\theta} = 0, \theta \neq e_\mu])\\
-&\frac{1}{4n}\sum_{i=1}^{n}(\mathsf{Pr}[\mathbf{Exp}^{\text{hiding-1}}_{\mathsf{Com_{NI}},\mathsf{R_{NI}}^*(\mathsf{R_{ext}}^*)} \to 1|\mu = i, b_\mu^{1-\theta} = 1, \theta \neq e_\mu]\\
&+ \mathsf{Pr}[\mathbf{Exp}^{\text{hiding-1}}_{\mathsf{Com_{NI}},\mathsf{R_{NI}}^*(\mathsf{R_{ext}}^*)} \to 1|\mu = i, b_\mu^{1-\theta} = 0, \theta \neq e_\mu])\\
=&\frac{1}{4n}\sum_{i=1}^{n}(\mathsf{Pr}[\mathsf{Hyb}^{i}_{\mathsf{ExtCom},\mathsf{R_{ext}}^*} \to 0] + \mathsf{Pr}[\mathsf{Hyb}^{i-1}_{\mathsf{ExtCom},\mathsf{R_{ext}}^*} \to 1])\\
&-\frac{1}{4n}\sum_{i=1}^{n}(\mathsf{Pr}[\mathsf{Hyb}^{i-1}_{\mathsf{ExtCom},\mathsf{R_{ext}}^*} \to 0] + \mathsf{Pr}[\mathsf{Hyb}^{i}_{\mathsf{ExtCom},\mathsf{R_{ext}}^*} \to 1])\\
=&\frac{1}{4n}(\mathsf{Pr}[\mathsf{Hyb}^{0}_{\mathsf{ExtCom},\mathsf{R_{ext}}^*} \to 1] - \mathsf{Pr}[\mathsf{Hyb}^{n}_{\mathsf{ExtCom},\mathsf{R_{ext}}^*} \to 1])\\
&+\frac{1}{4n}(\mathsf{Pr}[\mathsf{Hyb}^{1}_{\mathsf{ExtCom},\mathsf{R_{ext}}^*} \to 0] + \mathsf{Pr}[\mathsf{Hyb}^{0}_{\mathsf{ExtCom},\mathsf{R_{ext}}^*} \to 0])\\
\leq&\frac{2\delta}{4n} = \frac{\delta}{2n}
\end{aligned}
$$

That proves the hiding property of ExtCom.

Binding. We show that if $\mathsf{Com_{NI}}$ is statistically binding, then ExtCom also statistically binding; i.e., we show that if an adversarial sender $\mathsf{S_{ext}}^*$ can major-decommit to two distinct bits, then we construct an adversary $\mathsf{S_{NI}}^*$ that breaks the binding property of $\mathsf{Com_{NI}}$. In fact, we prove (and later use) a stronger statement, which, instead of requiring two inconsistent major decommitments, only requires one major decommitment together with minor decommitments for one of the n sharings of the bit $b = b_i^0 \oplus b_i^1$. The details follow.

$\mathsf{S_{NI}}^*$ interacts with $\mathsf{S_{ext}}^*$ as follows.

1. Execute $\mathsf{R_{ext}}$ in $\langle \mathsf{S_{ext}}^*(\mathtt{com}), \mathsf{R_{ext}}(\mathtt{recv})\rangle$ resulting in transcript $\tau_{\mathtt{com}} = ((c_1^0, c_1^1), \ldots, (c_n^0, c_n^1)), e, \beta(e))$. Let a minor decommitment for $c_1^{e_i}$ in $\tau_{\mathtt{com}}$, $\beta(e_i)$, be an opening to bit $b_i^{e_i}$.
2. Given a valid ExtCom decommitment $\tau_{\mathtt{open}} \leftarrow \langle \mathsf{S_{ext}}^*(\mathtt{open}, b), \mathsf{R_{ext}}(\mathtt{open})\rangle$ to a bit b and a valid minor-decommitment γ to $c_1^{1-e_i}$ to a bit $\tilde{b}$, check whether $b = b_i^{e_i} \oplus \tilde{b}$. If so, then abort. Otherwise, output the minor opening $\gamma(e_i)$ in $\tau_{\mathtt{open}}$ and γ, and halt.

If $b \neq b_i^{e_i} \oplus \tilde{b}$, then it implies that the openings $\gamma(e_i)$ and γ are openings to distinct bits for $c_1^{1-e_i}$. Thus, $\mathsf{S_{NI}}^*$ breaks the the binding property of $\mathsf{Com_{NI}}$ with the same probability that $\mathsf{S_{ext}}^*$ breaks binding of ExtCom. That proves the binding property of ExtCom.

Extractability. We show that ExtCom satisfies extractability by constructing an expected polynomial-time extractor E that having black-box access to the adversary $\mathsf{S_{ext}}^*$ in the commitment phase generates an output that is statistically indistinguishable from $\langle \mathsf{R_{ext}}(\mathtt{recv}), \mathsf{S_{ext}}^*(\mathtt{com}) \rangle$ and (if the commitment phase transcript passes the consistency check of $\mathsf{R_{ext}}$) a bit b that $\mathsf{S_{ext}}^*$ would open in the opening phase with all but negligible probability.

The extractor works as follows:

Extractor E

Initialization phase. Choose random tapes $\mathsf{ran_S}$ and ran_E, respectively, for $\mathsf{S_{ext}}^*$ and for the main thread below.

Invoke $\mathsf{S_{ext}}^*(\mathsf{ran_S})$.

Main thread (E1). Run $\mathsf{R_{ext}}$ with randomness ran_E in $\langle \mathsf{S_{ext}}^*(\mathtt{com}), \mathsf{R_{ext}}(\mathtt{recv}) \rangle$ to result in a transcript $\tau_{\mathtt{com}} = (\alpha, e, \beta(e))$. If the consistency check of $\mathsf{R_{ext}}$ fails, then output $(\tau_{\mathtt{com}}, \perp)$ and halt.

Rewinding threads (E2). If $\tau_{\mathtt{com}}$ is accepting, then keep running $\mathsf{R_{ext}}$ with $\mathsf{S_{ext}}^*(\mathsf{ran_S})$ in $\langle \mathsf{S_{ext}}^*(\mathtt{com}), \mathsf{R_{ext}}(\mathtt{recv}) \rangle$, each time with freshly chosen randomness, until we receive another accepting response $\beta(e')$ from $\mathsf{S_{ext}}^*(\mathsf{ran_S})$ with some challenge e'. If $e' = e$, then output $(\tau_{\mathtt{com}}, \perp)$ and halt. Otherwise, run the following procedure:

1. Choose $i \xleftarrow{\$} [n]$ such that $e_i \neq e'_i$.
2. Let b_i and b'_i be the bits opened to in the openings $\beta(e_i)$ and $\beta(e'_i)$, respectively.
3. Output $(\tau_{\mathtt{com}}, b_i \oplus b'_i)$ [7].

Now, let q be the probability over e that we obtain an accepting transcript $\tau_{\mathtt{com}}$. Then the expected number of queries E makes to $\mathsf{S_{ext}}^*$ is $(1-q) + q \cdot 1/q \leq 2$. Also, the probability that E fails to extract (i.e., the probability that $\tau_{\mathtt{com}}$ is accepting and $e = e'$) is at most 2^{-n}. Furthermore, if $\mathsf{S_{ext}}^*$ can open $\tau_{\mathtt{com}}$ to a different value from $b_i \oplus b'_i$, then we can combine this with the opening to $b_i \oplus b'_i$ obtained by E to contradict the binding property of ExtCom. From the above proof of binding property this event occurs only with negligible probability. This completes the description and analysis of the extractor. □

Remark 4 (Extractable string-commitment scheme)**.** *The extractable bit-commitment scheme in Protocol 4 can be trivially extended to an extractable string-commitment scheme just by replacing the shares of the bit to be committed to shares of the string to be committed and by using a non-interactive statistically-binding string-commitment scheme in place of the non-interactive bit-commitment scheme. The expected number of rewindings by the extractor and its failure probability will remain the same and the other properties - hiding and binding - would also trivially follow.*

B Round Optimality of Our $(3,1)$-Round Protocols

We observe that even though as we have shown previously, there is an issue in the impossibility of [Xia11] for $(3,1)$-round SOA-secure commitments, the same result can be used to claim the impossibility of $(2,1)$-round SOA-secure commitments. Indeed, as we have already discussed, the issue in the proof concerns the fact that only the case where the sender speaks first is considered. However, the case in which the

[7] In Prot. 3 we assume a modified version of this extractor that along with the transcript and the extracted bit, outputs also the openings of the pair of the minor commitments (the commitments of the shares) that gives the bit extracted, i.e., $\tau_{\mathtt{com}}$, $\beta(e_i)$, $\beta(e'_i)$, $b_i \oplus b'_i$.

sender speaks first, the receiver answers back and then the sender completes the communication properly contain the two possible 2-round communications: 1) sender first, then receiver; 2) receiver first, then sender. Therefore the (incomplete) proof given in [Xia11] for the impossibility of $(3,1)$-round SOA-secure commitments proves the impossibility of SOA-secure $(2,1)$-round commitments.

C Further Protocols

C.1 (4,1)-Round SOA-Secure Commitments under Any BB Weak Trapdoor Commitment

Let us denote as $\mathsf{wTCom} = (\mathsf{wTCgen}, \mathsf{S_{TC}}, \mathsf{R_{TC}}, \mathsf{TCfakeCom}, \mathsf{TCfakeDec})$ a weak-trapdoor commitment scheme and let n be the security parameter. In the following we show a construction $\mathsf{SOACom} = (\mathsf{S_{soa}}, \mathsf{R_{soa}})$ that uses wTCom as a black-box. If wTCom is (2,1)-round weak trapdoor commitment scheme the following construction is a (4,1)-round commitment scheme. As in the previous construction, we indicate with $\mathsf{S_{TC}}_i^{\sigma}, \mathsf{R_{TC}}_i^{\sigma}$ the i-th invocation of the commitment/decommitment algorithm of the protocol wTCom using the public key pk_i^{σ}. The sender $\mathsf{S_{soa}}$ has in input a bit b.

Protocol 5. *[(4,1)-round SOA-secure commitment scheme based on weak trapdoor commitment]*

Commitment phase.

$\mathsf{R_{soa}}$**:** *for* $i = 1, \ldots, n$*:*

1. r_i^0, $r_i^1 \xleftarrow{\$} \{0,1\}^n$; $(\mathsf{pk}_i^0, \mathsf{sk}_i^0) \leftarrow \mathsf{wTCgen}(r_i^0)$;$(\mathsf{pk}_i^1, \mathsf{sk}_i^1) \leftarrow \mathsf{wTCgen}(r_i^1)$;
2. *send* $\{\mathsf{pk}_i^0, \mathsf{pk}_i^1\}_{i\in[n]}$ *to* $\mathsf{S_{soa}}$;

$\mathsf{S_{soa}}$**:** *Upon receiving* $\{\mathsf{pk}_i^0, \mathsf{pk}_i^1\}_{i\in[n]}$*: send* $d_1, \ldots, d_n$ *to* $\mathsf{R_{soa}}$ *where* $d_i \xleftarrow{\$} \{0,1\}$ $i = 1, \ldots, n$;

$\mathsf{R_{soa}}$**:** *Upon receiving* $d_1, \ldots, d_n$ *send* $\{\mathsf{sk}_i^{d_i}\}_{i\in[n]}$ *to* $\mathsf{S_{soa}}$;

$\mathsf{S_{soa}}$**:** *Upon receiving* $\{r_i^{d_i}\}_{i\in[n]}$*:*

1. **check consistency***: for* $i = 1, \ldots, n$*:* $(\mathsf{pk}_i'^{d_i}, \mathsf{sk}_i'^{d_i}) \leftarrow \mathsf{wTCgen}(r_i^{d_i})$*; if* $\mathsf{pk}_i'^{d_i} \neq \mathsf{pk}_i^{d_i}$ `ABORT`.
2. *secret share the bit* b*: for* $i = 1, \ldots, n$*:* $b_i \xleftarrow{\$} \{0,1\}$, $b_n \leftarrow \bigoplus_{i=1}^{n-1} b_i \oplus b$;
3. *run* $\langle \mathsf{S_{TC}}_i^{\bar{d}_i}(\mathsf{pk}_i^{\bar{d}_i}, \mathtt{com}, b_i), \mathsf{R_{TC}}_i^{\bar{d}_i}(\mathsf{pk}_i^{\bar{d}_i}, \mathtt{recv})\rangle$ *with* $\mathsf{R_{soa}}$;

Decommitment phase.

$\mathsf{S_{soa}}$**:** *for* $i = 1, \ldots, n$*; run* $b_i' \leftarrow \langle \mathsf{S_{TC}}_i^{\bar{d}_i}(\mathtt{open}), \mathsf{R_{TC}}_i^{\bar{d}_i}(\mathtt{open})\rangle$ *with* $\mathsf{R_{soa}}$;

$\mathsf{R_{soa}}$**:** *If all opening phases were successful completed output* $b' \leftarrow \bigoplus_{i=1}^{n} b_i'$*. Else output* $\perp$*.*

Theorem 7 (Protocol 5 is secure under selective opening attacks)**.** *If* $\mathsf{wTCom} = (\mathsf{wTCgen}, \mathsf{S_{TC}}, \mathsf{R_{TC}}, \mathsf{TCfakeCom}, \mathsf{TCfakeDec})$ *is a weak-trapdoor commitment scheme, then protocol 5 is a commitment scheme secure under selective opening attacks.*

The formal proof can be found in Appendix D.2.

C.2 (5,1)-Round SOA-Secure Commitment Scheme on Any BB OWP

In this section we present a $(5,1)$-round SOA-secure commitment scheme based on a BB use of a OWP. This scheme is a slight modification of our $(3,3)$-round scheme given in Protocol 2. More specifically, recall that in the previous construction the sender sends n pairs of extractable commitments to b to the receiver and simultaneously engages in a coin-flipping protocol. Finally, the outcome of coin-flipping would dictate which one of the two commitments in every pair would be opened by the sender. The modification here is the sender is allowed to take either the outcome of coin-flipping or its binary-negation. Intuitively, we introduce this modification particularly since the decommitment phase begins *after* the sender sends the string d to the receiver. The simulator would proceed similarly as $(3,3)$-round counterpart, but here it crafts its random-string as to point to the extractable-commitments of a random bit θ. Once it receives the bit b to which it needs to open, depending on θ it will either open as per the outcome of the coin-flipping protocol or the binary-negation. The rest of the protocol remains the same as Protocol 2. Details to follow in Protocol 6.

Protocol 6. $/\mathsf{SOACom} = (\mathsf{S}_{\mathsf{soa}}, \mathsf{R}_{\mathsf{soa}})/$

$\mathsf{S}_{\mathsf{soa}}$**'s input:** $b \in \{0,1\}$

Commitment phase.

$\mathsf{R}_{\mathsf{soa}}$: *For* $i = 1, \ldots, n$*:*

1. $a_i \xleftarrow{\$} \{0,1\}$*;*
2. *run* $\langle {\mathsf{S}_{\mathsf{ext}}}^i(\mathtt{com}, a_i), {\mathsf{R}_{\mathsf{ext}}}^i(\mathtt{recv})\rangle$ *with* $\mathsf{S}_{\mathsf{soa}}$*;*

$\mathsf{S}_{\mathsf{soa}}$: *For* $i = 1, \ldots, n$*:*

1. *run* $\langle {\mathsf{S}_{\mathsf{ext}}}^{i,0}(\mathtt{com}, b), {\mathsf{R}_{\mathsf{ext}}}^{i,1}(\mathtt{recv})\rangle$ *with* $\mathsf{R}_{\mathsf{soa}}$*;*
2. *run* $\langle {\mathsf{S}_{\mathsf{ext}}}^{i,1}(\mathtt{com}, b), {\mathsf{R}_{\mathsf{ext}}}^{i,1}(\mathtt{recv})\rangle$ *with* $\mathsf{R}_{\mathsf{soa}}$*;*

$\mathsf{S}_{\mathsf{soa}}$: *If all commitment phases played with* $\mathsf{R}_{\mathsf{soa}}$ *are successfully completed, send* $d \xleftarrow{\$} \{0,1\}^n$ *to* $\mathsf{R}_{\mathsf{soa}}$*;*

$\mathsf{R}_{\mathsf{soa}}$: *Open all commitments:*

for $i = 1 \ldots, n$*: run* $\langle {\mathsf{S}_{\mathsf{ext}}}^i(\mathtt{open}), {\mathsf{R}_{\mathsf{ext}}}^i(\mathtt{open})\rangle$ *with* $\mathsf{S}_{\mathsf{soa}}$*;*

Decommitment Phase.

$\mathsf{S}_{\mathsf{soa}}$: *If all openings provided by* $\mathsf{R}_{\mathsf{soa}}$ *are valid, sample and send to* $\mathsf{R}_{\mathsf{soa}}$*,* $\theta \xleftarrow{\$} \{0,1\}$*, and for* $i = 1, \ldots, n$*:*

1. $\sigma_i \leftarrow d_i \oplus a_i \oplus \theta$*;*
2. *run* $\langle {\mathsf{S}_{\mathsf{ext}}}^{i,\sigma_i}(\mathtt{open}), {\mathsf{R}_{\mathsf{ext}}}^{i,\sigma_i}(\mathtt{open})\rangle$ *with* $\mathsf{R}_{\mathsf{soa}}$*;*

$\mathsf{R}_{\mathsf{soa}}$: *For every* i*, if* $\sigma_i = d_i \oplus a_i \oplus \theta$ *and all the corresponding openings provided by* $\mathsf{S}_{\mathsf{soa}}$ *open to the same bit* b*, output* b*. Otherwise output* $\perp$*.*

Theorem 8 (Protocol 6 is secure under selective opening attacks)**.** *If* ExtCom *is an extractable commitment scheme, then Protocol 6 is a commitment scheme secure against selective opening attacks.*

The proof of the theorem is provide in Appendix D.4.

D Proofs

D.1 Proof of Theorem 1

Proof. In the following we will refer as commitment/decommitment the commitment/decommitment phase of the main protocol SOACom, while we will refer as sub-commitment/sub-decommitment the black-box invocation of the commitment/decommitment procedure of sub-protocol TC. The malicious receiver $\mathsf{R}^*_{\mathsf{soa}}$ plays k concurrent sessions of SOACom, more precisely she will run k concurrent executions of the commitment phase, and up to $m = |I|$ concurrent decommitment phases.

Completeness. It follows from the completeness of the sub-protocol TC.

Binding. For the binding property is sufficient to consider the protocol in the stand-alone setting. Therefore we focus on one single session of the protocol SOACom.

We have to show that any PPT malicious sender $\mathsf{S}^*_{\mathsf{soa}}$ is not able to provide two distinct openings for the same commitment transcript with non-negligible probability. Note that the decommitment phase consists only of the opening of n sub-commitments for which $\mathsf{S}^*_{\mathsf{soa}}$ has not seen the secret keys. Therefore, if $\mathsf{S}^*_{\mathsf{soa}}$ is able to provide two distinct valid openings, it must be the case that $\mathsf{S}^*_{\mathsf{soa}}$ is able to open at least one of the sub-commitments as 0 *and* 1, therefore breaking the binding of TC. Due to the binding of TC this event happens with negligible probability.

Formally the reductions goes as follows. Assume that there exists $\mathsf{S}^*_{\mathsf{soa}}$ who breaks the binding of Protocol 1 with non-negligible probability δ. Then there exists at least a pair i, σ for which $\mathsf{S}^*_{\mathsf{soa}}$, in the i-th sub-commitments uses the public key pk_i^σ and is able to run two accepting decommitments: $b_i \leftarrow \langle \mathsf{S}_{\mathsf{TC}i}(\mathtt{open}, 0), \mathsf{R}_{\mathsf{TC}i}(\mathtt{open})\rangle$ and $b'_i \leftarrow \langle \mathsf{S}_{\mathsf{TC}i}(\mathtt{open}, 1), \mathsf{R}_{\mathsf{TC}i}(\mathtt{open})\rangle$ such that $b_i \neq b'_i$. Thus we construct a sender $\mathsf{S}^*_{\mathsf{TC}}$ breaking the binding of the protocol TC with probability $\delta/2n$.

$\mathsf{S}^*_{\mathsf{TC}}$ plays in the experiment $\mathbf{Exp}^{\text{binding}}_{\mathsf{TC}}$ receiving as input the public key pk and interacting with the honest receiver R_{TC}. It runs $\mathsf{S}^*_{\mathsf{soa}}$ as subroutine simulating the receiver $\mathsf{R}_{\mathsf{soa}}$: it honestly generates $2n-1$ pairs of public/secret parameters running TCgen, while for randomly selected $i \in [n]$, $\sigma \in \{0,1\}$ it sets $\mathsf{pk}_i^\sigma = pk$ and finally it sends the $2n-1$ public keys along with pk_i^σ to $\mathsf{S}^*_{\mathsf{soa}}$. Note that this message is distributed identically as the one generated by the honest receiver $\mathsf{R}_{\mathsf{soa}}$. Next $\mathsf{S}^*_{\mathsf{TC}}$ engages in n sub-commitments with $\mathsf{S}^*_{\mathsf{soa}}$, except that the messages for the sub-commitment in position i, σ are forwarded to the honest receiver R_{TC}. If $d_i = \sigma$ then $\mathsf{S}^*_{\mathsf{TC}}$ aborts (indeed it is not able to provide the randomness used to generate $pk = pk_i^\sigma$). Otherwise, $\mathsf{S}^*_{\mathsf{TC}}$ replies as the honest receiver, concluding the commitment phase.

In the opening phase, $\mathsf{S}^*_{\mathsf{TC}}$ is invoked to execute the opening phase with bit 0 and 1, thus it invokes $\mathsf{S}^*_{\mathsf{soa}}$ as well first with bit 0 and then with bit 1. Each time, $\mathsf{S}^*_{\mathsf{TC}}$ forwards to R_{TC} the i-th sub-decommitment received by $\mathsf{S}^*_{\mathsf{soa}}$. $\mathsf{S}^*_{\mathsf{TC}}$ wins the binding experiment for protocol TC if $\mathsf{S}^*_{\mathsf{soa}}$ will provide two distinct openings exactly in the i-th sub-commitment. This happens with probability $1/2n$. Therefore $\mathsf{S}^*_{\mathsf{TC}}$ wins the binding game with probability $\delta/2n$.

Hiding under selective opening attack. We show a PPT simulator Sim that having black-box access to the adversary $\mathsf{R}^*_{\mathsf{soa}}$ generates an output that is distributed as the output generated by the interaction between $\mathsf{R}^*_{\mathsf{soa}}$ and the real world sender $\mathsf{S}_{\mathsf{soa}}$. The simulator works as follows:

SOA-simulator Sim

Initialization phase. Choose random tapes $\mathsf{ran}_{\mathsf{R}}, \mathsf{ran}_{\mathsf{Sim}}$ respectively for $\mathsf{R}^*_{\mathsf{soa}}$ and for the commitment phase. Activate $\mathsf{R}^*_{\mathsf{soa}}(\mathsf{ran}_{\mathsf{R}})$.

Commitment phase (S1). (main thread)

- Upon receiving public keys $\{\mathsf{pk}_i^0, \mathsf{pk}_i^1\}_{i\in[n]}$ from $\mathsf{R}^*_{\mathsf{soa}}$ for some session $j \in [k]$ do:
  1. randomly choose bits $b_1, \ldots, b_n$; $d_1, \ldots, d_n$;
  2. for $i = 1, \ldots, n$: commit to b_i with $\mathsf{pk}_i^{\bar{d}_i}$ engaging in n sub-commitments (invoking S_{TC}) with $\mathsf{R}^*_{\mathsf{soa}}$. Send challenge $d_1, \ldots, d_n$ to $\mathsf{R}^*_{\mathsf{soa}}$. If $\mathsf{R}^*_{\mathsf{soa}}$ aborts, `ABORT` session j.
- Upon receiving $\{r_i^{d_i}\}_{i\in[n]}$ for some session j, check their consistency running $\mathsf{sk}_i^{d_i}, \mathsf{pk}_i^{d_i} \leftarrow \mathsf{TCgen}(r_i^{d_i})$. If the check fails `ABORT` session j. Otherwise store the secret keys $\{\mathsf{sk}_i^{d_i}\}_{i\in[n]}$ for session j.

Commitment phase completion. When the commitment phase of all k sessions is completed, Sim obtains the set of indexes I from $\mathsf{R}^*_{\mathsf{soa}}$ and obtains $\{\mathbf{b}[j]\}_{j\in I}$ from the experiment.

Extraction phase. (rewinding thread)

For $\ell = 1, \ldots, m$; if j_ℓ was not-aborted in the main thread do:

1. activate $\mathsf{R}^*_{\mathsf{soa}}$ with randomness $\mathsf{ran}_{\mathsf{R}}$ and executes all the sessions excepts j_ℓ using $\mathsf{ran}_{\mathsf{Sim}}$.
2. random string $d'_1, \ldots, d'_n$ using fresh randomness;
3. in the j-th session, executes all the commitments using randomness $\mathsf{ran}_{\mathsf{Sim}}$ as in the main thread but change the challenge to $d'_1, \ldots, d'_n$ and thus in turn the public keys used for the commitment (note that the view of $\mathsf{R}^*_{\mathsf{soa}}$ generated in this step is distributed identically as in step (**S1**). If $\mathsf{R}^*_{\mathsf{soa}}$ aborts then go to step 1. If $\mathsf{R}^*_{\mathsf{soa}}$ starts new sessions, follow instructions as in (**S1**)).
4. When $\mathsf{R}^*_{\mathsf{soa}}$ replies with secrets $\{r_i^{d'_i}\}_{i\in[n]}$: if the strings $d'_1 \ldots d'_n$, $d_1 \ldots d_n$ are equals then abort the simulation. Else, if there exists an r_i generating a valid pair $\mathsf{sk}_i^{d'_i}, \mathsf{pk}_i^{d'_i}$ where $\mathsf{pk}_i^{d'_i}$ appeared in the j_ℓ-th session of main thread, and $d'_i \neq d_i$ then return $\mathsf{sk}_i^{d'_i}$. Otherwise go to step 1.

Extraction completion. If Sim reaches this point, then for each (not-aborted) session $j \in I$ there is at least one i such that Sim obtained both sk_i^0 and sk_i^1.

Decommitment phase (S2) (main thread). Run $\mathsf{R}^*_{\mathsf{soa}}(\mathsf{ran}_{\mathsf{R}})$ till the completion of the commitment phase using $\mathsf{ran}_{\mathsf{Sim}}$. Then for non-aborted sessions $j_\ell \in I$, with $\ell \in [m]$, let $\mathbf{b}[j_\ell]$ be the bit to decommit to in the session j_ℓ, let i the index such that Sim obtained $\mathsf{sk}_i^0, \mathsf{sk}_i^1$ for the session j_ℓ.

When $\mathsf{R}^*_{\mathsf{soa}}$ asks for the decommitment of the j_ℓ-th session do as follows:

1. for all $l \neq i$ honestly run the sub-decommitment algorithm: $b_l \xleftarrow{\$} \langle \mathsf{S}_{\mathsf{TC}}{}_l^{\bar{d}_l}(\mathsf{open}), \mathsf{R}_{\mathsf{TC}}{}_l^{\bar{d}_l}(\mathsf{open})\rangle$; where d_l is the challenge sent in the commitment phase of the main thread. Compute $b'_i \leftarrow (\bigoplus_{l=1}^{n-1} b_l) \oplus \mathbf{b}[j]$.
2. to open the i-th sub-commitment run the sub-fake-decommitment algorithm using the trapdoor information $\mathsf{sk}_i^{\bar{d}_i}$: $b'_i \leftarrow \langle \mathsf{TCfakeDec}(\mathsf{sk}_i^{\bar{d}}, \mathsf{open}, b'_i), \mathsf{R}_{\mathsf{TC}}{}_i^{\bar{d}_i}(\mathsf{open})\rangle$; If $\mathsf{R}^*_{\mathsf{soa}}$ aborts, then aborts this session.

Finally, outputs whatever $\mathsf{R}^*_{\mathsf{soa}}$ outputs.

Proposition 1. *The simulator* Sim *runs in expected polynomial time in* n.

Proof. Sim consists of three phases: commitment phase, extraction phase, decommitment phase. Let us denote as t_c, t_d, t_{fd}, t_g the running time required to execute respectively the sub-commitment, the sub-decommitment, the fake-decommitment and the generator algorithm of the protocol TC. By definition of TC all these running time are polynomial in n.

In the commitment phase, for each session, Sim executes $2n$ sub-commitments and verifies the validity of the answer of $\mathsf{R}^*_{\mathsf{soa}}$ running the generation algorithm TCgen n times. Plus there is a linear time due to the choice of the random challenge. Thus, the running time of the commitment phase for one session is: $\mathsf{t}_{\mathtt{SimCom}} = \mathsf{t}_c \cdot 2n + \mathsf{t}_g \cdot n + \Theta(n)$. Hence, the total running time for the commitment phase is $k \cdot \mathsf{t}_{\mathtt{SimCom}}$, that is polynomial.

After the completion of the commitment phase Sim launches the rewinding threads in order to extract one trapdoor for each session that has been asked for the decommitment. The number of decommitment asked by $\mathsf{R}^*_{\mathsf{soa}}$ is $m = |I|$.

Sim extracts the trapdoors one session at time, hence it runs the extraction procedure at most m times. For each (non-aborting) session j Sim forces upon $\mathsf{R}^*_{\mathsf{soa}}$ the same transcript generated in the main thread, and it changes only the random challenge and the public keys used in the commitment phase of session j. Thus, the view of $\mathsf{R}^*_{\mathsf{soa}}$ is distributed identically as in the main thread. Then it repeats this procedure rewinding $\mathsf{R}^*_{\mathsf{soa}}$ until either a secret has been extracted or the fresh challenge chosen in a rewinding thread is identical to the challenge sent in the main thread. The probability of the latter event is $\mathtt{poly}(n)/2^n$, thus is negligible.

Let us denote as $\zeta^j = \zeta(\mathsf{ran}_{\mathsf{R}}, j)$ the probability that $\mathsf{R}^*_{\mathsf{soa}}$, activated with randomness $\mathsf{ran}_{\mathsf{R}}$, correctly answered to the challenge sent by Sim in the commitment phase of the j-th session. At each rewind, Sim keeps the transcript generated in the commitment phase and it only changes the random challenge, thus the view of $\mathsf{R}^*_{\mathsf{soa}}$ in the rewinding thread is distributed identically as in the commitment phase. Hence, the probability to get a correct answer is still bounded by ζ^j.

Thus, the expected number of rewinds to extract the secret key is $1/\zeta^j$ rewinds for each session j. Moreover, upon receiving a new challenge, $\mathsf{R}^*_{\mathsf{soa}}$ may initiate new sessions (i.e., sessions that did not appear in the main thread). In this case Sim follows the same procedure of the commitment phase, therefore each possibly new session initiated during the rewind takes time at most $\mathsf{t}_{\mathtt{SimCom}}$. Upon each rewind $\mathsf{R}^*_{\mathsf{soa}}$ may initiate at most a polynomial number of sessions, therefore the additional work of the simulator for each rewind that we indicate with $\mathsf{t}_{\mathtt{rew}}$ in bounded by $\mathtt{poly}(n) \cdot \mathsf{t}_{\mathtt{SimCom}}$. Obviously, once the simulator extracts the trapdoor for the target session, the new sessions are discarded.

In the decommitment phase Sim executes $n-1$ sub-decommitments plus one execution of the fake-decommitment algorithm, for at most m sessions. Each decommitment phase takes running time: $\mathsf{t}_{\mathtt{SimDec}} = (n-1) \cdot \mathsf{t}_d + \mathsf{t}_{fd}$, that is polynomial in n.

Hence, the total expected running time is:

$$\mathsf{t}_{\mathsf{Sim}} = k \cdot \mathsf{t}_{\mathtt{SimCom}} + \sum_{j=1}^{m} \zeta^j \Big[\frac{1}{\zeta^j} \cdot \mathsf{t}_{\mathtt{rew}} + \mathsf{t}_{\mathtt{SimDec}} \Big] = \mathtt{poly}(n).$$

□

Proposition 2. *The distribution of the output of the simulator* Sim *having black-box access to* $\mathsf{R}^*_{\mathsf{soa}}$ *is computationally indistinguishable from the output of* $\mathsf{R}^*_{\mathsf{soa}}$ *interacting with the real sender* $\mathsf{S}_{\mathsf{soa}}$.

Proof. Consider the following hybrids:

H_0**:** In this experiment Sim has in input the bit-vector $\mathbf{b} \leftarrow \mathcal{B}$ and follows the code of the honest sender $\mathsf{S}_{\mathsf{soa}}$. This is the real game.

H_1**:** This hybrid is the same as H_0 except that here, after the completion of the commitment phase, Sim launches the extraction phase. That is, for each non-aborted session asked by $\mathsf{R}^*_{\mathsf{soa}}$ to be opened, Sim rewinds $\mathsf{R}^*_{\mathsf{soa}}$ keeping the same transcript of the commitment phase (main thread) but changes the challenge string, till it obtains the trapdoor $\mathsf{sk}_i^{\bar{d}_i}$ for at least one $i \in [n]$, or it aborts. Possible new sessions initiated by $\mathsf{R}^*_{\mathsf{soa}}$ are handled running the honest sender procedure using the knowledge of $\mathbf{b}$. The extracted trapdoor is never used in the protocol. We now argue that H_0 and H_1 are indistinguishable. First note that, the extraction phase is initiated only for non-aborting sessions,

therefore, only for those in which $\mathsf{R}^*_{\mathsf{soa}}$ correctly completed the commitment phase with non-zero probability. Then note that the view of $\mathsf{R}^*_{\mathsf{soa}}$ in the rewinding thread is distributed identically to her view in the commitment phase (main thread). Thus the only differences between the two experiments is that 1) in H_1 Sim runs in expected polynomial time, this is not a problem since we are only interested in the output of the experiment (and it will not be a problem in the reductions shown for the next hybrids since rewinding threads that take more time than a fixed polynomial, can be truncated, without perturbing the non-negligible probability of success); 2) in H_1 Sim aborts with higher probability due to the possible aborts in the rewinding threads. These aborts happen when Sim picks a random challenge that is equal to the challenge sent in the main thread. This event happens with negligible probability ($\mathtt{poly}(n)/2^n$). Thus H_0 and H_1 are statistically indistinguishable.

We now fix some notation. Let $m = |I|$ be the number of sessions required by $\mathsf{R}^*_{\mathsf{soa}}$ to be opened. In order to associate the indexes to the sessions we use the following notation: we refer to j_ℓ the session corresponding to the ℓ-th index, where $\ell = 1, \ldots, m$. Note that the extraction phase is executed only for sessions belonging to the set I. In the following experiments we first deal with the (potential) new sessions initiated by $\mathsf{R}^*_{\mathsf{soa}}$ in the rewinding threads. Recall that Sim tries to extract the secret for one session at time. For each rewinding attempt for the extraction of a session j_ℓ, $\mathsf{R}^*_{\mathsf{soa}}$ may initiate several new sessions. We indicate with max_{j_ℓ} the maximum number of new sessions started during the rewinding threads for the session j_ℓ. Note that, for new sessions started during rewinding threads, Sim is never required to provide the decommitment (therefore those sessions do not need to be rewound).

$H_2^{j_\ell,s+1}$: for $\ell = 1, \ldots, m$; for $s = 0, \ldots, \mathsf{max}_{j_\ell} - 1$. In this sequence of hybrids Sim works as in experiment H_1 except that, in the *new* sessions started in the rewinding threads (recall that in each rewinding thread the view of $\mathsf{R}^*_{\mathsf{soa}}$ changes) in the commitment phase it commits to a random bit. Toward showing the indistinguishability of $H_2^{j_\ell,s}$ and $H_2^{j_\ell,s+1}$, we first show that $H_2^{j_\ell,s}$ is indistinguishable from an hybrid $\bar{H}_2^{j_\ell,s+1}$ where the bit committed in the $(s+1)$-th new session is the opposite bit used in $H_2^{j_\ell,s}$.

More precisely hybrid $\bar{H}_2^{j_\ell,s+1}$ is the same as hybrid $H_2^{j_\ell,s}$ except that in the commitment phase of the $(s+1)$-th new session, which index (in the range between 1 and k of all sessions played in the current view) we denote by t, initiated by $\mathsf{R}^*_{\mathsf{soa}}$ in the extraction phase of session j_ℓ, one of the sub-commitments hides the opposite bit such that the sum of the shares of all sub-commitments gives $1 - \mathbf{b}[t]$. More specifically, let $b_1, \ldots, b_n$ the shares of bit $\mathbf{b}[t]$, in experiment $\bar{H}_2^{j_\ell,s+1}$, Sim flips one of the shares, i.e. there exists one i such that Sim, differently from the experiment $H_2^{j_\ell,s}$ commits to $\bar{b}_i$, thus in turn committing to $\bar{\mathbf{b}}[t]$.

Assume that there exists a distinguisher D_{soa} that is able to tell apart hybrid $\bar{H}_2^{j_\ell,s+1}$ from $H_2^{j_\ell,s}$ then it is possible to construct a distinguisher who breaks the hiding of the commitment scheme TC. The reduction works as follows. $\mathsf{R}^*_{\mathsf{TC}}$ runs $\mathsf{R}^*_{\mathsf{soa}}$ as a subroutine and simulates Sim as in experiment $H_2^{j_\ell,s}$ except that in the new session t, it proceeds as follows: it receives the public keys from $\mathsf{R}^*_{\mathsf{soa}}$, then it prepares the shares of $\mathbf{b}[t]$: $b_1, \ldots, b_n$ and prepares the sub-commitments with the following strategy: it honestly computes sub-commitments for $n-1$ shares while for the computation of one of the sub-commitments it asks to the external sender S_{TC}. Namely, $\mathsf{R}^*_{\mathsf{TC}}$ forwards one the public keys received by $\mathsf{R}^*_{\mathsf{soa}}$ to S_{TC}, and then in turn, it forwards to $\mathsf{R}^*_{\mathsf{soa}}$ the commitment received by S_{TC} (along with $n-1$ sub-commitments honestly computed). Now if the bit committed by S_{TC} matches the share computed then the view generated by $\mathsf{R}^*_{\mathsf{TC}}$ is distributed identically to $H_2^{j_\ell,s}$, otherwise, if

S_{TC} committed to the flipped share, then the view generated is distributed identically to $\bar{H}_2^{j_\ell,s+1}$. The technical problem in this reduction is that $\mathsf{R}^*_{\mathsf{TC}}$ playing with S_{TC} must provide a well formed public key (i.e. a public generated by running TCgen), while $\mathsf{R}^*_{\mathsf{soa}}$, among the $2n$ public keys, could send public keys that are not well formed. However, since in the third round $\mathsf{R}^*_{\mathsf{soa}}$ must prove the correctness of some of them, $\mathsf{R}^*_{\mathsf{TC}}$ can run a preliminary execution in which it interacts with $\mathsf{R}^*_{\mathsf{soa}}$ exactly as Sim in $H_2^{j_\ell,s}$ and in the third round obtains the randomness of half of the public keys. Then, running the consistency check, it detects which public keys are well formed. At this point $\mathsf{R}^*_{\mathsf{TC}}$ rewinds $\mathsf{R}^*_{\mathsf{soa}}$ and starts the execution again, this time knowing which keys are well formed and can be safely forwarded to S_{TC}. Assume that i is the index of one of the public keys that have been proved to be well formed in the preliminary test. The formal reduction works as follows: $\mathsf{R}^*_{\mathsf{TC}}$ runs $\mathsf{R}^*_{\mathsf{soa}}$ as a subroutine and simulates Sim as in experiment $H_2^{j_\ell,s}$ except that, in the new session t, upon receiving the public keys from $\mathsf{R}^*_{\mathsf{soa}}$ it forwards $\mathsf{pk}_i^{\bar{d}_i}$ to the external sender S_{TC}. Then it performs the secret sharing of the bit $\mathbf{b}[t]$ as $b_1, \ldots, b_n$ and honestly executes $n-1$ sub-commitments using the remaining public parameters received by $\mathsf{R}^*_{\mathsf{soa}}$. Upon receiving the sub-commitment from S_{TC} for the public key $\mathsf{pk}_i^{\bar{d}_i}$, $\mathsf{R}^*_{\mathsf{TC}}$ forwards it along with the remaining honestly computed sub-commitments to $\mathsf{R}^*_{\mathsf{soa}}$. Finally $\mathsf{R}^*_{\mathsf{TC}}$ forwards the output of the experiment to D_{soa} and output whatever D_{soa} outputs.

Now, if S_{TC} has committed to the share b_i, then the view generated by $\mathsf{R}^*_{\mathsf{TC}}$ is distributed identically to hybrid $H_2^{j_\ell,s}$. Otherwise, if S_{TC} has committed to bit $1-b_i$ then the view generated is distributed identically to hybrid $\bar{H}_2^{j_\ell,s+1}$. By the hiding of protocol TC, we have that $H_2^{j_\ell,s}$ and $\bar{H}_2^{j_\ell,s+1}$ are indistinguishable.

Now, in $H_2^{j_\ell,s+1}$ the bit committed in session t (i.e., the $(s+1)$-th new session) is a random bit, and therefore the output of any distinguisher on $H_2^{j_\ell,s+1}$ will be indistinguishable from the one of $H_2^{j_\ell,s}$ and $\bar{H}_2^{j_\ell,s+1}$.

Therefore, $H_1 = H_2^{1,0}$ and $H_2^{j_m,\mathsf{max}_{j_m}}$ are indistinguishable.

$H_3^{j_{\ell+1}}$**: for** $\ell = 0, \ldots, m-1$**:** In this sequence of hybrids Sim performs the decommitment phase of the sessions that have been asked by $\mathsf{R}^*_{\mathsf{soa}}$, using the trapdoor extracted in the extraction phase, therefore, technically Sim can open to any bit.

More precisely, hybrid $H_3^{j_{\ell+1}}$ is the same as hybrid $H_3^{j_\ell}$ except that in $H_3^{j_{\ell+1}}$ in the decommitment phase of the $j_{\ell+1}$-th session Sim uses the trapdoor $\mathsf{sk}_i^{\bar{d}_i}$ (for some $i \in [n]$) extracted for this session. More specifically, in session $j_{\ell+1}$ Sim honestly performs $n-1$ sub-decommitments while the i-th sub-decommitment is executed using the trapdoor information and invoking the trapdoor algorithm $\mathsf{TCfakeDec}$ on input the bit b_i that is computed as follows: $b_i \leftarrow \bigoplus_{l\in[n],l\neq i} b_l^{\bar{d}_l} \oplus \mathbf{b}[j_{\ell+1}]$. Note that now in the opening, b_i depends of the actual input of the sender. However, in this experiment the share b_i computed in the commitment phase is identical to the share b_i given in input to the algorithm $\mathsf{TCfakeDec}$. More precisely, $\mathsf{TCfakeDec}$ is not used to open to a different bit, but to the very same bit. The difference therefore in this experiment consists in the use of $\mathsf{TCfakeDec}$ for the opening of the commitment.

Assume that there exists a distinguisher D_{soa} able to tell apart $H_3^{j_{\ell+1}}$ from $H_3^{j_\ell}$ then it is possible to construct an adversary for the trapdoor property of TC that we denote as $\mathsf{R}^*_{\mathsf{TC}}$. $\mathsf{R}^*_{\mathsf{TC}}$ is running in the experiment $\mathbf{Exp}_{\mathsf{TC}}^{\mathsf{Trap/Com}}$ against a sender S_{TC}. It works as follows: $\mathsf{R}^*_{\mathsf{TC}}$ runs $\mathsf{R}^*_{\mathsf{soa}}$ as subroutine simulating the honest sender $\mathsf{S}_{\mathsf{soa}}$. $\mathsf{R}^*_{\mathsf{TC}}$ simulates all sessions as in experiment $H_3^{j_\ell}$ except that the

session $j_{\ell+1}$ is handled as follows: $\mathsf{R}^*_{\mathsf{TC}}$ after having received the public keys from $\mathsf{R}^*_{\mathsf{soa}}$, performs the secret sharing of the bit $\mathbf{b}[j_{\ell+1}] = b_1 \oplus \cdots \oplus b_n$, picks challenge $d_1, \ldots, d_n$, an index $i \in [n]$, and forwards $\mathsf{pk}_i^{\bar{d}_i}$,$b_i$ to S_{TC}. Then it engages in $n-1$ sub-commitments of TC with $\mathsf{R}^*_{\mathsf{soa}}$ for the commitment of b_l, with $l \neq i$ while for the i-th sub-commitment it forwards the messages received by S_{TC}. Once the commitment phase is over, $\mathsf{R}^*_{\mathsf{TC}}$ runs the extraction phase. If it does not get the secret key $\mathsf{sk}_i^{\bar{d}_i}$ it aborts. Otherwise, it continues executing the decommitment phase. In the $j_{\ell+1}$-th decommitment phase $\mathsf{R}^*_{\mathsf{TC}}$ sends $\mathsf{sk}_i^{\bar{d}_i}$ to S_{TC} and forwards the decommitment received by S_{TC} to $\mathsf{R}^*_{\mathsf{soa}}$. $\mathsf{R}^*_{\mathsf{TC}}$ honestly computes the remaining $n-1$ decommitments. Finally $\mathsf{R}^*_{\mathsf{TC}}$ forwards the output of $\mathsf{R}^*_{\mathsf{soa}}$ to D_{soa} and outputs what D_{soa} outputs. Now, if the $j_{\ell+1}$-th sub-decommitment was computed by using the trapdoor secret $\mathsf{sk}_i^{\bar{d}_i}$, then the view of $\mathsf{R}^*_{\mathsf{soa}}$ is distributed identically as hybrid $H_3^{j_{\ell+1}}$, otherwise, if the sub-decommitment was honestly computed then the view is distributed according to hybrid $H_3^{j_\ell}$. Therefore, if D_{soa} distinguishes the two experiments with non-negligible advantage δ then $\mathsf{R}^*_{\mathsf{TC}}$ wins the game $\mathbf{Exp}^{\mathsf{Trap/Com}}_{\mathsf{TC}}$ with advantages $\delta/2n$ that is still non-negligible therefore breaking the trapdoor property of TC. Hence, $H_3^{j_{\ell+1}}$ and $H_3^{j_\ell}$ are computationally indistinguishable.

Therefore $H_2^{j_m,\mathsf{max}_{j_m}} = H_3^{j_0}$ and $H_3^{j_m}$ are computationally indistinguishable.

H_4^{j+1}**: for** $j = 0, \ldots, k-1$**.** In this sequence of hybrids Sim performs the commitment phase committing to random bits instead of using the vector $\mathbf{b}$.

More precisely, hybrid H_4^{j+1} is the same as H_4^j except that in H_4^{j+1} Sim performs the commitment phase of the session j committing to a random bit instead of $\mathbf{b}[j]$. Due to hiding of the commitment scheme TC, and following the same arguments shown for hybrids $H_2^{j_\ell,s}$ and $H_2^{j_\ell,s+1}$, hybrids H_4^{j+1} and H_4^j are indistinguishable.

By noticing $H_4^0 = H_3^{j_m}$ and that H_4^k corresponds to the game played by the simulator, we have that the claim holds.

□

This concludes the proof of the Theorem.

□

D.2 Proof of Theorem 7

Proof. In the following we use the same notation used in the proof of Pro. 1.
Completeness. It follows from the completeness of the sub-protocol wTCom.
Binding. The binding proof follows the same logic of the one provided for the protocol 1, therefore is omitted.

Hiding under selective opening attack. We show a PPT simulator Sim that having black box access to the adversary $\mathsf{R}^*_{\mathsf{soa}}$ generates an output that is distributed as the output of generated by the interaction between $\mathsf{R}^*_{\mathsf{soa}}$ and the real world sender $\mathsf{S}_{\mathsf{soa}}$. The simulator works as follows:

SOA-simulator Sim
Initialization phase. Choose random tape $\mathsf{ran}_{\mathsf{R}}$ and activate $\mathsf{R}^*_{\mathsf{soa}}(\mathsf{ran}_{\mathsf{R}})$.
Commitment phase. Main thread.

1. Upon receiving public keys $\{\mathsf{pk}_i^0, \mathsf{pk}_i^1\}_{i\in[n]}$ for some session $j \in [k]$: pick a random n-bit string $d_1, \ldots, d_n$ and send it to $\mathsf{R}^*_{\mathsf{soa}}$. Label this point as **1-j**.
2. Upon receiving $\{r_i^{d_i}\}_{i\in[n]}$ for some session j, check their consistency, running $\mathsf{sk}_i^{d_i}, \mathsf{pk}_i^{d_i} \leftarrow \mathsf{TCgen}(r_i^{d_i})$. If the check fails ABORT session j. If there exists a pair $(\mathsf{sk}_i^{\bar{d}_i}, j)$ for some $i \in [n]$, go to step 4. Else go to step 3.
3. Extraction of secret keys for session j: start rewinding threads to extract $\mathsf{sk}_i^{\bar{d}_i}$ for some $i \in [n]$:
   (a) rewind $\mathsf{R}^*_{\mathsf{soa}}$ up to point **1-j**.
   (b) send a randomly chosen challenge string $d'_1, \ldots, d'_n$ to $\mathsf{R}^*_{\mathsf{soa}}$. If $\mathsf{R}^*_{\mathsf{soa}}$ aborts, rewind her up to step 3a.
   (c) if $\mathsf{R}^*_{\mathsf{soa}}$ starts new commitment sessions, follow the commitment procedure of the honest sender $\mathsf{S}_{\mathsf{soa}}$ committing to a random bit.
   (d) when $\mathsf{R}^*_{\mathsf{soa}}$ replies with secrets $\{r_i^{d'_i}\}_{i\in[n]}$ for the session j: if the random strings $d'_1, \ldots, d'_n$ and $d_1, \ldots, d_n$ are equal ABORT. Else, if there exists at least one r_i generating a valid pair $\mathsf{sk}_i^{d'_i}, \mathsf{pk}_i^{d'_i}$ where $\mathsf{pk}_i^{d'_i}$ was received in step **1-j** and $d'_i \neq d_i$ store the pair $(\mathsf{sk}_i^{d'_i}, j)$, rewind $\mathsf{R}^*_{\mathsf{soa}}$ up to step 2 and return. Otherwise go to step 3a.
4. Commitment of session j: On input the pair $\mathsf{sk}_i^0, \mathsf{sk}_i^1, j$ proceeds with the commitments for the session j;
   (a) randomly choose random bits $b_1, \ldots, b_n$;
   (b) for all bit b_l s.t. $l \neq i$ honestly run the sub-commitment algorithm: $b_l \overset{\$}{\leftarrow} \langle \mathsf{S}_{\mathsf{TC}_l}^{\bar{d}_l}(\mathtt{open}), \mathsf{R}_{\mathsf{TC}_l}^{\bar{d}_l}(\mathtt{open})\rangle$; where d_l is the challenge sent in step **1-j**.
   (c) for the i-th sub-commitment, run the fake commitment algorithm using the secret key $\mathsf{sk}_i^{\bar{d}_i}$: $\langle\ \mathsf{TCfakeCom}(\mathsf{pk}_i^{\bar{d}_i}, \mathsf{sk}_i^{\bar{d}_i}, \mathtt{com}), \mathsf{R}_{\mathsf{TC}_i}^{\bar{d}_i}(\mathsf{pk}_i^{\bar{d}_i}, \mathtt{recv})\ \rangle$; If $\mathsf{R}^*_{\mathsf{soa}}$ aborts, then aborts this session.

Commitment phase completion. When the commitment phase of all k sessions is completed, Sim obtains the set of indexes I from $\mathsf{R}^*_{\mathsf{soa}}$ and obtains $\{\mathbf{b}[j]\}_{j\in I}$ from the experiment.

Decommitment phase. When $\mathsf{R}^*_{\mathsf{soa}}$ asks for the opening of session $j \in I$, run n openings as follows:

1. for all sub-commitments $l \neq i$ honestly run the sub-decommitment algorithm: $b_l \overset{\$}{\leftarrow} \langle \mathsf{S}_{\mathsf{TC}_l}^{\bar{d}_l}(\mathtt{open}), \mathsf{R}_{\mathsf{TC}_l}^{\bar{d}_l}(\mathtt{open})\rangle$; where d_l is the challenge sent in the commitment phase. Compute $b'_i \leftarrow (\bigoplus_{l\in[n-1]} b_l) + \mathbf{b}[j]$.
2. for the i-th sub-commitment run the fake-sub-decommitment algorithm using the trapdoor information $\mathsf{sk}_i^{\bar{d}_i}$: $b'_i \leftarrow \langle \mathsf{TCfakeDec}(\mathsf{sk}_i^{\bar{d}}, \mathtt{open}, b_i), \mathsf{R}_{\mathsf{TC}_i}^{\bar{d}_i}(\mathtt{open})\rangle$; If $\mathsf{R}^*_{\mathsf{soa}}$ aborts, then aborts this session.

Finally, outputs whatever $\mathsf{R}^*_{\mathsf{soa}}$ outputs.

Proposition 3. *The simulator* Sim *runs in expected polynomial time in* n.

Proof. The simulator strategy mainly follows the strategy of the simulator shown in Appendix. D.1 therefore the analysis of the running time follows from Proposition 1. □

Proposition 4. *The distribution of the output of simulator* Sim *having black box access to* $\mathsf{R}^*_{\mathsf{soa}}$ *is computationally close to the output of* $\mathsf{R}^*_{\mathsf{soa}}$ *interacting with real sender* $\mathsf{S}_{\mathsf{soa}}$.

Proof. Consider the following hybrids:

H_0**:** In this experiment Sim has in input the bit-vector $\mathbf{b} \leftarrow \mathcal{B}$ and follows the code of the honest sender $\mathsf{S}_{\mathsf{soa}}$. This is the real game.

$H_1^{j_{\ell+1}}$ **(for** $\ell = 0, \ldots, k-1$**):** Experiment $H_1^{j_{\ell+1}}$ is the same as experiment $H_1^{j_\ell}$ except that in the $(\ell+1)$-th session for which Sim obtains the answer to the challenge string (second step of the sender in the description of the protocol) Sim launches the extraction phase to extract the trapdoor $\mathsf{sk}_i^{\bar{d}_i}$ for some $i \in [n]$ for the session $j_{\ell+1}$. In the new sessions initiated by $\mathsf{R}^*_{\mathsf{soa}}$ during the rewinds, Sim runs as the honest sender using the knowledge of $\mathbf{b}$. However, the extracted trapdoor is never used in the protocol. We now argue that $H_1^{j_\ell}$ and $H_1^{j_{\ell+1}}$ are indistinguishable. First note that, the extraction phase is initiated only if $\mathsf{R}^*_{\mathsf{soa}}$ correctly completed the second step of the protocol with non-zero probability. Then note that the view of $\mathsf{R}^*_{\mathsf{soa}}$ in the rewinding thread is distributed identically to her view in the commitment phase. Thus the only differences between the two experiments is that 1) in $H_1^{j_\ell+1}$ Sim runs in expected polynomial time, this is not a problem since we are only interested in the output of the experiment (and it will not be a problem in the reductions shown for the next hybrids since rewinding threads that take more time than a fixed polynomial, can be truncated, without perturbing the non-negligible probability of success); 2) in $H_1^{j_\ell+1}$ Sim aborts with higher probability due to the possible aborts in the rewinding threads. These aborts happen when Sim picks a random challenge that is equal to the challenge sent in the commitment phase. This event happens with negligible probability ($\mathtt{poly}(n)/2^n$). Thus $H_1^{j_\ell}$ and $H_1^{j_{\ell+1}}$ are statistically indistinguishable.

$H_2^{j_\ell,s+1}$**: for** $\ell = 1, \ldots, m$**; for** $s = 0, \ldots, \mathsf{max}_{j_\ell} - 1$**.** In this sequence of hybrids Sim works as in experiment H_1 except that, in the *new* sessions started in the rewinding threads (recall that in each rewinding thread the view of $\mathsf{R}^*_{\mathsf{soa}}$ changes) in the commitment phase it commits to a random bit. Toward showing the indistinguishability of $H_2^{j_\ell,s}$ and $H_2^{j_\ell,s+1}$, we first show that $H_2^{j_\ell,s}$ is indistinguishable from an hybrid $\bar{H}_2^{j_\ell,s+1}$ where the bit committed in the $(s+1)$-th new session is the opposite bit used in $H_2^{j_\ell,s}$.

More precisely hybrid $\bar{H}_2^{j_\ell,s+1}$ is the same as hybrid $H_2^{j_\ell,s}$ except that in the commitment phase of the $(s+1)$-th new session, which index (in the range between 1 and k of all sessions played in the current view) we denote by t, initiated by $\mathsf{R}^*_{\mathsf{soa}}$ in the extraction phase of session j_ℓ, the last sub-commitment hides the opposite bit such that the sum of the shares of all sub-commitments gives $1 - \mathbf{b}[t]$. More specifically, let $b_1, \ldots, b_n$ the shares of bit $\mathbf{b}[t]$, in experiment $\bar{H}_2^{j_\ell,s+1}$, Sim flips one of the shares, i.e. there exist one i such that Sim, differently from the experiment $H_2^{j_\ell,s}$ commits to $\bar{b}_i$, thus in turn committing to $\bar{\mathbf{b}}[t]$.

Assume that there exists a distinguisher D_{soa} that is able to tell apart hybrid $\bar{H}_2^{j_\ell,s+1}$ from $H_2^{j_\ell,s}$ then it is possible to construct a distinguisher who breaks the hiding of the commitment scheme wTCom. The reduction works as follows. $\mathsf{R}^*_{\mathsf{wTCom}}$ simulates Sim as in experiment $H_2^{j_\ell,s}$ except that in the new session t, it proceeds as follows: plays the first three rounds of the protocol as Sim, i.e. obtains the $2n$ public keys, replies with challenge $d_1^{\mathtt{test}}, \ldots, d_n^{\mathtt{test}}$ and then obtains $r_{d_1^{\mathtt{test}}}, \ldots, r_{d_n^{\mathtt{test}}}$ and checks the consistency of half of the public keys. Let $\mathsf{pk}_i^{d_i^{\mathtt{test}}}$ the index of one of the public keys that passed the consistency test (i.e. $\mathsf{pk}_i^{d^{\mathtt{test}}}$ has been computed running wTCgen). Then $\mathsf{R}^*_{\mathsf{wTCom}}$ rewinds $\mathsf{R}^*_{\mathsf{soa}}$ and starts the execution of the session t again: in the first round it receives the same public keys from $\mathsf{R}^*_{\mathsf{soa}}$ (due to the fact that $\mathsf{R}^*_{\mathsf{soa}}$ has been activated with the same randomness and that it speaks first), then to answer to the second round it picks a random n-bit string $d_i, \ldots, d_n$ such that $d_i \neq d_i^{\mathtt{test}}$. Then it continues as follows: it forwards $\mathsf{pk}_i^{\bar{d}_i}$ (recall that $\bar{d}_i = d_i^{\mathtt{test}}$) to the external

sender S_{TC}. Then it performs the secret sharing of the bit $\mathbf{b}[t]$ as $b_1, \ldots, b_n$ and honestly executes $n-1$ sub-commitments using the remaining public parameters received by $\mathsf{R}^*_{\mathsf{soa}}$. Upon receiving the sub-commitment from S_{TC} for the public key $\mathsf{pk}_i^{\bar{d}_i}$, $\mathsf{R}^*_{\mathsf{TC}}$ forwards it along with the remaining honestly computed sub-commitments to $\mathsf{R}^*_{\mathsf{soa}}$. Finally $\mathsf{R}^*_{\mathsf{TC}}$ forwards the output of the experiment to D_{soa} and output whatever D_{soa} outputs.

Now, if $\mathsf{S}_{\mathsf{wTCom}}$ has committed to the share b_i, then the view generated by $\mathsf{R}^*_{\mathsf{TC}}$ is distributed identically to hybrid $H_2^{j_\ell,s}$. Otherwise, if $\mathsf{S}_{\mathsf{wTCom}}$ has committed to bit $1-b_i$ then the view generated is distributed identically to hybrid $\bar{H}_2^{j_\ell,s+1}$. By the hiding of protocol TC, we have that $H_2^{j_\ell,s}$ and $\bar{H}_2^{j_\ell,s+1}$ are indistinguishable.

Now, in $H_2^{j_\ell,s+1}$ the bit committed in session t (i.e., the $(s+1)$-th new session) is a random bit, and therefore the output of any distinguisher on $H_2^{j_\ell,s+1}$ will be indistinguishable from the one of $H_2^{j_\ell,s}$ and $\bar{H}_2^{j_\ell,s+1}$.

Therefore, $H_1 = H_2^{j_1,0}$ and $H_2^{j_m,\mathsf{max}_{j_m}}$ are indistinguishable.

$H_3^{j_{\ell+1}}$**: for** $\ell = 0, \ldots, k-1$**:** In this sequence of hybrids Sim uses the trapdoor extracted in the extraction phase. In each session, it performs the commitment/decommitment phase by using the algorithms $\mathsf{TCfakeCom}/\mathsf{TCfakeDec}$ for one of the n the sub-commitment. Therefore, in this hybrid Sim does not use the knowledge of $\mathbf{b}$ anymore.

More precisely, hybrid $H_3^{j_{\ell+1}}$ is the same as hybrid $H_3^{j_\ell}$ except that in $H_3^{j_{\ell+1}}$ in the decommitment phase of the $j_{\ell+1}$-th session Sim uses the trapdoor $\mathsf{sk}_i^{\bar{d}_i}$ (for some $i \in [n]$) extracted for this session. That is, in session $j_{\ell+1}$ Sim honestly performs $n-1$ sub-decommitments while the i-th sub-commitment is computed invoking the fake-sub-commitment algorithm $\mathsf{TCfakeCom}$ and the sub-decommitment (if session $j_{\ell+1}$ will be asked to be opened) is computed invoking the trapdoor algorithm $\mathsf{TCfakeDec}$ on input the bit b_i computed as follows: $b'_i \leftarrow \bigoplus_{l\in[n],l\neq i} b_l^{\bar{d}_l} \oplus \mathbf{b}[j_{\ell+1}]$.

Note that now in the opening, b_i depends of the actual input of the sender.

Assume there exists a distinguisher D_{soa} who is able to tell apart experiment $H_3^{j_{\ell+1}}$ from $H_3^{j_\ell}$ then it is possible to construct a distinguisher $\mathsf{R}^*_{\mathsf{wTCom}}$ for the weak trapdoor property of wTCom. $\mathsf{R}^*_{\mathsf{wTCom}}$ is running in the experiment $\mathbf{Exp}^{\mathsf{wTrap/Com}}_{\mathsf{wTCom}}$ trying to distinguish whether the messages received by sender $\mathsf{S}_{\mathsf{wTCom}}$ are computed using the honest or the fake algorithm. It works as follows: $\mathsf{R}^*_{\mathsf{TC}}$ runs $\mathsf{R}^*_{\mathsf{soa}}$ as subroutine simulating Sim as in experiment $H_3^{j_\ell}$ except that in session $j_{\ell+1}$, after having obtained (from the extraction phase) the trapdoor $\mathsf{sk}^{\bar{d}_i}$ for some $i \in [n]$ proceeds as follows. It performs the secret sharing of the bit $\mathbf{b}[j_{\ell+1}] = b_1, \ldots, b_n$ and forwards $\mathsf{pk}_i^{\bar{d}_i}$,$\mathsf{sk}_i^{\bar{d}_i}, b_i$ to $\mathsf{S}_{\mathsf{wTCom}}$ (note that if the sender $\mathsf{S}_{\mathsf{wTCom}}$ is running $\mathsf{TCfakeCom}$ the bit b_i is ignored and is given only in the decommitment phase to the algorithm $\mathsf{TCfakeDec}$.). Then $\mathsf{R}^*_{\mathsf{wTCom}}$ honestly computes the sub-commitments for bits b_l, with $l \neq i$, while for the i-th sub-commitment it forwards the commitment received by S_{TC}. When the commitment phase is completed $\mathsf{R}^*_{\mathsf{wTCom}}$ obtains the set I from $\mathsf{R}^*_{\mathsf{soa}}$, if the set does not contain the session $j_{\ell+1}$, it aborts. Otherwise, $\mathsf{R}^*_{\mathsf{wTCom}}$ performs the decommitment phase of the session $j_{\ell+1}$ as follows: it honestly opens the sub-commitments in position $i \neq l$, while it forwards the decommitment received by $\mathsf{S}_{\mathsf{wTCom}}$ in position i. Finally $\mathsf{R}^*_{\mathsf{TC}}$ forwards the output of $\mathsf{R}^*_{\mathsf{soa}}$ to D_{soa} and outputs whatever D_{soa} outputs. Now, if the i-th sub-commitment/sub-decommitment was computed by using the trapdoor $\mathsf{sk}_i^{\bar{d}_i}$, then the view of $\mathsf{R}^*_{\mathsf{soa}}$ is distributed identically as hybrid $H_3^{j_{\ell+1}}$, otherwise, if the sub-commitment/sub-decommitment was honestly computed then the view

is distributed according to hybrid $H_3^{j_\ell}$. Therefore, if D_{soa} distinguishes the two experiments with non-negligible advantage δ then $\mathsf{R}^*_{\mathsf{TC}}$ win the $\mathbf{Exp}^{\mathsf{Trap/Com}}_{\mathsf{TC}}$ game with advantages $\frac{\delta m}{k}$ that is still non-negligible therefore breaking the trapdoor property of TC.

Hence, $H_3^{j_{\ell+1}}$ and $H_3^{j_\ell}$ are computationally indistinguishable.

Therefore $H_2^{j_m,\mathsf{max}_{j_m}} = H_3^{j_0}$ and $H_3^{j_k}$ are computationally indistinguishable.

By noticing that $H_3^{j_k}$ corresponds to the game played by the simulator, we have that the claim holds.

□

This concludes the proof of the Theorem. □

D.3 Proof of Theorem 2

Proof. We present our proof with the extractable commitment in the protocol being instantiated with Protocol 4. For notations, we continue to use those in Protocol 4. In order to distinguish between the extractable commitments of the sender and the receiver, we use a prefix of $\mathsf{S}.$ or $\mathsf{R}.$, and to identify the messages in different parallel executions of Protocol 4 performed by the sender and the receiver we use an index in the superscript of the messages - for instance, in the (i,j)th parallel session of ExtCom performed by the sender, the first round message is denoted by $\mathsf{S}.\alpha^{(1,1)}$. The proof of completeness, binding, and hiding under selective opening attacks, follow.

Completeness. It follows from the completeness of the sub-protocol ExtCom.

Binding. In the following we reduce the binding property of SOACom (in the stand-alone setting) to the binding property of ExtCom (due to the ExtCom commitments by $\mathsf{S}_{\mathsf{soa}}$) and to the hiding property of ExtCom (due to the ExtCom commitments by $\mathsf{R}_{\mathsf{soa}}$).

Suppose there exists a PPT adversary ${\mathsf{S}_{\mathsf{soa}}}^*$ that breaks binding of SOACom with probability $\delta > 1/P(n)$, where $P(\cdot)$ is a polynomial; i.e., it outputs a commitment phase $\tau_{\mathtt{com}}$, and two valid opening phase transcripts, $\tau^0_{\mathtt{open}}$ and $\tau^1_{\mathtt{open}}$, that are openings to 0 and 1, respectively. Then we construct efficient adversaries ${\mathsf{S}_{\mathsf{ext}}}^*$ and ${\mathsf{R}_{\mathsf{ext}}}^*$ that break binding and hiding of ExtCom, respectively, such that,

$$\begin{aligned}&\mathsf{Pr}[\mathbf{Exp}^{\text{binding}}_{\mathsf{SOACom},{\mathsf{S}_{\mathsf{soa}}}^*} \to 1] \\ &\quad \leq \frac{1}{10 \cdot n \cdot P(n)}(\mathbf{Adv}^{\text{hiding}}_{\mathsf{ExtCom},{\mathsf{R}_{\mathsf{ext}}}^*} + \mathsf{Pr}[\mathbf{Exp}^{\text{binding}}_{\mathsf{ExtCom},{\mathsf{S}_{\mathsf{ext}}}^*} \to 1])\end{aligned} \tag{1}$$

We begin with a high-level sketch of the reduction.

1. $\underline{{\mathsf{R}_{\mathsf{ext}}}^*}$: We begin by observing that if ${\mathsf{S}_{\mathsf{soa}}}^*$ can somehow know $a = (a_1, \ldots, a_n)$ before sending d, then it can easily break binding of SOACom: ${\mathsf{S}_{\mathsf{soa}}}^*$ would begin by generating the n pairs of commitments such that in every pair a random one is a commmitment to 0 and the other is a commmitment to 1. Then, once it knows a, it chooses $\tau^0_{\mathtt{open}}.d$ in such a way that $a \oplus \tau^0_{\mathtt{open}}.d$ points to positions that are commitments to 0. For $\tau^1_{\mathtt{open}}$, it would set $\tau^1_{\mathtt{open}}.d = \mathbf{1} \oplus \tau^0_{\mathtt{open}}.d$, thus being able to generate valid $\tau^0_{\mathtt{open}}$ and $\tau^1_{\mathtt{open}}$. Intuitively, since a is random, ${\mathsf{S}_{\mathsf{soa}}}^*$ is able to craft its d this way with any noticeable probability only by knowing a, i.e., by breaking hiding of ExtCom. This is the case that ${\mathsf{R}_{\mathsf{ext}}}^*$ takes advantage of. We now describe how ${\mathsf{R}_{\mathsf{ext}}}^*$ works at a high level. ${\mathsf{R}_{\mathsf{ext}}}^*$ begins by trying to identify this case first by extracting all the $2n$ commitments with some noticeable probabilty and then by observing if the extracted bits and d sent in the first of the openings, say $\tau^{\hat{j}}_{\mathtt{open}}$, are

favorable to this case. Meanwhile, $\mathsf{R_{ext}}^*$ would have committed to one of the a_is, say a_m, by using its interaction with its challenger in the hiding experiment of ExtCom, and the rest of the a_is by itself. Thus, finally it predicts a_m using the extracted bits, d, and the rest of the a_is.

2. $\underline{\mathsf{S_{ext}}^*}$: Note that $\mathsf{R_{ext}}^*$ above leaves unutilized the cases where either $\mathsf{S_{soa}}^*$ provides openings in the opening phases that are inconsistent with the extracted openings or there exists i such that the ith pair of commitments is extracted to equal bits. To this end, we build $\mathsf{S_{ext}}^*$ that breaks binding of ExtCom under these cases.

More detailed descriptions of $\mathsf{R_{ext}}^*$ and $\mathsf{S_{ext}}^*$ follow.

$\underline{\mathsf{R_{ext}}^*}$: Let $q(n)$ be the probability over $2n$ challenges, $(\mathsf{S}.e^{(1,0)}, \mathsf{S}.e^{(1,1)}), \ldots, (\mathsf{S}.e^{(n,0)}, \mathsf{S}.e^{(n,1)})$, sent in $\langle \mathsf{S_{soa}}^*(\mathtt{open}), \langle \mathsf{R_{soa}}(\mathtt{open})\rangle_2\rangle$ that $\mathsf{S_{soa}}^*$ does not abort. Note that $q(n) \geq \delta$. $\mathsf{R_{ext}}^*$ interacts with $\mathsf{S_{soa}}^*$ as follows.

1. Sample $m \xleftarrow{\$} [n]$.
2. Once α is received from the challenger in $\mathbf{Exp}^{\text{hiding-}a_m}_{\mathsf{Com},\mathsf{R}^*}$, for a random bit a_m, set $\mathsf{R}.\alpha^{(m)} := \alpha$ and send it to $\mathsf{S_{soa}}^*$. $\forall i \in [n] - \{m\}$, sample $a_i \xleftarrow{\$} \{0,1\}$ and run $(\mathsf{R}.\alpha^{(1)}, \cdot) \xleftarrow{\$} \langle {\mathsf{S_{ext}}}^i(\mathtt{com}, a_i), {\mathsf{R_{ext}}}^i(\mathtt{recv})\rangle_1$ with $\mathsf{S_{soa}}^*$.
3. Upon receiving $(\mathsf{R}.e^{(1)}, \ldots, \mathsf{R}.e^{(n)})$ and $(\mathsf{S}.\alpha^{(1,0)}, \mathsf{S}.\alpha^{(1,1)}), \ldots, (\mathsf{S}.\alpha^{(n,0)}, \mathsf{S}.\alpha^{(n,1)})$ from $\mathsf{S_{soa}}^*$, set $e := \mathsf{R}.e^{(m)}$ and send it to the challenger.
4. Once the challenger responds via $\beta(e)$, set $\mathsf{R}.\beta(e^{(m)}) := \beta(e)$, and send it to $\mathsf{S_{soa}}^*$. $\forall i \in [n] - \{m\}$, run $(\mathsf{R}.\beta(e^{(i)}), \cdot) \xleftarrow{\$} \langle {\mathsf{S_{ext}}}^i(\mathtt{com}, a_i),\ {\mathsf{R_{ext}}}^i(\mathtt{recv})\rangle_2$, and $\forall i \in [n]$, $\forall j \in \{0,1\}$, run $(\cdot, \mathsf{S}.e^{(i,j)}) \xleftarrow{\$} \langle {\mathsf{S_{ext}}}^{i,j}(\mathtt{com}, b), {\mathsf{R_{ext}}}^{i,j}(\mathtt{recv})\rangle_1$ with $\mathsf{S_{soa}}^*$.
5. If $\mathsf{S_{soa}}^*$ aborts, then output $\tilde{b} \xleftarrow{\$} \{0,1\}$ and halt; otherwise, once it enters one of the opening phases, say $\tau^{\hat{j}}_{\mathsf{open}}$, and sends $\tau^{\hat{j}}_{\mathsf{open}}.d$ and $(\beta(e^{(1,0)}), \mathsf{S}.\beta(e^{(1,1)})), \ldots, (\mathsf{S}.\beta(e^{(1,0)}), \beta(e^{(n,1)}))$, then proceed as follows in order to extract the $2n$ bits committed by $\mathsf{S_{soa}}^*$ using ExtCom.
   - Forall $i \in [n]$, $j \in \{0,1\}$, run $2n$ parallel invocations of $E2$ of E to extract bits $b^{(i,j)}$ with an upper-bound on the number of rewindings to be $2P^2(n)$.
   - If E fails to extract $b^{(i,j)}$ for any i, j within the specified number of rewindings, then output $\tilde{b} \xleftarrow{\$} \{0,1\}$ and halt.
6. Perform the following checks in the order; if any check fails, then output $\perp$ and halt.
   1. Check whether, $\forall i \in [n]$, one of the bits $b^{(i,0)}$, $b^{(i,1)}$ is 1 and the other is 0.
   2. Check if either $\forall i \in [n] - \{m\}$, $j = \tau^{\hat{j}}_{\mathsf{open}}.d_i \oplus a_i$, $b^{(i,j)} = 0$, or $\forall i \in [n] - \{m\}$, $j = \tau^{\hat{j}}_{\mathsf{open}}.d_i \oplus a_i$, $b^{(i,j)} = 1$.

   If all the above conditions check out, then output bit a'_m such that $b^{(m,j)}$, for $j = \tau^{\hat{j}}_{\mathsf{open}}.d_m \oplus a'_m$, is equal to $b^{(i,j)}$ for every $i \in [n] - \{m\}$ and $j = \tau^{\hat{j}}_{\mathsf{open}}.d_i \oplus a_i$.

Let $\mathsf{E.fail}$ denote the event that $\exists i, j$ such that E failed to extract $b^{(i,j)}$ in $2P^2(n)$ rewindings, and let $\mathsf{E.fail}^{(i,j)}$ denote the event $\mathsf{E.fail}$ conditioned on $b^{(i,j)}$ not being extracted. In the following we bound the probability of $\mathsf{E.fail}$.

Note that the view of $\mathsf{S_{soa}}^*$ in the rewinding threads is identical to that in the main-thread. Thus, its abort probability in the rewinding threads also continues to be $(1 - q(n))$. Since $q(n) \geq \delta > 1/P(n)$, we have,

$$\mathsf{Pr}[\mathsf{E.fail}] \leq \frac{1}{2P(n)} + \sum_{i=1}^{n} \frac{1}{2^n} \tag{2}$$

The first term in the above bound corresponds to the event when $\mathsf{S_{soa}}^*$ aborts in all the rewinding threads and the bound is derived from the Markov's inequality. The second term corresponds to the event where, for at least one of the $2n$ commitments of $\mathsf{S_{soa}}^*$, the challenge in the rewinding thread for which $\mathsf{S_{soa}}^*$ did not abort is equal to the one in the main thread. Thus, we have,

$$\mathsf{Pr}[\neg \mathsf{E.fail}] > \frac{1}{4P(n)}$$

Now consider an algorithm that interacts with $\mathsf{S_{soa}}^*$ by running $\mathsf{R_{soa}}$ (i.e., commits to all a_is by itself) except that it also tries to extract the $2n$ bits committed by $\mathsf{S_{soa}}^*$ (like $\mathsf{R_{ext}}^*$). Then denote the event that the extracted bits are the same as the opened bits in $\tau^0_{\mathtt{open}}$ and $\tau^1_{\mathtt{open}}$ by $\mathtt{extracted} = \mathtt{opened}$. Also, let $\mathtt{extracted}^{(i)} \neq \mathtt{opened}^{(i)}$ denote the event that at least one bit in the ith pair of extracted bits is inconsistent with the openings in $\tau_{\mathtt{com}}$, $\tau^0_{\mathtt{open}}$ or $\tau^1_{\mathtt{open}}$.

We have,

$$\begin{aligned}&\mathsf{Pr}[\mathbf{Exp}^{\text{hiding-1}}_{\mathsf{ExtCom},\mathsf{R_{ext}}^*}(n) \to 1)] \\ &\geq \mathsf{Pr}[\mathbf{Exp}^{\text{binding}}_{\mathsf{SOACom},\mathsf{S_{soa}}^*}(n) \to 1 \wedge \mathtt{extracted} = \mathtt{opened} \wedge \forall i, b^{(i,0)} \neq b^{(i,1)}] \cdot \mathsf{Pr}[a_m = 1] \cdot \mathsf{Pr}[\neg \mathsf{E.fail}]\end{aligned} \tag{3}$$

$$\begin{aligned}&\mathsf{Pr}[\mathbf{Exp}^{\text{hiding-0}}_{\mathsf{ExtCom},\mathsf{R_{ext}}^*}(n) \to 1] \\ &\leq \mathsf{Pr}[\mathbf{Exp}^{\text{binding}}_{\mathsf{SOACom},\mathsf{S_{soa}}^*}(n) \to 1 \wedge \mathtt{extracted}^{(m)} \neq \mathtt{opened}^{(m)} \wedge \forall i, b^{(i,0)} \neq b^{(i,1)}] \cdot \mathsf{Pr}[a_m = 0] \cdot \mathsf{Pr}[\neg \mathsf{E.fail}]\end{aligned} \tag{4}$$

$\underline{\mathsf{S_{ext}}^*}$: $\mathsf{S_{ext}}^*$ interacts with $\mathsf{S_{soa}}^*$ as follows.

1. Run $\mathsf{R_{soa}}$ and interact with $\mathsf{S_{soa}}^*$ to obtain $\tau_{\mathtt{com}}$, $\tau^0_{\mathtt{open}}$ and $\tau^1_{\mathtt{open}}$.
2. Forall $i \in [n]$, $j \in \{0,1\}$, run $2n$ parallel invocations of $E2$ of E to extract bits $b^{(i,j)}$ with an upper-bound on the number of rewindings to be $2P^2(n)$. If E fails to extract $b^{(i,j)}$ for any i, j within the specified number of rewindings, then output $\perp$ and halt.
3. Check whether $\exists i, j$ such that the (i,j)th extracted minor opening is inconsistent with the one in $\tau^0_{\mathtt{open}}/\tau^1_{\mathtt{open}}$. If so, then output the (i,j)th ExtCom commitment, its major opening, together with the inconsistent minor opening and halt.
4. If $\forall i, j$, the extracted minor openings are consistent with the major openings, then check whether $\tau^1_{\mathtt{open}}.d \neq \mathbf{1} \oplus \tau^0_{\mathtt{open}}.d$. If not, then output $\perp$ and halt. Otherwise, choose $i \xleftarrow{\$} [n]$ such that $\tau^0_{\mathtt{open}}.d_i = \tau^1_{\mathtt{open}}.d_i$. Output the part of $\tau_{\mathtt{com}}$, $\tau^0_{\mathtt{open}}$ and $\tau^1_{\mathtt{open}}$ corresponding to the (i,j)th commitment by $\mathsf{S_{soa}}^*$, where $j = \tau^0_{\mathtt{open}}.d_i \oplus a_i$ and halt.

We have,

$$\begin{aligned}&\mathsf{Pr}[\mathbf{Exp}^{\text{binding}}_{\mathsf{ExtCom},\mathsf{S_{ext}}^*}(n) \to 1] \\ &\geq \mathsf{Pr}[\mathbf{Exp}^{\text{binding}}_{\mathsf{SOACom},\mathsf{S_{soa}}^*}(n) \to 1 \wedge (\mathtt{extracted} \neq \mathtt{opened} \vee \tau^1_{\mathtt{open}}.d \neq \mathbf{1} \oplus \tau^0_{\mathtt{open}}.d)] \cdot \mathsf{Pr}[\neg \mathsf{E.fail}]\end{aligned} \tag{5}$$

From the bounds (3),(4), and (5), the bound (1) follows. This proves the binding property of SOACom.

Hiding under selective opening attack. We construct an expected polynomial-time simulator Sim that having black-box access to the adversary $\mathsf{R^*_{soa}}$ generates an output that is computationally indistinguishable from the output generated by the interaction between $\mathsf{R^*_{soa}}$ and the honest sender $\mathsf{S_{soa}}$. At a high level, the simulator first interacts with $\mathsf{R^*_{soa}}$ in the commitment phase of every session as follows. Unlike $\mathsf{S_{soa}}$ which commits to the same bit in all its n pairs of commitments, the simulator, in every pair, commits to

0 in one commitment and to 1 in the other. Further, when it receives the bits to which it needs to open, it equivocates in the opening phase by carefully crafting the string d so that the outcome of coin-flipping points to the n commitments to 0 if the bit it needs to open to is 0 or to the n commitments to 1 if the bit it needs to open to is 1. A formal description follows.

SOA-simulator Sim

Initialization phase. Choose random tapes $\mathsf{ran}_{\mathsf{R}}, \mathsf{ran}_{\mathsf{Sim}}$ respectively for $\mathsf{R}^*_{\mathsf{soa}}$ and for the commitment phase below.
Invoke $\mathsf{R}^*_{\mathsf{soa}}(\mathsf{ran}_{\mathsf{R}})$.

Commitment phase (S1). (Main thread)

- Upon receiving $\mathsf{R}.\alpha^{(1)}, \ldots, \mathsf{R}.\alpha^{(n)}$ from $\mathsf{R}^*_{\mathsf{soa}}$ for some session $j \in [k]$:
  1. Sample $\sigma' \xleftarrow{\$} \{0,1\}^n$.
  2. (Unlike $\mathsf{S}_{\mathsf{soa}}$,) $\forall\ i \in [n]$, $j \in \{0,1\}$ set $b^{(i,j)} := 0$ for $j = \sigma'_i$ and set $b^{(i,j)} := 1$ for $j = 1 \oplus \sigma'_i$. $\forall i, j$, run $(\mathsf{S}.\alpha^{(i,j)}, \cdot) \xleftarrow{\$} \langle\ {\mathsf{S}_{\mathsf{ext}}}^{i,j}(\mathtt{com}, b^{(i,j)}), recext^{i,j}(\mathtt{recv})\rangle_1$ with $\mathsf{R}^*_{\mathsf{soa}}$.
  3. (Like $\mathsf{S}_{\mathsf{soa}}$,) $\forall\ i \in [n]$, run $(\cdot, \mathsf{R}.e^{(i)}) \xleftarrow{\$} \langle {\mathsf{S}_{\mathsf{ext}}}^{i}(\mathtt{com}, a_i), {\mathsf{R}_{\mathsf{ext}}}^{i}(\mathtt{recv})\rangle_1$ with $\mathsf{R}^*_{\mathsf{soa}}$.
  4. If $\mathsf{R}^*_{\mathsf{soa}}$ aborts, then abort session j.
- Upon receiving $(\mathsf{R}.\beta(e^{(1)}), \ldots, \mathsf{R}.\beta(e^{(n)}))$ and $(\mathsf{S}.e^{(1,0)}, \mathsf{S}.e^{(1,1)}), \ldots, (\mathsf{S}.e^{(n,0)}, \mathsf{S}.e^{(n,1)})$, check whether $\mathsf{S}_{\mathsf{soa}}$ aborts; if it does, then abort session j.

Commitment phase completion. When the commitment phases of all k sessions are completed, Sim obtains the set of indices I, where $|I| \leq m$, from $\mathsf{R}^*_{\mathsf{soa}}$ and obtains $\{\mathbf{b}[j]\}_{j \in I}$ from the experiment.

Extraction phase. (rewinding threads)

- For $\ell = 1, \ldots, m$: if the j_ℓth session was not-aborted in the main thread, then:
- For $c = 1, \ldots, n$: Run $E2$ of E in order to extract a_c. During this extraction all the other messages of the same session and the messages in the other sessions (including those in sessions that may be newly begun by $\mathsf{R}^*_{\mathsf{soa}}$ in the rewinding threads) are computed as in (**S1**).

- If E fails to extract any of a_cs, then abort the simulation and halt.

Opening phase (S2)(Main thread). $\forall j_\ell \in I$, if the j_ℓth session is not aborted, then open it to $\mathbf{b}[j_\ell]$ as follows:

1. Let σ' be the random string chosen in the j_ℓth session in (**S1**). If $\mathbf{b}[j_\ell] = 0$ then set $d := \sigma' \oplus a$; otherwise, set $d := \sigma' \oplus a \oplus \mathbf{1}$. Send d to $\mathsf{R}^*_{\mathsf{soa}}$. Also, $\forall i, j$, run $(\mathsf{S}.\beta(e^{(i,j)}), \cdot) \xleftarrow{\$} \langle {\mathsf{S}_{\mathsf{ext}}}^{i,j}(\mathtt{com}, b^{(i,j)}), {\mathsf{R}_{\mathsf{ext}}}^{i,j}(\mathtt{recv})\rangle_2$ with $\mathsf{R}^*_{\mathsf{soa}}$. If $\mathsf{R}^*_{\mathsf{soa}}$ aborts, then abort this session.
2. Upon receiving $(\mathsf{R}.\gamma(e^{(1)}), \ldots, \mathsf{R}.\gamma(e^{(n)}))$, check whether $\mathsf{S}_{\mathsf{soa}}$ would have aborted; if so, then abort the session. Also, check whether the bits opened to from this response are equal to the extracted bits $a_1, \ldots, a_n$. If not, then abort the session; otherwise, define $\sigma := d \oplus a$, and $\forall i \in [n]$, run $(\gamma(e^{(i,\sigma_i)}), \cdot) \xleftarrow{\$} \langle {\mathsf{S}_{\mathsf{ext}}}^{i,\sigma_i}(\mathtt{com}, b^{i,\sigma_i}), {\mathsf{R}_{\mathsf{ext}}}^{i,\sigma_i}(\mathtt{recv})\rangle_3$ with $\mathsf{R}^*_{\mathsf{soa}}$.

Finally, output the commitment phase transcript from (**S1**) and the opening phase transcript from (**S2**) together with the final output of $\mathsf{R}^*_{\mathsf{soa}}$.

Proposition 5. *The simulator* Sim *runs in expected polynomial time in* n.

Proof. Sim consists of three phases: commitment phase, extraction phase, opening phase. The commitment phase (**S1**) basically is the commitment phase of SOACom (with the only difference in how the bits $b^{(i,j)}$ committed to using ExtCom are chosen). Thus the running time to Sim in the commitment phase, $\mathtt{t}_{\mathtt{SimCom}}$, is polynomial in n.

After the completion of the commitment phase Sim is asked for openings of $m = |I|$ sessions, and for each such session that is also not aborted, Sim runs the extractor E n times to extract $a = (a_1, \ldots, a_n)$.

Let us denote as $\zeta^j = \zeta(\mathsf{ran}_\mathsf{R}, j)$ the probability that $\mathsf{R}^*_\mathsf{soa}$, activated with randomness ran_R, correctly answered to the challenge sent by Sim in the commitment phase of the j-th session. While running E, since Sim generates the messages other than those generated by E the same way as in the commitment phase, the view of $\mathsf{R}^*_\mathsf{soa}$ in the rewinding threads is distributed identically as in the commitment phase. Hence, in the rewinding threads also, the probability that $\mathsf{R}^*_\mathsf{soa}$ responds without aborting is also bounded by ζ^j. As noted in the analysis of E, its expected running time is $\zeta^j \cdot 1/\zeta^j$. Further, in each rewinding performed by E, since the new sessions initiated by $\mathsf{R}^*_\mathsf{soa}$ is bounded by a polynomial, the extra running time of Sim due to the newly initiated sessions for each rewinding is bounded by $\mathtt{poly}(n) \cdot \mathsf{t}_{\mathtt{SimCom}}$. (Obviously, once the simulator extracts the trapdoor for the target session, the new sessions are discarded.) With $\mathsf{t}_{\mathtt{rew}}$ denoting the running time of Sim for each rewinding of E, expected running time of Sim in the extraction phase is:

$$\sum_{j=1}^{m} \zeta^j \Big[\frac{1}{\zeta^j} \cdot \mathsf{t}_{\mathtt{rew}} \Big] = \mathtt{poly}(n)$$

In the decommitment phase Sim just executes the decommitment algorithm of SOACom for at most m sessions. Since the running time of S_soa in the decommitment phase of SOACom and m are both polynomial in n, the running time of Sim in the decommitment phase, $\mathsf{t}_{\mathtt{SimDec}}$, is also polynomial in n.

Thus, the total expected running time:

$$\mathsf{t}_\mathsf{Sim} = k \cdot \mathsf{t}_{\mathtt{SimCom}} + \sum_{j=1}^{m} \zeta^j \Big[\frac{1}{\zeta^j} \cdot \mathsf{t}_{\mathtt{rew}} + \mathsf{t}_{\mathtt{SimDec}} \Big]$$

is also polynomial in n. □

Proposition 6. *The distribution of the output of the simulator* Sim *having black-box access to* $\mathsf{R}^*_\mathsf{soa}$ *is computationally close to the output of* $\mathsf{R}^*_\mathsf{soa}$ *interacting with the real sender* S_soa.

Proof. The proof is by a series of hybrid arguments, where we prove the indistinguishability of consecutive hybrids using the binding and hiding properties of the commitment scheme ExtCom.

Consider the following sequence of hybrids:

H_0**:** In this experiment Sim has as input the bit-vector $\mathbf{b} \leftarrow \mathcal{B}$ before it begins interacting with $\mathsf{R}^*_\mathsf{soa}$ and follows the code of the honest sender S_soa in committing to these bits. This is the real game.

$H_1^{j_\ell, i}$**:** This hybrid is the same as H_0 except for the following changes:

- For every j_cth session, where $c > \ell$, Sim behaves the same way as in H_0.
- For every j_cth session, where $0 \leq c < \ell$, Sim also gets as input $a_1, \ldots, a_n$ that $\mathsf{R}^*_\mathsf{soa}$ would later open to in the opening phase of the j_cth session. Then it proceeds as follows: Choose $\sigma' \xleftarrow{\$} \{0,1\}^n$; $\forall i' \in [n]$, set $b^{(i',j')} := 0$ for $j' = \sigma'_{i'}$ and set $b^{(i',j')} := 1$ for $j' = 1 - \sigma'_i$; $\forall i', j'$, run $(\mathsf{S}.\alpha^{(i,j)}, \cdot) \xleftarrow{\$} \langle {\mathsf{S}_\mathsf{ext}}^{i,j}(\mathtt{com}, b^{(i',j')}), {\mathsf{R}_\mathsf{ext}}^{i,j}(\mathtt{recv}) \rangle_1$ with $\mathsf{R}^*_\mathsf{soa}$. Also, set $d := a \oplus \sigma' \oplus \mathbf{b}[j_c]$. The rest of the messages are computed the same way as in H_0.
- In the j_ℓth session, $\forall i' > i$, Sim computes the i'th pair of commitments the same way as in H_0. For every i'th pair of commitments, where $1 \leq i' \leq i$, Sim chooses $\sigma'_i \xleftarrow{\$} \{0,1\}$, $b^{(i',j')} := 0$ for $j' = \sigma'_{i'}$, sets $b^{(i',j')} := 1$ for $j' = 1 - \sigma'_i$; $\forall i', j'$, and runs $(\mathsf{S}.\alpha^{(i,j)}, \cdot) \xleftarrow{\$} \langle {\mathsf{S}_\mathsf{ext}}^{i,j}(\mathtt{com}, b^{(i',j')}), {\mathsf{R}_\mathsf{ext}}^{i,j}(\mathtt{recv}) \rangle_1$ with $\mathsf{R}^*_\mathsf{soa}$. Also, set $d_i := a_i \oplus \sigma'_i \oplus \mathbf{b}[j_c]$.

Note the the hybrids $H_1^{0,0}$ and H_0 are identical. We now show that if there exists a distinguisher D_{soa} that distinguishes any two consecutive hybrids $H_1^{j'_\ell,i'}$ and $H_1^{j_\ell,i}$, where $(j'_\ell,i') < (j_\ell,i)$, then we construct an efficient adversary $\mathsf{R_{ext}}^*$ that breaks hiding of ExtCom. $\mathsf{R_{ext}}^*$ interacts with $\mathsf{R^*_{soa}}$ the same way as Sim in $H_1^{j_\ell,i}$, except the following change: To compute the ith pair of commitments in the j_ℓth session, $\mathsf{R_{ext}}^*$ proceeds as follows: Choose $\sigma'_i \xleftarrow{\$} \{0,1\}$, set $d_i := a_i \oplus \sigma'_i \oplus \mathbf{b}[j_\ell]$. If $\mathbf{b}[j_\ell] = 0$, then for $j = \sigma'_i$, run $(\mathsf{S}.\alpha^{(i,j)},\cdot) \xleftarrow{\$} \langle {\mathsf{S_{ext}}}^{i,j}(\mathtt{com},0), {\mathsf{R_{ext}}}^{i,j}(\mathtt{recv})\rangle_1$ with $\mathsf{R^*_{soa}}$ and compute the rest of the messages corresponding to this commitment as per $\mathsf{S_{ext}}$; for $j = 1-\sigma'_i$, upon receiving α from the challenger, set $\mathsf{S}.\alpha^{(i,j)} := \alpha$ and send it to $\mathsf{R^*_{soa}}$; upon receiving $\mathsf{S}.e^{(i,j)}$ from $\mathsf{R^*_{soa}}$, set $e := \mathsf{S}.e^{(i,j)}$ and send it to the challenger; upon receiving $\beta(e)$ from the challenger, set $\mathsf{S}.\beta(e^{(i,j)}) := \beta(e)$ and send it to $\mathsf{R^*_{soa}}$. Note that $\mathsf{R_{ext}}^*$ will not be asked to provide opening for this commitment since $j = 1-\sigma'_i$. If $\mathbf{b}[j_\ell] = 1$, then proceed as in the case of $\mathbf{b}[j_\ell] = 0$ except that, for $j = 1-\sigma'_i$, run $(\mathsf{S}.\alpha^{(i,j)},\cdot) \xleftarrow{\$} \langle {\mathsf{S_{ext}}}^{i,j}(\mathtt{com},1), {\mathsf{R_{ext}}}^{i,j}(\mathtt{recv})\rangle_1$ with $\mathsf{R^*_{soa}}$, and $j = \sigma'_i$, the (i,j)th ExtCom commitment is performed using the interaction with the challenger as explained for the case of $\mathbf{b}[j_\ell] = 0$. Similarly, in this case too, $\mathsf{R_{ext}}^*$ will not be asked to provide opening for the (i,j)th commitment for $j = \sigma'_i$. Finally, when D_{soa} outputs a bit b', then $\mathsf{R_{ext}}^*$ outputs $b' \oplus \mathbf{b}[j_\ell]$.

Note here that if the hiding experiment of $\mathsf{R_{ext}}^*$ is $\mathbf{Exp}^{\text{hiding-}b}_{\mathsf{ExtCom},{\mathsf{R_{ext}}^*}}(n)$ with $b \neq \mathbf{b}[j_\ell]$ then $\mathsf{R_{ext}}^*$ is running $H_1^{j_\ell,i}$ with $\mathsf{R^*_{soa}}$; otherwise, it is running $H_1^{j'_\ell,i'}$. Thus, $\mathsf{R_{ext}}^*$ breaks hiding of ExtCom with the same probability that D_{soa} distinguishes $H_1^{j'_\ell,i'}$ and $H_1^{j_\ell,i}$.

H_2**:** This experiment is same as $H_1^{j_m,n}$, except that Sim also extracts each of the bits $a_1,\ldots,a_n$ using the extractor E for every non-aborted session that it is asked to provide opening for. The extraction is performed for one session after the other as explained in the description of the original simulator Sim. If new sessions are initiated by $\mathsf{R^*_{soa}}$ in the rewinding threads, then Sim proceeds the same way as in the commitment phase of the main threads. Also, Sim will not be given $a_1,\ldots,a_n$ for these new sessions. However, this will not affect the indistinguishability since it was only used to set the value of d in the main thread, and as Sim discards these new sessions and hence never enters their opening phases, it does not affect the view of $\mathsf{R^*_{soa}}$. Furthermore, Sim aborts the simulation if it fails to extract any of the bits $a_1,\ldots,a_n$ in any such non-aborted session.

Now note that the statistical distance between the hybrids $H_1^{j_m,n}$ and H_2 is at most the probability that Sim aborts due to failure in the extraction. Since this probability is at most $\mathtt{poly}(n)/2^n$, we observe that the two hybrids are negligibly close.

H_3**:** This experiment is same as H_2, except that Sim also aborts the simulation if in any of the sessions, any of the extracted bits $a_1,\ldots,a_n$ is inconsistent with the corresponding opening provided by $\mathsf{R^*_{soa}}$ in the opening phase.

Assume that there exists a distinguisher D_{soa} that distinguishes H_2 and H_3. Then we construct an adversary $\mathsf{S_{ext}}^*$ that breaks binding of ExtCom. $\mathsf{S_{ext}}^*$ interacts with $\mathsf{R^*_{soa}}$ the same way as Sim in H_3. Whenever $\mathsf{R^*_{soa}}$ provides an opening to a bit a'_i that is inconsistent with it corresponding opening, then outputs the ExtCom commitment phase transcript corresponding to this bit along with the extracted opening and the opening provided by $\mathsf{R^*_{soa}}$ in the opening phase, thus breaking the binding of ExtCom with the same probability that D_{soa} that distinguishes H_2 and H_3, and in turn, implying the indistinguishability of the two hybrids.

H_4**:** This experiment is same as H_3, except that Sim does not receive as input $a_1, \ldots, a_n$ for any session, but uses the extracted bits in setting the value of d for the same.

If Sim does not abort in H_3 due to openings that are inconsistent with the extractions, then the extracted bits are equal to those opened by $\mathsf{R}^*_{\mathsf{soa}}$ in the opening phase. Thus, we have that the hybrids H_3 and H_4 are identical. Also, H_4 corresponds to the ideal game. This completes the proof of indistinguishability of the output of Sim from the output of $\mathsf{R}^*_{\mathsf{soa}}$. □

□

D.4 Proof of Theorem 8

Proof. **Completeness.** It follows from the completeness of the sub-protocol ExtCom.

Binding. The proof of binding follows on the same lines as the that of the Protocol 2 and is therefore omitted.

Hiding under selective opening attack. We construct an expected polynomial-time simulator Sim that having black-box access to the adversary $\mathsf{R}^*_{\mathsf{soa}}$ generates an output that is computationally indistinguishable from the output generated by the interaction between $\mathsf{R}^*_{\mathsf{soa}}$ and the honest sender $\mathsf{S}_{\mathsf{soa}}$. At a high level, the simulator works in a way similar to the simulator of Protocol 2, with a slight difference. Namely, the simulator first interacts with $\mathsf{R}^*_{\mathsf{soa}}$ in the commitment phase of every session as follows. Unlike $\mathsf{S}_{\mathsf{soa}}$ which commits to the same bit in all its n pairs of commitments, the simulator, in every pair, commits to 0 in one commitment and to 1 in the other. Now recall that the simulator for Protocol 2 first receives the bits to which it needs to open and then crafts its d so that the outcome of coin-flipping points to the n commitments to 0 if the bit it needs to open to is 0 or to the n commitments to 1 if the bit it needs to open to is 1. Now, since the simulator would receive the bits to which it needs to open only after the completion of all the commitment phases, the same strategy cannot work directly as in this protocol, the sender sends d in the commitment phase itself. However, we exploit the flexibility of the sender in choosing the outcome of coin-flipping or its binary-negation to dictate which commitments it finally opens. More specifically, the simulator first samples a random bit θ and crafts its d so as to get the outcome of coin-flipping to point to point to the commitments of θ. Then, once it receives the bits to which it needs to open, it equivocates in the opening phase by choosing between the outcome of coin-flipping and its binary-negation depending on the θ and the bit to be opened to. A formal description follows.

SOA-simulator Sim

Initialization phase. Choose random tapes $\mathsf{ran}_{\mathsf{R}}, \mathsf{ran}_{\mathsf{Sim}}$ respectively for $\mathsf{R}^*_{\mathsf{soa}}$ and for the commitment phase below.
Invoke $\mathsf{R}^*_{\mathsf{soa}}(\mathsf{ran}_{\mathsf{R}})$.

Commitment phase (S1). (Main thread)

- Upon receiving $\mathsf{R}.\alpha^{(1)}, \ldots, \mathsf{R}.\alpha^{(n)}$ from $\mathsf{R}^*_{\mathsf{soa}}$ for some session $j \in [k]$:
  1. Sample $\sigma' \xleftarrow{\$} \{0,1\}^n$.
  2. (Unlike $\mathsf{S}_{\mathsf{soa}}$,) $\forall\ i \in [n]$, $j \in \{0,1\}$ set $b^{(i,j)} := 0$ for $j = \sigma'_i$ and set $b^{(i,j)} := 1$ for $j = 1 \oplus \sigma'_i$.
     $\forall i, j$, run $(\mathsf{S}.\alpha^{(i,j)}, \cdot) \xleftarrow{\$} \langle {\mathsf{S}_{\mathsf{ext}}}^{i,j}(\mathtt{com}, b^{(i,j)}), {\mathsf{R}_{\mathsf{ext}}}^{i,j}(\mathtt{recv})\rangle_1$ with $\mathsf{R}^*_{\mathsf{soa}}$.
  3. (Like $\mathsf{S}_{\mathsf{soa}}$,) $\forall\ i \in [n]$, run $(\cdot, \mathsf{R}.e^{(i)}) \xleftarrow{\$} \langle {\mathsf{S}_{\mathsf{ext}}}^{i}(\mathtt{com}, a_i), {\mathsf{R}_{\mathsf{ext}}}^{i}(\mathtt{recv})\rangle_1$ with $\mathsf{R}^*_{\mathsf{soa}}$.
  4. If $\mathsf{R}^*_{\mathsf{soa}}$ aborts, then abort session j.

Extraction phase. (rewinding threads)

- For $\ell = 1, \ldots, m$: if the j_ℓth session was not-aborted in the main thread, then:
- For $c = 1, \ldots, n$: Run $E2$ of E in order to extract a_c. During this extraction all the other messages of the same session and the messages in the other sessions (including those in sessions that may be newly begun by $\mathsf{R}^*_{\mathsf{soa}}$ in the rewinding threads) are computed as in (**S1**).

- If E fails to extract any of a_cs, then abort the simulation and halt.

Commitment phase continued. (Main thread)

- Upon receiving $(\mathsf{R}.\beta(e^{(1)}), \ldots, \mathsf{R}.\beta(e^{(n)}))$ and $(\mathsf{S}.e^{(1,0)}, \mathsf{S}.e^{(1,1)}), \ldots, (\mathsf{S}.e^{(n,0)}, \mathsf{S}.e^{(n,1)})$, check whether $\mathsf{S}_{\mathsf{soa}}$ aborts; if it does, then abort session j. Also, check whether the bits opened to from this response are equal to the extracted bits $a_1, \ldots, a_n$. If not, then abort the session; otherwise, sample $\theta' \xleftarrow{\$} \{0,1\}$, set $d := \theta' \oplus \sigma' \oplus a$, and send d to $\mathsf{R}^*_{\mathsf{soa}}$. Also, $\forall i, j$, run $(\mathsf{S}.\beta(e^{(i,j)}), \cdot) \xleftarrow{\$} \langle {\mathsf{S}_{\mathsf{ext}}}^{i,j}(\mathtt{com}, b^{(i,j)}), {\mathsf{R}_{\mathsf{ext}}}^{i,j}(\mathtt{recv})\rangle_2$ with $\mathsf{R}^*_{\mathsf{soa}}$. If $\mathsf{R}^*_{\mathsf{soa}}$ aborts, then abort this session.

Commitment phase completion. When the commitment phases of all k sessions are completed, Sim obtains the set of indices I, where $|I| \leq m$, from $\mathsf{R}^*_{\mathsf{soa}}$ and obtains $\{\mathbf{b}[j]\}_{j \in I}$ from the experiment.

Opening phase (S2)(Main thread). $\forall j_\ell \in I$, if the j_ℓth session is not aborted, then open it to $\mathbf{b}[j_\ell]$ as follows:

1. If $\mathbf{b}[j_\ell] = 0$ then set $\theta = \theta'$; otherwise set $\theta = 1 - \theta'$. Define $\sigma := \theta \oplus d \oplus a$, and $\forall i \in [n]$, run $(\gamma(e^{(i,\sigma_i)}), \cdot) \xleftarrow{\$} \langle {\mathsf{S}_{\mathsf{ext}}}^{i,\sigma_i}(\mathtt{com}, b_{i\sigma_i}), {\mathsf{R}_{\mathsf{ext}}}^{i,\sigma_i}(\mathtt{recv})\rangle_3$ with $\mathsf{R}^*_{\mathsf{soa}}$.

Finally, output the commitment phase transcript from (**S1**) and the opening phase transcript from (**S2**) together with the final output of $\mathsf{R}^*_{\mathsf{soa}}$.

Proposition 7. *The simulator* Sim *runs in expected polynomial time in* n.

Proof. The proof follows on the same lines as in Proposition 5 and is hence omitted. □

Proposition 8. *The distribution of the output of the simulator* Sim *having black-box access to* $\mathsf{R}^*_{\mathsf{soa}}$ *is computationally close to the output of* $\mathsf{R}^*_{\mathsf{soa}}$ *interacting with the real sender* $\mathsf{S}_{\mathsf{soa}}$.

Proof. The proof follows on the same lines as in Proposition 6 with a few changes. We however describe the hybrids for clarity and completion.

Consider the following sequence of hybrids:

H_0**:** In this experiment Sim has as input the bit-vector $\mathbf{b} \leftarrow \mathcal{B}$ before it begins interacting with $\mathsf{R}^*_{\mathsf{soa}}$ and follows the code of the honest sender $\mathsf{S}_{\mathsf{soa}}$ in committing to these bits. This is the real game.

$H_1^{j_\ell, i}$**:** This hybrid is the same as H_0 except for the following changes:

- For every j_cth session, where $c > \ell$, Sim behaves the same way as in H_0.
- For every j_cth session, where $0 \leq c < \ell$, Sim also gets as input $a_1, \ldots, a_n$ that $\mathsf{R}^*_{\mathsf{soa}}$ would later open to in the opening phase of the j_cth session. Then it proceeds as follows: Choose $\sigma' \xleftarrow{\$} \{0,1\}^n$; $\forall i' \in [n]$, set $b^{(i', j')} := 0$ for $j' = \sigma'_{i'}$ and set $b^{(i', j')} := 1$ for $j' = 1 - \sigma'_i$; $\forall i', j'$, run $(\mathsf{S}.\alpha^{(i,j)}, \cdot) \xleftarrow{\$} \langle {\mathsf{S}_{\mathsf{ext}}}^{i,j}(\mathtt{com}, b^{(i', j')}), {\mathsf{R}_{\mathsf{ext}}}^{i,j}(\mathtt{recv})\rangle_1$ with $\mathsf{R}^*_{\mathsf{soa}}$. Also, ample $\theta' \xleftarrow{\$} \{0,1\}$, set $d := a \oplus \sigma' \oplus \theta'$, $\sigma := d \oplus a$, and $\theta := \theta' \oplus \mathbf{b}[j_c]$. The rest of the messages are computed the same way as in H_0.
- In the j_ℓth session, sample $\theta' \xleftarrow{\$} \{0,1\}$. $\forall i' > i$, Sim computes the i'th pair of commitments the same way as in H_0. For every i'th pair of commitments, where $1 \leq i' \leq i$, Sim chooses

$\sigma'_i \xleftarrow{\$} \{0,1\}$, $b^{(i',j')} := 0$ for $j' = \sigma'_{i'}$, sets $b^{(i',j')} := 1$ for $j' = 1 - \sigma'_i$; $\forall i', j'$, and runs $(\mathsf{S}.\alpha^{(i,j)}, \cdot) \xleftarrow{\$} \langle {\mathsf{S}_{\mathsf{ext}}}^{i,j}(\mathtt{com}, b^{(i',j')}), {\mathsf{R}_{\mathsf{ext}}}^{i,j}(\mathtt{recv})\rangle_1$ with $\mathsf{R}^*_{\mathsf{soa}}$.

Note the the hybrids $H_1^{0,0}$ and H_0 are identical. We now show that if there exists a distinguisher D_{soa} that distinguishes any two consecutive hybrids $H_1^{j'_\ell,i'}$ and $H_1^{j_\ell,i}$, where $(j'_\ell, i') < (j_\ell, i)$, then we construct an efficient adversary ${\mathsf{R}_{\mathsf{ext}}}^*$ that breaks hiding of ExtCom. ${\mathsf{R}_{\mathsf{ext}}}^*$ interacts with $\mathsf{R}^*_{\mathsf{soa}}$ the same way as Sim in $H_1^{j_\ell,i}$, except the following change: To compute the ith pair of commitments in the j_ℓth session, ${\mathsf{R}_{\mathsf{ext}}}^*$ proceeds as follows: Choose $\sigma'_i \xleftarrow{\$} \{0,1\}$, set $d_i := a_i \oplus \sigma'_i \oplus \mathbf{b}[j_\ell]$. If $\mathbf{b}[j_\ell] = 0$, then for $j = \sigma'_i$, run $(\mathsf{S}.\alpha^{(i,j)}, \cdot) \xleftarrow{\$} \langle {\mathsf{S}_{\mathsf{ext}}}^{i,j}(\mathtt{com}, 0), {\mathsf{R}_{\mathsf{ext}}}^{i,j}(\mathtt{recv})\rangle_1$ with $\mathsf{R}^*_{\mathsf{soa}}$ and compute the rest of the messages corresponding to this commitment as per $\mathsf{S}_{\mathsf{ext}}$; for $j = 1 - \sigma'_i$, upon receiving α from the challenger, set $\mathsf{S}.\alpha^{(i,j)} := \alpha$ and send it to $\mathsf{R}^*_{\mathsf{soa}}$; upon receiving $\mathsf{S}.e^{(i,j)}$ from $\mathsf{R}^*_{\mathsf{soa}}$, set $e := \mathsf{S}.e^{(i,j)}$ and send it to the challenger; upon receiving $\beta(e)$ from the challenger, set $\mathsf{S}.\beta(e^{(i,j)}) := \beta(e)$ and send it to $\mathsf{R}^*_{\mathsf{soa}}$. Note that ${\mathsf{R}_{\mathsf{ext}}}^*$ will not be asked to provide opening for this commitment since $j = 1 - \sigma'_i$. If $\mathbf{b}[j_\ell] = 1$, then proceed as in the case of $\mathbf{b}[j_\ell] = 0$ except that, for $j = 1 - \sigma'_i$, run $(\mathsf{S}.\alpha^{(i,j)}, \cdot) \xleftarrow{\$} \langle {\mathsf{S}_{\mathsf{ext}}}^{i,j}(\mathtt{com}, 1), {\mathsf{R}_{\mathsf{ext}}}^{i,j}(\mathtt{recv})\rangle_1$ with $\mathsf{R}^*_{\mathsf{soa}}$, and $j = \sigma'_i$, the (i,j)th ExtCom commitment is performed using the interaction with the challenger as explained for the case of $\mathbf{b}[j_\ell] = 0$. Similarly, in this case too, ${\mathsf{R}_{\mathsf{ext}}}^*$ will not be asked to provide opening for the (i,j)th commitment for $j = \sigma'_i$. Finally, when D_{soa} outputs a bit b', then ${\mathsf{R}_{\mathsf{ext}}}^*$ outputs $b' \oplus \mathbf{b}[j_\ell]$.

Note here that if the hiding experiment of ${\mathsf{R}_{\mathsf{ext}}}^*$ is $\mathbf{Exp}^{\text{hiding-}b}_{\mathsf{ExtCom},{\mathsf{R}_{\mathsf{ext}}}^*}(n)$ with $b \neq \mathbf{b}[j_\ell]$ then ${\mathsf{R}_{\mathsf{ext}}}^*$ is running $H_1^{j_\ell,i}$ with $\mathsf{R}^*_{\mathsf{soa}}$; otherwise, it is running $H_1^{j'_\ell,i'}$. Thus, ${\mathsf{R}_{\mathsf{ext}}}^*$ breaks hiding of ExtCom with the same probability that D_{soa} distinguishes $H_1^{j'_\ell,i'}$ and $H_1^{j_\ell,i}$.

$H_{2,0}$: This experiment is same as $H_1^{j_m,n}$, except that Sim also extracts each of the bits $a_1, \ldots, a_n$ using the extractor E for every non-aborted session that it is asked to provide opening for. The extraction is performed for one session after the other as explained in the description of the original simulator Sim. If new sessions are initiated by $\mathsf{R}^*_{\mathsf{soa}}$ in the rewinding threads, then Sim will be given $a_1, \ldots, a_n$ as input for these new sessions and it will proceed the same way as in the commitment phase of the main threads. Sim aborts the simulation if it fails to extract any of the bits $a_1, \ldots, a_n$ in any such non-aborted session.

Now note that the statistical distance between the hybrids $H_1^{j_m,n}$ and H_2 is at most the probability that Sim aborts due to failure in the extraction. Since this probability is at most $\mathtt{poly}(n)/2^n$, we observe that the two hybrids are negligibly close.

$H_{2,1}$: This experiment is same as $H_{2,0}$, except that Sim will not be given $a_1, \ldots, a_n$ for the newly initiated sessions in the rewinding threads, and Sim will choose d at random from $\{0,1\}^n$.

From the same argument as before, if there exists an distinguisher D_{soa} that distinguishes $H_{2,1}$ and $H_{2,0}$, then we construct an efficient adversary ${\mathsf{R}_{\mathsf{ext}}}^*$ that breaks hiding of ExtCom.

H_3: This experiment is same as H_2, except that Sim also aborts the simulation if in any of the sessions, any of the extracted bits $a_1, \ldots, a_n$ is inconsistent with the corresponding opening provided by $\mathsf{R}^*_{\mathsf{soa}}$ in the opening phase.

Assume that there exists a distinguisher D_{soa} that distinguishes H_2 and H_3. Then we construct an adversary ${\mathsf{S}_{\mathsf{ext}}}^*$ that breaks binding of ExtCom. ${\mathsf{S}_{\mathsf{ext}}}^*$ interacts with $\mathsf{R}^*_{\mathsf{soa}}$ the same way as Sim in H_3.

Whenever $\mathsf{R}^*_{\mathsf{soa}}$ provides an opening to a bit a'_i that is inconsistent with it corresponding opening, then outputs the ExtCom commitment phase transcript corresponding to this bit along with the extracted opening and the opening provided by $\mathsf{R}^*_{\mathsf{soa}}$ in the opening phase, thus breaking the binding of ExtCom with the same probability that D_{soa} that distinguishes H_2 and H_3, and in turn, implying the indistinguishability of the two hybrids.

H_4**:** This experiment is same as H_3, except that Sim does not receive as input $a_1, \ldots, a_n$ for any session, but uses the extracted bits in setting the value of d for the same.

If Sim does not abort in H_3 due to openings that are inconsistent with the extractions, then the extracted bits are equal to those opened by $\mathsf{R}^*_{\mathsf{soa}}$ in the opening phase. Thus, we have that the hybrids H_3 and H_4 are identical. Also, H_4 corresponds to the ideal game. This completes the proof of indistinguishability of the output of Sim from the output of $\mathsf{R}^*_{\mathsf{soa}}$. □

□

D.5 Proof of Theorem 3

Proof.
Completeness. It follows from the completeness of sub-protocols $\mathsf{Com}_{\mathsf{NI}}$, ExtCom and FLS.

Binding. For the binding property it is sufficient to consider the protocol in the stand-alone setting. Therefore we focus on one single session of the protocol SOACom. Intuitively the binding property holds due to the statistical binding of the scheme $\mathsf{Com}_{\mathsf{NI}}$ (i.e., $\mathsf{S}^*_{\mathsf{soa}}$ cannot open C or $\mathsf{c}_{\mathtt{wi}}$ in two ways), to the hiding of ExtCom (i.e., $\mathsf{S}^*_{\mathsf{soa}}$ cannot guess the value committed by $\mathsf{R}_{\mathsf{soa}}$), and to the proof of knowledge property of the FLS protocol, (i.e., a valid witness can be extracted from what $\mathsf{S}^*_{\mathsf{soa}}$ is able to prove).

The formal proof follows. Assume that there exists a PPT malicious sender $\mathsf{S}^*_{\mathsf{soa}}$ that, keeping the same commitment transcript, is able to provide two accepting openings with non-negligible probability δ. Then we can construct an adversary ${\mathsf{S}_{\mathsf{NI}}}^*$ for the binding of $\mathsf{Com}_{\mathsf{NI}}$. The reduction works as follows: ${\mathsf{S}_{\mathsf{NI}}}^*$, that is playing the binding experiment interacting with the honest receiver R_{NI}, internally runs $\mathsf{S}^*_{\mathsf{soa}}$ as subroutine simulating the honest receiver to her. When $\mathsf{S}^*_{\mathsf{soa}}$ runs the commitment phase of $\mathsf{Com}_{\mathsf{NI}}$, ${\mathsf{S}_{\mathsf{NI}}}^*$ forwards the message to the external receiver R_{NI}. When the commitment phase is over, ${\mathsf{S}_{\mathsf{NI}}}^*$ is activated with bit 0 and he has to provide the first opening. In order to do it, ${\mathsf{S}_{\mathsf{NI}}}^*$ extracts the witness from the $\mathcal{WIPOK}$ FLS provided by $\mathsf{S}^*_{\mathsf{soa}}$ (also activated with input 0). By the correctness of FLS the witness extracted is either the opening of commitment C or of $\mathsf{c}_{\mathtt{wi}}$. Note that the extraction does not affect the execution of the parallel protocol ExtCom, indeed, it only requires to change $\mathsf{FLS2}$, leaving everything else untouched. If the witness is the opening of C, then ${\mathsf{S}_{\mathsf{NI}}}^*$ forwards the opening, and repeat the same process when activated with bit 1. If the witness is the opening of $\mathsf{c}_{\mathtt{wi}}$ then ${\mathsf{S}_{\mathsf{NI}}}^*$ aborts, otherwise ${\mathsf{S}_{\mathsf{NI}}}^*$ has got two openings which however can not happen by the statistical binding of $\mathsf{Com}_{\mathsf{NI}}$. Therefore the only remaining case to prove is that the probability of abort (due to the extraction of an opening of C) is negligible.

Assume that after running the extractor for FLS given by $\mathsf{S}^*_{\mathsf{soa}}$, ${\mathsf{S}_{\mathsf{NI}}}^*$ obtains the opening for $\mathsf{c}_{\mathtt{wi}}$ with non-negligible probability. Then, it is possible to show an adversary for the hiding of ${\mathsf{R}_{\mathsf{ext}}}^*$ for the scheme ExtCom. The reduction works as follows.

${\mathsf{R}_{\mathsf{ext}}}^*$ run $\mathsf{S}^*_{\mathsf{soa}}$ internally, while it is interacting with the honest sender $\mathsf{S}_{\mathsf{ext}}$ in the external experiment. Thus, it forwards all messages belonging to commitment phase of ExtCom between $\mathsf{S}_{\mathsf{ext}}$ and $\mathsf{S}^*_{\mathsf{soa}}$ (note that this is possible since $\mathsf{R}_{\mathsf{soa}}$ does not open the commitment), while it will play honestly for messages belonging only to SOACom protocol. When the protocol is completed ${\mathsf{R}_{\mathsf{ext}}}^*$ runs the FLS extractor to

obtain the witness, and output it. By the hiding of ExtCom this event happens with negligible probability. Hence, $\mathsf{S_{NI}}^*$ breaks the binding of $\mathsf{Com_{NI}}$ with probability δ- ϵ.

Hiding under selective opening attack. Let us fix some notation that will be used in the proof. The malicious receiver $\mathsf{R}^*_{\mathsf{soa}}$ plays k concurrent sessions of SOACom, more precisely she will run k concurrent executions of the commitment phase, and up to $m = |I|$ concurrent decommitment phases. We use the index j to refer to the sessions opened by $\mathsf{R}^*_{\mathsf{soa}}$ in the main thread, (i.e., $0 \leq j \leq k$). We use t to refer to the new sessions (i.e., not opened in the main thread) that $\mathsf{R}^*_{\mathsf{soa}}$ may open in the extraction phase (rewinding threads) due to the fact that during the extraction attempts her view changes. The simulator proceeds with the extraction only for sessions that have been asked to be opened. In particular, since I is a set of indexes there are at most $|I| = m$ sessions to be extracted. We refer to those sessions using indexes j_ℓ for $\ell \in [m]$. We indicate with max_{j_ℓ} the maximum number of new sessions opened in the rewinding threads for the session j_ℓ.

Very informally, the simulator's strategy is the following: in the commitment phase it honestly follows the sender's strategy except that it commits to bit 0. Note that Sim can be honest in computing $\mathsf{FLS1}, \mathsf{FLS2}$ since in this $\mathcal{WIPOK}$ the knowledge of the witness is required only in the last round $\mathsf{FLS3}$. When the commitment phase is over, Sim starts the rewinding threads in order to extract the secrets committed by the malicious verifier. Note that the extraction is done sequentially, running the extractor of ExtCom scheme, one session at time. This allow Sim to be expected polynomial time. Note also, that during the extraction, since the view of $\mathsf{R}^*_{\mathsf{soa}}$ changes, she can require to open new sessions that were not opened in the main thread. In this case Sim follows the same procedure of the sender committing the bit 0. Nevertheless, Sim will not need to extract in these new sessions, since they did not appear in the main thread. Notice that here we are crucially using the fact that there is a separation between the commitment and the decommitment phase. Furthermore, note that the transcript of the commitment phase of the main thread is not changed after the extraction phase. Finally, in the decommitment phase, Sim can continue the main thread and can honestly compute $\mathsf{FLS3}$ using as witness the secrets extracted in the extraction phase.

SOA-simulator Sim

Initialization phase. Choose random tapes $\mathsf{ran_R}, \mathsf{ran_{Sim}}$ respectively for $\mathsf{R}^*_{\mathsf{soa}}$ and for the commitment phase. Activate $\mathsf{R}^*_{\mathsf{soa}}(\mathsf{ran_R})$.

Commitment phase (S1). (main thread)

- Honestly run ExtCom and honestly computes $\mathsf{FLS1}, \mathsf{FLS2}$.
- Run $\mathsf{Com_{NI}}$ to commit to bit 0. Let C be the output.

Commitment phase completion. When the commitment phase of all k sessions is completed, Sim obtains the set of indexes I from $\mathsf{R}^*_{\mathsf{soa}}$ and obtains $\{\mathbf{b}[j_\ell]\}_{j_\ell \in I}$ from the experiment.

Extraction phase. (rewinding thread)

For $\ell = 1, \ldots, m$; if j_ℓ was not-aborted in the main thread do:

1. activate $\mathsf{R}^*_{\mathsf{soa}}$ with randomness $\mathsf{ran_R}$ and executes all the sessions excepts j_ℓ using $\mathsf{ran_{Sim}}$.
2. in session j_ℓ run Step $E2$ of the extractor ExtCom to obtain one of the α_i along with the opening [8]. Furthermore, each rewinding attempt changes the view of $\mathsf{R}^*_{\mathsf{soa}}$ that could thus open new sessions (i.e., sessions that did not appear in the main thread). If $\mathsf{R}^*_{\mathsf{soa}}$ starts new sessions, follow instructions as in (**S1**). If no α_i is obtained, repeat Step 2.

Extraction completion. If Sim reaches this point, then for each (not-aborted) session $j_\ell \in I$ there is

[8] Recall that since $\mathsf{R}^*_{\mathsf{soa}}$ does not open the commitments, she could cheat in ExtCom and instead of committing to the same string α in all n pairs, it can commit to arbitrary strings. However, we are guaranteed that, if the commitment phase is valid, the extractor does extract one of the string committed in the minor commitments, along with the opening data.

at least one i such that Sim obtained α_i^0, α_i^1 along with the openings.

Decommitment phase(S2)(main thread). Run $\mathsf{R}^*_{\mathsf{soa}}(\mathsf{ran}_{\mathsf{R}})$ till the completion of the commitment phase using $\mathsf{ran}_{\mathsf{Sim}}$. Then for non-aborted sessions $j_\ell \in I$, with $\ell \in [m]$, let $\mathbf{b}[j_\ell]$ be the bit to decommit to in the session j_ℓ, let i the index such that Sim obtained (α_i^0, α_i^1) along with the openings, for the session j_ℓ.

When $\mathsf{R}^*_{\mathsf{soa}}$ asks for the decommitment of the j_ℓ-th session do as follows:

1. Commit to $\alpha_i^0 \oplus \alpha_i^1$ using $\mathsf{Com}_{\mathsf{NI}}$. Let be $\mathsf{c}_{\mathtt{wi}}$ the output. Computes $\mathsf{FLS3}$ using as witness the openings associated with (α_i^0, α_i^1) and the opening of $\mathsf{c}_{\mathtt{wi}}$. Send $\mathbf{b}[j_\ell]$ to $\mathsf{R}^*_{\mathsf{soa}}$. If $\mathsf{R}^*_{\mathsf{soa}}$ abort, then aborts this session.

Finally, outputs whatever $\mathsf{R}^*_{\mathsf{soa}}$ outputs.

Proposition 9. *The simulator* Sim *runs in expected polynomial time in* n.

Proof. Sim consists of three phases: commitment phase, extraction phase, opening phase. The commitment phase (**S1**) is the honest commitment phase of SOACom (for committing the bit 0). Thus the running time to Sim in the commitment phase, $\mathsf{t}_{\mathtt{SimCom}}$, is polynomial in n.

After the completion of the commitment phase Sim is asked for openings of $m = |I|$ sessions, and for each such session that is also not aborted, Sim runs the extractor E.

Let us denote as $\zeta^j = \zeta(\mathsf{ran}_{\mathsf{R}}, j)$ the probability that $\mathsf{R}^*_{\mathsf{soa}}$, activated with randomness $\mathsf{ran}_{\mathsf{R}}$, correctly answered to the challenge sent by Sim in the commitment phase of the j-th session. While running E, since Sim generates the messages other than those generated by E the same way as in the commitment phase, the view of $\mathsf{R}^*_{\mathsf{soa}}$ in the rewinding threads is distributed identically as in the commitment phase. Hence, the probability that $\mathsf{R}^*_{\mathsf{soa}}$ responds without aborting is also bounded by ζ^j. As noted in the analysis of E, its expected running time is $\zeta^j \cdot 1/\zeta^j$. Further, in each rewinding performed by E, since the number of new sessions initiated by $\mathsf{R}^*_{\mathsf{soa}}$ is bounded by a polynomial, the extra running time of Sim due to the newly initiated sessions for each rewinding performed by E is bounded by $\mathtt{poly}(n) \cdot \mathsf{t}_{\mathtt{SimCom}}$. (Obviously, once the simulator extracts the trapdoor for the target session, the new sessions are discarded.) With $\mathsf{t}_{\mathtt{rew}}$ denoting the running time of Sim each time it runs the step $E2$ the expected running time of Sim in the extraction phase is:

$$\sum_{j=1}^{m} \zeta^j \Big[\frac{1}{\zeta^j} \cdot \mathsf{t}_{\mathtt{rew}}\Big] = \mathtt{poly}(n)$$

In the decommitment phase Sim just executes the decommitment algorithm of SOACom for at most m sessions. Since the running time of $\mathsf{S}_{\mathsf{soa}}$ in the decommitment phase of SOACom and m are both polynomial in n, the running time of Sim in the decommitment phase, $\mathsf{t}_{\mathtt{SimDec}}$, is also polynomial in n. Thus, the total expected running time:

$$\mathsf{t}_{\mathsf{Sim}} = k \cdot \mathsf{t}_{\mathtt{SimCom}} + \sum_{j=1}^{m} \zeta^j \Big[\frac{1}{\zeta^j} \cdot \mathsf{t}_{\mathtt{rew}} + \mathsf{t}_{\mathtt{SimDec}}\Big]$$

is also polynomial in n. □

Proposition 10. *The distribution of the output of the simulator* Sim *having black-box access to* $\mathsf{R}^*_{\mathsf{soa}}$ *is computationally close to the output of* $\mathsf{R}^*_{\mathsf{soa}}$ *interacting with the real sender* $\mathsf{S}_{\mathsf{soa}}$.

Intuitively, the output of Sim is indistinguishable to an efficient adversary for the following reasons. In the commitment phase the only difference between Sim and $\mathsf{S}_{\mathsf{soa}}$ is that Sim commits to the bit 0. By

the computational hiding of $\mathsf{Com_{NI}}$ this is not detected by any efficient $\mathsf{R}^*_{\mathsf{soa}}$. In the decommitment phase, Sim commits to a secret extracted from $\mathsf{R}^*_{\mathsf{soa}}$ (instead of committing to a junk string) and provides the special $\mathcal{WIPOK}$ using another witness. Again, by the hiding of $\mathsf{Com_{NI}}$ and the witness indistinguishability of FLS, the view of $\mathsf{R}^*_{\mathsf{soa}}$ in the decommitment phase is indistinguishable from the one generated by the honest sender. Clearly, Sim is able to perform the decommitment phase by relying on to the extractability of ExtCom.

A formal proof follows. Consider the following hybrids:

H_0**:** In this hybrid Sim has in input the bit-vector $\mathbf{b} \leftarrow \mathcal{B}$ and follows the code of the honest sender $\mathsf{S}_{\mathsf{soa}}$. This is the real game.

H_1**:** This hybrid is the same as H_0 except that here, after the completion of the commitment phase, for all non-aborted sessions that are asked to be opened (that we denote as j_ℓ), Sim executes the extraction phase. The extraction is performed one session at time. Possible new sessions initiated by $\mathsf{R}^*_{\mathsf{soa}}$ (that we denote as t) are handled running the honest sender procedure using the knowledge of $\mathbf{b}$. The extracted secrets are never used in the protocol. We now argue that H_0 and H_1 are indistinguishable. First note that, the extraction phase is initiated only for non-aborted sessions, therefore, only for those in which $\mathsf{R}^*_{\mathsf{soa}}$ correctly completed the commitment phase. Then note that the view of $\mathsf{R}^*_{\mathsf{soa}}$ in the rewinding threads is distributed identically to her view in the commitment phase (main thread). Thus the only differences between the two experiments is that 1) in H_1 Sim runs in expected polynomial time, this is not a problem since we are only interested in the output of the experiment (and it will not be a problem in the reductions shown for the next hybrids since rewinding threads that take more time than a fixed polynomial, can be truncated, without perturbing the non-negligible probability of success); 2) in H_1 Sim aborts with higher probability due to the possible aborts in the rewinding threads. Due to the extractability property of ExtCom this event happens with negligible probability. Thus H_0 and H_1 are statistically indistinguishable.

$H_2^{j_\ell,s+1}$**:** (**for** $\ell = 1,\ldots,m$**; for** $s = 0,\ldots,\mathsf{max}_{j_\ell}$)**.** In this sequence of hybrids Sim follows the strategy of the honest sender in the main thread, while in the (possible) *new* sessions that are opened by $\mathsf{R}^*_{\mathsf{soa}}$ in the rewinding threads it commits to 0 (recall that for these sessions the decommitment phase is not executed). Toward showing the indistinguishability of $H_2^{j_\ell,s}$ and $H_2^{j_\ell,s+1}$, we first show that $H_2^{j_\ell,s}$ is indistinguishable from an hybrid $\bar{H}_2^{j_\ell,s+1}$ where the bit committed in the $(s+1)$-th new session is the opposite bit used in $H_2^{j_\ell,s}$.

More precisely hybrid $\bar{H}_2^{j_\ell,s+1}$ is the same as hybrid $H_2^{j_\ell,s}$ except that in the commitment phase of the $(s+1)$-th new session, which index (in the range between 1 and k of all sessions played in the current view) we denote by t, initiated by $\mathsf{R}^*_{\mathsf{soa}}$ in the extraction phase of session j_ℓ, in the commitment phase it runs the procedure of $\mathsf{S_{NI}}$ for the bit $(1-\mathbf{b}[t])$.

Assume that there exists a distinguisher D_{soa} that is able to tell apart hybrid $\bar{H}_2^{j_\ell,s+1}$ from $H_2^{j_\ell,s}$ then it is possible to construct an adversary for the hiding of the scheme $\mathsf{Com_{NI}}$, that we denote as $\mathsf{R_{NI}}^*$. The reduction works as follows. $\mathsf{R_{NI}}^*$, playing in the hiding experiment, (wlog since $\mathsf{Com_{NI}}$ is non-interactive) receives the commitment C from $\mathsf{S_{NI}}$. Then it runs $\mathsf{R}^*_{\mathsf{soa}}$ as sub-routine. It picks a vector $\mathbf{b} \leftarrow \mathcal{B}$, and runs as Sim in the hybrid $H^{j_\ell,s}$ except that, when $\mathsf{R}^*_{\mathsf{soa}}$ initiates the commitment phase for session t $\mathsf{R_{NI}}^*$ forwards the commitment received by $\mathsf{S_{NI}}$. At the end of the simulation, $\mathsf{R_{NI}}^*$ hands to D_{soa} the output of $\mathsf{R}^*_{\mathsf{soa}}$. Now, if C is a commitment of $\mathbf{b}[t]$ then the view of $\mathsf{R}^*_{\mathsf{soa}}$ is distributed as in hybrid $H^{j_\ell,s}$, otherwise, if C is a commitment of $1-\mathbf{b}[t]$ then the view of $\mathsf{R}^*_{\mathsf{soa}}$ is

distributed as in hybrid $H^{j_\ell,s+1}$. Thus, by the computational hiding of $\mathsf{Com_{NI}}$ $H_2^{j_\ell,s}$ and $H_2^{j_\ell,s+1}$ are indistinguishable.

Now, in $H_2^{j_\ell,s+1}$ the bit committed in session t (i.e., the $(s+1)$-th new session) is 0, and therefore the output of any distinguisher on $H_2^{j_\ell,s+1}$ will be indistinguishable from the one of $H_2^{j_\ell,s}$ and $\bar{H}_2^{j_\ell,s+1}$.

Therefore, $H_1 = H_2^{j_1,0}$ and $H_2^{j_m,\mathsf{max}_{j_m}}$ are indistinguishable.

$H_3^{j_{\ell+1}}$ **(for $\ell = 1,\ldots,\ell = m$):** Hybrid $H_3^{j_{\ell+1}}$ is the same as $H_3^{j_\ell}$ except that in $H_3^{j_{\ell+1}}$, in the decommitment phase of the session j_ℓ, Sim commits to the secret extracted for session j_ℓ instead of committing to the junk string 0^n. By the computational hiding of $\mathsf{Com_{NI}}$, following the same arguments shown above, $H_3^{j_\ell}$ and $H_3^{j_{\ell+1}}$ are computational indistinguishable.

Therefore $H_2^{j_m,\mathsf{max}_{j_m}} = H_3^{j_1}$ and $H_3^{j_k}$ are computationally indistinguishable.

$H_4^{j_{\ell+1}}$ (for $\ell = 1,\ldots,\ell = m$): Hybrid $H_4^{j_{\ell+1}}$ is the same as $H_4^{j_\ell}$ except that here, in the decommitment phase of the session j_ℓ, Sim computes $\mathsf{FLS3}$ using as witness the secret extracted. Assume that there exists a distinguisher D between $H_4^{j_{\ell+1}}$ and $H_4^{j_\ell}$ then it is possible to construct an adversary V^* for the WI of FLS protocol. The reduction works as follows: V^* plays the WI experiment running with a hones prover of FLS. It runs $\mathsf{R}^*_{\mathsf{soa}}$ as sub-routine executing the procedure of Sim as in experiment $H_4^{j_\ell}$ except that in session $j_{\ell+1}$ the messages belonging to FLS protocol are forwarded to the external prover. Note that, by the special property of FLS for which the first two rounds $\mathsf{FLS1}$ and $\mathsf{FLS2}$ are independent from the theorem, V^* can forward the first two messages to the prover without having established the theorem, and chosen the witness, to prove yet. Indeed, V^* obtains the witness only after the extraction phase. Hence, in the decommitment phase, V^* provides the external prover with the theorem to prove and both witnesses, i.e, the extracted secret along with the opening of the commitment $\mathsf{c_{wi}}$ and the opening of the commitment C along with b. Finally, it forwards the last message $\mathsf{FLS3}$ to $\mathsf{R}^*_{\mathsf{soa}}$. At the end of the experiment it forward the output of $\mathsf{R}^*_{\mathsf{soa}}$ to D and outputs whatever D outputs. If the external prover used the extracted secret as witness, then the view of $\mathsf{R}^*_{\mathsf{soa}}$ is distributed identically to hybrid $H_4^{j_{\ell+1}}$, otherwise if the prover used as witness the opening of the commitment C then the view of $\mathsf{R}^*_{\mathsf{soa}}$ is distributed identically to hybrid $H_4^{j_\ell}$. Hence, by the witness indistinguishability property of FLS hybrids $H_4^{j_{\ell+1}}$ and $H_4^{j_\ell}$ are indistinguishable.

Therefore $H_3^{j_m,\mathsf{max}_{j_m}} = H_4^{j_1}$ and $H_4^{j_k}$ are computationally indistinguishable.

H_5^{j+1} (for $j = 1,\ldots,j = k$): Hybrid $H_5^{j_{j+1}}$ is the same as H_5^{j} except that in $H_5^{j_{j+1}}$, in the commitment phase of the j-th session Sim commits to the bit 0. Following the same arguments of hybrid $H_2^{j_\ell,s}$ and $H_2^{j_\ell,s+1}$, hybrids $H_5^{j_{j+1}}$ and H_5^{j} are computationally indistinguishable.

Hence $H_4^{j_k} = H_5^{j}$ and $H_5^{j_k}$ are indistinguishable. By noticing that $H_5^{j_k}$ corresponds to the game played by the simulator, we have that the claim holds. This concludes the proof of the Theorem. □

D.6 Proof of Theorem 5

Proof. In our proof we assume the existence of OWFs, which is implied by the existence of any commitment scheme. Let Π be a r-round string commitment protocol that is SOA-secure under concurrent composition. By definition there exists a black-box simulator Sim that for all R^* is able to produce a view $(\tau^k, I, \{b_i, w_i\}_{i\in I})$ that is indistinguishable from the view of the interaction between R^* and S. In the next part of the proof we will use Sim as a subrouting to contradict in strict polynomial time some hardness assumptions. Since Sim is only expected poylomial time, we are implicity assuming that we run it up to

some strict polynomial number of steps (obviously greater than the expected polynomial time), and our results will still work since Sim is often successful before that poylonomial number of steps.

The formal proof consists of the following steps. First, we define the family of message distributions $\mathcal{B}$. Then we define a family of adversaries $\{\mathsf{R}^*_p\}_{p\in\{0,1\}}$ and we show that such adversaries make the rewinding strategy of any black-box Sim ineffective for one protocol execution. Still, by the security of Π it must be the case that Sim is able to successfully carry out the simulation of such execution even without rewinding R^*_p. Finally we construct a malicious sender that runs such a simulator as sub-routine to break the binding of Π, thus contradicting the hypothesis that Π is a commitment scheme.

Distribution $\mathcal{B}$. Recall that $\mathcal{B}$ is the set of distribution over $(\{0,1\}^n)^k$, where k is the number of sessions. In order to define our particular set $\mathcal{B}$ we need to define signature scheme. A signature scheme is a triplet of algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ where Gen is the generation algorithm that on input the security parameter outputs a pair vk, sk where sk is the secret key and vk is the verification key. Sign is a randomized signing algorithm and Vrfy is the verification algorithm. The correctness requirement states that for all $(vk, sk) \stackrel{\$}{\leftarrow} \mathsf{Gen}(1^n)$, all messages x, all randomness r, $\mathsf{Vrfy}(vk, x, \mathsf{Sign}(sk, x, r)) = 1$. The security requirement (called unforgeability) states that no efficient algorithm M, even after having seen polynomially many signatures of messages of her choice is able to produce a pair (x^*, σ^*) such that $\mathsf{Vrfy}(vk, x^*, \sigma^*) = 1$ without knowing sk. We define our set $\mathcal{B} = \{\mathcal{B}_{sk}\}_{sk\stackrel{\$}{\leftarrow}\mathsf{Gen}(1^n)}$ where:

$$\mathcal{B}_{sk} = \{\sigma : \exists i \in \{1, \ldots, k\}, r \in \{0,1\}^n : \sigma = \mathsf{Sign}(sk, i, r)\}$$

Thus we define the message space as a set of signatures under a secret key sk, in particular the message committed in session j is the signature of j under sk. Then we assume that the adversarial receiver is given in auxiliary input the verification key vk, such that, it is allowed to check whether messages obtained in the opening phase belong to the distribution $\mathcal{B}_{sk}$. Notice that, having defined $\mathcal{B}$ in this way, we have that a query made by Sim to oracle $\mathcal{O}$ to receive the opening of an index j, correspond to a query to a signing oracle $\mathcal{O}^{sk}$ for the signature $\mathsf{Sign}(sk, j, r)$. Such definition of $\mathcal{B}$ allows us to formally claim that in order to simulate the honest sender Sim is forced to ask $\mathcal{O}$, unless it is able to forge the signatures. Having fixed $\mathcal{B}_{sk}$ we are ready to define the adversaries $\{\mathsf{R}^*_p\}_{p\in\{0,1\}}$.

Adversary's strategy. Let Π be a r-round string commitment concurrent SOA-secure protocol. R^*_p runs $k = r2n + 1$ protocol executions. Let us denote by $\Pi_1, \ldots, \Pi_k$ the k protocol executions, by $\Pi_j(i)$ the i-th round of the protocol Π_j, and by $\Pi_j(i)_{\mathsf{S}}$ and $\Pi_j(i)_{\mathsf{R}}$ the messages sent by S and R in that round of the protocol. Let $F_k : \{0,1\}^* \to \{0,1\}^n$ a PRF for $k \stackrel{\$}{\leftarrow} \{0,1\}^n$.

The adversary's strategy is the following.

R^*_p plays the first round of Π_1 (i.e $\Pi_1(1)$) following the procedure of the honest receiver and then it starts a block of $2n$ executions in parallel $(\Pi_2, \ldots, \Pi_{2n+1})$. In this block of executions R^*_p honestly completes the $2n$ commitment phases while the selection for the sessions to open (denoted as I_1) is computed as follows: consider the $2n$ executions as a sequence of n pairs (each pair consists of left and right execution) then: 1. R^*_p computes a n-bit string $s_1 \leftarrow F_k(\Pi_1(1)_{\mathsf{S}})$; 2. consider the ℓ-th pair with $\ell \in [n]$, among the n pairs of executions, R^* asks to open the left execution of the ℓ-th pair if the ℓ-th bit of s_1 is 0 and the right execution otherwise. We denote this process of selecting the set of positions I according to the output of F_k by $I_1 \rightleftharpoons F_k(\Pi_1(1)_{\mathsf{S}})$. Then R^*_p sends I_1 to S and obtains the openings $\sigma_{j_1}, \ldots, \sigma_{j_n}$ with $j_i \in I_1$ and checks if $\mathsf{Vrfy}(vk, j_i, \sigma_{j_i}) = 1$ for all $j_i \in I_1$. If any check fails, then it aborts. Otherwise, R^* runs $\Pi_1(2)$ (the second round of Π_1) and starts another block of $2n$ executions till completion as described before. In general, after the ith round of Π_1, R^*_p initiates a block of $2n$ parallel executions of Π $(\Pi_{2(i-1)n+i+1}, \ldots, \Pi_{2in+i})$ and selects the subset of positions I_i according to $F(\Pi_1(i)_{\mathsf{S}})$.

Finally, when the commitment phase of the execution Π_1 is completed, R^*_p ask for the opening with probability p. In Fig. 1 we show the diagram of the schedule.

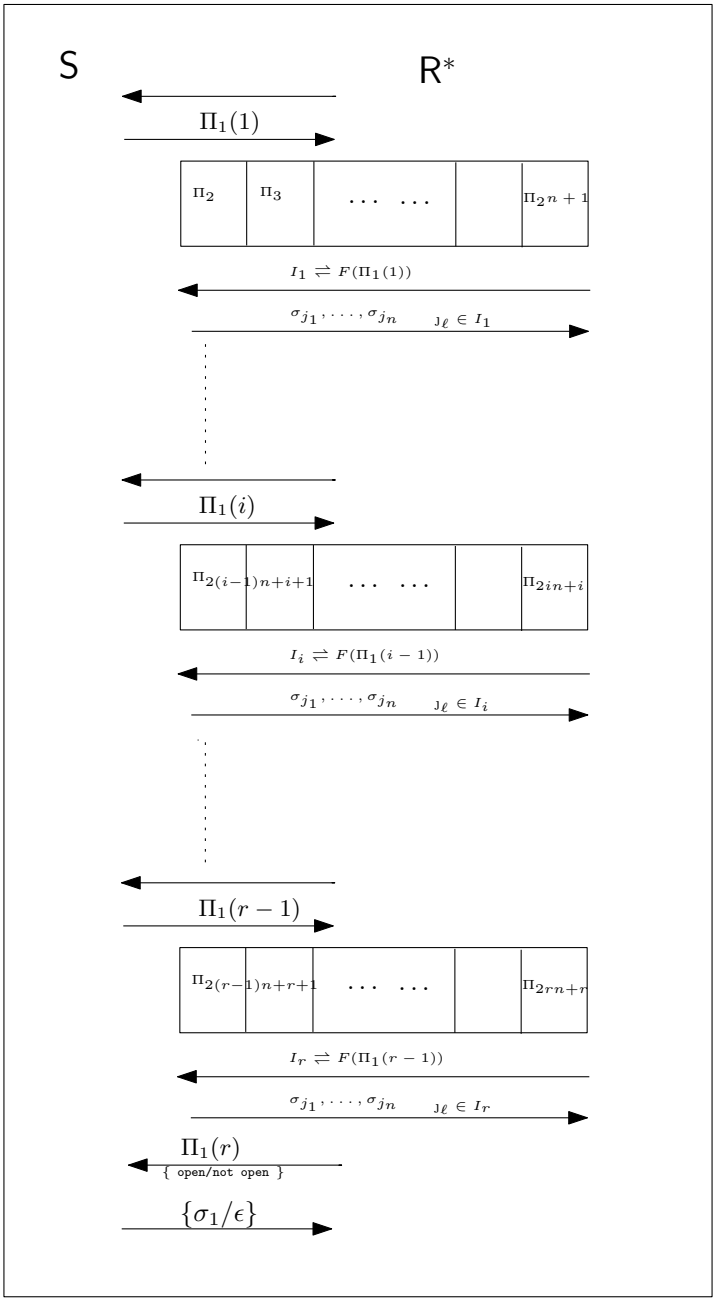

Figure 1: Adversarial strategy of $\mathsf{R}^*_0/\mathsf{R}^*_1$.

Ideal-World Simulator. By the assumption that Π is black-box secure we have that there exists a black-box simulator Sim such that the output of the ideal execution with Sim is indistinguishable from the result of a real execution of $\{\mathsf{R}^*_p\}_{p\in\{0,1\}}$ running $\Pi_1, \ldots, \Pi_k$ with S. As black-box simulator Sim must work for all malicious R^* and therefore also for R^*_0 and R^*_1 defined above. Recall that Sim is given oracle access

to R_p^*, namely Sim is given a next message function that receives a sequence of messages and computes R_p^*'s response. If any prefix of the query is such that R_p^* would abort upon that message, then the output for the entire query is $\perp$. We now prove a special property of *all* oracle calls in which R_p^* does *not* abort.

Claim 1. *For every i let Q_i be the set of all queries sent by Sim to R_p^* during the ideal world execution which include all messages from the block of executions activated after the message $\Pi_1(i)_\mathsf{S}$ (i.e., from $\Pi_{2(i-1)n+i+1}$ to $\Pi_{2(i)n+i}$) and where R_p^* does not abort[9]. Then, the same message $\Pi_1(i)_\mathsf{S}$ appears in every $q \in Q_i$, except with negligible probability.*

Proof. The proof of this claim follows almost identically the claim proved in [Lin03] except that here we use signature scheme instead of one-time signature scheme and between each round of the protocol Π_1 here the adversary opens a bunch of $2n$ new executions instead of only one. As discussed in the paragraph of the proof intuition, the main reason we have these differences is that in the SOA-secure setting we cannot exploit the fact that both parties have private inputs. The proof is based on the unforgeability of the signature scheme and the collision resistance property of the PRF.

First, we claim that Sim does not produce two oracle queries q and q' containing $\Pi_1(i)_\mathsf{S} \neq \Pi_1(i)'_\mathsf{S}$ such that $F_k(\Pi_1(i)_\mathsf{S}) = F_k(\Pi_1(i)'_\mathsf{S})$ with non negligible probability. Otherwise we can construct a machine M that has oracle access to a random function and distinguishes if the oracle is F_k (for a random k) or a truly random function. Machine M works by emulating the entire experiment between R^* and Sim except that instead of R^* computing $s_i = F_k(\Pi_1(i)_\mathsf{S})$, machine M queries the oracle with $\Pi_1(i)_\mathsf{S}$. Now, if the oracle is F_k then the emulation is perfect. Therefore with non-negligible probability M obtains from Sim two messages $\Pi_1(i)_\mathsf{S} \neq \Pi_1(i)'_\mathsf{S}$ such that the oracle responses is the same. On the other hand, if the oracle is a truly random function then the collision happens with negligible probability. Thus M distinguishes with non-negligible probability.

Now we prove that Sim cannot produce two non-aborting queries q, q' such that q contains message $\Pi_1(i)_\mathsf{S}$ and q' contains message $\Pi_1(i)'_\mathsf{S}$. This claim follows from the unforgeability of the signature scheme, and by the fact that Sim cannot ask to the oracle more openings that R_p^* would ask in the real world attack. The intuition is the following. By the proof given above, if there exist q, q' such that $\Pi_1(i)_\mathsf{S} \neq \Pi_1(i)'_\mathsf{S}$ then it must be that $s_i \leftarrow F_k(\Pi_1(i)_\mathsf{S}) \neq s'_i \leftarrow F_k(\Pi_1(i)'_\mathsf{S})$. Hence, since s_i, s'_i differ in at least one bit, due to the way R_p^* chooses I_i, we have that I_i chosen after query q is distinct from I'_i for the query q' in at least one index. Note also that, by construction $|I_i| = |I'_i| = n$, thus there exist at least one index j_ℓ that is in I'_i but is not in I_i. If both queries q, q' were not aborting, then Sim was able to provide signatures for both sets of indexes, thus Sim provided signature for at least $n+1$ indexes. Recall that, Sim cannot ask the oracle with more indexes that R_p^* would ask in the real world experiment. In particular, for each block of parallel executions, Sim must asks exactly n signatures, according to R_p^* strategy (indeed the strategy of R_p^* is such that R_0^* always asks exactly n opening at each block and rn openings in total, while R_1^* asks for a total of $rn+1$ openings). Thus if R_p^* did not abort in both q, q' this means that Sim was able to provide to R_p^* a total of at least $n+1$ valid signatures, still asking only n signatures to $\mathcal{O}$, thus Sim has generated a forgery.

Formally the reduction works as follows. We construct a signature-forgery algorithm M that given vk simulates R_p^* and the oracle $\mathcal{O}$ to Sim. M perfectly emulates R_p^* and answers the queries that Sim makes to the oracle by forwarding them to its signing oracle. Note that the emulation is perfect. Now assume that with non-negligible probability there exist $q, q' \in Q_i$ with different messages $\Pi_1(i)_\mathsf{S}$, $\Pi_1(i)'_\mathsf{S}$ yielding

[9] That is, we consider all of the oracle queries made by Sim to R_p^* throughout the simulation and take the subset of those queries which include all the messages belonging to the executions of $\Pi_{2(i-1)n+i+1}$ to Π_{2in+i}. By the scheduling described in Fig. 1, such a query defines the ith message of Π_1 that is sent by R^*, (i.e., $\Pi_1(i)_\mathsf{R}$.)

to different set of indexes I_i, I'_i. Since R^* does not abort it must have seen $(\sigma_{j_1}, \ldots, \sigma_{j_n})$ with $j_\ell \in I_i$ and $(\sigma_{j'_1}, \ldots, \sigma_{j'_n})$ with $j'_\ell \in I'_i$. Since there exist at least an index j^* that is not in both sets, R^* gets at least $n+1$ signatures. Recall that for each block Sim is allowed to ask only for n signatures. Thus there exists at least a signature $\sigma^*_{j^*}$ that was not provided by the oracle, hence M outputs $(j^*, \sigma^*_{j^*})$ thus contradicting the security of the signature scheme. □

In Claim 1 we proved that against adversaries $\{\mathsf{R}^*_p\}_{p\in\{0,1\}}$ any Sim cannot make effective rewinds for the execution Π_1, and thus if Sim exists then it is able to equivocate the first execution without rewinding R^*_p. Since R^*_p in first execution basically follows the procedure of the honest receiver we want to argue that the same strategy used by Sim to equivocate can be adopted by a malicious sender that wants to equivocate but cannot rewind the honest party.

The idea is to construct a malicious sender S^* that runs Sim as subroutine and simulates the same attack of R^*_1 except that for the execution Π_1 it forwards the messages to the honest R such that, when Sim asks the oracle for the opening of session 1, S^* replies with the string that it wants to open to R. In particular, it sends the first string and obtains the opening by Sim, then it rewinds Sim and it sends a distinct string, for which it will obtain another valid opening (this is true due to the existence of Sim). Note however that in this reduction we are assuming that Sim asks for the queries only after the execution of Π_1 is completed. Indeed, if Sim queries the oracle for the opening of Π_1 any time before the commitment phase of Π_1 is completed we cannot use Sim as a sub-routine to break binding since Sim could change the transcript of the commitment phase according to the response received from $\mathcal{O}$, and thus in turn if S^* rewinds Sim and changes the string to open, it could obtain a distinct commitment transcript, therefore not violating the binding property. Therefore, before proceeding with the construction of the malicious adversary we need to prove another claim on Sim.

Claim 2. *In the execution Π_1, except with negligible probability, Sim queries the oracle $\mathcal{O}$ only after receiving the query by $\{\mathsf{R}^*_p\}_{p\in\{0,1\}}$.*

Proof. Since Sim is black-box, then it must work for all R^*. In particular Sim must successfully simulate adversaries R^*_0 and R^*_1. Adversary R^*_0 does not query for the opening the first execution Π_1 (i.e., the probability of opening is $p = 0$), therefore Sim is not required to provide an opening and can simulate Π_1 without interacting with $\mathcal{O}$. Adversary R^*_1 always asks to see the opening of Π_1. In this case, Sim could ask for the opening of session 1 to $\mathcal{O}$ at the beginning of the simulation and thus run the first execution as an honest sender (i.e., with no need of equivocation). The output of Sim would be indistinguishable from the output of S. However, the definition of black-box simulation requires that the same simulation strategy should work for all adversarial strategies. Thus the decision on whether to ask for the opening to $\mathcal{O}$ can be made only after the R^*_p has sent the request of opening. Indeed, if Sim asks the oracle any time *before* it has received the query from R^*_0, then the set of indexes I asked by Sim in the ideal execution is clearly distinguishable since in the real world execution the set I requested by R^*_0 does not contain index of Π_1.

The last case to consider is the case that, Sim does not query the oracle for session 1. (Recall that if Sim does not ask the oracle then the malicious sender in the reduction would not be able to exploit Sim to open two distinct strings). Due to the unforgeability of the signature scheme, this case happens with negligible probability. The reduction works as in Claim 1.

□

Claim 3. *In the execution Π_1, when dealing with adversary R^*_1, Sim provides a valid opening with all but negligible probability.*

Proof. In Π_1 the adversary R_1^* is playing as the honest receiver, thus always gets a valid opening when interacting with S. By the assumption that Π is SOA-secure under concurrent composition it must holds that also Sim provides a valid opening with all but negligible probability. □

Now we are ready to show the formal construction of the adversary for binding S^* that uses Sim as a sub-routine.

Malicious Sender. S^* playing the binding game, externally interacts with an honest receiver R while internally interacts with Sim. S^* generates a pair $(vk, sk) \xleftarrow{\$} \mathsf{Gen}(1^n)$, thus defining the set $\mathcal{B}_{sk}$ and gives vk to Sim and R. S^* emulates R_1^* and the oracle $\mathcal{O}$ to Sim for all executions $\Pi_2, \ldots, \Pi_k$. This emulation can be perfectly carried out since it has all the secrets. S^* plays the execution Π_1 by forwarding the messages to the external R. That is, let q be an oracle query from Sim to R^* such that R_1^* response is the i-th message of execution Π_1. Then, if R_1^* would abort receiving q or any prefix of q, S^* emulates it by aborting. Otherwise, if q is such that R_1^* does not abort but rather replies with the i-th message of Π_1 then S^* extracts the message of the simulator $\Pi_1(i)_{\mathsf{S}}$ from q, and then, if R has already sent the response $\Pi_1(i)_{\mathsf{R}}$ according to the extracted message, then R^* replays this same message to Sim. If R has already sent the answer $\Pi_1(i)_{\mathsf{R}}$ according to another message $\Pi_1'(i-1)_{\mathsf{S}}$ then S^* halts. We call this event as *failure*. Finally, if R did not replay to $\Pi_1(i)_{\mathsf{S}}$ yet, then S^* forwards it to R and stores the pair $\Pi_1(i)_{\mathsf{S}}$, $\Pi_1(i)_{\mathsf{R}}$. Finally, when Sim makes queries to the oracle $\mathcal{O}$ for a set of indexes I_i, the sender responds via the signatures $\sigma_j = \mathsf{Sign}(j, sk)$ for all $j \in I$.

When Sim queries the oracle for the opening of the execution Π_1, if the commitment phase of the execution with the honest receiver is not completed yet, then S^* aborts and halts. We call this event too as *failure*. Else, S^* provides a string $\alpha_0 \leftarrow \mathsf{Sign}(1, sk, r_0)$ and obtains the opening α_0, w_0 from Sim. Then S^* rewinds Sim up to the point in which it asks the opening for session Π_1 and provides a distinct string $\alpha_1 \leftarrow \mathsf{Sign}(1, sk, r_1)$ to obtain the opening α_1, w_1 from Sim. If Sim never asks the oracle for session Π_1 then S^* aborts and halts. Again we call this event as *failure*.

Finally S^* obtains two openings for the protocol executions Π_1 played with the honest R.

First note that if S^* does not abort then S^* perfectly emulates the attack of R_1^*. Indeed it plays all but the first execution following R_1^* procedure, while for the message of the first execution it forwards the messages receiver by the honest receiver. Again, this is consistent with the strategy of R_1^* that in the first execution is honest.

By Claim 1 we have that the event *failure* happens only with negligible probability. By Claim 2 we have that, except with negligible probability, Sim makes the oracle query for the opening of execution Π_1 only after the commitment phase have been completed. By Claim 3 we have that with all but negligible probability S^* gets two valid openings when interacting with Sim. Thus with high probability S^* provides two valid openings for the commitment phase transcript obtained by playing with R.

Corollary 1. *There exists no bit commitment protocol that is SOA-secure under strong concurrent composition.*

Proof. Toward a contradiction assume that there exists a bit commitment scheme $\Gamma = (\mathsf{S}_\Gamma, \mathsf{R}_\Gamma)$ that is SOA-secure under concurrent composition. Then it is possible to construct a string commitment scheme $\Pi = (\mathsf{S}, \mathsf{R})$ as follows. Let $m = m_0 | \ldots | m_n$ be the n-bit message that S wants to commit to. The commitment phase consists of n-parallel executions of the commitments phase of Γ, (i.e., $\langle \mathsf{S}_\Gamma^i(\mathtt{com}, m_i), \mathsf{R}_\Gamma^i(\mathtt{recv}) \rangle$ for $i = 1, \ldots, n$). The commitment phase of Π is over when all commitments phases of Γ are successfully completed. The opening phase consists of the parallel execution of the n decommitment phases of Γ

(i.e., $\langle \mathsf{S}^i_\Gamma(\mathtt{open}), \mathsf{R}^i_\Gamma(\mathtt{open})\rangle$ for $i = 1, \ldots, n$). Basically, Π consists only of executions of Γ, and since Γ is SOA-secure under concurrent composition, so is Π. □

□