Revisiting Lower and Upper Bounds for Selective Decommitments

Rafail Ostrovsky [*] Vanishree Rao [†] Alessandra Scafuro [‡] Ivan Visconti [§]

Abstract

In [DNRS99, DNRS03], Dwork et al. opened the fundamental question of 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 problem positively by presenting a scheme which is based on non-black-box use of a one-way permutation and which has super-constant number of rounds. The achieved solution however opened other challenging questions on improvements of round complexity and on possibility of obtaining fully black-box schemes where access to an underlying primitive and to an adversary are black-box only. Recently, in TCC 2011, Xiao ([Xia11a]) investigated on how to achieve (nearly) optimal SOA-secure commitment schemes where optimality is in the sense of 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, the various results in [Xia11a] focus only on either parallel or concurrent SOA.

In this work we first point out various issues in the claims of [Xia11a] that actually re-open several of the questions left open in [BHY09, Hof11]. Then, we provide new lower bounds and concrete constructions that produce a very different state-of-the-art compared to the one given in [Xia11a].

More specifically, denoting by (x, y) the round complexity of a scheme that requires x rounds in the commitment phase and y rounds in the decommitment phase, and by considering only (like in [Xia11a]) the setting of black-box simulation for SOA-security, we show that:

1. There is an issue in the result of [Xia11a] on the existence of $(3, 1)$-round schemes for parallel SOA; in fact, we are able to contradict their impossibility result by presenting a $(3, 1)$-round scheme based on black-box use of trapdoor commitments. Moreover, we can instantiate such a scheme with a non-black-box use of a one-way function, thus producing a $(3, 1)$-round scheme based on any one-way function that improves the result of [BHY09, Hof11] from logarithmic round complexity to 3 (optimal), also under optimal complexity assumptions. We also show a $(3, 3)$-round scheme based on black-box use of any one-way permutation.
2. There is an issue in the proof of security for parallel composition of the $(4, 1)$-round scheme given in [Xia11a]; thus such scheme may not be secure. 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 [Xia11a]. This issue emerges under the case where the simulator cannot itself efficiently sample from

[*]Departments of Computer Science and Department of Mathematics, UCLA, 3732D Boelter Hall, Los Angeles CA 90095-1596, U.S. Email: rafail@cs.ucla.edu

[†]Department of Computer Science, UCLA, 3771 Boelter Hall, Los Angeles CA 90095-1596, U.S. Email: vanishri@cs.ucla.edu

[‡]Dipartimento di Informatica, University of Salerno, Italy. Email. scafuro@dia.unisa.it

[§]Dipartimento di Informatica, University of Salerno, Italy. Email. visconti@dia.unisa.it

the distribution of committed messages. In fact, we contradict the claimed security of this scheme by showing that there can not exist such a scheme, regardless of its round complexity and of the (black-box or non-black-box) use of cryptographic primitives.

All our schemes are secure for parallel SOA composition and also secure for concurrent SOA composition under the restriction that all commitment phases are played before any decommitment 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. While their binding property guarantees that a committed message can not be opened to two distinct messages, their hiding property guarantees that before the decommitment 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 will not be able to open a committed message in two ways, and even a concurrent malicious receiver will not be able to detect any relevant information about committed messages as long as only commitment phases have been played so far.

In [DNRS99], Dwork et al. pointed out a more subtle definition of security for hiding 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, thus opening the fundamental question of existence of commitment schemes that are secure against selective opening attacks (SOA, for short). We stress that the question is particularly important since commitments are often used in larger protocols, where often 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]) who presented a SOA-secure 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 on round optimality and (black-box) use of the underlying cryptographic primitives. The notion of black-box use of cryptographic primitives has attracted much attention and significant progress has been achieved in recent years [CDSMW09, PW09, Wee10].

In TCC 2011 [Xia11a], 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 and all his results concern a simulation-based definition. The subsequent work [Xia12b], shows a black-box construction of 4-round statistically-binding SOA commitment secure only for parallel composition. As we shall see later, our $(3, 1)$-round and $(4, 1)$-round schemes are only computationally binding, but in the stronger setting of concurrent-with-barrier composition.

In [BDWY12] Bellare et al., showed that the existence of CRHFs implies that non-interactive SOA-secure commitments are impossible. This holds, even if the simulator is non black-box and knows the distribution of the message space. An implication of such results is that, standard security

[1]In [GM06] some forms of zero-knowledge sets were proposed, and their strongest definition required SOA-secure commitments.

does not imply SOA-security. Previous results [BHY09, Hof11] only showed the impossibility for the case of black-box reductions. [BDWY12] also studied the SOA-security notions for public-key encryption schemes. In particular, they showed that for public-key encryption schemes, IND-CPA security does not imply simulation-based SOA-security.

Continuing on this line of research, recently, [BHK12] almost completed the picture of the relationship between different notions of SOA-security of public-key encryption schemes. In particular, they showed that indistinguishability-based SOA-security and simulation-based SOA-security do not imply each other.

1.1 Our Contribution

In this work we focus on black-box simulation-based SOA-secure commitment schemes. Firstly we point out various issues in the claims of [Xia11a]. These issues essentially re-open all the major open questions that were supposed to be answered in [Xia11a]. We next show how to solve (in many cases in a nearly optimal way) all of them. Interestingly, our final claims render quite a different state-of-the-art from (and in some cases also in contrast to) the state-of-the-art set by the claims of [Xia11a].

In detail, by specifying as (x, y) the round complexity of a commitment scheme when the commitment phase takes x rounds and the decommitment phase takes y rounds, and by considering only the definition of BB simulation for SOA security, we revisit [Xia11a] claims and re-open some challenging open questions as follows:

1. There is an issue in the impossibility result of [Xia11a] on the existence of $(3, 1)$-round schemes secure under parallel composition. This re-opens the question of the achievability of $(3, 1)$-round SOA-secure schemes.
2. There are issues in the proof of security of the $(4, 1)$-round scheme of [Xia11a] for parallel composition, and thus this scheme may not be secure. This re-opens the question of obtaining a constant-round scheme that is provably SOA-secure.
3. There is an issue in the proof of security of the construction of [Xia11a] that is claimed to be SOA-secure under concurrent composition in the strong sense; i.e., composition can be fully concurrent, allowing even the commitment and decommitment phases to be interleaved together. This issue arises for the distributions where the simulator by itself cannot efficiently sample from the distribution of messages committed to by the honest sender (but needs to query an external party for it).[2] An example of such a distribution can be signatures on some public verification key (the simulator will not be able to efficiently sample from this distribution as it does not have the corresponding secret key). This issue in [Xia11a] re-opens the possibility of achieving schemes that are SOA-secure under fully 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 paper we solve the above open problems (still sticking to the notion of black-box simulation as formalized in [Xia11a]) 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 trapdoor commitment (wTCom, for short)[3], and a $(5, 1)$-round scheme

[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. We call it "weak" because any TCom is also a wTCom.

based on BB use of any OWP.

2. We show that when the simulator does not know the distribution of the messages committed to by the honest sender, there exists no scheme that achieves fully 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 [Xia11a].
3. As a corollary of our $(3,1)$-round scheme based on BB use of any TCom, there exists a $(3,1)$-round scheme based on NBB use of any one-way function (OWF). Moreover, since we show that there does not exist a $(2,1)$-round scheme regardless of the use of the underlying cryptographic primitive, our $(3,1)$-round scheme is essentially round-optimal. This improves the round complexity in [BHY09] from logarithmic in the security parameter to only 3 rounds and using minimal complexity-theoretic assumptions.

Notice that both our $(3,1)$-round protocols - the one based on BB use of TCom and the other based on NBB use of OWFs - contradict the impossibility given in [Xia11a]. Moreover, note that our $(3,1)$-round protocol based on BB use of TCom (as well as our $(4,1)$-round protocol based on BB use of wTCom) does not require **NP** reductions, in contrast to our $(3,1)$-round protocol that is based on NBB use of OWFs.

All our constructions are in fact secure under concurrent composition as long as all commitment phases are played before the beginning of any decommitment phase; we shall refer to this form of composition as "concurrency-with-barrier" and it obviously implies parallel composition too. Furthermore, our simulators do not need to know the distribution of the messages committed to by the honest sender. In light of our impossibility for the fully 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 [Xia11a] (see the paragraph "Stronger definitions of hiding" in [Xia11a]) and in [Hof11] (see Item 3 in paragraph "Discussion of the Definitional Choices" in [Hof11]).

As an additional application, we also show that our $(3,1)$-round schemes can be used to obtain non-interactive (concurrent) zero knowledge [DNS98] with 3 rounds of pre-processing. This improves upon [CO99] where (at least) 3 rounds of interactions are needed both in the pre-processing phase and in the proof phase. Moreover, the simulator of [CO99] works only with non-aborting verifiers, while our simulator does not have this limitation.

We further compare our results with the previous state-of-the-art in the table below. Here, for instance, $(3,1)$ PAR in the "Impossible" row under [Xia11a] means that [Xia11a] claims impossibility for a $(3,1)$-round scheme that is SOA-secure under parallel composition; NBB $(\log n, 1)$ CwB OWP in the "Achieved" 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 [Xia11a]), t-SH refers to a $(t,1)$-round statistically-hiding commitment scheme. In the last column, we list the results of [Xia11a] that we contradict.

	[BHY09, Hof11]	[Xia11a]	This Paper	This Paper on [Xia11a]
Impossible	BB $(1,1)$	$(3,1)$ PAR $(o(\log n/\log\log n),1)$ CC	(any, any) 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 $(3,1)$ TCom; NBB $(3,1)$ OWF BB $(4,1)$ wTCom ; BB $(3,3)$ OWP BB $(5,1)$ OWP (all CwB)	~~BB $(4,1)$ PAR OWP~~ ~~BB $(\omega(t\log n),1)$ CC t-SH~~ – *unknown distribution* –

We stress that all issues we point out in this submission about the claims of [Xia11a] have been later confirmed by Xiao in his last revision of his work [Xia12a].

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 (negl., for short) 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 $\{1, \ldots, k\}$; $\mathtt{poly}(n)$ 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 $x \overset{\$}{\leftarrow} \mathsf{A}$ 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 $v \overset{\$}{\leftarrow} \mathsf{A}()$ when the algorithm A is randomized. Finally, let P_1 and P_2 be two parties running a protocol that uses protocol $\langle A, B \rangle$ as a sub-routine. 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.

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). *A commitment scheme is a tuple of PPT algorithms* $\mathsf{Com} = (\mathsf{Gen}, \mathsf{S}, \mathsf{R})$ *implementing the following two-phase functionality.* Gen *takes as input a random n-bit string r and outputs the public parameters pk. Given to* S *an input $b \in \{0, 1\}$, in the first phase (*`commitment phase`*)* S *interacts with* R *to commit to the bit b; we denote this interaction as* $\langle \mathsf{S}(pk, \mathtt{com}, b), \mathsf{R}(\mathtt{recv}) \rangle$. *In the second phase (*`opening phase`*)* S *interacts with* R *to reveal the bit b, we denote this interaction as* $\langle \mathsf{S}(\mathtt{open}), \mathsf{R}(\mathtt{open}) \rangle$ *and* R *finally outputs a bit b' or $\perp$. Consider the following two experiments:*

Experiment $\mathbf{Exp}^{\text{binding}}_{\mathsf{Com},\mathsf{S}^*}(n)$:	**Experiment** $\mathbf{Exp}^{\text{hiding-}b}_{\mathsf{Com},\mathsf{R}^*}(n)$:
R *runs* $(pk) \leftarrow \mathsf{Gen}(r)$ *and sends* pk *to* S^*;	$pk^* \leftarrow \mathsf{R}^*(1^n)$;
$\langle \mathsf{S}^*(pk, \mathtt{com}, b), \mathsf{R}(\mathtt{recv}) \rangle$;	$(\cdot, b') \overset{\$}{\leftarrow} \langle \mathsf{S}(pk^*, \mathtt{com}, b), \mathsf{R}^*(\mathtt{recv}) \rangle$;
$(\cdot, b_0) \overset{\$}{\leftarrow} \langle \mathsf{S}^*(\mathtt{open}, 0), \mathsf{R}(\mathtt{open}) \rangle$;	*output* b'.
rewind S^* *and* R *back after the second step;*	
$(\cdot, b_1) \overset{\$}{\leftarrow} \langle \mathsf{S}^*(\mathtt{open}, 1), \mathsf{R}(\mathtt{open}) \rangle$;	
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:*

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

Binding. *For any PPT malicious sender* S^* *there exists a negl. function ϵ such that:* $\Pr[\mathbf{Exp}^{\text{binding}}_{\mathsf{Com},\mathsf{S}^*} \to 1] \leq \epsilon(n)$.

The above probabilities are taken over the choice of the randomness r for the algorithm Gen *and the random coins of the parties. A commitment scheme is statistically hiding (resp., binding) if 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 used by R to generate the parameters for the commitment. Such a definition is convenient when one aims to use commitment schemes as sub-protocols in a black-box way. However, for better readability, when we construct or use as sub-protocol a commitment scheme that does not use public parameters we refer to it only as $\mathsf{Com} = (\mathsf{S}, \mathsf{R})$ omitting the algorithm Gen. In particular we shall denote by $\mathsf{Com}_{\mathsf{NI}} = (\mathsf{S}_{\mathsf{NI}}, \mathsf{R}_{\mathsf{NI}})$ a non-interactive commitment scheme. Such commitment schemes exist based on any OWP [GL89].

Remark 1 (Hiding definition)**.** *We stress that, the definition of hiding formalized through the hiding experiment* $\mathbf{Exp}^{\text{hiding-}b}_{\mathsf{Com},\mathsf{R}^*}(n)$*, guarantees that indistinguishability holds even when the public parameter* pk^* *is adversarially chosen (by* R^**). As a consequence, in our proofs we deal with possibly bad parameters* pk^* *by relying on the fact that hiding is guaranteed even for such possibly bad* pk^*s *(i.e., as can be seen in the proofs, no additional procedure for verifying the correctness of the parameters will be required).*

Remark 2 (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-negl. probability. Since we consider also interactive decommitments, we formalize this notion as a game following the definition given in [BHY09, Hof11]. That is,* S^* *is run twice in the decommitment phase, but with an additional input necessary to obtain two distinct openings (indeed* S^* *is run twice with the same randomness), i.e.,* S^* *is invoked as* $\mathsf{S}^*(\mathsf{open}, b)$.

For the definitions of trapdoor commitments we borrow some notation from [MY04, Rey01].

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* TCGen*, on input a random n-bit string r, 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* $(\mathsf{S}, \mathsf{TCFakeDec})$ *are such that:*

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

$$\mathbf{Adv}^{\mathsf{trapdoor}}_{\mathsf{TC},\mathsf{R}^*} = \Pr[\mathbf{Exp}^{\mathsf{Trap}}_{\mathsf{TC}}(n) \to 1] - \Pr[\mathbf{Exp}^{\mathsf{Com}}_{\mathsf{TC}}(n) \to 1] \le \epsilon(n).$$

The probability is taken over the choice of r for the algorithm TCGen *and the random coins of the players.*

Experiment $\mathbf{Exp}^{\mathsf{Com}}_{\mathsf{TC}}(n)$ *:*	*Experiment* $\mathbf{Exp}^{\mathsf{Trap}}_{\mathsf{TC}}(n)$*:*
R^* *chooses a bit b;*	R^* *chooses a bit b;*
$\langle \mathsf{S}(pk, \mathtt{com}, b), \mathsf{R}^*(pk, sk, b, \mathtt{recv})\rangle$*;*	$(\xi, \cdot) \leftarrow \langle \mathsf{S}(pk, \mathtt{com}, b^\star), \mathsf{R}^*(pk, sk, b, \mathtt{recv})\rangle$*;*
$(\cdot, b') \overset{\$}{\leftarrow} \langle \mathsf{S}(\mathsf{open}), \mathsf{R}^*(\mathsf{open})\rangle$*;*	$(\cdot, b') \overset{\$}{\leftarrow} \langle \mathsf{TCFakeDec}(sk, \mathsf{open}, b, \xi), \mathsf{R}^*(\mathsf{open})\rangle$*;*
output b';	*output b';*

In the experiment $\mathbf{Exp}_{\mathsf{TC}}^{\mathsf{Trap}}(n)$ S runs the procedure of the honest sender on input $b^\star$. The variable ξ contains the randomness used by S to compute the commitment phase and it is used by $\mathsf{TCFakeDec}$ to compute the decommitment. The knowledge of the trapdoor is required only in decommitment phase. In the trapdoor commitment of Pedersen [Ped92], the trapdoor property holds for any $b^\star$, namely one can use the honest sender procedure to commit an arbitrary bit $b^\star$ and use the trapdoor to decommit to any $b \neq b^\star$. Instead, in the trapdoor commitment proposed by Feige and Shamir [FS89], as we show next, the trapdoor property holds only if the honest procedure was used to commit to bit $b^\star = 0$. In both commitment schemes the trapdoor is used only in the decommitment phase.

We stress that, according to the standard definition, while the hiding property must hold for all pk possibly maliciously generated by R^*, the trapdoor property must hold only for the pairs (pk, sk) honestly generated. In some definitions [Rey01] it is required that hiding holds for all the malicious keys that pass the test of an additional verification algorithm TCVer, however, w.l.o.g. one can assume that the commitment procedure runs the verification algorithm as a first step. Note that implementations of a trapdoor commitment enjoying all the properties above do exist, one example is Pedersen's Trapdoor Commitment [Ped92], in which once the public parameter pk are given, the commitment procedure is non-interactive. We mention below a construction based on any OWF.

It is possible to consider a weaker[4] definition of trapdoor commitment (see Appendix A.1.1) in which, in order to be able to later equivocate, the trapdoor is needed already in the commitment phase. The trapdoor commitment proposed in [PW09] uses such a definition.

Trapdoor Commitment Scheme based on Non-black-box use of a OWF. In [FS89] Feige and Shamir presented a construction of trapdoor commitments based on NBB access to OWFs. The commitment procedure consists of running the simulator of Blum's protocol [Blu86] for Graph Hamiltonicity (HC) where the challenge is the bit to commit to. For completeness we recall the construction. Let $f : \{0,1\}^n \to \{0,1\}^*$ be a OWF.

- $(G, C) \leftarrow \mathsf{TCGen}(r)$: pick a random x and compute $y \leftarrow f(x)$. From y obtain a hard instance $G \in \mathsf{HC}$ and let C be one of the Hamiltonian cycles of G. This transformation requires non-black-box access to f. Set the public key $pk = G$ and the trapdoor $sk = C$.

- $\mathsf{S}(G, \mathtt{com}, b)$: if $b = 0$, pick a random permutation π and commit to $\pi(G)$. If $b = 1$, commit to a random n-vertex cycle H. Both commitments are performed using Naor Commitment [Nao91] that is based on BB access to OWFs.

- $\langle \mathsf{S}(\mathtt{open}, b), \mathsf{R}(\mathtt{open}) \rangle$: S sends b and the opening according to b, i.e., if $b = 0$ it sends π and opens all commitments, if $b = 1$ it opens the cycle H. R checks whether the openings are correct according to challenge b and the procedure of the verifier of Blum's protocol.

- $\xi \leftarrow \mathsf{S}(G, \mathtt{com}, b^\star)$: S runs as $\mathsf{S}(G, \mathtt{com}, 0)$. The variable ξ contains the randomness used to run S.

- $\langle \mathsf{TCFakeDec}(C, \mathtt{open}, b, \xi), \mathsf{R}(\mathtt{open}) \rangle$: to open to 0 send π and open all the commitments, to open to 1 open the cycle C in $\pi(G)$. The opening information is taken from ξ.

[4]The definition is weaker in the sense that it is implied by the previous definition, but could be a strictly weaker primitive achievable under better assumptions and with better efficiency.

Hiding comes from the hiding of Naor commitments, and binding from the hardness of the OWF. A commitment can be equivocated only if it was computed following the procedure to commit 0. Thus, the above protocol satisfies the trapdoor property for $b^\star = 0$.

Definition 3 (Hiding in the presence of Selective Opening Attacks (slight variation of [BHY09, Hof11])). *Let $k = \texttt{poly}(n)$, 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 negl. function ϵ such that:*

$$\mathbf{Adv}^{\mathsf{soa}}_{\mathsf{Com}} = \left|\mathsf{Pr}[\mathbf{Exp}^{\mathrm{real}}_{\mathsf{Com},\mathsf{S},\mathsf{R}^*}(n) \to 1] - \mathsf{Pr}[\mathbf{Exp}^{\mathrm{ideal}}_{\mathsf{Com},\mathsf{Sim},\mathsf{R}^*}(n) \to 1]\right| \le \epsilon(n).$$

The probability is taken over the choice of the random coins of the parties.

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

We denote by $(\cdot, ext) \xleftarrow{\$} \langle \mathsf{S}_i(\cdot), \mathsf{R}^*(\cdot)\rangle$ the output of R^* after having interacted concurrently with k instances of S each one denoted by S_i. In the paper an instance of the protocol is called session. A malicious receiver R^* can run many sessions in concurrency with the following limitation. R^* runs commitment phases concurrently for polynomially many sessions, but it can initiate the first decommitment phase only after the commitment phases of all the sessions have been completed (and therefore after 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 [Xia11a], 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 3. *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.*

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

Concurrency-with-barrier composition vs. Parallel and Concurrent Composition. In [Xia11a] 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: R^* 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, depending on the transcript of the commitment *and* decommitment of other sessions. This definition is pretty general, but unfortunately, as we show in this paper, achieving this result is impossible (under the assumption that the simulator does not know the distribution of the messages committed to by the honest sender); this is in contrast to [Xia11a] where it is claimed that this definition is achievable. The concurrent-with-barrier composition that we adopted (following [Hof11]) implies security under parallel composition while due to the barrier between commitment and decommitment phase, it is weaker than the fully concurrent definition of [Xia11a].

Decommitment Phase can be interactive. Following [Hof11] our definition is more general than the one of [Xia11a] since it allows also the decommitment phase to be interactive.

Honest Party Behaviour. We follow [Xia11a] 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 of pairs of shares of a message but finally only one share per pair is required to be opened in the commitment phase.

Concerning the honest sender, we assume that R^* interacts with k independent senders, that are oblivious to each other, and play with input $\mathbf{b}[j]$, while [Xia11a] considers a single sender S^k who gets as input the complete k-bit string and plays k independent sessions with R^*. This variation is cosmetic only.

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, 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 the (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 [Xia11a] 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 honest receiver to choose the commitments to be opened. However, our definition is weaker than the concurrent definition of [Xia11a] that however we show to be impossible to achieve (when the distribution of the messages committed by S is unknown to Sim).

3 Upper Bounds

3.1 SOA-secure Commitment Scheme based on BB use of Trapdoor Commitments

We present a construction of a round-optimal SOA-secure commitment scheme based on BB use of trapdoor commitments. In particular we show that if 2-round (where the first round only serves for the receiver to send the public parameters) trapdoor commitment schemes exist[5] then a 3-round commitment scheme that is secure under selective opening attack exists. Under the assumption that *weak* trapdoor commitment schemes exist, in Appendix C.1 we present a 4-round construction.

The main idea behind both protocols is to require the sender to commit to its private input using a trapdoor commitment scheme and to make the trapdoor information extractable to the black-box simulator. This allows the simulator to cheat in the opening phase without changing the transcript of the commitment phase. Obviously, the parameters of the trapdoor commitment are generated by the receiver (if this was not the case then a malicious sender can cheat in decommitment phase using the trapdoor), and are made extractable through cut-and-choose techniques. In more details, the protocol goes as follows. 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 allow extraction of the trapdoor, we require that R provides $2n$ public parameters and S asks to "reveal" the trapdoor of a random n-size subset of them. S will use the remaining n parameters (for which the trapdoors are still hidden) to commit to n shares of its secret input. In this way the equivocation requires the knowledge of one trapdoor only among the set of the remaining n keys that were not revealed. Thus, the simulator first commits to n random bits, then through rewinding threads it will extract from R at least one trapdoor of the remaining unrevealed keys. One trapdoor is sufficient to equivocate one of the shares already committed, and in turn, to decommit to any bit.

In Protocol 1, that uses trapdoor commitments, the simulator can commit without knowing the trapdoor, thus the commitment of the shares can be merged with the cut-and-choose phase, therefore yielding a 3-rounds commitment phase. In the protocol that uses *weak* trapdoor commitments (see Protocol 4 in Appendix C.1), the simulator needs to extract the trapdoor before committing (since it will be able to equivocate only commitments that are computed using the trapdoor), therefore the commitment must be postponed after the completion of the cut-and-choose phase. This adds one more round to the commitment phase.

Finally, binding follows straight-forwardly from the binding of the trapdoor commitment scheme used as sub-protocol.

(3,1)-round SOA-secure Scheme based on BB use of Trapdoor Commitments. Let us denote as $\mathsf{TC} = (\mathsf{TCGen}, \mathsf{S}_{\mathsf{TC}}, \mathsf{R}_{\mathsf{TC}}, \mathsf{S}, \mathsf{TCFakeDec})$ a trapdoor commitment scheme. 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 commitment phase of TC is non-interactive (given the public parameters in input) then the following construction yields a (3,1)-round commitment scheme. We denote by $\langle {\mathsf{S}_{\mathsf{TC}}}_i^{\bar{d}_i}, {\mathsf{R}_{\mathsf{TC}}}_i^{\bar{d}_i} \rangle$ the i-th invocation of sub-protocol TC run with public key $\mathsf{pk}^{\bar{d}_i}$. Here d_i denotes the i^{th} challenge for the cut-and-choose, i.e., $\mathsf{S}_{\mathsf{soa}}$ computes the trapdoor associated to the key pk^{d_i}, while it commits to the i^{th} share of the input using key $\mathsf{pk}^{\bar{d}_i}$ (for which the trapdoor will not be revealed).

Protocol 1. *[(3,1)-Round SOA-Secure Commitment Scheme] [$\mathsf{SOACom} = (\mathsf{S}_{\mathsf{soa}}, \mathsf{R}_{\mathsf{soa}})$]*

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* $(\mathsf{pk}_i^0, \mathsf{pk}_i^1)$ *to* $\mathsf{S}_{\mathsf{soa}}$;

[5][Ped92] is an example of a trapdoor commitment scheme where the public parameters pk are generated by the receiver and sent to the sender in the first round. Given pk, the commitment procedure is non-interactive.

$\mathsf{S_{soa}}$: *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 $i = 1, \ldots, n$: $b_i \xleftarrow{\$} \{0,1\}$, such that $b = (\bigoplus_{i=1}^n b_i)$;*
2. *for $i = 1, \ldots, n$ do in parallel:*
    - *send $d_i \xleftarrow{\$} \{0,1\}$ to $\mathsf{R_{soa}}$;*
    - *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 $\mathsf{R_{soa}}$;*

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

$\mathsf{S_{soa}}$: *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 $\mathsf{pk}_i'^{d_i} \neq \mathsf{pk}_i^{d_i}$ then abort.*

Decommitment phase.

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

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

Theorem 1 (Protocol 1 is secure under selective opening attacks). *If $\mathsf{TC} = (\mathsf{TCGen}, \mathsf{S_{TC}}, \mathsf{R_{TC}}, \mathsf{TCFakeDec})$ is a trapdoor commitment scheme, then Protocol 1 is a commitment scheme secure against selective opening attacks.*

The proof appears in Appendix D.1. The case of a $(4,1)$-round construction is very similar, and only deviates in the fact that the commitments of the shares are sent in the 4th round instead of the 2nd round. Further details appear in Appendix C.1.

(3,1)-round SOA-secure Scheme based on NBB use of OWFs. We observe that, by instantiating Protocol 1 with the Feige-Shamir trapdoor commitment scheme described in Section 2.1, one can obtain a (3,1) SOA-secure scheme with non-black-box access to OWFs.

3.2 (3,3)-round SOA-secure Scheme based on BB use of OWPs

In this section we present a $(3,3)$-round SOA-secure commitment scheme based on BB use of any OWP. As a main ingredient, we use an extractable commitment scheme ExtCom. As shown in Protocol 3 (Appendix A.4), ExtCom can be constructed with a BB use of statistically-binding commitments that in turn can be constructed with a BB use of OWPs.

The idea behind the protocol 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 reveals its random string. 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). In the decommitment phase, at the completion of the coin-flipping protocol, the sender opens only one of the commitments in each pair according to the outcome of the coin-flipping.

To allow for simulation (while arguing hiding), the commitment of the receiver in the coin-flipping protocol is implemented via extractable commitment scheme, so that the simulator can extract the receiver's string in the commitment phase itself. Furthermore, we require that the sender sends its random string for the coin-flipping only in the decommitment phase; by the beginning of the decommitment phase, the simulator will have received the bit b to open to, and this gives the simulator an opportunity to craft its random string to point to the commitments of b. To see why, first note that if the simulator 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, with the knowledge of the receiver's random string and the bit b, it can craft its own random string such that the xor with the string of R points to the commitments of b. Since the receiver commits via an extractable commitment scheme, the simulator is able to extract the receiver's random string 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 (resp., decommitments) always *after* it receives commitments (resp., 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.

We prove binding of SOACom by reducing it to the statistical binding property of ExtCom (due to the ExtCom commitments played by $\mathsf{S_{soa}}$) and to the computational hiding property of ExtCom (due to the ExtCom commitments played by $\mathsf{R_{soa}}$). At a high level, we show that if an adversarial sender breaks binding, then it should have been able to bias outcome of the coin-flipping by predicting the randomness committed to by the receiver using ExtCom, before the sender sends its own ExtCom commitments. Then in the reduction, we make use of this fact to break computational hiding of ExtCom. Here, we would like to give a heads-up to the reader that there would be a few subtleties that need attention in constructing the reduction; the subtleties and the new techniques that we will use to resolve them would become clear as we proceed along the proof.

We prove the binding property of SOACom using the statistical binding property of ExtCom (due to the ExtCom commitments played by $\mathsf{S_{soa}}$) and the computational hiding property of ExtCom (due to the ExtCom commitments played by $\mathsf{R_{soa}}$).

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

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

Commitment phase.

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

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

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

$\mathsf{R}_{\mathsf{soa}}$: *If all the corresponding openings provided by* $\mathsf{S}_{\mathsf{soa}}$ *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 (Protocol 2) basically consists of running the commitment phase of an extractable commitment scheme ExtCom in both directions (i.e. from $\mathsf{S}_{\mathsf{soa}}$ to $\mathsf{R}_{\mathsf{soa}}$ and vice versa). Implementing ExtCom using the $(3, 1)$-round extractable commitment scheme described in Protocol 3 (Appendix A.4), it seems that the commitment phase requires 4-rounds. However, by merging the third round of the extractable commitment played by $\mathsf{S}_{\mathsf{soa}}$ with the first round of the opening phase (played by $\mathsf{S}_{\mathsf{soa}}$ 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.*

The proof can be found in Appendix D.3.

$(5, 1)$-round SOA-secure Scheme based on BB use of OWPs. Our $(5, 1)$-round SOA-secure commitment scheme on BB use of any OWP is very similar to the $(3, 3)$-round scheme presented in Protocol 2 and is essentially based on shifting the first two rounds of the opening phase of Protocol 2 to the commitment phase. However, the opening strategy is slightly different to allow for simulation. The opening phase indeed is such that sender can open either the extractable commitments that are always in the positions defined by the coin flipping or the extractable commitments that are always in the positions defined by the binary negation of it.

Intuitively, this modification in the opening strategy is due to the following fact. Note that in the $(3, 3)$-round scheme the sender sends its share of randomness d in the first round of the decommitment phase. Thus, in the proof of hiding of our $(3, 3)$-round scheme, the simulator knows the bit to be opened to before it sends its share of randomness d. However, when we shift the first two rounds of the decommitment phase of the $(3, 3)$-round scheme to the commitment phase, the simulator in the hiding experiment, (which needs to output commitment phase transcripts before receiving the bits it needs to open them to), when it needs to send d it does not know yet as to which bit to open to and hence it does not know whether to bias the outcome of coin-flipping to the ExtCom commitments of 1 or to the ExtCom commitments of 0. Hence, the aforementioned modification of giving the sender the freedom of opening at either the outcome of coin flipping or its negation later facilitates simulation, as explained in further detail in the proof.

Further details appear in Appendix C.2.

4 Issues in Some of the Claims of [Xia11a]

In this section we point out some issues regarding some of the main results in [Xia11a].

Revisiting proof of Theorem 3.3 in [Xia11a]. Theorem 3.3 in [Xia11a] claims that their $(4, 1)$-round protocol is SOA-secure under parallel composition with BB use of OWPs. The protocol recalls the equivocal commitment scheme of [CO99]. There is a preamble for coin flipping followed by Naor's commitment. In the preamble, firstly, the receiver commits to a random string α using a non-interactive (therefore computationally hiding only) commitment scheme; secondly, the sender sends a random string β in the clear to the receiver; finally, the receiver opens its commitment. Theorem 3.3 in [Xia11a] claims that the resulting protocol is a computationally hiding, computationally binding SOA-secure under parallel composition with BB use of a OWP. The first problem is that it is not clear how one can prove the binding of such commitment scheme. The authors mention that binding follows from the same arguments of Naor's commitment [Nao91]. However it does not seem to be the case. While in Naor's commitment scheme the receiver sends a random string, here there is a coin flipping where the receiver first commits in a computationally hiding way. Therefore the malicious sender could have an advantage in biasing the outcome of the coin flipping, due to the computational hiding only. Therefore if one wants to prove computational binding of the SOA scheme, there should be a reduction to the hiding of the commitment played by the receiver in the coin flipping. Such a reduction seems to be very unlikely since the reduction should be completed without opening the commitment played in the first round of the coin flipping, therefore only 2 rounds can be played. From 2 rounds the only information that an adversary for the hiding of the commitment of the coin flipping can get is the random string received from the adversarial sender of the SOA scheme. With this sole information, it is not possible to check in polynomial time if the xor of such a string received from the sender with one of the possible strings committed in the first round of the coin flipping produces a string that is the output of the pseudo-random generator (this is indeed the sole way that allows a malicious sender of the SOA scheme of Theorem 3.3 to equivocate). We do not see how this reduction can be completed. The proof of binding for Theorem 3.3 in [Xia11a] is essentially missing, indeed one can not rely on the arguments of [CO99] since they are based on the use of a perfectly hiding commitment in the first round of the coin flipping. However such a commitment scheme can not be implemented in one round in the standard model (not to mention the issue of using OWPs only in a BB manner).

Beyond the above major problem with the proof of binding, there are also issues with the proof of SOA security due to difficulties about applying the Goldreich and Kahan [GK96] simulation strategy when multiple sessions are played in parallel with possibly different abort probabilities. For further details, see Appendix E.

We remark that although the $(4,1)$-round scheme of [Xia11a] is not simulatable directly via the Goldreich-Kahan simulation strategy, the author of [Xia11a], elaborated an alternative simulation strategy for the same protocol [Xia11b]. The proof of binding however as remarked above is still missing and unlikely to exist.

Revisiting proof of Theorem 3.5 in [Xia11a]. Theorem 3.5 of [Xia11a] claims that if a coin-flipping preamble implemented via the $\omega(\log(n))$-round preamble of [PRS02], is followed by Naor's commitment, 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, if the coin-flipping is implemented with [PRS02]'s preamble, the claim of [Xia11a] is that the simulator shown in [PRS02] obtains the random string committed to by the receiver, by the end of the coin-flipping, and this values can be used by the SOA-simulator to equivocate. Now, firstly, like we mentioned in the discussion above, there could exist adversarial receivers who always abort some specific sessions thus rendering the above claim to be immediately untrue. This itself 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 [Xia11a]) 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. In 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 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 a deviation in the distribution of the resulting set of indexes 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 3) that establishes the impossibility of schemes (with any round complexity) that satisfy SOA-security under concurrent composition, unknown message distributions, and black-box simulation.

Revisiting proof of Theorem 4.4 in [Xia11a]. Theorem 4.4 in [Xia11a] 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 the commitment phase is such that the sender speaks first. However, we argue that this assumption loses generality. In fact, we present a $(3, 1)$-round commitment scheme (Protocol 1) in which the receiver speaks first, such that security in the concurrent-with-barrier setting (that is strictly stronger than the parallel composition setting [Xia11a]) 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 [Xia11a] implies the impossibility of a 2-round protocol.

5 Impossibility of Fully Concurrent Black-Box SOA-Security

The protocols presented in our paper achieve security under concurrent-with-barrier composition in the "strong" sense, that is, assuming that the simulator does not know the distribution of the messages committed to by the sender. The last question to answer is whether there exist protocols that actually achieve the definition of security under strong fully (i.e., without barrier) concurrent composition (as defined in [Xia11a]), or if the concurrent-with-barrier security definition is the best one can hope to achieve (when black-box simulation is taken into account). In this section we show that in contrast to the claim of Theorem 3.5 of [Xia11a], the strong fully concurrent security definition of [Xia11a] is impossible to achieve. This holds regardless of the round complexity of the

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

protocol [7] and of the black-box use of cryptographic primitives. Under the assumption that OWFs exist, the only requirements 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. Both requirements are already specified in the strong fully concurrent security definition of [Xia11a]. In the following, we first recall the definition provided in [Xia11a] for completeness, then we give the intuitions behind the proof.

Definition of hiding under SOA - concurrent composition (from [Xia11a]). Let $\mathcal{B}$, $\mathcal{I}$, k be as defined in Definition 3 and $\mathbf{b} \xleftarrow{\$} \mathcal{B}$ be the input given to the honest sender S.

Security is defined as 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. For example, the receiver can generate one commit-phase transcript, ask the sender to decommit that instance, then use this information in its interaction to generate the second commit-phase transcript, and so forth. The output of this experiment is defined as $\langle \mathsf{S}^k(\mathbf{b}), \mathsf{R}^* \rangle = (\tau^k, I, \{b_i, w_i\}_{i \in I})$, 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 game, 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 answers to a query j with the value $\mathbf{b}[j]$. The output $(\mathsf{Sim}_k^{\mathsf{R}^*} | \mathbf{b})$ 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 every $\mathcal{I}, \mathcal{B}$ and k, there exists Sim such that for all R^* it holds that $\langle \mathsf{S}^k(\mathbf{b}), \mathsf{R}^* \rangle$ and $(\mathsf{Sim}_k^{\mathsf{R}^*} | \mathbf{b})$ are computationally indistinguishable. As stated in [Xia11a], 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 [Xia11a] stronger. If instead there exists Sim that works for all $\mathcal{B}, \mathcal{I}$ and R^* then the protocol is said SOA-secure under fully concurrent composition. All the constructions shown in [Xia11a] are claimed to achieve this strong(er) definition in which the message distribution $\mathcal{B}$ is not known by Sim. The same definition can be extended to concurrent SOA-secure string commitment scheme.

From the definition shown above note the following. The set I given as output in the ideal game 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 very 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 3. *If OWF exists, then no string commitment scheme can be SOA-secure under fully concurrent composition.*

Proof Idea. Our proof consists in adapting a proof provided by Lindell in [Lin03].[Lin03] shows that there exist functionalities for which proving that a protocol is secure under m-concurrent

[7]This is therefore different from the case of concurrent zero knowledge [CKPR01, PRS02].

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, such as, 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. In our proof, we convert the role of the oracle $\mathcal{O}$ into the role of the functionality, and when deriving the contradiction we do not break the privacy of the receiver but the correctness of the protocol (i.e. the binding).

The full proof is shown in Appendix D.5 and 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 the adversary R^* chooses the sessions to decommitment adaptively on the transcript, and in order to obtain the string to provide the decommitment, Sim must query an external oracle (recall that we are considering the strong definition in which the simulator does not know the message distribution). Thus, changing the transcript in the rewinding yields to different sessions asked by R^*, and in turns more queries made by Sim to the oracle. Such additional queries are caused only by the rewinding attempts and they do not appear in the real world execution. However, the distribution of I due to Sim in the ideal game should not be distinguishable from the one due to R^* 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. The formal proof can be found in Appendix D.5. In Appendix D.5, as a corollary, we show that this result holds also for bit commitment schemes.

6 Application to cZK with Pre-processing

Additionally, we show how to use SOA-secure commitment schemes to construct concurrent zero-knowledge (cZK) protocol with pre-processing by using OWFs only, therefore improving a previous result of [CO99]. We combine our $(3, 1)$-round SOA-secure computationally binding scheme based on NBB use of OWPs with the use of the special $\mathcal{WIPoK}$ of [LS90]. The preprocessing takes 3 rounds and is composed by two subprotocols played in parallel. The first subprotocol is a coin-flipping protocol where the prover commits to a random string using the SOA commitment that ends with the 3rd round of the verifier. In the 3rd round the verifier also sends his random string and the xor of the two strings is the outcome of this subprotocol. The second subprotocol is a special $\mathcal{WIPoK}$ to prove that $x \in L$ or the output of the coin flipping is also the output of a PRG. Only two rounds of this subprotocol are played during the preprocessing.

At the end of the above preprocessing the prover knows the result of the coin flipping and later non-interactively can complete the proof by opening his SOA commitment and sending the last round of the special $\mathcal{WIPoK}$. The simulator will get advantage of the simulator of the SOA commitment to bias the outcome of all coin-flipping protocols, therefore being able to complete all proofs running the prover of the special $\mathcal{WIPoK}$ using the trapdoor witness.

Acknowledgments

Supported in part by NSF grants 0830803, 09165174,1065276, 1118126 and 1136174, US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. This material is also based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. The work of the third and the fourth authors has been done while visiting UCLA and is supported in part by the European Commission through the FP7 programme under contract 216676 ECRYPT II.

References

[BDWY12] Mihir Bellare, Rafael Dowsley, Brent Waters, and Scott Yilek. Standard security does not imply security against selective-opening. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 645–662. Springer, 2012.

[BHK12] Florian Böhl, Dennis Hofheinz, and Daniel Kraschewski. On definitions of selective opening security. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *Lecture Notes in Computer Science*, pages 522–539. Springer, 2012.

[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.

[Blu86] Manuel Blum. How to Prove a Theorem So No One Else Can Claim It. In *Proceedings of the International Congress of Mathematicians*, pages 1444–1451, 1986.

[CDSMW09] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Simple, Black-Box Constructions of Adaptively Secure Protocols. In *Theory of Cryptography, 6th Theory of Cryptography Conference, TC C 2009*, pages 387–402, 2009.

[CKPR01] Ran Canetti, Joe Kilian, Erez Petrank, and Alon Rosen. Black-box concurrent zero-knowledge requires omega∼(log n) rounds. In *STOC*, pages 570–579, 2001.

[CO99] Giovanni Di Crescenzo and Rafail Ostrovsky. On concurrent zero-knowledge with pre-processing. 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.

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. In *STOC*, pages 409–418, 1998.

[FS89] Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 526–544. Springer, 1989.

[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.

[Rey01] Leonid Reyzin. *Zero-Knowledge with Public Keys, Ph.D. Thesis.* MIT, 2001.

[Wee10] Hoeteck Wee. Black-Box, Round-Efficient Secure Computation via Non-malleability Amplification. In *Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '10, pages 531–540, 2010.

[Xia11a] David Xiao. (Nearly) round-optimal black-box constructions of commitments secure against selective opening attacks. In *TCC*, pages 541–558, 2011.

[Xia11b] David Xiao. Unpublished manuscript. Personal communication, October 2011.

[Xia12a] David Xiao. On the round complexity of black-box constructions of commitments secure against selective opening attacks. Cryptology ePrint Archive, Report 2009/513 - Revision May 29, 2012, 2012. `http://eprint.iacr.org/`.

[Xia12b] David Xiao. Round-optimal black-box statistically binding selective-opening secure commitments. In *Progress in Cryptology - AFRICACRYPT 2012 - 5th International Conference on Cryptology in Africa, Ifrance, Morocco, July 10-12, 2012. Proceedings*, volume 7374 of *Lecture Notes in Computer Science*, pages 395–411. Springer, 2012.

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* $\mathsf{wTCom}=$ (wTCGen, S, R, $\mathsf{TCFakeCom}$, $\mathsf{TCFakeDec}$) *is a weak trapdoor commitment scheme if* TCGen *on input a random n-bit string r, 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* $(pk, sk) \leftarrow \mathsf{TCGen}(r)$*, for any PPT malicious receiver* R^* *there exists a negligible function* ϵ *such that the following holds:*

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

The probability is taken over the choice of the randomness r for the algorithm TCGen *and the random coins of the parties.*

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

```
Experiment $\mathbf{Exp}^{\mathsf{wTrap}}_{\mathsf{wTCom}}(n)$:
run $(\xi, \cdot) \xleftarrow{\$} \langle \mathsf{TCFakeCom}(pk, sk, \mathtt{com}), \mathsf{R}^*(pk, sk, b, \mathtt{recv})\rangle$;
$b' \xleftarrow{\$} \langle \mathsf{TCFakeDec}(\mathtt{open}, b, \xi), \mathsf{R}^*(\mathtt{open})\rangle$;
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 exists a non-interactive weak trapdoor commitment schemes that is *not* a "regular" non-interactive trapdoor commitment scheme as follows. Take any "regular" trapdoor commitment scheme in which the decommitment phase is non-interactive. A non-interactive weak trapdoor commitment scheme can be constructed by using the regular trapdoor commitment scheme to commit to a bit, and then by adding two (perfectly binding) commitments of the openings. The honest sender will open one of the two perfectly binding commitment chosen at random. Instead knowledge of the trapdoor from the commitment phase allows one to commit both to a decommitment of 0 and to a decommitment of 1 (in random order), therefore allowing equivocation in the opening. The interesting point is that this scheme is not a "regular" trapdoor commitment scheme, which implies that a weak trapdoor commitment scheme could be in theory constructed under better assumptions, or with better efficiency. Notice that in [PW09] it is shown a construction of an interactive weak trapdoor commitment scheme (they called it "look-ahead" trapdoor commitment) from any black-box use of a one-way permutation.

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 $\mathsf{M}[i][j] = 0$.

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

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

$\mathsf{FLS3}$ ($\mathcal{P} \rightarrow \mathcal{V}$):

- if $b = 0$, $\mathcal{P}$ opens all the commitments, showing that the matrix committed in step $\mathsf{FLS1}$ 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 $\mathsf{FLS3}$: 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 $\mathsf{FLS3}$. Looking ahead this allows the simulator to equivocate in the decommitment phase in which it will be required to perform $\mathsf{FLS3}$, without changing the transcript of the commitment phase. This property turns out to be very important to achieve SOA-secure protocols.

A.3 Concurrent Zero-Knowledge with Pre-processing

Concurrent zero knowledge (cZK, for short) with pre-processing is a variant of concurrent zero knowledge that consists of two phases as described in [CO99]. In the first phase, called the *pre-processing* phase, the prover and the verifier interact possibly without the knowledge of the theorem statement. After the completion of the pre-processing phase, both the parties are given a theorem statement and the prover is also given the witness. Then they interact in the next phase called the *proof* phase. The requirements are completeness, soundness and concurrent zero-knowledge, where the notion of concurrency is that an adversarial verifier can interact with the provers in polynomially many executions, with the pre-processing phases of all the executions being completed before the beginning of the proof phase of any execution. The following definition is borrowed from [CO99].

We first give a description of an interactive protocol with pre-processing, and then give a definition of cZK with pre-processing. *A*n interactive protocol with pre-processing is a pair of interactive protocols $(\langle A1, B1\rangle, \langle A2, B2\rangle)$. The mechanics of an interactive protocol with pre-processing is divided in two phases, as follows. In the first phase, called the pre-processing phase, the first pair $\langle A1, B1\rangle$ is executed; at the end of this phase a string state_A is output by $A1$ and given as private input to $A2$, and a string state_B is output by $B1$ and given as private input to $B2$. Now, an input string x is given as common input to $A2$ and $B2$, and each of $A2$ and $B2$ is given a private input corresponding to x, and the second pair $\langle A2, B2\rangle$ is executed. A2 runs on input a valid witness for x.

Definition 7 (Concurrent Zero Knowledge with Pre-processing ([CO99] generalized))**.** *Let* $\langle \mathcal{P}, \mathcal{V}\rangle = (\langle \mathcal{P}1\ \mathcal{V}1\rangle, \langle \mathcal{P}2, \mathcal{V}2\rangle)$ *be an interactive protocol with pre-processing. We say that* $\langle \mathcal{P}, \mathcal{V}\rangle$ *is a concurrent computational (resp., statistical, perfect) zero-knowledge proof system with pre-processing for language* L *if the following conditions hold:*

- *Completeness.* $\forall x \in L, \forall w \in \mathcal{R}_L(x)$ *it holds that* $\mathsf{Pr}[(\mathsf{state}_{\mathcal{P}}, \mathsf{state}_{\mathcal{V}}) \overset{\$}{\leftarrow} \langle \mathcal{P}1, \mathcal{V}1\rangle(1^{|x|}); (\cdot, \mathsf{accept}) \overset{\$}{\leftarrow} \langle \mathcal{P}2(w, \mathsf{state}_{\mathcal{P}}), \mathcal{V}2(\mathsf{state}_{\mathcal{V}})\rangle(x)] = 1$.
- *Soundness. For any* $x \notin L$*, and any* $(\mathcal{P}1^*, \mathcal{P}2^*)$*, the following probability is negligible:*
$$\mathsf{Pr}[(\mathsf{state}_{\mathcal{P}}, \mathsf{state}_{\mathcal{V}}) \overset{\$}{\leftarrow} \langle \mathcal{P}1^*, \mathcal{V}1\rangle(1^{|x|}); (\cdot, \mathsf{accept}) \overset{\$}{\leftarrow} \langle \mathcal{P}2^*(\mathsf{state}_{\mathcal{P}}), \mathcal{V}2(\mathsf{state}_{\mathcal{V}})\rangle(x)]$$
.
- *Concurrent Zero-Knowledge: There exists an expected polynomial-time simulator algorithm* Sim *such that, for each probabilistic polynomial-time algorithm* $\mathcal{V}^* = (\mathcal{V}1^*, \mathcal{V}2^*)$*, for any polynomial* $q = q(n)$*, for each* $x_1, \ldots, x_q \in L$*, for each* $w_i \in \mathcal{R}_L(x_i)$ *where* $i \in [q]$*, where* $|x_1| = \ldots = |x_q| = n$*, the two distributions* $\mathsf{Sim}^{\mathcal{V}^*}(\mathbf{x})$ *and* $\mathsf{View}_{\mathcal{V}^*}(\mathbf{x})$ *are computationally (resp., statistically, perfectly) indistinguishable, where* $\mathsf{View}_{\mathcal{V}^*}(\mathbf{x})$ *is the output of* $\mathcal{V}^*$ *after playing concurrently with polynomially many honest provers in the pre-processing phase, and then subsequently in the concurrent phase.*

Remark 4. *In the above definition, we consider a black-box simulator, in contrast to [CO99] wherein the definition only requires, for every adversarial verifier, the existence of a simulator.*

A.4 Constant-round Extractable Commitment Schemes

A key component of three of our protocols - Protocol 2 and Protocol 5 - 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 to by the sender. Following are the definition of extractable commitment schemes and a constant-round construction for the same. Since we would only need the scheme to be statistically-binding, we shall focus only on the statistically-binding variant.

Definition 8 (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 valid opening to 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. Moreover, if $b' \neq \perp$ then it is statistically impossible to open τ to any value other than 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, we require only a commitment scheme that is extractable in the stand-alone setting (as it will be clearer later in the proofs), and such a scheme can be constructed in constant number of rounds. The crucial difference lies in the fact that even in the case that 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 the execution of rewinding threads. This simplification essentially comes from the setting of concurrency with barrier (while the setting in [MOSV06] is full-fledged concurrency).

Protocol 3. *[Extractable Commitment Scheme,* 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 can be constructed based on black-box use of any one-way permutations.

$\mathsf{S}_{\mathsf{ext}}$**'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 $(c_i^0, \cdot) \xleftarrow{\$} \langle \mathsf{S}_{\mathsf{NI}}(\mathtt{com}, b_i^0), \mathsf{R}_{\mathsf{NI}}(\mathtt{recv})\rangle$ and $(c_i^1, \cdot) \xleftarrow{\$} \langle \mathsf{S}_{\mathsf{NI}}(\mathtt{com}, b_i^1), \mathsf{R}_{\mathsf{NI}}(\mathtt{recv})\rangle$ with $\mathsf{R}_{\mathsf{ext}}$;*

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

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

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

Decommitment phase.

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

$\mathsf{R}_{\mathsf{ext}}$: *If any of the openings is invalid, then output $\perp$. Otherwise, if there exists a bit $\tilde{b}$ such that, $\forall i \in [n]$, $\beta(e_i) \oplus \gamma(e_i) = \tilde{b}$, then output $\tilde{b}$. Otherwise, output $\perp$.*

Remark 5. *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 a sub-protocol. In the following we prove the above protocol is an extractable commitment scheme.

Theorem 4 (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. Let $\mathsf{Com}_{\mathsf{NI}}$ be a statistically-binding commitment scheme. We prove that ExtCom satisfies hiding, binding, and extractability as follows.

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 generated 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 $\mathsf{Com}_{\mathsf{NI}}$ satisfies hiding, then ExtCom also satisfies hiding.

Suppose there exists a PPT adversary $\mathsf{R}^*_{\mathsf{ext}}$ that breaks hiding of ExtCom with probability δ (i.e., $\mathbf{Adv}^{\text{hiding}}_{\mathsf{ExtCom},\mathsf{R}^*_{\mathsf{ext}}} = \delta$). Then we construct an efficient adversary $\mathsf{R}^*_{\mathsf{NI}}$ that breaks hiding of $\mathsf{Com}_{\mathsf{NI}}$ 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 the same as $\mathsf{S}_{\mathsf{ext}}$ with the only exception that during the first round message of the commit phase, for $j > i$ it chooses b_j^0, $b_j^1 \xleftarrow{\$} \{0,1\}$ such that of $b_j^0 \oplus b_j^1 = 0$, and for $j \le i$ it chooses b_j^0, $b_j^1 \xleftarrow{\$} \{0,1\}$ such that of $b_j^0 \oplus b_j^1 = 1$. Finally, when $\mathsf{R}^*_{\mathsf{ext}}$ outputs a bit b', the same is set to be the output of the experiment. We denote by $\mathsf{Hyb}^i_{\mathsf{ExtCom},\mathsf{R}^*_{\mathsf{ext}}} \to 1$ the event that the output of this hybrid experiment is 1.

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

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

1. Sample $\mu \xleftarrow{\$} [n]$.
2. Compute the first round message 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 $b_i^0 \oplus b_i^1 = 1$.
   - Sample $\theta \xleftarrow{\$} \{0,1\}$ and $b_\mu^{1-\theta} \xleftarrow{\$} \{0,1\}$.
3. For $\forall i \in [n]-\{\mu\}$, $\forall j \in \{0,1\}$, and for $(i,j) = (\mu, 1-\theta)$, run $(c_i^j, \cdot) \xleftarrow{\$} \langle \mathsf{S}_{\mathsf{NI}}(\mathtt{com}, b_i^j), \mathsf{R}_{\mathsf{NI}}(\mathtt{recv})\rangle$ with $\mathsf{R}^*_{\mathsf{ext}}$. Also, set $c_i^\theta \leftarrow c^*$, the commitment from the external sender, and send it to $\mathsf{R}^*_{\mathsf{ext}}$.
4. Upon receiving a challenge e from $\mathsf{R}^*_{\mathsf{ext}}$, check whether $e_\mu = \theta$. If so, then output a random bit and halt; otherwise, proceed as per the protocol and once $\mathsf{R}^*_{\mathsf{ext}}$ outputs a bit b', output $b' \oplus b_\mu^{1-\theta}$.

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

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

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

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

That proves the hiding property of ExtCom.

Binding. Breaking binding of ExtCom (i.e., producing two valid openings, one for 0 and another for 1, for a single commitment phase transcript) necessarily means producing two valid openings 0 and 1 respectively of at least one of the $\mathsf{Com_{NI}}$ commitments, thus breaking statistical binding of $\mathsf{Com_{NI}}$. Hence, such an event can occur with at most negligible probability.

Extractability. We show that ExtCom satisfies extractability by constructing an expected polynomial-time extractor E that having black-box access to the adversary $\mathsf{S}^*_{\mathsf{ext}}$ in the commitment phase outputs a transcript that is statistically indistinguishable from the transcript of the interaction $\langle \mathsf{S}^*_{\mathsf{ext}}(\mathtt{com}), \mathsf{R}_{\mathsf{ext}}(\mathtt{recv})\rangle$ and (if the commitment phase transcript passes the consistency check of $\mathsf{R}_{\mathsf{ext}}$) a bit b so that it is statistically impossible to open it to any other value.

The extractor works as follows:

Extractor E

Initialization phase. Choose random tapes $\mathsf{ran}_{\mathsf{S}}$ and ran_E, respectively, for $\mathsf{S}^*_{\mathsf{ext}}$ and for the main thread below.
Invoke $\mathsf{S}^*_{\mathsf{ext}}(\mathsf{ran}_{\mathsf{S}})$.

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

Rewinding threads (E2). If τ_{com} is accepting, then keep running $\mathsf{R}_{\mathsf{ext}}$ with $\mathsf{S}^*_{\mathsf{ext}}(\mathsf{ran}_{\mathsf{S}})$ in $\langle \mathsf{S}^*_{\mathsf{ext}}(\mathtt{com}), \mathsf{R}_{\mathsf{ext}}(\mathtt{recv})\rangle$, each time with freshly chosen randomness, until it receives another accepting response $\beta(e')$ from $\mathsf{S}^*_{\mathsf{ext}}(\mathsf{ran}_{\mathsf{S}})$ with some challenge e'. If $e' = e$, then output $(\tau_{\mathsf{com}}, \perp)$ 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 $(\tau_{\mathsf{com}}, \beta(e_i), b_i \oplus b'_i)$.

Now, let q be the probability over e that we obtain an accepting transcript τ_{com}. Then the expected number of queries E makes to $\mathsf{S}^*_{\mathsf{ext}}$ is $(1-q)+q\cdot 1/q \leq 2$. Also, the probability that E fails to extract (i.e., the probability that τ_{com} is accepting and $e = e'$) is at most 2^{-n}. Furthermore, an opening to a bit different from the extracted bit directly breaks statistical binding of ExtCom, and hence can be produced with at most negligible probability. This completes the description and analysis of the extractor. □

Remark 6 (Extractable string-commitment scheme)**.** *The extractable bit-commitment scheme in Protocol 3 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 proof of impossibility of [Xia11a] for $(3,1)$-round SOA-secure commitments, the arguments in the proof 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 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 [Xia11a] 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 Scheme based on BB use of Weak Trapdoor Commitments

Let us denote as wTCom = (wTCGen, $\mathsf{S_{TC}}$, $\mathsf{R_{TC}}$, $\mathsf{TCFakeCom}$, $\mathsf{TCFakeDec}$) a weak-trapdoor commitment scheme. 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 $\langle \mathsf{S_{TC}}{}_i^{\sigma}, \mathsf{R_{TC}}{}_i^{\sigma} \rangle$ the i-th invocation of the algorithms of protocol wTCom using the public key pk_i^{σ}. The sender $\mathsf{S_{soa}}$ has as input a bit b.

Protocol 4. *[(4,1)-round SOA-secure commitment scheme based on BB use of weak trapdoor commitment] [*$\mathsf{SOACom} = (\mathsf{S_{soa}}, \mathsf{R_{soa}})$*]*

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* $\{\mathsf{pk}_i^0, \mathsf{pk}_i^1\}$ *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* $d_i \xleftarrow{\$} \{0,1\}$*;*

$\mathsf{R_{soa}}$**:** *Upon receiving* $d_1, \ldots, d_n$ *send* $\{r_i^{d_i}\}_{i \in [n]}$ *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* $\mathsf{pk}_i'^{d_i} \neq \mathsf{pk}_i^{d_i}$ *then* `ABORT`.
2. *secret share the bit b: for* $i = 1, \ldots, n$*:* $b_i \xleftarrow{\$} \{0,1\}$*, such that* $b = (\bigoplus_{i=1}^n b_i)$*;*
3. *run* $\langle \mathsf{S_{TC}}{}_i^{\bar{d}_i} (\mathsf{pk}_i^{\bar{d}_i}, \texttt{com}, b_i), \mathsf{R_{TC}}{}_i^{\bar{d}_i}(\mathsf{pk}_i^{\bar{d}_i}, \texttt{recv})\rangle$ *with* $\mathsf{R_{soa}}$*;*

Decommitment phase.

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

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

The above protocol can be instantiated with the *weak* trapdoor commitment scheme based on BB access to OWPs shown in [PW09]. In such a protocol the commitment is interactive, and follows the commit-challenge-response structure. The commitment is such that if the sender knows the challenge in advance, it can commit in a way that allows equivocation. In such a scheme the trapdoor is the challenge sent by the receiver, and in turn, the public parameter is the (statistically hiding) commitment of the challenge. One can plug such protocol in our (4,1)-round SOA-secure protocol and obtain a (6,1)-round protocol (the commitment phase in [PW09] is interactive).

Theorem 5 (Protocol 4 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 4 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 Scheme on BB use of OWPs.

In this section we present a $(5,1)$-round SOA-secure commitment scheme based on 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 that 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 so 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 its binary-negation. The rest of the protocol remains the same as Protocol 2. Details to follow in Protocol 5.

Protocol 5. *[(5,1)-round SOA-secure Scheme based on BB use of OWPs.] [*$\mathsf{SOACom} = (\mathsf{S_{soa}}, \mathsf{R_{soa}})$*]*
$\mathsf{S_{soa}}$**'s input:** $b \in \{0,1\}$

Commitment phase.

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

```
    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 $\mathsf{S_{soa}}$;
```

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

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

$\mathsf{S_{soa}}$: *If all extractable commitments 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 $\mathsf{S_{soa}}$;
```

Decommitment phase.

$\mathsf{S_{soa}}$: *If all openings provided by* $\mathsf{R_{soa}}$ *are valid, sample* $\theta \xleftarrow{\$} \{0,1\}$ *and send* θ *to* $\mathsf{R_{soa}}$*. Also, for* $i = 1, \ldots, n$*:*

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

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

Theorem 6 (Protocol 5 is secure under selective opening attacks)**.** *If* ExtCom *is an extractable commitment scheme, then Protocol 5 is a commitment scheme secure against selective opening attacks.*

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

D Proofs

D.1 Proof of Theorem 1

Proof. In the following, we prove completeness, binding and hiding under selective-opening-attacks of the $(3, 1)$ round protocol presented in Protocol 1.

We refer to the execution of the commitment (resp., decommitment) procedure of the sub-protocol TC as sub-commitment (resp., sub-decommitment). 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 it 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 valid 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 [8]. 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 to both 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 SOACom with non-negligible probability δ. Then there exists at least one pair (i, σ) such that $\mathsf{S}^*_{\mathsf{soa}}$ opens the commitment computed using the public key pk_i^σ in two ways; more formally such that: $(\cdot, b_i) \leftarrow \langle \mathsf{S}_{\mathsf{TC}i}(\mathsf{open}, 0), \mathsf{R}_{\mathsf{TC}i}(\mathsf{open})\rangle$ and $(\cdot, b_i') \leftarrow \langle \mathsf{S}_{\mathsf{TC}i}(\mathsf{open}, 1), \mathsf{R}_{\mathsf{TC}i}(\mathsf{open})\rangle$ such that $\perp \neq b_i \neq b_i' \neq \perp$. 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 randomly picks $i \in [n]$, $\sigma \in \{0, 1\}$ and sets $\mathsf{pk}_i^\sigma = pk$, while it honestly generates $2n - 1$ pairs of public/secret parameters running TCGen. 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}. When $\mathsf{S}^*_{\mathsf{soa}}$ sends the challenge $d_1, \ldots, d_n$: 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, $\mathsf{S}^*_{\mathsf{TC}}$ answers as the honest receiver $\mathsf{R}_{\mathsf{soa}}$, concluding the commitment phase.

In the opening phase, $\mathsf{S}^*_{\mathsf{TC}}$ is invoked to execute the opening phase with bits 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 from $\mathsf{S}^*_{\mathsf{soa}}$. $\mathsf{S}^*_{\mathsf{TC}}$ wins the binding experiment for protocol TC if $\mathsf{S}^*_{\mathsf{soa}}$ provides two distinct openings for the i-th sub-commitment. Therefore $\mathsf{S}^*_{\mathsf{TC}}$ wins the binding game with probability at least $\delta/2n$.

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

[8]We assume that if $\mathsf{S}^*_{\mathsf{soa}}$ computes a commitment using a public key for which she later asks to see the secret key, she will be caught by the honest receiver.

interaction between $\mathsf{R}^*_{\mathsf{soa}}$ and $\mathsf{S}_{\mathsf{soa}}$ in the real game.

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 opened we use the following notation: we refer to j_ℓ the session corresponding to the ℓ-th index, where $\ell = 1, \ldots, m$. 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 $\{\mathsf{pk}^0_i, \mathsf{pk}^1_i\}_{i\in[n]}$ 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 $\mathsf{pk}^{\bar{d}_i}_i$ by 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, then abort session j.
- Upon receiving $\{r^{d_i}_i\}_{i\in[n]}$ for some session j, check their consistency running $(\mathsf{sk}^{d_i}_i, \mathsf{pk}^{d_i}_i) \leftarrow \mathsf{TCGen}(r^{d_i}_i)$. If the check fails, abort session j. Otherwise store the secret keys $\{\mathsf{sk}^{d_i}_i\}_{i\in[n]}$ for session j.

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

Extraction phase (rewinding thread).

For $\ell = 1, \ldots, m$; for session $j_\ell \in I$ that was not-aborted in the main thread do:

1. activate $\mathsf{R}^*_{\mathsf{soa}}$ with randomness $\mathsf{ran}_{\mathsf{R}}$ and use $\mathsf{ran}_{\mathsf{Sim}}$ to execute all the sessions except j_ℓ.
2. in session j_ℓ, uniformly choose bits $d'_1, \ldots, d'_n$ and compute the sub-commitments as in Step **S1** (note that the view of $\mathsf{R}^*_{\mathsf{soa}}$ generated in this step is distributed identically as in the main thread). If $\mathsf{R}^*_{\mathsf{soa}}$ aborts then go to Step 1. If $\mathsf{R}^*_{\mathsf{soa}}$ starts new sessions, follow instructions as in Step **S1**.
3. When $\mathsf{R}^*_{\mathsf{soa}}$ replies with $\{r^{d'_i}_i\}_{i\in[n]}$: if strings $d'_1 \ldots d'_n$, $d_1 \ldots d_n$ (the challenge used for session j_ℓ in the main thread) are equal then abort the simulation. Else, if there exists an $r^{d'_i}_i$ generating a valid pair $(\mathsf{sk}^{d'_i}_i, \mathsf{pk}^{d'_i}_i)$ where $\mathsf{pk}^{d'_i}_i$ appeared in the j_ℓ-th session of main thread, and $d'_i \neq d_i$ then return $\mathsf{sk}^{d'_i}_i$. Otherwise go to Step 1.

Extraction completion. If Sim reaches this point, then for every(not-aborted) session $j_\ell \in I$ there is at least one i for which Sim obtained both trapdoors sk^0_i and sk^1_i.

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 be the index such that Sim obtained $\mathsf{sk}^0_i, \mathsf{sk}^1_i$ for the session j_ℓ.

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

1. for all $l \neq i$ honestly run the sub-decommitment algorithm, i.e., $\langle \mathsf{S}_{\mathsf{TC}}{}^{\bar{d}_l}_l(\mathsf{open}), \mathsf{R}_{\mathsf{TC}}{}^{\bar{d}_l}_l(\mathsf{open})\rangle$, 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 $\mathsf{sk}^{\bar{d}_i}_i$, i.e., $(\cdot, b'_i) \leftarrow \langle \mathsf{TCFakeDec}(\mathsf{sk}^{\bar{d}}_i, \mathsf{open}, b'_i), \mathsf{R}_{\mathsf{TC}}{}^{\bar{d}_i}_i(\mathsf{open})\rangle$; If $\mathsf{R}^*_{\mathsf{soa}}$ aborts, then aborts this session.

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

For simplicity we are assuming that the underlying trapdoor commitment TC satisfies the trapdoorness property for any $b^\star \in \{0,1\}$ (Pedersen commitment [Ped92] achieves this property). In case the trapdoorness holds only for a specific bit $b^\star$, then the above simulator should be tweaked only in the extraction phase adding a further condition. That is, when extracting a new secret key $\mathsf{sk}_i^{d_i}$ for a session j, the simulator considers the extraction phase successfully completed for such session, only if the commitment in position (i, d_i) is a commitment of $b^\star$. If this is not the case, then Sim continues the rewinding threads. It is easy to see that this further condition is satisfied w.h.p and similar analysis that we show for the simpler simulator apply.

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 times 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 times are polynomial in n.

In the commitment phase, for each session, Sim executes $2n$ sub-commitments and verifies the validity of the response 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, so that it extracts at least one trapdoor for each session j_ℓ that has been asked for the decommitment. The number of decommitments 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_ℓ, 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 ℓ. 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 in any of the sessions is at most $\mathtt{poly}(n)/2^n$ and thus is negligible.

More formally, let us denote by $\zeta^{j_\ell} = \zeta(\mathsf{ran}_\mathsf{R}, j_\ell)$ the probability that $\mathsf{R}^*_{\mathsf{soa}}$, activated with randomness ran_R, correctly responded to the challenge (i.e., $d_1, \ldots, d_n$) sent by Sim in the commitment phase of the j_ℓ-th session in the main thread. In each rewinding thread the probability to get another correct answer (for some $d'_i, \ldots, d'_n$) is still ζ^{j_ℓ}.

Thus, for each session j_ℓ, the expected number of rewinds needed for the extraction of the trapdoor is bounded by $1/\zeta^{j_\ell}$. 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 denote by $\mathsf{t}_{\mathtt{rew}}$, 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.

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:

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

□

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 as 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 Sim receives the set I from $\mathsf{R}^*_{\mathsf{soa}}$ (upon the completion of the commitment phase), it launches the extraction phase. That is, for each non-aborted session $j_\ell \in I$, Sim launches the extraction phase to obtain the trapdoor $\mathsf{sk}_i^{\bar{d}_i}$ for at least one $i \in [n]$. Possible new sessions initiated by $\mathsf{R}^*_{\mathsf{soa}}$ in the rewinding attempts of the extraction phase are handled by running the honest sender procedure using the knowledge of $\mathbf{b}$. The extracted trapdoors are never used by Sim. 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 of the main thread. Thus the only differences between H_0 and H_1 are: 1) in H_1 Sim runs in expected polynomial time and 2) in H_1 Sim aborts with higher probability due to the possible aborts in the rewinding threads.

Concerning 1), the expected running time is not a problem since we are only interested in the output of the experiment, and it will not be an issue in the reductions shown for the next hybrids since rewinding threads that take longer than a fixed polynomial can be truncated without perturbing the non-negligible probability of success. Concerning 2), observe that in the rewinding threads an abort happens when Sim picks a random challenge that is equal to the challenge sent in the main thread, and it happens with at most negligible probability $\mathtt{poly}(n)/2^n$. Therefore, hybrids H_0 and H_1 are statistically indistinguishable.

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 hybrid $H_2^{j_\ell,s+1}$ Sim works as in experiment $H_2^{j_\ell,s}$ except that, in the $(s+1)$th *new* session started by $\mathsf{R}^*_{\mathsf{soa}}$ during the extraction phase launched for session j_ℓ, Sim 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 a hybrid $\bar{H}_2^{j_\ell,s+1}$ where the bit committed to in the $(s+1)$-th new session is the negation of the 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 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 $1 - \mathbf{b}[t]$. We denote the index of the $(s+1)$-th new session by t, where $1 \le t \le k$. More specifically, let $b_1, \dots, b_n$ be the shares of bit $\mathbf{b}[t]$ in the 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, upon receiving the public keys from $\mathsf{R}^*_{\mathsf{soa}}$ it forwards $\mathsf{pk}_i^{\bar{d}_i}$ to the external sender S_{TC}. We stress that $\mathsf{pk}_i^{\bar{d}_i}$ could be maliciously chosen. As we explained in Remark 1, the hiding experiment is defined for any public parameter pk^* maliciously chosen by R^*.

Upon receiving the sub-commitment from S_{TC} for the public key $\mathsf{pk}_i^{\bar{d}_i}$, $\mathsf{R}^*_{\mathsf{TC}}$ randomly chooses $n-1$ random shares $b_1, \dots, b_n - 1$ and honestly executes $n-1$ sub-commitments using the remaining public parameters received from $\mathsf{R}^*_{\mathsf{soa}}$. Then it forwards all the sub-commitments to $\mathsf{R}^*_{\mathsf{soa}}$. Finally $\mathsf{R}^*_{\mathsf{TC}}$ forwards the output of the experiment to D_{soa} and outputs whatever D_{soa} outputs xored with $\bigoplus_{l \in [n], l \neq i} b_l$.

Now, let b_i such that $\bigoplus_{l \in [n]} b_l = \mathbf{b}[t]$, 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 $\ell = 0, \dots, m-1$: In this sequence of hybrids Sim performs the decommitment phase of the sessions that have been asked for 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. That is, in session $j_{\ell+1}$ Sim honestly performs $n-1$ sub-decommitments while the i-th sub-decommitment is executed running $\mathsf{TCFakeDec}$ 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.

Assume that there exists a distinguisher D_{soa} able to tell apart $H_3^{j_{\ell+1}}$ from $H_3^{j_\ell}$ with non-negligible probability δ, then it is possible to construct an adversary $\mathsf{R}^*_{\mathsf{TC}}$ against the trapdoor property of TC. $\mathsf{R}^*_{\mathsf{TC}}$ runs in the experiment $\mathbf{Exp}_{\mathsf{TC}}^{\mathsf{Trap/Com}}$ against a sender S_{TC} and works as follows.

It runs $\mathsf{R}^*_{\mathsf{soa}}$ as subroutine, it randomly chooses a session s, and then proceeds as follows: for the commitment phase it simulates all sessions as in experiment $H_3^{j_\ell}$ except the session s. In such session, $\mathsf{R}^*_{\mathsf{TC}}$ 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 decommitment phase $\mathsf{R}^*_{\mathsf{soa}}$ asks the opening of m sessions: $\{j_1, \ldots, j_m\}$; if $s \neq j_{\ell+1}$ then $\mathsf{R}^*_{\mathsf{TC}}$ aborts. Otherwise it computes the decommitment phase of the first ℓ sessions (i.e., $j_1, \ldots, j_\ell$) as in hybrid $H_3^{j_\ell}$, while for the decommitment of session s (that is the $j_{\ell+1}$-th session asked for opening) $\mathsf{R}^*_{\mathsf{TC}}$ sends $\mathsf{sk}_i^{\bar{d}_i}$ to S_{TC} and forwards the decommitment received by S_{TC} to $\mathsf{R}^*_{\mathsf{soa}}$ along with the remaining $n-1$ sub-decommitments honestly computed. Finally $\mathsf{R}^*_{\mathsf{TC}}$ forwards the output of $\mathsf{R}^*_{\mathsf{soa}}$ to D_{soa} and outputs what D_{soa} outputs. Now, if in the $j_{\ell+1}$-th session, the i-th sub-decommitment was computed by algorithm $\mathsf{TCFakeDec}$, then the view of $\mathsf{R}^*_{\mathsf{soa}}$ is distributed identically as hybrid $H_3^{j_{\ell+1}}$, otherwise, if it was computed using the honest sender procedure, 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 $\mathbf{Exp}_{\mathsf{TC}}^{\mathsf{Trap/Com}}$ with advantage at least $\delta \cdot \frac{m}{k} \cdot \frac{1}{2n}$ 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 for distinguishability of 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 Theorem 1.

□

D.2 Proof of Theorem 5

Proof. In the following, we prove completeness, binding and hiding under selective-opening-attacks of the $(4, 1)$ round protocol presented in Protocol 4. We use the same notation used in the proof for Protocol 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 Protocol 1, and is therefore 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 generated from 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 $\{\mathsf{pk}^0_i, \mathsf{pk}^1_i\}_{i\in[n]}$ for some session $j \in [k]$: pick a random n-bit string $d_1, \dots, d_n$ and send it to $\mathsf{R}^*_{\mathsf{soa}}$. Label this point as **1-j**.
2. Upon receiving $\{r^{d_i}_i\}_{i\in[n]}$ for some session j, check their consistency by running $(\mathsf{sk}^{d_i}_i, \mathsf{pk}^{d_i}_i) \leftarrow \mathsf{TCGen}(r^{d_i}_i)$. If the check fails, abort session j. In case there exists a secret key $(\mathsf{sk}^{\bar{d}_i}_i)$ (for some $i \in [n]$) already stored for session j, then the trapdoor for session j has been extracted, thus go to step 4. Else go to step 3.
3. Extraction of secret keys for session j: start **rewinding threads** to extract $\mathsf{sk}^{\bar{d}_i}_i$ for some $i \in [n]$:
   (a) rewind $\mathsf{R}^*_{\mathsf{soa}}$ up to point **1-j**.
   (b) send a randomly chosen challenge string $d'_1, \dots, d'_n$ to $\mathsf{R}^*_{\mathsf{soa}}$. If $\mathsf{R}^*_{\mathsf{soa}}$ aborts, go 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 $\{r^{d'_i}_i\}_{i\in[n]}$ for the session j: if the random strings $d'_1, \dots, d'_n$ and $d_1, \dots, d_n$ are equal, abort. Else, if there exists at least one $r^{d'_i}_i$ generating a valid pair $(\mathsf{sk}^{d'_i}_i, \mathsf{pk}^{d'_i}_i)$ where $\mathsf{pk}^{d'_i}_i$ was received in Step **1-j** and $d'_i \neq d_i$ store the secret key $(\mathsf{sk}^{d'_i}_i)$ for session 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 $(\mathsf{sk}^0_i, \mathsf{sk}^1_i)$ proceeds with the commitments for the session j;
   (a) randomly choose bits $b_1, \dots, b_n$;
   (b) for all bits b_l s.t. $l \neq i$ honestly run the sub-commitment algorithm: $(\cdot, b_l) \overset{\$}{\leftarrow} \langle \mathsf{S_{TC}}^{\bar{d}_l}_l(\mathtt{open}), \mathsf{R_{TC}}^{\bar{d}_l}_l(\mathtt{open})\rangle$ with $\mathsf{R}^*_{\mathsf{soa}}$ where d_l is the challenge sent in Step **1-j**.
   (c) for the i-th sub-commitment, run the fake commitment procedure using the secret key $\mathsf{sk}^{\bar{d}_i}_i$: $\langle \mathsf{TCFakeCom}\ (\mathsf{pk}^{\bar{d}_i}_i,\ \mathsf{sk}^{\bar{d}_i}_i,\ \mathtt{com}),\ \mathsf{R_{TC}}^{\bar{d}_i}_i(\mathsf{pk}^{\bar{d}_i}_i, \mathtt{recv})\rangle$. If $\mathsf{R}^*_{\mathsf{soa}}$ aborts, then aborts this session.

Commitment phase completion. When the commitment phase 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 sub-decommitments as follows:

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

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

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

Proof. As the simulator strategy mainly follows the strategy of the simulator shown in Appendix. D.1, the analysis of the running time follows the same arguments shown in 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.

In the following hybrids, we denote by κ the number of sessions opened by $\mathsf{R}^*_{\mathsf{soa}}$ in the main-thread only. We denote by j_ℓ with $\ell = 0, \ldots, \kappa - 1$ the ℓ-th session opened in the main thread for which Sim obtains a valid second message (i.e., the values $\{r_i^{d_i}\}_{i\in[n]}$) from $\mathsf{R}^*_{\mathsf{soa}}$, and thus it has to extract the secret key in order to be able to compute the sub-commitments in such session.

$H_1^{j_{\ell+1}}$ **(for $\ell = 0, \ldots, \kappa - 1$):** 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 values $\{r_i^{d_i}\}_{i\in[n]}$ from $\mathsf{R}^*_{\mathsf{soa}}$, it 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 by Sim. 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 an issue 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. Note that $H_1^{j_0} = H_0$ and $H_1^{j_\kappa} = H_2^{j_0}$

$H_2^{j_\ell, s+1}$**: for $\ell = 1, \ldots, \kappa$; for $s = 0, \ldots, \mathsf{max}_{j_\ell} - 1$.** In hybrid $H_2^{j_\ell, s+1}$ Sim works as in experiment $H_2^{j_\ell, s}$ except that, in the $(s+1)$th *new* session started by $\mathsf{R}^*_{\mathsf{soa}}$ in the rewinding threads (recall that in each rewinding thread the view of $\mathsf{R}^*_{\mathsf{soa}}$ changes) for session j_ℓ, Sim 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: after having received the public keys from $\mathsf{R}^*_{\mathsf{soa}}$, it picks a random n-bit string $d_i, \ldots, d_n$ and an index i and it forwards $\mathsf{pk}_i^{\bar{d}_i}$ to the external sender $\mathsf{S}_{\mathsf{wTCom}}$.

Upon receiving the sub-commitment from $\mathsf{S}_{\mathsf{wTCom}}$ for the public key $\mathsf{pk}_i^{\bar{d}_i}$, $\mathsf{R}^*_{\mathsf{TC}}$ randomly chooses $n-1$ random bits $b_1, \ldots, b_n - 1$ and honestly executes $n-1$ sub-commitments using the remaining public parameters received from $\mathsf{R}^*_{\mathsf{soa}}$. Then it forwards all the sub-commitments to $\mathsf{R}^*_{\mathsf{soa}}$. Finally $\mathsf{R}^*_{\mathsf{TC}}$ forwards the output of the experiment to D_{soa} and outputs whatever D_{soa} outputs xored with $\bigoplus_{l \in [n], l \neq i} b_l$.

Now, let b_i such that $\bigoplus_{l \in [n]} b_l = \mathbf{b}[t]$, 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 wTCom, it holds hat $H_2^{j_\ell,s}$ 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** $\ell = 0, \ldots, \kappa - 1$**:** 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-commitments. 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 from sender $\mathsf{S}_{\mathsf{wTCom}}$ are computed using the honest or the fake algorithm. $\mathsf{R}^*_{\mathsf{wTCom}}$ works as follows: it 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, for $l \neq i$, while for the i-th sub-commitment it forwards the commitment received from $\mathsf{S}_{\mathsf{wTCom}}$. 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 from $\mathsf{S}_{\mathsf{wTCom}}$ in position i. Finally $\mathsf{R}^*_{\mathsf{wTCom}}$ 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 $\mathsf{R}^*_{\mathsf{wTCom}}$ wins the game $\mathbf{Exp}^{\mathsf{Trap/Com}}_{\mathsf{wTCom}}$ with advantage $\frac{\delta m}{k}$ that is still non-negligible, therefore breaking the trapdoor property of wTCom.

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 5. □

D.3 Proof of Theorem 2

Proof. The proof of completeness, binding, and hiding under selective opening attack of the $(3, 3)$-round protocol (Protocol 2) follow.

Completeness. It follows from the completeness of the sub-protocol for extractable commitment.

Binding. We now prove the binding property of SOACom using the statistical binding property of ExtCom (due to the ExtCom commitments played by $\mathsf{S}_{\mathsf{soa}}$) and the computational hiding property of ExtCom (due to the ExtCom commitments played by $\mathsf{R}_{\mathsf{soa}}$).

In the following we reduce the binding property of SOACom to the hiding property of ExtCom (due to the ExtCom commitments by $\mathsf{R}_{\mathsf{soa}}$) with the assumption that the underlying extractable commitment is statistically binding (due to the ExtCom commitments by $\mathsf{S}_{\mathsf{soa}}$).

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 transcript $\tau_{\mathtt{com}}$, 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 an efficient adversary $\mathsf{R}^*_{\mathsf{ext}}$ that breaks hiding of ExtCom, such that,

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

where, the negligible function $\mathsf{negl}(\cdot)$ corresponds to breaking binding of the statistically-binding extractable commitment.

We shall refer to any message msg sent in a decommitment phase $\tau^{\hat{j}}_{\mathtt{open}}$ as $\tau^{\hat{j}}_{\mathtt{open}}.\mathsf{msg}$, where $\hat{j} \in \{0,1\}$. We begin with a high-level sketch of $\mathsf{R}^*_{\mathsf{ext}}$.

$\underline{\mathsf{R}^*_{\mathsf{ext}}}$: We first give a simplified description of $\mathsf{R}^*_{\mathsf{ext}}$, then describe the technical issue that would arise, and then describe our solution to fix this issue.

We first observe that if $\mathsf{S}^*_{\mathsf{soa}}$ can somehow know $a = (a_1, \ldots, a_n)$ before sending the string 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 one is a commitment to 0 and the other is a commitment 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. In $\tau^1_{\mathtt{open}}$, 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, $\mathsf{S}^*_{\mathsf{soa}}$ can 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 favorable case first by extracting all the $2n$ commitments with some noticeable probability 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, $\mathsf{R}^*_{\mathsf{ext}}$ 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 all the extracted bits, $d = d_1, \ldots, d_n$, and all the a_is except a_m, with a non-negligible probability since the extracted bits are the same as what $\mathsf{S}^*_{\mathsf{soa}}$ would have opened to by extractability of ExtCom.

While this completes the high-level description of $\mathsf{R}^*_{\mathsf{ext}}$, we point out that we need to design $\mathsf{R}^*_{\mathsf{ext}}$ more carefully; this is because of the following fact that will lead to a technical issue: $\mathsf{R}^*_{\mathsf{ext}}$ cannot decide whether $\mathsf{S}^*_{\mathsf{soa}}$ would have broken binding of SOACom, had $\mathsf{R}^*_{\mathsf{ext}}$ continued the interaction. This is because $\mathsf{R}^*_{\mathsf{ext}}$ cannot proceed beyond the step where the sender sends d, as in the next step $\mathsf{R}^*_{\mathsf{ext}}$ is required to send the openings of all a_is (including a_m). Due to this fact, $\mathsf{R}^*_{\mathsf{ext}}$ fails to exploit the advantage of $\mathsf{S}^*_{\mathsf{soa}}$ if it proceeds the way as described in the simplified description above. We fix this issue with the following modification to the above version of $\mathsf{R}^*_{\mathsf{ext}}$: Instead of using all known a_is in the checks it makes, $\mathsf{R}^*_{\mathsf{ext}}$ chooses a random subset of size $n/2$ from the set of all known a_is and uses only these a_is in the checks.

The details follow.

$\underline{\mathsf{R}^*_{\mathsf{ext}}}$: $\mathsf{R}^*_{\mathsf{ext}}$ interacts with $\mathsf{S}^*_{\mathsf{soa}}$ in the commitment phase as follows.

1. Sample $m \xleftarrow{\$} [n]$.
2. For the extractable commitment of a_m, use the messages from the external sender in $\mathbf{Exp}^{\text{hiding-}a_m}_{\mathsf{Com},\mathsf{R}^*}$ and send them to $\mathsf{S}^*_{\mathsf{soa}}$. Also, messages from $\mathsf{S}^*_{\mathsf{soa}}$ corresponding to this extractable commitment are forwarded to the external sender.

3. To commit to the rest of the a_is, follow the honest receiver's code.

At any point in time till now, if $\mathsf{S}^*_{\mathsf{soa}}$ aborts, $\mathsf{R}^*_{\mathsf{ext}}$ outputs a random bit and halts. Otherwise, $\mathsf{R}^*_{\mathsf{ext}}$ interacts with $\mathsf{S}^*_{\mathsf{soa}}$ in the decommitment phase as follows.

1. Let $\mathsf{S}^*_{\mathsf{soa}}$ enter a decommitment phase, $\tau^{\hat{j}}_{\mathtt{open}}$, for some bit $\hat{j} \in \{0,1\}$. Once $\mathsf{S}^*_{\mathsf{soa}}$ sends the random string $\tau^{\hat{j}}_{\mathtt{open}}.d$ and completes the commitment phase of the extractable commitments, run $2n$ parallel invocations of the extractor E to extract the $2n$ bits $b^{(i,j)}$ committed to by $\mathsf{S}^*_{\mathsf{soa}}$ with an upper-bound on the number of rewindings for extraction to be $2P^2(n)$.
2. If extraction fails for any of the bits, then output a random bit and halt. Otherwise, perform the following checks in the order; if any check fails, then output a random bit 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. Choose a random subset $\mathcal{Q}_{\mathsf{check}}$ of $n/2$ indices $i \in [n]$ such that $i \neq m$; i.e., choose $\mathcal{Q}_{\mathsf{check}} \xleftarrow{\$} 2^{[n]}$ such that $|\mathcal{Q}_{\mathsf{check}}| = n/2$ and $m \notin \mathcal{Q}_{\mathsf{check}}$. Then, check if $\exists b_{\mathsf{same}} \in \{0,1\}$ such that $\forall i \in \mathcal{Q}_{\mathsf{check}}$, $j = \tau^{\hat{j}}_{\mathtt{open}}.d_i \oplus a_i$, $b^{(i,j)} = b_{\mathsf{same}}$.

   Let $\mathsf{CHECK\text{-}YES}$ denote the event that both the above checks go through. If $\mathsf{CHECK\text{-}YES}$ occurs, then output bit a'_m such that, for $j = \tau^{\hat{j}}_{\mathtt{open}}.d_m \oplus a'_m$, $b^{(m,j)} = b_{\mathsf{same}}$.

This completes the description of $\mathsf{R}^*_{\mathsf{ext}}$.

Let $q(n)$ be the probability that $\mathsf{S}^*_{\mathsf{soa}}$ does not abort before entering the opening phase. Note that $q(n) \geq \delta$. 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. In the following we bound the probability of $\mathsf{E.fail}$.

Note that the view of $\mathsf{S}^*_{\mathsf{soa}}$ 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,

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

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

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

Now consider an algorithm that interacts with $\mathsf{S}^*_{\mathsf{soa}}$ by running $\mathsf{R}_{\mathsf{soa}}$ (i.e., commits to all a_is by itself) except that it also tries to extract the $2n$ bits committed to by $\mathsf{S}^*_{\mathsf{soa}}$ (like $\mathsf{R}^*_{\mathsf{ext}}$). 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 $\mathtt{extracted} = \mathtt{opened}$.

Let $\mathcal{Q}_{\mathsf{agree}} \subset (2^{[n]} - \mathcal{Q}_{\mathsf{check}})$ such that $i \in \mathcal{Q}_{\mathsf{agree}}$ if and only if, for $j = \tau^{\hat{j}}_{\mathtt{open}}.d_i \oplus a_i$, $b^{(i,j)} = b_{\mathsf{same}}$. We have,

$$\begin{aligned}
&\mathsf{Pr}[\mathbf{Exp}^{\text{hiding-1}}_{\mathsf{ExtCom},\mathsf{R}^*_{\mathsf{ext}}}(n) \to 1)] \\
&\geq \mathsf{Pr}[\mathbf{Exp}^{\text{binding}}_{\mathsf{SOACom},\mathsf{S}^*_{\mathsf{soa}}}(n) \to 1 \wedge \textsf{CHECK-YES} \wedge \texttt{extracted} = \texttt{opened}] \cdot \mathsf{Pr}[a_m = 1] \cdot \mathsf{Pr}[\neg \mathsf{E.fail}] && (3)\\
&\quad + \mathsf{Pr}[\mathbf{Exp}^{\text{binding}}_{\mathsf{SOACom},\mathsf{S}^*_{\mathsf{soa}}}(n) \to 0 \wedge \textsf{CHECK-YES} \wedge m \in \mathcal{Q}_{\mathsf{agree}}] \cdot \mathsf{Pr}[a_m = 1] \cdot \mathsf{Pr}[\neg \mathsf{E.fail}] && (4)\\
&\quad + \mathsf{Pr}[\mathbf{Exp}^{\text{hiding-1}}_{\mathsf{ExtCom},\mathsf{R}^*_{\mathsf{ext}}}(n) \to 1 | \neg\textsf{CHECK-YES} \vee \mathsf{E.fail}] \cdot \mathsf{Pr}[a_m = 1] \cdot \mathsf{Pr}[\neg \textsf{CHECK-YES}] \cdot \mathsf{Pr}[\mathsf{E.fail}] && (5)
\end{aligned}$$

and

$$\begin{aligned}
&\mathsf{Pr}[\mathbf{Exp}^{\text{hiding-0}}_{\mathsf{ExtCom},\mathsf{R}^*_{\mathsf{ext}}}(n) \to 1] \\
&\leq \mathsf{Pr}[\textsf{CHECK-YES} \wedge \neg(\texttt{extracted} = \texttt{opened})] \cdot \mathsf{Pr}[a_m = 0] \cdot \mathsf{Pr}[\neg \mathsf{E.fail}] && (6)\\
&\quad + \mathsf{Pr}[\mathbf{Exp}^{\text{binding}}_{\mathsf{SOACom},\mathsf{S}^*_{\mathsf{soa}}}(n) \to 0 \wedge \textsf{CHECK-YES} \wedge m \notin \mathcal{Q}_{\mathsf{agree}}] \cdot \mathsf{Pr}[a_m = 0] \cdot \mathsf{Pr}[\neg \mathsf{E.fail}] && (7)\\
&\quad + \mathsf{Pr}[\mathbf{Exp}^{\text{hiding-0}}_{\mathsf{ExtCom},\mathsf{R}^*_{\mathsf{ext}}}(n) \to 1 | \neg\textsf{CHECK-YES} \vee \mathsf{E.fail}] \cdot \mathsf{Pr}[a_m = 0] \cdot \mathsf{Pr}[\neg \textsf{CHECK-YES}] \cdot \mathsf{Pr}[\mathsf{E.fail}] && (8)
\end{aligned}$$

Thus,

$$\mathbf{Adv}^{\text{hiding}}_{\mathsf{ExtCom},\mathsf{R}^*_{\mathsf{ext}}} \geq |Term(3) + Term(4) + Term(5) - Term(6) - Term(7) - Term(8)|$$

Since $\mathsf{R}^*_{\mathsf{ext}}$ outputs a random bit if $\neg$CHECK-YES$\vee$E.fail occurs, $Term(5) = Term(8)$. By statistical extractability of ExtCom, the event $\neg(\texttt{extracted} = \texttt{opened})$ is statistically impossible, and hence $Term(6)$ is negligible. Also, note that by statistical binding of ExtCom, $\mathbf{Exp}^{\text{binding}}_{\mathsf{SOACom},\mathsf{S}^*_{\mathsf{soa}}}(n) \to 1 \wedge \neg\textsf{CHECK-YES}$ occurs with only negligible probability. Thus we have that, in $Term(3)$, $\mathsf{Pr}[\mathbf{Exp}^{\text{binding}}_{\mathsf{SOACom},\mathsf{S}^*_{\mathsf{soa}}}(n) \to 1 \wedge \textsf{CHECK-YES} \wedge \texttt{extracted} = \texttt{opened}]$ is negligibly close to $\mathsf{Pr}[\mathbf{Exp}^{\text{binding}}_{\mathsf{SOACom},\mathsf{S}^*_{\mathsf{soa}}}(n) \to 1]$.

Hence, we will be done once we prove the following claim:

Claim 1. *There exists a negligible function* negl() *such that* $Term(4) + \mathsf{negl}(n) \geq Term(7)$.

Proof. Without loss of generality, $\mathsf{Pr}[a_m = 0] = \mathsf{Pr}[a_m = 1] = \frac{1}{2}$.

Let $p = \frac{1}{2} \cdot \mathsf{Pr}[\neg\mathsf{E.fail}]$.

$$\begin{aligned}
&\left(\frac{1}{p}\right) \cdot (Term(4) - Term(7)) \\
&= \mathsf{Pr}[\mathbf{Exp}^{\text{binding}}_{\mathsf{SOACom},\mathsf{S}^*_{\mathsf{soa}}}(n) \to 0 \wedge \textsf{CHECK-YES} \wedge m \in \mathcal{Q}_{\mathsf{agree}} \wedge (|\mathcal{Q}_{\mathsf{agree}}| \geq \left(\frac{n}{4} + 1\right))] \\
&\quad + \mathsf{Pr}[\mathbf{Exp}^{\text{binding}}_{\mathsf{SOACom},\mathsf{S}^*_{\mathsf{soa}}}(n) \to 0 \wedge \textsf{CHECK-YES} \wedge m \in \mathcal{Q}_{\mathsf{agree}} \wedge (|\mathcal{Q}_{\mathsf{agree}}| < \left(\frac{n}{4} + 1\right))] && (9)\\
&\quad - \mathsf{Pr}[\mathbf{Exp}^{\text{binding}}_{\mathsf{SOACom},\mathsf{S}^*_{\mathsf{soa}}}(n) \to 0 \wedge \textsf{CHECK-YES} \wedge m \notin \mathcal{Q}_{\mathsf{agree}} \wedge (|\mathcal{Q}_{\mathsf{agree}}| \geq \left(\frac{n}{4} + 1\right))] \\
&\quad - \mathsf{Pr}[\mathbf{Exp}^{\text{binding}}_{\mathsf{SOACom},\mathsf{S}^*_{\mathsf{soa}}}(n) \to 0 \wedge \textsf{CHECK-YES} \wedge m \notin \mathcal{Q}_{\mathsf{agree}} \wedge (|\mathcal{Q}_{\mathsf{agree}}| < \left(\frac{n}{4} + 1\right))]
\end{aligned}$$

Ignoring $Term(9)$, we have

$$\left(\frac{1}{p}\right) \cdot (Term(4) - Term(7))$$

$$\geq \mathsf{Pr}[\mathbf{Exp}^{\text{binding}}_{\mathsf{SOACom},\mathsf{S}^*_{\mathsf{soa}}}(n) \to 0 \wedge \mathsf{CHECK\text{-}YES} \wedge m \in \mathcal{Q}_{\mathsf{agree}} \wedge (|\mathcal{Q}_{\mathsf{agree}}| \geq \left(\frac{n}{4}+1\right))] \tag{10}$$

$$- \mathsf{Pr}[\mathbf{Exp}^{\text{binding}}_{\mathsf{SOACom},\mathsf{S}^*_{\mathsf{soa}}}(n) \to 0 \wedge \mathsf{CHECK\text{-}YES} \wedge m \notin \mathcal{Q}_{\mathsf{agree}} \wedge (|\mathcal{Q}_{\mathsf{agree}}| \geq \left(\frac{n}{4}+1\right))] \tag{11}$$

$$- \mathsf{Pr}[\mathbf{Exp}^{\text{binding}}_{\mathsf{SOACom},\mathsf{S}^*_{\mathsf{soa}}}(n) \to 0 \wedge \mathsf{CHECK\text{-}YES} \wedge m \notin \mathcal{Q}_{\mathsf{agree}} \wedge (|\mathcal{Q}_{\mathsf{agree}}| < \left(\frac{n}{4}+1\right))] \tag{12}$$

We note that, if both the events CHECK-YES and $|\mathcal{Q}_{\mathsf{agree}}| \geq \left(\frac{n}{4}+1\right)$ occur, then,

$$\mathsf{Pr}[m \in \mathcal{Q}_{\mathsf{agree}}] \geq \frac{\left(\frac{n}{4}+1\right)}{\frac{n}{2}} \tag{13}$$

$$\mathsf{Pr}[m \notin \mathcal{Q}_{\mathsf{agree}}] \leq 1 - \frac{\left(\frac{n}{4}+1\right)}{\frac{n}{2}} \tag{14}$$

From the bounds (13) and (14), $Term(10) - Term(11) \geq 0$.

Now, to complete the proof of Claim 1, it remains to prove that $Term(12)$ is negligible. For this, roughly, we argue that, for some $\mathcal{Q}_{\mathsf{check}}$ if CHECK-YES occurs and $|\mathcal{Q}_{\mathsf{agree}}| < \left(\frac{n}{4}+1\right)$, then for "most other" sets $\mathcal{Q}'_{\mathsf{check}}$ (chosen as per Check 2 of $\mathsf{R}^*_{\mathsf{ext}}$), $\exists i \in \mathcal{Q}'_{\mathsf{check}}$ such that $i \notin \mathcal{Q}_{\mathsf{check}} \cup \mathcal{Q}_{\mathsf{agree}}$. For all such $\mathcal{Q}'_{\mathsf{check}}$, CHECK-YES does not occur. Since only such $\mathcal{Q}'_{\mathsf{check}}$ are chosen with all but negligible probability, we have that Term (12) is negligible. More concretely:

$$\begin{aligned}
&\mathsf{Pr}[\mathsf{CHECK\text{-}YES} \wedge |\mathcal{Q}_{\mathsf{agree}}| < \left(\frac{n}{4}+1\right)] \\
&\quad = \mathsf{Pr}_{\mathcal{Q}'_{\mathsf{check}}}[\mathcal{Q}'_{\mathsf{check}} \cap ([n] - (\mathcal{Q}_{\mathsf{check}} \cup \mathcal{Q}_{\mathsf{agree}})) = \{\}] \\
&\quad = \frac{(3n/4)!\ (n/2)!}{(n/4)!\ (n)!} \\
&\quad = \frac{\prod_{i=1}^{n/4}(n/4+i)}{\prod_{i=1}^{n/4}(3n/4+i)} \\
&\quad \leq (1/2)^{(n/4)}
\end{aligned}$$

which is negligible in n. The last inequality follows from the fact that, for any $i \in [n/4]$, the fraction $\frac{(n/4+i)}{(3n/4+i)} \leq \frac{1}{2}$. To see this, observe that $\frac{(n/4+i)}{(3n/4+i)} \leq \frac{1}{2}$ iff $(n/2+2i) \leq (3n/4+i)$ iff $i \leq n/4$ (which is the case we are interested in).

This completes the proof of Claim 1 and hence the proof of binding of SOACom given by Protocol 2.

□

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 first interacts with $\mathsf{R}^*_{\mathsf{soa}}$ in the commitment phase of each 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. At the completion of commitment phases of all sessions, in each session, Sim extracts $a_1, \ldots, a_n$ committed to by $\mathsf{R}^*_{\mathsf{soa}}$. Further, once it receives the bits to which it needs to open, it can equivocate in the opening phase by carefully crafting the string d (using the knowledge of extracted $a_1, \ldots, a_n$) so that the outcome of coin-flipping points to the n commitments of 0 (resp., 1) if the bit it needs to open to is 0 (resp., 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)

- In session $j \in [k]$:
  1. Sample $\sigma' \xleftarrow{\$} \{0,1\}^n$.
  2. $\forall\ i \in [n]$, set $b^{(i,j)} := 0$ for $j = \sigma'_i$ and set $b^{(i,j)} := 1$ for $j = 1 \oplus \sigma'_i$.
  3. Interact with $\mathsf{R}^*_{\mathsf{soa}}$ as per the commitment phase of the protocol except for the choice of $b^{(i,j)}$ as above (This is unlike the honest sender which sets every $b^{(i,j)}$ to the same bit, namely the bit it wishes to commit to).
  4. If $\mathsf{R}^*_{\mathsf{soa}}$ aborts, then abort session j.

Commitment phase completion. When the commitment phases of all k sessions are completed, Sim obtains the set of indices $I = \{j_1, j_2, \ldots\}$, where $|I| \leq m$, from $\mathsf{R}^*_{\mathsf{soa}}$ and obtains $\{\mathbf{b}[j_\ell]\}_{j_\ell \in I}$ from the experiment.

Extraction phase. (rewinding threads)

$\forall j_\ell \in I$ if the j_ℓth session was not-aborted in the main thread, then:

- For $c = 1, \ldots, n$: Run the extractor E of the underlying extractable commitment 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 a_c, 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:

- Follow the decommitment phase of the protocol except for the following change. 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 $\mathsf{R}^*_{\mathsf{soa}}$.

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 of 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 not aborted, Sim runs the extractor n times to extract $a = (a_1, \ldots, a_n)$.

Let us denote by $\zeta^{j_\ell} = \zeta(\mathsf{ran}_\mathsf{R}, j_\ell)$ the probability that $\mathsf{R}^*_{\mathsf{soa}}$, initialized with randomness ran_R, successfully completes the commitment phase of the j_ℓ-th session. While running the extractor, 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 ζ^{j_ℓ}. As noted in the analysis of the extractor E of ExtCom, its expected running time is $\zeta^{j_\ell} \cdot 1/\zeta^{j_\ell}$. Further, in each rewinding performed by the extractor, 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 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 the extractor, expected running time of Sim in the extraction phase is:

$$\sum_{\ell=1}^{m} \zeta^{j_\ell} \left[\frac{1}{\zeta^{j_\ell}} \cdot \mathsf{t}_{\mathtt{rew}} \right] = \mathtt{poly}(n)$$

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:

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

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* $\mathsf{S}_{\mathsf{soa}}$.

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

Consider the following sequence of hybrids:

H_1**:** 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_2**:** This experiment is same as H_1, except that Sim also extracts each of the bits $a_1, \ldots, a_n$ using the extractor 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 proceeds the same way as in the commitment phase of the main threads, but does not extract $a_1, \ldots, a_n$ for these new sessions.) 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 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 $\mathsf{R}^*_{\mathsf{soa}}$ in the opening phase.

H_2 and H_3 are statistically close since an extracted bit is inconsistent with the opening only with negligible probability by the (statistical) extractability of ExtCom.

$H_4^{s,i}$ **for** $s = 0, \ldots, k$ **and** $i = 0, \ldots, n$: $H_4^{s,i}$ differs from H_3 as follows:

- For every s'-th session, where $s' > s$, Sim behaves the same way as in H_3.
- For every s'-th session, where $0 \leq s' < s$, Sim chooses the bits to be committed 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 $j' = 1 - \sigma'_i$. After the commitment phases of all the sessions, if $\mathsf{R}^*_{\mathsf{soa}}$ requests for the decommitment of the s'-th session, then extract $a = a_1, \ldots, a_n$ as in H_3, and also, d is set as $d := a \oplus \sigma' \oplus \mathbf{b}[s']$. The rest of the messages are computed the same way as in H_3.
- In the s-th session, $\forall i' > i$, Sim computes the i'th pair of commitments the same way as in H_3. For every i'th pair of commitments, where $1 \leq i' \leq i$, Sim chooses $\sigma'_i \xleftarrow{\$} \{0,1\}$ and sets $b^{(i',j')} := 0$ for $j' = \sigma'_{i'}$, $b^{(i',j')} := 1$ for $j' = 1 - \sigma'_{i'}$. Also, after the commitment phases of all the sessions, if $\mathsf{R}^*_{\mathsf{soa}}$ requests for the decommitment of the s-th session, then extract $a = a_1, \ldots, a_i$ as in H_3, and set $d_{i'} := a_{i'} \oplus \sigma'_{i'} \oplus \mathbf{b}[s]$. The rest of the messages are computed the same way as in H_3.

Let $\preceq$ be a total ordering on the set $T = \{s,i\}_{1 \leq s \leq k, 1 \leq i \leq n} \cup \{(0,0)\}$ such that $(s',i') < (s,i)$ iff $s' < s$ OR ($s' = s$ AND $i' < i$). Note the the hybrids $H_4^{0,0}$ and H_3 are identical. We now show that if there exists a distinguisher D_{soa} that distinguishes any two consecutive hybrids $H_4^{s',i'}$ and $H_4^{s,i}$, where $(s',i') < (s,i)$, 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 $H_4^{s,i}$, except for the following change. If $\mathbf{b}[s] = 0$, then commit to $b^{(i,j)}$ for $j = 1 - \sigma'_i$ using the interaction from the external challenger in $\mathbf{Exp}^{\text{hiding-}b}_{\mathsf{ExtCom},\mathsf{R}^*_{\mathsf{ext}}}(n)$. (Note that $\mathsf{R}^*_{\mathsf{ext}}$ will not be asked to provide opening for this commitment since $j = 1 - \sigma'_i$.) The rest of the messages are computed the same way as in $H_4^{s,i}$. On the other hand, if $\mathbf{b}[s] = 1$, then commit to $b^{(i,j)}$ for $j = \sigma'_i$ using the interaction from the external challenger in $\mathbf{Exp}^{\text{hiding-}b}_{\mathsf{ExtCom},\mathsf{R}^*_{\mathsf{ext}}}(n)$. (Similarly, in this case too, $\mathsf{R}^*_{\mathsf{ext}}$ will not be asked to provide opening for this commitment since $j = \sigma'_i$.) Again, rest of the messages are computed the same way as in $H_4^{s,i}$. Finally, when D_{soa} outputs a bit b', then $\mathsf{R}^*_{\mathsf{ext}}$ outputs $b' \oplus \mathbf{b}[s]$.

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 $b \neq \mathbf{b}[s]$ then $\mathsf{R}^*_{\mathsf{ext}}$ is running $H_4^{s,i}$ with $\mathsf{R}^*_{\mathsf{soa}}$; otherwise, it is running $H_4^{s',i'}$. Thus, $\mathsf{R}^*_{\mathsf{ext}}$ breaks hiding of ExtCom with the same probability that D_{soa} distinguishes between $H_4^{s',i'}$ and $H_4^{s,i}$.

Note that the final hybrid is identical to the described simulator. This completes the proof of indistinguishability of the outputs the real and the simulated experiments. □

□

D.4 Proof of Theorem 6

Proof. In the following, we prove completeness, binding and hiding of the $(5,1)$ round protocol presented in Protocol 5.

Completeness. It follows from the completeness of the sub-protocol for extractable commitment.

Binding. The proof of binding follows on the same lines as the $(3,3)$ round protocol presented in 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. Like the simulator for Protocol 2, the simulator, in every pair, commits to 0 in one commitment and to 1 in the other (unlike $\mathsf{S}_{\mathsf{soa}}$ which commits to the same bit in all its n pairs of commitments). Next, 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 round in which the sender sends d is in the commitment phase for this protocol, the simulator needs to send d (and complete the entire commitment phase) in every session before it receives the bits, the same strategy as in the Protocol 2 cannot work directly. Instead, here, we exploit the flexibility of the sender in choosing either the outcome of coin-flipping or its binary-negation to dictate which commitments it finally opens (i.e., the sender first chooses either the outcome of the outcome of coin-flipping or its binary-negation, and the i-th bit of it dictates whether to open the 0-th or the 1-st commitment in the i-th pair). 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 the commitments of θ. Then, once it receives the bits to which it needs to open, it equivocate in the opening phase by choosing between the outcome of coin-flipping and its binary-negation depending on the value of 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 (S1a). (Main thread)

In session $j \in [k]$:

1. Sample $\sigma' \xleftarrow{\$} \{0,1\}^n$.
2. $\forall\, i \in [n]$, set $b^{(i,j)} := 0$ for $j = \sigma'_i$ and set $b^{(i,j)} := 1$ for $j = 1 \oplus \sigma'_i$.
3. Interact with $\mathsf{R}^*_{\mathsf{soa}}$ as per the commitment phase of the protocol until the point just before it needs to send d, except for the choice of $b^{(i,j)}$ as above (This is unlike the honest sender which sets all $b^{(i,j)}$ to the bit it wishes to commit to).
4. If $\mathsf{R}^*_{\mathsf{soa}}$ aborts, then abort session j. Otherwise, execute the extraction phase of the j-th session as described below.

Extraction phase. (Rewinding threads)

If the jth session was not aborted in (**S1a**), then:

- For $c = 1, \ldots, n$: Run the extractor of the underlying extractable commitment scheme in order to extract a_c. During this extraction all the other messages of the same session before sending d 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) before sending d are computed as in (**S1a**) and the rest of the messages are computed the same way as in the protocol.

- If E fails to extract any a_c, then abort the simulation and halt. Otherwise, continue with the interaction in the main thread as described below.

Commitment phase continued (S1b). (Main thread)

- Continue with the commitment phase of the jth session by proceeding as follows: Sample $\theta' \xleftarrow{\$} \{0,1\}$ and set $d := \theta' \oplus \sigma' \oplus a$. Rest of the messages are computed the same way as in the protocol. Also, check whether the bits opened to by $\mathsf{R}^*_{\mathsf{soa}}$ in the main thread are equal to the extracted bits $a_1, \ldots, a_n$; if not, then abort the simulation. 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 $I = \{j_1, j_2, \ldots\}$, where $|I| \leq m$, from $\mathsf{R}^*_{\mathsf{soa}}$ and obtains $\{\mathbf{b}[j_\ell]\}_{j_\ell \in I}$ 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 $\sigma := \theta \oplus d \oplus a$. The rest of the messages are computed the same way as in the protocol.

Finally, output the commitment phase transcripts from (**S1a**) and (**S1b**) as well as 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_1**:** 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_2**:** This experiment is same as H_1, except that Sim also extracts each of the bits $a_1, \ldots, a_n$ using the extractor for every non-aborted session. It launches these extractors just before it needs to send d in that session. During extraction, if new sessions are initiated by $\mathsf{R}^*_{\mathsf{soa}}$ in the rewinding threads, then Sim proceeds the same way as in the commitment phase of the main threads, but does not extract $a_1, \ldots, a_n$ for these new sessions. 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 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 $\mathsf{R}^*_{\mathsf{soa}}$ in the opening phase.

H_2 and H_3 are statistically close since an extracted bit is inconsistent with the opening only with negligible probability by the (statistical) extractability of ExtCom.

$H_4^{s,i}$ **for** $s = 0, \ldots, k$ **and** $i = 0, \ldots, n$: $H_4^{s,i}$ differs from H_3 as follows:

- For every s'-th session, where $s' > s$, Sim behaves the same way as in H_3.
- For every s'-th session, where $0 \leq s' < s$, Sim chooses the bits to be committed 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 $j' = 1 - \sigma'_i$. After the commitment phases of all the sessions, if $\mathsf{R}^*_{\mathsf{soa}}$ requests for the decommitment of the s'-th session, then extract $a = a_1, \ldots, a_n$ as in H_3, and also, sample $\theta' \xleftarrow{\$} \{0,1\}$, 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_3.
- In the s-th session, sample $\theta' \xleftarrow{\$} \{0,1\}$. $\forall i' > i$, Sim computes the i'th pair of commitments the same way as in H_3. For every i'th pair of commitments, where $1 \leq i' \leq i$, Sim chooses $\sigma'_{i'} \xleftarrow{\$} \{0,1\}$, sets $b^{(i',j')} := 0$ for $j' = \sigma'_{i'}$ and $b^{(i',j')} := 1$ for $j' = 1 - \sigma'_{i'}$. Also, after the commitment phases of all the sessions, if $\mathsf{R}^*_{\mathsf{soa}}$ requests for the decommitment of the s-th session, then it extracts $a = a_1, \ldots, a_i$ as in H_3, and sets $d_{i'}$ as $d_{i'} := a_{i'} \oplus \sigma'_{i'} \oplus \theta'$ and $\theta := \theta' \oplus \mathbf{b}[s]$. The rest of the messages are computed the same way as in H_3.

Let $\preceq$ be a total ordering on the set $T = \{s, i\}_{1 \leq s \leq k, 1 \leq i \leq n} \cup \{(0,0)\}$ such that $(s', i') < (s, i)$ iff $s' < s$ OR ($s' = s$ AND $i' < i$). Note the the hybrids $H_4^{0,0}$ and H_3 are identical. We now show that if there exists a distinguisher D_{soa} that distinguishes any two consecutive hybrids $H_4^{s',i'}$ and $H_4^{s,i}$, where $(s', i') < (s, i)$, then we construct an efficient adversary $\mathsf{R}^*_{\mathsf{ext}}$ that breaks hiding of the extractable commitment. $\mathsf{R}^*_{\mathsf{ext}}$ interacts with $\mathsf{R}^*_{\mathsf{soa}}$ the same way as Sim in $H_4^{s,i}$, except for the following change. In the sth session: Choose $\sigma'_i \xleftarrow{\$} \{0,1\}$, set $d_i := a_i \oplus \sigma'_i \oplus \mathbf{b}[s]$. If $\mathbf{b}[s] = 0$, commit to $b^{(i,j)}$ for $j = 1 - \sigma'_i$ using the interaction from the external challenger in $\mathbf{Exp}^{\text{hiding-}b}_{\mathsf{ExtCom},\mathsf{R}^*_{\mathsf{ext}}}(n)$. (Note that $\mathsf{R}^*_{\mathsf{ext}}$ will not be asked to provide opening for this commitment since $j = 1 - \sigma'_i$.) The rest of the messages are computed the same way as in $H_4^{s,i}$. On the other hand, if $\mathbf{b}[s] = 1$, then commit to $b^{(i,j)}$ for $j = \sigma'_i$ using the interaction from the external challenger in $\mathbf{Exp}^{\text{hiding-}b}_{\mathsf{ExtCom},\mathsf{R}^*_{\mathsf{ext}}}(n)$. (Similarly, in this case too, $\mathsf{R}^*_{\mathsf{ext}}$ will not be asked to provide opening for this commitment since $j = \sigma'_i$.) Again, rest of the messages are computed the same way as in $H_4^{s,i}$. Finally, when D_{soa} outputs a bit b', then $\mathsf{R}^*_{\mathsf{ext}}$ outputs $b' \oplus \mathbf{b}[s]$.

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 $b \neq \mathbf{b}[s]$ then $\mathsf{R}^*_{\mathsf{ext}}$ is running $H_4^{s,i}$ with $\mathsf{R}^*_{\mathsf{soa}}$; otherwise, it is running $H_4^{s',i'}$. Thus, $\mathsf{R}^*_{\mathsf{ext}}$ breaks hiding of ExtCom with the same probability that D_{soa} distinguishes between $H_4^{s',i'}$ and $H_4^{s,i}$.

Note that the final hybrid is identical to the described simulator. This completes the proof of indistinguishability of the outputs the real and the simulated experiments.

□

□

D.5 Proof of Theorem 3

Proof. In the following we prove the impossibility of fully concurrent SOA-security. In our proof we assume the existence of OWFs, which is implied by the existence of any commitment scheme. Let $\Pi = (\mathsf{S}, \mathsf{R})$ be a r-round string-commitment protocol that is SOA-secure (with a black-box simulation strategy) 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 sub-routine to contradict in strict polynomial time some hardness assumptions. Since Sim is only expected polynomial time, we are implicitly 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 in that polynomial 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 pair of adversaries $\mathsf{R}_0^*, \mathsf{R}_1^*$ and we show that such adversaries make the rewinding strategy of any black-box Sim ineffective for one protocol execution. Still, by the concurrent SOA-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^*, for $p = 0, 1$. 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 use a signature scheme. A signature scheme is a tuple 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 $(vk, sk) \overset{\$}{\leftarrow} \mathsf{Gen}(1^n)$, 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 new pair (x^*, σ^*) such that $\mathsf{Vrfy}(vk, x^*, \sigma^*) = 1$ without knowing sk. We define our set $\mathcal{B} = \{\mathcal{B}_{sk}\}_{sk \overset{\$}{\leftarrow} \mathsf{Gen}(1^n)}$ where:

$$\mathcal{B}_{sk} = \{\sigma_1, \ldots, \sigma_k : \sigma_i = \mathsf{Sign}(sk, i, r), \text{for } r \in \{0,1\}^n\}$$

Thus we define the message space as a set of signatures under a secret key sk, in particular the message committed to 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. R_p^*, for $p = 0, 1$ runs $k = r(2n) + 1$ protocol executions, where r is the number of rounds of protocol Π. 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 $F_k : \{0,1\}^* \rightarrow \{0,1\}^n$ be a PRF for $k \overset{\$}{\leftarrow} \{0,1\}^n$.

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 an 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_p^* 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 i^{th} 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^* asks for the opening with probability p. In Fig. 1 we show the diagram of the schedule.

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 2. *For every i let Q_i be the set of all queries sent by Sim to R_p^* during the ideal world execution which includes 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.

[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 i^{th} message of Π_1 that is sent by R^*, (i.e., $\Pi_1(i)_\mathsf{R}$.)

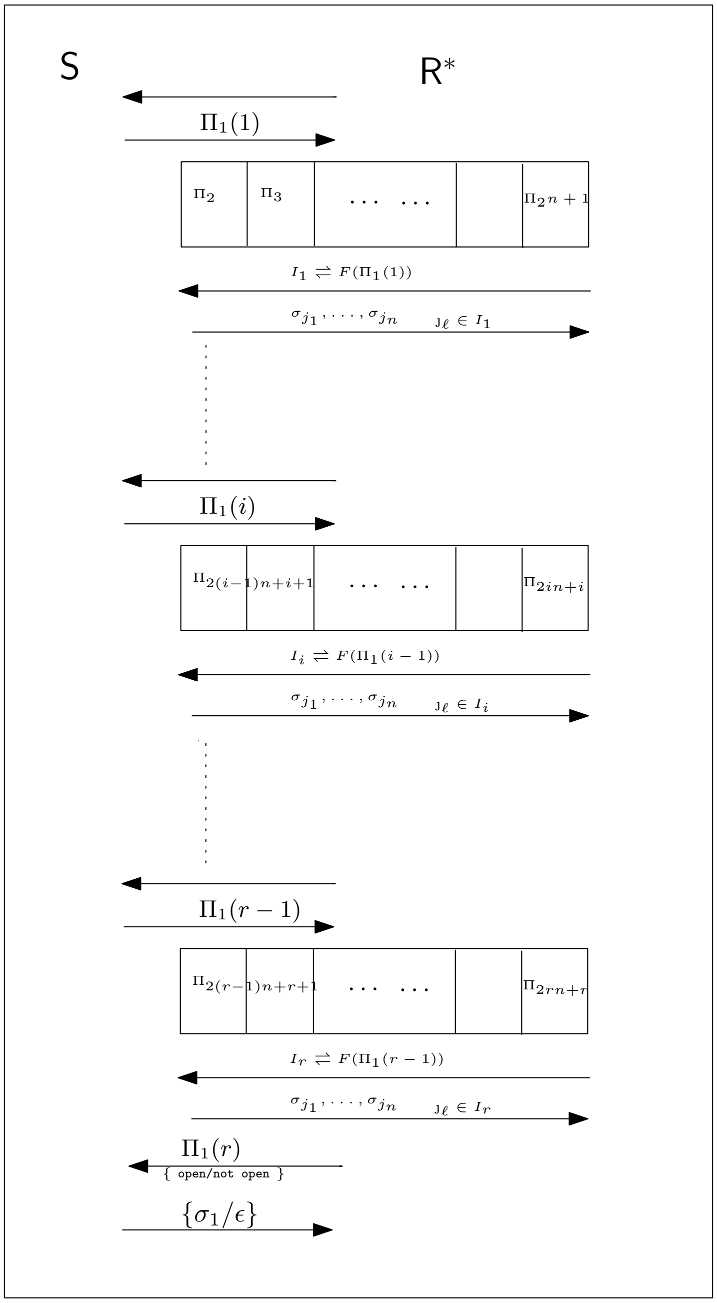
S
R*
$\Pi_1(1)$
Π_2
Π_3
$\ldots\ \ldots$
$\Pi_2 n+1$
$I_1 \rightleftharpoons F(\Pi_1(1))$
$\sigma_{j_1},\ldots,\sigma_{j_n} \quad j_\ell \in I_1$
$\Pi_1(i)$
$\Pi_{2(i-1)n+i+1}$
$\ldots\ \ldots$
Π_{2in+i}
$I_i \rightleftharpoons F(\Pi_1(i-1))$
$\sigma_{j_1},\ldots,\sigma_{j_n} \quad j_\ell \in I_i$
$\Pi_1(r-1)$
$\Pi_{2(r-1)n+r+1}$
$\ldots\ \ldots$
Π_{2rn+r}
$I_r \rightleftharpoons F(\Pi_1(r-1))$
$\sigma_{j_1},\ldots,\sigma_{j_n} \quad j_\ell \in I_r$
$\Pi_1(r)$
{ open/not open }
$\{\sigma_1/\epsilon\}$

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

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 ask exactly n signatures, according to R^*_p strategy (indeed the strategy of R^*_p is such that R^*_0 always asks exactly n openings 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 a total of at least $n+1$ valid signatures to R^*_p, still by 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 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 exists at least one 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 2 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 3. *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, 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 very 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 2.

□

Claim 4. *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 $(vk, sk) \overset{\$}{\leftarrow} \mathsf{Gen}(1^n)$, 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 receiver R. That is, let q be an oracle query from Sim to R_1^* such that R_1^*'s 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 $(\Pi_1(i)_\mathsf{S})$ of the simulator from q, and then, if R has already sent the response $\Pi_1(i)_\mathsf{R}$ according to the extracted message, then R_p^* replies 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 reply 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^* since she plays the first execution of Π as an honest receiver.

By Claim 2 we have that the event *failure* happens only with negligible probability. By Claim 3 we have that, except with negligible probability, Sim makes the oracle query for the opening of execution Π_1 only after the commitment phase has been completed. By Claim 4 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., $\langle \mathsf{S}^i_\Gamma(\texttt{com}, m_i), \mathsf{R}^i_\Gamma(\texttt{recv})\rangle$ 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., $\langle \mathsf{S}^i_\Gamma(\texttt{open}), \mathsf{R}^i_\Gamma(\texttt{open})\rangle$ 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. $\square$

$\square$

E Further Potential Issues in the Proof of Theorem 3.3 of [Xia11a]

The proof of Theorem 3.3. in [Xia11a] also 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] can not 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 can not 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 adaptively based on the commitment-phase transcript. However for the non-aborted session the simulator is still required to carry-out the simulation. In fact, as also explained in [CO99] (see paragraph "The Simulator Sim", Section 5.1), the simulator in [CO99] first checks whether the verifier decommits all the commitments in a proper way. If this is the case, then the simulator continues the simulation; otherwise, namely, if there exists at least one commitment that is not opened properly by the verifier, simulator outputs the transcript seen until then and halts. Therefore the above simulator works only against a non-aborting concurrent verifier, 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 [Xia11a], 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.

Subsequently to announcement of our results, a different and more involved simulation strategy has been presented in [Xia12b].

F Application to Concurrent Zero-Knowledge with Pre-processing

As an interesting application of SOA-secure commitment schemes, we present a construction of cZK with pre-processing. We first present a construction that uses in a black-box manner any SOA-secure commitment scheme with non-interactive opening phase. This construction also uses the 3-round FLS protocol defined in Appendix A.2.1 as a main ingredient. If the underlying SOA-secure scheme is statistically binding, then resulting protocol is a cZK proof system with pre-processing, while, if it is only computationally binding, then resulting protocol is a cZK argument system with pre-processing.

As a corollary, due to our $(3, 1)$-round SOA-secure computationally binding scheme based on NBB use of OWPs, we have a cZK argument system with pre-processing, where the pre-processing takes 3 rounds and the proof phase is non-interactive.

The construction $(\mathcal{P}, \mathcal{V})$ follows below as Protocol 6 for a language L.

Let $\mathsf{SOACom} = (\mathsf{S_{soa}}, \mathsf{R_{soa}})$ be a SOA-secure commitment scheme with non-interactive opening phase. Let G be a pseudo-random generator (PRG). 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 $\Lambda_{L,\mathtt{G}}$: $(x, y) \in \Lambda_{L,\mathtt{G}}$ if $x \in L$ OR there exists a seed s such that $y = G(s)$.

Protocol 6. *[$(\mathcal{P}, \mathcal{V})$]*

Pre-processing phase.

Common input: 1^n.

$\mathcal{P}1$: *1. Sample* $\sigma_P \xleftarrow{\$} \{0,1\}^{f(n)}$.
2. Run $\langle \mathsf{S_{soa}}(\mathtt{com}, \sigma_P), \mathsf{R_{soa}}(\mathtt{recv})\rangle$ *with* $\mathcal{V}1$*; if* $\mathsf{S_{soa}}$ *aborts, then abort.*
3. Run $\mathsf{FLS1}$ *with* $\mathcal{V}1$.

$\mathcal{V}1$: *1. If* $\mathsf{R_{soa}}$ *aborts, then abort; otherwise, sample* $\sigma_V \xleftarrow{\$} \{0,1\}^{f(n)}$ *and send it to* $\mathcal{P}1$.
2. Run $\mathsf{FLS2}$ *with* $\mathcal{P}1$.

Set $\mathsf{state}_{\mathcal{P}} = \mathsf{state}_S$ *and* $\mathsf{state}_{\mathcal{V}} = (\sigma_V, \mathsf{state}_R)$*, where* state_S *and* state_R *are the states of* $\mathsf{S_{soa}}$ *and* $\mathsf{R_{soa}}$*, respectively, after the commitment phase.*

Proof Phase.

Common input: $x \in L$.
$\mathcal{P}2's$ **input:** $w \in \mathcal{R}_L(x), \mathsf{state}_{\mathcal{P}}$.
$\mathcal{V}2's$ **input:** $\mathsf{state}_{\mathcal{V}}$.

$\mathcal{P}2$: *If the pre-processing phase is successfully completed, then:*

1. *run* $\langle \mathsf{S}_{\mathsf{soa}}(\mathsf{open}), \mathsf{R}_{\mathsf{soa}}(\mathsf{open})\rangle$ *with* $\mathcal{V}2$*;*
2. *send* $\sigma = \sigma_P \oplus \sigma_V$ *to* $\mathcal{V}2$*;*
3. *run* FLS3 *for the theorem* $(x, \sigma) \in \Lambda_{L,\mathtt{G}}$ *using the witness* w *with* $\mathcal{V}2$.

$\mathcal{V}2$: *Check whether* $\mathsf{R}_{\mathsf{soa}}$ *did not abort in the opening phase,* FLS3 *is accepting, and* $\sigma = \sigma'_P \oplus \sigma_V$*, where* σ'_P *is the string received correctly by* $\mathsf{R}_{\mathsf{soa}}$*. If so, then output* accept*; otherwise, output* $\perp$.

Assuming that FLS is a special $\mathcal{WIP}oK$ for $\Lambda_{L,\mathtt{G}}$, in the following, we prove that if SOACom is a statistically binding SOA-secure commitment scheme, then $(\mathcal{P}, \mathcal{V})$ is a cZK proof system with pre-processing for the language L. Also, on the other hand, if SOACom is an SOA-secure commitment scheme that is only computationally binding, then $(\mathcal{P}, \mathcal{V})$ is a cZK argument system with pre-processing for the language L.

Theorem 7 ($(\mathcal{P}, \mathcal{V})$ is a cZK proof/argument system with pre-processing.)**.** *If* SOACom *is a statistically binding (resp., computationally binding) SOA-secure commitment scheme and* FLS *is a special* $\mathcal{WIP}oK$ *for* $\Lambda_{L,\mathtt{G}}$*, then* $(\mathcal{P}, \mathcal{V})$ *is a cZK proof (resp., argument) system with pre-processing.*

Proof Sketch:

- **Completeness.** Completeness directly follows from that of the commitment scheme SOACom and the $\mathcal{WIP}oK$, FLS.

- **Soundness.** Soundness follows from the binding property of SOACom and the proof of knowledge property of the FLS protocol, (i.e., a valid witness can be extracted from what any prover in FLS protocol is able to prove). Assume for contradiction that there exists an adversarial prover $\mathcal{P}^*$ that breaks soundness of $(\mathcal{P}, \mathcal{V})$ with non-negligible probability; i.e., $\mathcal{P}^*$ interacts with $\mathcal{V}$ and produces an accepting transcript for $x \notin L$. Then we can construct an adversarial sender $\mathsf{S}^*_{\mathsf{soa}}$ that can break binding of SOACom also with non-negligible probability. Intuitively, the reduction works as follows. $\mathsf{S}^*_{\mathsf{soa}}$ first interacts with $\mathcal{P}^*$ by running the code of the honest verifier $\mathcal{V}$. From the proof of knowledge property of the FLS protocol, if $\mathcal{V}$ accepts the proof then, with all but negligible probability, there exists a witness for $(x, \sigma) \in \Lambda_{L,\mathtt{G}}$ for which the FLS proof is given. Since $x \notin L$, the witness is s such that $\sigma = G(s)$. Now, $\mathsf{S}^*_{\mathsf{soa}}$ rewinds $\mathcal{P}^*$ to a point just before $\mathcal{V}$ sent σ_V and continues to interact with the same messages as in the main thread except for using $\sigma'_V \overset{\$}{\leftarrow} \{0,1\}^{f(n)}$ sampled with fresh randomness. $\mathsf{S}^*_{\mathsf{soa}}$ continues to rewind $\mathcal{P}^*$ for at most η many times, until $\mathcal{P}^*$ opens to a string different from the one it opened in the main thread. We can argue that if $\eta = \eta(n)$ is set to be a polynomial that is large enough (depending on the success probability of $\mathcal{P}^*$), then, with non-negligible probability, $\mathsf{S}^*_{\mathsf{soa}}$ obtains openings to two distinct values within η rewinds. This is because, since in each rewinding $\mathsf{S}^*_{\mathsf{soa}}$ chooses σ'_V freshly at random, and if the opened string σ_P remains the same in every rewinding, then $\mathcal{P}^*$ can succeed at most with negligible probability as $\sigma = G(s)$ for random σ only with negligible probability. Since we assume that $\mathcal{P}^*$ succeeds with non-negligible probability, we have that $\mathsf{S}^*_{\mathsf{soa}}$ obtains openings to two distinct strings σ_P and σ'_P in η rewinds with non-negligible probability, thus breaking binding of SOACom.

- **Concurrent Zero-Knowledge.** We argue that $(\mathcal{P}, \mathcal{V})$ satisfies the property of concurrent zero-knowledge by showing the existence of an expected polynomial time simulator algorithm Sim. With any adversarial verifier $\mathcal{V}^* = (\mathcal{V}1^*, \mathcal{V}2^*)$, Sim interacts in the pre-processing phase by running the simulator of SOACom in the commitment phase. Meanwhile, Sim also interacts in the first two rounds of the FLS protocol in the same way as in $(\mathcal{P}, \mathcal{V})$. Once pre-processing phases of all q concurrent executions are completed, to complete the proof phase of the i-th session, Sim first samples a random seed s'_i, sets $\sigma'_i = G(s'_i)$, and computes a proof π'_i for the statement $(x_i, \sigma'_i) \in \Lambda_{L,\mathsf{G}}$. Then, it computes the string it needs to open to, σ^i_P, as $\sigma^i_P = \sigma'_i \oplus \sigma^i_V$ and runs the simulator of SOACom in the decommitment phase to open the commitment run in the i-th session to σ^i_P. Finally, Sim outputs the output of the SOACom's simulator, $\{\sigma^i_V\}_{i\in[q]}$ sent by $\mathcal{V}^*$, the messages of the FLS protocol, and $\{(\sigma'_i, \pi'_i)\}_{i\in[q]}$. By a hybrid argument, we can show that the output of Sim is indistinguishable from the view output by $\mathcal{V}^*$. In particular, if the output of the simulator of SOACom is statistically (resp., computationally) indistinguishable from the interaction of the honest sender with any (possibly malicious) receiver, and if FLS proof is statistically (resp., computationally) witness-indistinguishable, then the simulated output produced by Sim is also statistically (resp., computationally) indistinguishable from the view of any (possibly malicious) verifier that interacts with the honest prover $\mathcal{P}$ in q concurrent sessions.

Observe that FLS can be run in parallel with SOACom (i.e., 1st round of FLS played along with the 2nd round of SOACom). Therefore we have that when SOACom is our $(3, 1)$-round scheme based on NBB use of OWFs the pre-processing phase can be run in 3 rounds and the opening phase in one round only. Moreover, in the 1st round of the pre-processing phase, the receiver can also send the first round of Naor's commitment scheme, therefore FLS can be run under the sole assumptions that OWFs exist. We have therefore the following corollary.

Corollary 2. *There exists a concurrent non-interactive zero-knowledge argument system with 3 rounds of pre-processing for L based on non-black-box use of OWFs.*

Then, by observing that in the 1st round of the pre-processing phase the receiver can send the first round of a 2-round statistically hiding commitment scheme, we have that FLS can be implemented so that it is a statistical $\mathcal{WIPoK}$ and that SOACom can be implemented to yield a statistically hiding SOA-secure commitment scheme. We have therefore the following corollary.

Corollary 3. *There exists a concurrent statistical non-interactive zero-knowledge argument system with 3 rounds of pre-processing for L based on the existence of collision-resistant hash functions.*