# Round-Optimal Black-Box Two-Party Computation

Rafail Ostrovsky[1], Silas Richelson[1], and Alessandra Scafuro[2(✉)]

[1] UCLA, Los Angeles, USA
[2] Boston University and Northeastern University, Boston, USA
{rafail,sirichel}@ucla.edu, scafuro@bu.edu

**Abstract.** In [Eurocrypt 2004] Katz and Ostrovsky establish the *exact* round complexity of secure two-party computation with respect to black-box proofs of security. They prove that 5 rounds are necessary for secure two-party protocols (4-round are sufficient if only one party receives the output) and provide a protocol that matches such lower bound. The main challenge when designing such protocol is to parallelize the proofs of consistency provided by both parties – necessary when security against malicious adversaries is considered– in 4 rounds. Toward this goal they employ specific proofs in which the statement can be unspecified till the last round but that require non-black-box access to the underlying primitives.

A rich line of work [1,9,11,13,24] has shown that the non-black-box use of the cryptographic primitive in secure two-party computation is not necessary by providing black-box constructions matching basically all the feasibility results that were previously demonstrated only via non-black-box protocols.

All such constructions however are far from being round optimal. The reason is that they are based on cut-and-choose mechanisms where one party can safely take an action only *after* the other party has successfully completed the cut-and-choose phase, therefore requiring additional rounds.

A natural question is whether round-optimal constructions do inherently require non-black-box access to the primitives, and whether the lower bound shown by Katz and Ostrovsky can only be matched by a non-black-box protocol.

In this work we show that round-optimality is achievable even with only black-box access to the primitives. We provide the first 4-round black-box oblivious transfer based on any enhanced trapdoor permutation. Plugging a parallel version of our oblivious transfer into the black-box non-interactive secure computation protocol of [12] we obtain the first round-optimal black-box two-party protocol in the plain model for any functionality.

## 1 Introduction

Secure two-party computation allows two mutually distrustful parties to compute a function of their secret inputs without revealing any information except


R. Gennaro and M. Robshaw (Eds.): CRYPTO 2015, Part II, LNCS 9216, pp. 339–358, 2015.
DOI: 10.1007/978-3-662-48000-7_17

what can be gathered from the output. It is known that achieving secure two-party computation information theoretically is impossible, and thus computation assumptions are required. In this work we are interested in construction for two-party computation based on general hardness assumptions in the plain model. Protocols based on general assumptions are flexible in that they allow the protocol to be implemented based on a variety concrete assumptions; even possibly ones which where not considered when the protocol was designed. Constructions based on general assumptions may use the cryptographic primitive based on the assumption in two ways: **black-box usage**, if the construction refers only to the input/output behavior of the underlying primitive; **non-black-box usage**, if the construction uses the code computing the functionality of the primitive. The advantage of black-box constructions is that their complexity is independent of the complexity of the implementation of the underlying primitive and are typically considered the first step towards practical constructions.

*Secure Two-Party Computation Under General Assumptions.* Yao [29] provided an elegant construction which securely realizes any two-party functionality, and which uses the underlying cryptographic primitives as black-box. This construction however guarantees security only against semi-honest adversaries, *i.e.*, adversaries that honestly follow the protocol. [5] show that semi-honest security is sufficient, as any protocol tolerating semi-honest adversaries can be compiled into one secure against malicious adversaries (*i.e.*, adversaries who can arbitrarily deviate from the protocol), by forcing the parties to prove, after each step, that they behaved honestly. Roughly, in this compiler, each party commits to his input at the very beginning and use a coin-flipping protocol to define the randomness that will be used in the semi-honest protocol. Then, for each protocol message, they add a zero-knowledge "proof of consistency" proving that the message was correctly computed according to the input committed and the randomness generated in the coin-flipping.

Unfortunately, this compiler is highly inefficient as the proofs of consistency require Karp reductions involving the circuits of the cryptographic primitives used. The exact complexity of these reductions grows more than linearly in the circuit complexity of the cryptographic primitive.[1] Researchers naturally began to wonder whether security against malicious adversaries could be achieved without relying on non-black-box use of cryptographic primitives.

*Black-Box Secure Two-Party Computation.* Ishai et al. [9,11] show that malicious security can be achieved without using expensive zero-knowledge proofs involving the code of the cryptographic primitives. Their work is based on the following observation: we can check that a party is honestly computing the protocol messages, by challenging the party to reveal the input and the randomness

[1] We note that a different approach altogether was taken by Kilian in [16], which does not require the use of cryptographic assumptions at all. However, his compiler works in the OT-hybrid model, thus we still need a protocol that implements the oblivious transfer functionality against malicious adversaries.

used in the computation. While this will certainly prove consistency[2], it is not zero-knowledge, as it leaks parties' entire inputs. Thus, the next idea is to have the parties engage in several parallel executions of the semi-honest protocol, where they run with random inputs. When a party is challenged on a random subset of protocol executions, she can safely reveal the randomness/inputs used in those executions which are independent of her actual inputs. If the party provides all convincing answers, then the challenger is guaranteed that the majority of the remaining executions are honestly computed as well. This step is repeated again in the opposite direction. Eventually after both parties have passed the tests, they run an additional step to "connect" the random inputs with their actual inputs and combine them across the remaining executions.

Following [9,11] subsequent work have shown black-box construction for adaptively secure two-party protocols [1], and constant-round black-box two-party protocols [13,24].

*Round-Optimal Secure Two-Party Computation.* In [15] Katz and Ostrovsky establish the *exact* round complexity of secure two-party computation from general assumptions. They show that 5 rounds are necessary and sufficient to compute any two-party functionality where both parties obtain the output, and 4 rounds are sufficient if only one party receives the output. To prove the upper bound they give a protocol that uses non-black-box proofs of consistency to enforce semi-honest behavior. The main technical difficulty they face is getting these proofs to complete in only 4 rounds − not an easy task as zero-knowledge in the standard model requires at least 4 rounds. Nevertheless, they manage to parallelize the proof and the computation into just 4 rounds using special properties of certain constructions of witness-indistinguishable (WI) proofs of knowledge. Namely, they crucially use the fact that in the WI proof of [19], the statement can be specified in the last round. Unfortunately, however, the statements to be proved concern values committed or computed in the protocol, and require the use of the circuits of the cryptographic primitives used in the protocol.

Previous results [1,11,24] have shown that essentially any feasibility result for two-party computation demonstrated using non-black-box techniques can also be obtained via black-box constructions. A natural question however, which so far has not been answered, is whether this is true for round optimal non-black-box constructions. This question is the focus of the current work. Namely,

> *Can we construct a round-optimal fully black-box protocol for two-party computation based on general assumptions?*

*Black-Box Round-Optimal Two-Party Computation?* When it comes to round optimality, the current state of the art suggests a negative answer. All known black-box protocols for secure computation achieve malicious security using a

[2] For sake of better clarity we are oversimplifying here. In [11] they introduce the definition of defensible adversaries and show how to use it in the cut-and-choose. We refer the reader to [11] for more details.

cut-and-choose mechanism that introduces additional rounds. The need for additional rounds seems inherent because in such mechanisms a party will take an action only *after* the other party has successfully completed the cut-and-choose phase. Additional rounds are used to combine and connect the random inputs used in the unopened sessions of the cut-and-choose with the real inputs.

An alternative to the traditional cut-and-choose approach for black-box construction was shown by Ishai et al. in [12], where they provide a black-box protocol for non-interactive secure two-party computation (NISC) based on the "MPC-in-the-head" paradigm of [13]. This approach however, following [14,16], works in the OT-hybrid model, and thus can only hope to achieve round optimality in the plain model if there exists a 4-round black-box oblivious transfer protocol in the plain model with parallel security.

One might hope that perhaps we can build a 4-round black-box oblivious transfer in the plain model starting from the 2-round OT protocol of Peikert et al. [25] − whose security is in the CRS model − by running a two-party coin flipping protocol to generate the CRS. We note that this approach seems doomed to fail because, as proved in [15], secure coin-flipping requires at least 5 rounds, *regardless of the use of the underlying cryptographic primitives.*

*Our Contribution.* In this paper we answer the above question positively by constructing a 4-round black-box oblivious transfer protocol based on the existence of (enhanced) trapdoor permutations. Our construction is easily extended to achieve parallel secure oblivious transfer which, using the compiler of [12], gives a round-optimal black-box protocol for two-party computation in the plain model.

### 1.1 Our Techniques

As mentioned above, it suffices to build a 4-round black-box oblivious transfer protocol based on general assumptions. We start with a high-level overview of the main ideas behind the construction.

Our starting point is the following basic 3-round protocol for OT based on black-box use of enhanced trapdoor permutations (TDP).

1. S chooses trapdoor permutation $(f, f^{-1}) \leftarrow \mathsf{Gen}(1^\kappa)$ and sends $f$ to R.
2. R chooses $x \overset{\mathrm{R}}{\leftarrow} \{0,1\}^\kappa$, and sends $(z_0, z_1)$ to S where $z_b = f(x)$ and where $z_{1-b} \overset{\mathrm{R}}{\leftarrow} \{0,1\}^\kappa$ is random.
3. S returns $(w_0, w_1)$ where $w_a = s_a \oplus \mathsf{hc}\big(f^{-1}(z_a)\big)$, $a \in \{0,1\}$

where $\mathsf{hc}(\cdot)$ is a hardcore bit of $f$. If both parties follow the protocol then S can't learn anything about R's input bit $b$ as both $z_0$ and $z_1$ are just random $\kappa-$bit strings. Similarly, the security of the TDP $f$ ensures that R cannot distinguish $w_{1-b}$ from random as long as $z_{1-b}$ was truly chosen randomly. Unfortunately, there are two serious problems with this protocol. First, there is nothing to stop a malicious R from sending $(z_0, z_1)$ such that he knows the pre-images of *both* values under $f$, thus allowing him to learn both $s_0$ and $s_1$. Indeed, the

above protocol only offers security against a semi-honest receiver. Second, while the above protocol leaks no information to S about R's input bit, it is not simulateably secure. Input indistinguishably is often sufficient if a protocol is to be executed once in isolation, however we aim to use our OT as a building block for general 2PC, as such, stronger security is required.

Katz and Ostrovsky [15] solve the first problem by having the parties engage in a secure coin-flipping protocol to produce a random $r \in \{0,1\}^\kappa$ and forcing R to prove that either $z_0 = r$ or $z_1 = r$ using a witness-indistinguishable proof of knowledge. This denies R the freedom to generate both $z_0$ and $z_1$. Such WI proofs, however, require using the underlying commitment scheme, used for the coin-flipping, in a non-black-box way. Our solution to this problem can be seen as implementing the coin-flipping idea of [15] while making only black-box use of the commitment scheme. For this we use an adaptation of the black-box commit-and-prove protocol of Kilian [17].

We solve the second problem by having S commit the inputs already in the second round and prove that such committed inputs are the ones used for the OT. Doing this naïvely would require making non-black-box use of cryptographic primitives, so, in typical cut-and-choose style, we instead have S commit to *shares* of the inputs, and play the protocol many times in parallel where R opens mostly the shares corresponding to his input bit (to enable reconstruction of $s_b$) but enough shares of $s_{1-b}$ to be convinced that S is playing fairly. This introduces several subtleties, that we discuss in the next paragraph.

We construct our OT protocol in two steps. First, we construct a 4-round OT protocol, $\Pi^{\mathrm{R}}_{\mathsf{OT}}$, that is simulatable only against a malicious receiver. Then, we use $\Pi^{\mathrm{R}}_{\mathsf{OT}}$ as a building block to build the final OT protocol that is simulatable for both parties. In the next two paragraphs we describe the ideas outlined above in greater details.

*A 4-Round Black-Box OT Secure Against Malicious Receivers.* We want to implement the coin-flipping that we mentioned above, without requiring R to give non-black-box proofs about the committed values. We do it by recasting the above problem in terms of equivocal and binding commitments, and having the output of the coin-flipping to be the pair of strings $(z_0, z_1)$ (instead of a random $r$ such that either $z_0 = r$ or $z_1 = r$). We provide a mechanism that allows R to compute one binding commitment and one equivocal commitment. In the coin-flipping, R first sends such commitments to S, then after seeing S's random strings, she opens both commitments. The crucial point is that R can control the output of one of the strings by equivocating one the commitments, while the other string will be truly random. With this tool we can directly obtain a black-box OT protocol that is simulatable for the receiver as follows.

1. R, on secret input $b$, chooses random strings $r_0, r_1$. Then sends commitments $C_0, C_1$ such that commitment $C_b$ is equivocal. R proves that one of the commitments is binding.
2. S chooses trapdoor permutation $(f, f^{-1}) \leftarrow \mathsf{Gen}(1^\kappa)$ and sends $f$ to R. Additionally S sends a random string $r$ to R.

3. R chooses $x \stackrel{R}{\leftarrow} \{0,1\}^\kappa$ and computes $z_b = f(x)$. Then it equivocates commitment $C_b$ so that it opens to $r_b = z_b \oplus r$, while it honestly opens value $r_{1-b}$.
4. S upon receiving $r_0, r_1$, computes $z_0 = r \oplus r_0$ and $z_1 = r \oplus r_1$ and sends $(w_0, w_1)$ where $w_a = s_a \oplus \mathsf{hc}\big(f^{-1}(z_a)\big)$.

If the proof in Step 1 is sound, R can only equivocate one string and thus knows the preimage of one value only. If the proofs and the commitments are hiding, the sender has no advantage in distinguishing which string is controlled by the receiver. Additionally, if we make the proof extractable, then the above protocol is simulatable against a malicious receiver.

Thus, what is left to do is to construct the tool that allows R to compute an equivocal commitments and a binding commitment and a WI proof of the binding of one of the two. This proof must be black-box and 3 rounds only. We implement this proof, by employing ideas from the black-box commit-and-prove protocol due to Kilian [17] which allows a party to commit to two bits $x_0$ and $x_1$ and prove the *equality* $x_0 = x_1$ without revealing the value of the committed bit. Kilian's protocol for proving equality of two committed bits goes as follows. (In the following matrix, think of each column of the matrix as the shares of one bit.)

1. R chooses $\mathbf{M} = \begin{pmatrix} x_{0,0} & x_{0,1} \\ x_{1,0} & x_{1,1} \end{pmatrix} \in \{0,1\}^{2\times 2}$ randomly such that $x_{0,a} \oplus x_{1,a} = x_a$ for $a \in \{0,1\}$. R then computes and sends $\mathsf{Com}(x_{a,a'})$ for $a, a' \in \{0,1\}$ over to S along with $v = x_{0,0} \oplus x_{0,1}$.
2. S sends a random $b \stackrel{R}{\leftarrow} \{0,1\}$ to R.
3. R sends to S the decommitments to $x_{b,0}$ and $x_{b,1}$. S verifies that $v = x_{b,0} \oplus x_{b,1}$.

Note that the sum of the columns of $\mathbf{M}$ are equal iff $x_0 = x_1$, in which case the sum of the rows of $\mathbf{M}$ are also equal, and so if R is honest the protocol will complete and S's verification will succeed. On the other hand, if $x_0 \neq x_1$ then the sum of the rows of $\mathbf{M}$ are different and so no matter which value $v$ was sent by R in Step 1, there is only a $1/2$ chance that S will ask for the row which sums to $v$. To decommit, R decommits to one of the remaining two values that he has not yet revealed, $x_{1-b,0}$ and $x_{1-b,1}$. Revealing one is enough since either one can be used to reconstruct $x_0$. The interesting feature of this protocol, which was already used in [4], is that opening only one of the columns, instead of two, can enable equivocality: assume R can guess the row S will ask to open, then R could commit to a matrix where each column sums to a different bit, and compute $v$ as the xor of the row that S will select. In this way S will be convinced and later R can adaptively choose whether to decommit to 0 or 1, by opening one column or another. This observation is particularly useful when combined with a standard trick for composing two $\Sigma$-protocols to compute the OR of two statements [2]. Recall that $\Sigma$-protocols satisfy the property that, if the challenge is known in advance, then one can simulate an accepting transcript without knowledge of the witness. The trick is to run two independent executions, say $\Sigma_0, \Sigma_1$, in parallel, but have the challenges $c_0, c_1$ derived in such a way that the prover can control

exactly one of the challenge $c_b$ while the other $c_{1-b}$ will be totally random, and the verifier cannot tell the difference. In this way, the prover can successfully finish both protocols $\Sigma_0, \Sigma_1$ by simulating one of the transcripts and computing the other one honestly.

Putting the two ideas together, we can build a protocol where R commits to a bit $x$ and a bit $y$, using two executions of the above protocol for equality proofs, and then using the trick for OR composition, R can cheat in one of the equality proofs. Thus one of the value between $x$ and $y$ is equivocal.

One can extend this idea to a string commitment having R commits to two strings $X$ and $Y$ by committing each single bit and then cheat in all the proof for bits belonging to one string, and being honest in all bits belonging to the other string, by using the OR trick as before. Note however, that in the string case we must show that a malicious committer, cannot gain advantage by committing equivocally only *some* of the bits of each string. We protect our protocol from such behavior by using error-correction: we expand each $\kappa$-bit string into a $3\kappa$ bit string, while having the committer being able to control in total only $\kappa$ bits for both strings. We are able to prove that due to error-correcting property, corrupting only some bits for each string is not enough to control the final value of the string. We provide more details on how this mechanism is implemented in Sect. 3.

*From One-Side Simulatable OT to Fully Simulatable OT.* The protocol $\Pi^{\mathrm{R}}_{\mathsf{OT}}$ is not simulatable against a malicious sender. Just as with the basic protocol, S is not committed to any value till the last round so any rewinding strategy will be ineffective. Therefore we have the sender commit to two secret keys in the second round via an extractable commitment[3] and then have him play $\Pi^{\mathrm{R}}_{\mathsf{OT}}$ using the decommitments as inputs. In the last round the server encrypts the actual inputs using the committed keys. This gives the simulator some hope of extracting S's secret inputs by rewinding. This idea by itself doesn't exactly work; the simulator has no guarantee that S used valid decommitments as inputs in the OT played with R. This opens the door to input-dependent abort attacks. We fix this by having S first secret share his inputs and commit to the shares. Then R and S run many executions of the OT protocol $\Pi^{\mathrm{R}}_{\mathsf{OT}}$, where in the $i$-th execution, S uses as input the decommitments to the $i-$th shares. Intuitively, this helps solving the input-depended abort attack, because now R will also check some of the shares corresponding to $s_{b-1}$. This check, however, must be done in such a way that the probability of S passing the check is independent of the bit $b$. A bit more in details, obtaining a fully secure protocol requires dealing with two types of malicious behavior. First, we need a mechanism that allows S to prove that he committed to valid shares of a secret. For this we use $t$-out-of-$\kappa$ Shamir secret sharing scheme and another variant of Kilian's commit-and-prove protocol. Our main observation is that Kilian's technique is actually quite general and can be used, not only to prove equality of committed values, but that the committed

[3] Note that extractable commitments can be built from black-box use of any one-way-permutation [21]. In particular it does *not* require trapdoor permutation.

values satisfy *any* linear relation. In particular, it can be used to prove that a committed vector is a set of valid shares of some secret according to Shamir secret sharing scheme.

Secondly, we must give R a strategy to detect the case in which S is not using valid decommitment of the shares in some of the OT executions. Consider, for example, what happens if S were to give correct decommitment in all of the parallel executions of $\Pi_{\mathsf{OT}}^{\mathrm{R}}$ except one, where he uses a wrong decommitment in correspondence of the bit 1. Then since R opens more of the $\Pi_{\mathsf{OT}}^{\mathrm{R}}$ using input $b$ he is noticeably more likely to notice the bad input $b = 1$. We fix this problem by having R performing first a test, which is independent on his secret bit $b$. R opens an equal number of execution of $\Pi_{\mathsf{OT}}^{\mathrm{R}}$, say $\kappa/4$, using inputs $b = 0$ and $b = 1$. This test is clearly independent on R's actual input and allows R to check that S is playing honestly in most of the OT executions for *both* inputs. If the test passes, then R is guaranteed that he will obtain at least $t - n/4$ more valid decommitments from the remaining OTs and will be able to reconstruct the secret.

### 1.2 Further Discussions

Following the OT protocol used in [15], our protocol is based only on *enhanced* trapdoor permutation. We do not require any additional assumption. Moreover, we stress that the lower bound of 4 rounds for secure two-party computation only applies in the plain model. Indeed, in the UC-setting we know how to construct 2-round OT [25] (although under different, standard, assumptions).

We also emphasize that aim of this paper is to match the upper bound of 4-round for two-party computation from general assumptions, that so far was achieved only with a non-black-box construction. As such, our result should be seen as a feasibility result rather than an attempt of building more efficient two-party protocols under general assumptions. It is an interesting direction to improve our techniques to achieve better efficiency.

### 1.3 Other Related Work on Black-Box Secure Computation

We mention additional related work that are less relevant for our result but that have contributed in the understanding of the power of black-box access to cryptographic primitives. In [3] Damgaard and Ishai show a constant round multi-party protocol where the party have only black-box access to a PRG. This work assumes honest majority. In [27], Wee shows the first black-box constructions with sub-linear round complexity for MPC, which Goyal [6] improves to obtain constant-round MPC constructions based on the black-box use of any OWF. In [7] black-box use of OWFs has been shown to be sufficient to construct constant-round concurrent non-malleable commitments. Other black-box constructions for commitment schemes have been considered w.r.t. selective opening attacks in [22,28]. In [20] Lin and Pass showed the first black-box construction for MPC in the standard model that satisfies a non-trivial form of concurrent security. Their construction requires a non-constant number of rounds. Very

recently, Kiyoshima et al. in [18] improved on the round complexity providing a constant- round construction for the same result. Finally, another line of research has looked at achieving black-box construction for protocols that requires non-black-box simulation, such as black-box public coin ZK [8] and resettably-sound ZK from OWF [23].

## 2 Preliminaries

*General Notation.* We denote by $\kappa$ the security parameter, and by PPT a machine running in probabilistic polynomial time. We denote vector using bold notation $\mathbf{v}$ and we denote the $i$-th coordinate of a vector $\mathbf{v}$ using notation $[v]_i$. We denote a matrix using capital and bold letters $\mathbf{M}$, and we denote the element in position $i, j$ of $\mathbf{M}^{\mathsf{index}}$ by $x_{i,j}^{\mathsf{index}}$. Let $[n]$ be the set $\{1, \ldots, n\}$ and $\mathbb{Z}_q$ be the integers mod $q$. For a bit $b \in \{0,1\}$ we write $\overline{b}$ as shorthand for $1 - b$. We write $\mathbf{negl}(\cdot)$ for an unspecified negligible function.

*Trapdoor Permutations.* Trapdoor permutations are permutations which are easy to compute and hard to invert *unless* you know the trapdoor, in which case they are easy to invert. The formal definition is as follows.

**Definition 1 (Trapdoor Permutation).** Let $\mathcal{F} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Invert})$ be three PPT algorithms such that

- $\mathsf{Gen}(1^\kappa)$ outputs a pair $(f, \mathsf{trap})$ where $f : \{0,1\}^\kappa \to \{0,1\}^\kappa$ is a permutation;
- $\mathsf{Eval}(f, \cdot) = f(\cdot)$ evaluates $f$; and
- $\mathsf{Invert}(f, \mathsf{trap}, \cdot) = f^{-1}(\cdot)$ evaluates $f^{-1}$.

We say that $\mathcal{F}$ is a *family of trapdoor permutations* (TDPs) if for any PPT algorithm R R

$$\Pr_{(f,\mathsf{trap})\leftarrow\mathsf{Gen}(1^\kappa), y\leftarrow\{0,1\}^\kappa}\big(\mathrm{R}(f, y) = f^{-1}(x)\big) = \mathbf{negl}(\kappa).$$

Additionally, we assume that our TDP families have a weak form of certifiability. Namely, we assume that given some $f$ output by $\mathsf{Gen}(1^\kappa)$ it is possible to tell in polynomial time whether $f$ is a permutation on $\{0,1\}^\kappa$ or not. It will be convenient for us to have trapdoor permutations which act on vector spaces over fields instead of just $\{0,1\}^\kappa$. This can be arranged by identifying $\{0,1\}^\kappa$ with $\mathbb{F}_2^\kappa$, or if we need a larger alphabet, we can identify $\{0,1\}^\kappa$ with $\mathbb{F}_{2^k}^{\kappa/k}$. When we are using this point of view we will write $(f, \mathsf{trap}) \xleftarrow{\mathrm{R}} \mathsf{Gen}(\mathbb{F}_2^\kappa)$.

*Hard-Core Bits.* We assume the reader is familiar with the notion of a hard-core bit of a oneway permutation. Briefly, we say that a family of predicates $\mathcal{H} = \big\{h : \{0,1\}^\kappa \to \{0,1\}\big\}$ is *hard-core* for the TDP family $\mathcal{F}$ if for random $(f, \mathsf{trap}) \xleftarrow{\mathrm{R}} \mathsf{Gen}(1^\kappa)$, $h \xleftarrow{\mathrm{R}} \mathcal{H}$, and $x \xleftarrow{\mathrm{R}} \{0,1\}^\kappa$, $h(x)$ is hard to predict given $f(x)$. A hardcore big can be extended to output a vector in a natural way: $\mathbf{h}(x) = h(x) \circ h\big(f(x)\big) \circ \cdots \circ h\big(f^{k-1}(x)\big)$, which is indistinguishable from random, given $f^k(x)$. When we identify the domain $\{0,1\}^\kappa$ of $f$ with a $\kappa-$dimensional vector space over $\mathbb{F}_2$, we will likewise identify the output of $\mathbf{h}(\cdot)$ with an $\mathbb{F}_2-$vector of the same dimension.

*Oblivious Transfer.* Oblivious Transfer (OT) is a two-party functionality $\mathcal{F}_{\mathsf{OT}}$, in which a sender S holds a pair of strings $(s_0, s_1)$, and a receiver R holds an a bit $b$, and wants to obtain the string $s_b$. The security requirement for the $\mathcal{F}_{\mathsf{OT}}$ functionality is that any malicious receiver does not learn anything about the string $s_{1-b}$ and any malicious sender does not learn which string has been transfered. This security requirement is formalized via the ideal/real world paradigm. In the ideal world, the functionality is implemented by a trusted party that takes the inputs from S and R and provides the output to R and is therefore secure by definition. A real world protocol $\Pi$ securely realizes the ideal $\mathcal{F}_{\mathsf{OT}}$ functionalities, if the following two conditions hold. (a) **Security against a malicious receiver.** The output of any malicious receiver $\mathrm{R}^*$ running one execution of $\Pi$ with an honest sender S can be simulated by a PPT simulator $\mathsf{Sim}$ that has only access to the ideal world functionality $\mathcal{F}_{\mathsf{OT}}$ and oracle access to $\mathrm{R}^*$. (b) **Security against a malicious sender.** The joint view of output of any malicious sender $\mathrm{S}^*$ running one execution of $\Pi$ with R and the output of R can be simulated by a PPT simulator $\mathsf{Sim}$ that has only access to the ideal world functionality functionality $\mathcal{F}_{\mathsf{OT}}$ and oracle access to $\mathrm{S}^*$. In this case the output of the malicious $\mathrm{S}^*$ is combined with the output of R in the ideal world.

We also consider a weaker definition of $\mathcal{F}_{\mathsf{OT}}$ that is called **one-sided** simulatable $\mathcal{F}_{\mathsf{OT}}$, in which we do not demand the existence of a simulator against a malicious sender, but we only require that a malicious sender cannot distinguish whether the honest receiver is playing with bit 0 or 1. A bit more formally, we require that for any PPT malicious sender $\mathrm{S}^*$ the view obtained from executing $\Pi$ when the receiver R plays with bit 0 is computationally indistinguishable from the view obtained when R is playing with bit 1.

Finally, we consider the $\mathcal{F}_{\mathsf{OT}}^m$ functionality where the sender S and the receiver R runs $m$ execution of OT in parallel.

*Secure Two-Party Computation.* Let $\mathcal{F}(x_1, x_2)$ be a two-party functionality run between parties $P_1$ holding input $x_1$ and $P_2$ holding input $x_2$. In the ideal world, $P_i$ (with $i \in \{1, 2\}$) sends its input $x_i$ to the $f$ and obtains only $y = \mathcal{F}(x_1, x_2)$. We say that a protocol $\Pi$ securely realizes $\mathcal{F}(\cdot, \cdot)$ if the view of any malicious $P_i^*$ executing $\Pi$ with an honest $P_j$ with $i \neq j$ combined with the output of $P_j$ (if any) can be simulated by a PPT simulator that has only access to $\mathcal{F}$ and has oracle access to $P_i^*$.

*Shamir Secret Sharing Scheme.* A $t$-out-of-$n$ secret sharing scheme gives a way to break a secret into shares in such a way so that any set of shares either reveals nothing about the secret, if the set has size less than $t$, or allows one to reconstruct the entire secret, if the set has size at least $t$. Shamir secret sharing [26] constructs such a scheme using polynomials. Fix a prime $q > n$. To share a secret field element $\alpha \in \mathbb{Z}_q$, the function $\mathsf{Share}$ chooses a random polynomial $f(x) \in \mathbb{Z}_q[x]$ of degree at most $t - 1$ and defines the vector $[\alpha] = \big([\alpha]_1, \ldots, [\alpha_n]\big) \in \mathbb{Z}_q^n$, by setting $[\alpha]_i = f(i)$. That this is a $t-$out$-$of$-n$ secret sharing scheme follows from basic properties of polynomials. To reconstruct a secret, the fuction $\mathsf{Recon}$ takes in input a set of $t + 1$ valid shares and uses

Lagrange interpolation to compute the unique $t$-degree polynomial $f$ defined by such shares and output the free coefficient of $f$.

We briefly comment on another property of this scheme that we will use in our protocol. The map which sends a degree $t-1$ polynomial to its vector of shares is linear over $\mathbb{Z}_q$. It follows that the set of vectors in $\mathbb{Z}_q^n$ which are valid sharings is a $t-$plane in $\mathbb{Z}_q^n$, or equivalently, that there exists a linear map $\psi : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n-t}$ such that $\psi(\mathbf{v}) = 0$ iff $\mathbf{v}$ is a valid sharing.

*Extractable Commitments.* A commitment scheme scheme is a two-party functionality run between a sender with input a secret message $m$ and a committer that has no input, and consists of two phase: commitment and decommitment phase. In the commitment phase the sender commits to its message $m$. A commitment scheme is hiding if any PPT malicious receiver cannot distinguish the secret message $m$ in this phase. In the decommitment phase the message reveals $m$ and the randomness used to compute the commitment. This phase is statistically binding if any malicious sender cannot successfully open to any message $m' \neq m$ in this phase.

We say that a commitment scheme is extractable if there exists an efficient extractor that having black-box access to any malicious sender that successfully performs the commitment phase, is able to efficiently extract the committed string. In the paper we employ the extractable commitment provided in [21]. The commitment phase consists of 3 rounds, that we denote by ($\mathsf{ExtCom1}$, $\mathsf{ExtCom2}$, $\mathsf{ExtCom3}$). The decommitment phase is non-interactive.

## 3 Four-Round Black-Box Oblivious Transfer

In this section we describe our 4-round black-box OT protocol $\Pi_{\mathsf{OT}}$ in details. We present it in two steps. First we give an OT protocol that is simulatable against a malicious receiver and provides only indistinguishability security against a malicious sender. We denote this protocol by $\Pi^{\mathrm{R}}_{\mathsf{OT}}$. We then show how to use (black-box) extractable commitments and Shamir secret sharing, to compile $\Pi^{\mathrm{R}}_{\mathsf{OT}}$ into a protocol that is fully simulatable.

### 3.1 Four-Round Black-Box OT Secure Against Malicious Receivers

The building block for $\Pi^{\mathrm{R}}_{\mathsf{OT}}$ is a protocol that allows the receiver to compute two string commitments, $C_0, C_1$, such that one commitment is equivocal, and prove that at least one commitment is binding.

As a warm up for our construction we show how to implement such building block for the simpler case where the receiver commits to two bits, and then he is able to equivocate one bit. The soundness of the warm up protocol is 1/2. The idea is to have two executions of Kilian's black-box commit-and-prove protocol (outlined in Sect. 1.1), and combine the two proofs using the OR trick of $\Sigma$-protocols. The details are shown in Protocol 1.

**Protocol 1. Compute One Biding and One Equivocal Commitment.**
**Input to R:** *A bit $b$ indicating which commitment should be equivocal.*

1. R *chooses two matrices* $\mathbf{M}^0$ *and* $\mathbf{M}^1$ *where each* $\mathbf{M}^a = \begin{pmatrix} x^a_{0,0} & x^a_{0,1} \\ x^a_{1,0} & x^a_{1,1} \end{pmatrix} \in \{0,1\}^{2\times 2}$ *is random such that:*
   - *Matrix* $\mathbf{M}^b$*: this matrix represent two different bits, therefore the xor of the first column is* 0*, and the xor of the second column is* 1*. Namely,* $x^b_{0,0} \oplus x^b_{1,0} = 0$ *and* $x^b_{0,1} \oplus x^b_{1,1} = 1$*;*
   - *Matrix* $\mathbf{M}^{1-b}$*: both columns are representing the same bit:* $x^{1-b}_{0,0} \oplus x^{1-b}_{1,0} = x^{1-b}_{0,1} \oplus x^{1-b}_{1,1} = r_{1-b}$ *for* $r_{1-b} \in \{0,1\}$*.*

   R *commits to all of the* $x^a_{a',a''}$ *in both matrixes and sends to* S *values* $v^0$ *and* $v^1$ *computed as follows:*
   - $v^{1-b}$ *is honestly computed as the xor of the first row (as in Kilian's protocol), namely,* $v^{1-b} = x^{1-b}_{0,0} \oplus x^{1-b}_{0,1}$*.*
   - $v^b$ *is a random bit.*
2. S *sends* R *a random* $r' \overset{R}{\leftarrow} \{0,1\}$ *and a challenge* $c \in \{0,1\}$*.*
3. R *computes challenges* $(c_0, c_1)$ *such that* $c_0 \oplus c_1 = c$ *and the challenge* $c_b$ *is pointing exactly to the row of* $\mathbf{M}^b$ *the xor of which is* $v_b$*. Namely,* $c_b$ *is such that* $v^b = x^b_{c_b,0} \oplus x^b_{c_b,1}$*. Note this is always possible as* $x^b_{0,0} \oplus x^b_{0,1} \neq x^b_{1,0} \oplus x^b_{1,1}$*. Next,* R *decommits to* $x^0_{c_0,0}$*,* $x^0_{c_0,1}$ *from matrix* $\mathbf{M^0}$ *as well as* $x^1_{c_1,0}$*,* $x^1_{c_1,1}$ *from matrix* $\mathbf{M^1}$*.*

   *Finally, for each matrix,* R *decommits to one column. For* $\mathbf{M^{1-b}}$*, which is honestly computed,* R *opens one column chosen at random (*R *will need to decommit one value of the column as the other one was already opened to answer the challenge). For* $\mathbf{M^b}$*,* R *will decommit to the column* $r_b$ *such that* $r_b \oplus r' = s_b$ *where* $s_b$ *is the bit that* R *wants to obtain out of the coin-flipping of the bit in position* $b$*. Formally,* R *decommits to one of* $x^{\overline{b}}_{c_{\overline{b}},0}$ *and* $x^{\overline{b}}_{c_{\overline{b}},1}$ *at random (using the shorthand* $\overline{b} = 1 - b$*), completing a decommitment to the value* $s_{1-b} = r_{1-b}$*.* R *decommits to* $x^b_{c_b,r_b}$ *(completing a decommitment to* $s_b = r_b \oplus r'$*).* R *also sends* $(c_0, c_1)$*.*
4. **Verification:** S *checks that* $c_0 \oplus c_1 = c$ *and that* $x^a_{c_a,0} \oplus x^a_{c_a,1} = v^a$ *for* $a \in \{0,1\}$*. If not* S *aborts.*
5. **Output:** *Both parties set output to* $(z_0, z_1)$ *where* $z_a = s_a \oplus r'$*.*

If R correctly follows the protocol then the output $(z_0, z_1)$ satisfies $z_b = r_b$ while $z_{1-b}$ is random. Furthermore, if R, in an attempt to cheat, chooses $M^0$ and $M^1$ both such that $x^a_{0,0} \oplus x^a_{1,0} \neq x^a_{1,0} \oplus x^a_{1,1}$, then S will abort whenever $c \neq d_0 \oplus d_1$ (which happens with probability 1/2), where $d_a \in \{0,1\}$ is such that $v^a = x^a_{d_a,0} \oplus x^a_{d_a,1}$. This protocol can be seen as partial coin-flipping protocol that output two coins and guarantee that at least one coin is fair.

The ability for R to completely control one but not both of the output bits in the above protocol is essentially exactly what we need in order to compile the OT which is secure only against a semi-honest R into one which is maliciously secure. The basic idea is to extend the above coin-flipping to strings and enable

R to obtain two strings $z_0, z_1 \in \{0,1\}^\kappa$ such that $z_b = f^k(x)$ for some value $x$ chosen by her. As mentioned, this forces $z_{1-b}$ to be random, and so R cannot know a preimage without breaking the trapdoor permutation.

However, when extending the above warm-up protocol to a string via bit-wise commit-and-proofs we must enforce that a malicious receiver cannot cheat by controlling *some* of the bits of both $z_0$ and $z_1$ and wind up knowing preimages of both values. We protect our protocol from such behavior by letting $\mathbf{A} \in \mathbb{Z}_q^{3\kappa \times \kappa}$ be a matrix with good error correcting properties (such as a Vandermonde matrix) and working in the image of $\mathbf{A}$.

This introduces some complications. Specifically it requires moving to a non-binary base field as we need an error correcting code with (constant but) large distance. In our actual protocol we use a variant of the unfair coin flipping described above, adapted to work over over $\mathbb{Z}_q$ for some prime power $q = \emptyset(\kappa)$. The major difference is that instead of committing to every entry in $2-$by$-2$ matrices, R commits to every entry in $2-$by$-q$ matrices. For each matrix R proves that the sum of the elements in each column is the same. In order to commit equivocally, R chooses the $q$ columns of the matrices corresponding to his bit $b$ to have distinct sums. Namely, for every $\alpha \in \mathbb{Z}_q$ there is exactly one column whose entries add to $\alpha$. The final protocol $\Pi_{\mathsf{OT}}^{\mathrm{R}}$ is formally described in Protocol 2.

**Protocol 2 ($\Pi_{\mathsf{OT}}^{\mathrm{R}}$). Public Input:** *A prime $q = \mathcal{O}(\kappa)$, a Vandermonde matrix $\mathbf{A} \in \mathbb{Z}_q^{3\kappa \times \kappa}$ and a statistically binding commitment scheme* $\mathsf{Com}$.
**Sender's Input:** $\mathbf{s}_0, \mathbf{s}_1 \in \mathbb{Z}_q^\kappa$. **Receiver's Input:** $b \in \{0,1\}$.

**1.** $(\mathrm{R} \longrightarrow \mathrm{S})$**:** R *chooses* $\mathbf{r}^{\bar{b}} \xleftarrow{R} \mathbb{Z}_q^\kappa$ *and sets* $\hat{\mathbf{r}}^{\bar{b}} = \mathbf{A}\mathbf{r}^{\bar{b}} \in \mathbb{Z}_q^{3\kappa}$. R *then chooses* $6\kappa$ *matrices* $\left\{(\mathbf{M}^{0,i}, \mathbf{M}^{1,i})\right\}_{i=1,\ldots,3\kappa}$ *where* $\mathbf{M}^{a,i} = \begin{pmatrix} x_{0,0}^{a,i} & x_{0,1}^{a,i} & \cdots & x_{0,q-1}^{a,i} \\ x_{1,0}^{a,i} & x_{1,1}^{a,i} & \cdots & x_{1,q-1}^{a,i} \end{pmatrix} \in \mathbb{Z}_q^{2\times q}$ *is random such that:*
- $x_{0,0}^{\bar{b},i} + x_{1,0}^{\bar{b},i} = \cdots = x_{0,q-1}^{\bar{b},i} + x_{1,q-1}^{\bar{b},i} = [\hat{\mathbf{r}}^{\bar{b}}]_i, \forall\ i.$
- $x_{0,0}^{b,i} + x_{1,0}^{b,i} = \sigma_i(0), \ldots, x_{0,q-1}^{b,i} + x_{1,q-1}^{b,i} = \sigma_i(q-1)\ \forall\ i$, *where the $\sigma_i$ are random permutations of $\mathbb{Z}_q$.*

R *commits to all of the* $x_{a',a''}^{a,i}$ *using* $\mathsf{Com}$. *Let* $\mathbf{x}_0^{a,i} = (x_{0,0}^{a,i}, \ldots, x_{0,q-1}^{a,i}) \in \mathbb{Z}_q^q$ *be the top row vector of* $\mathbf{M}^{a,i}$. *Similarly, let* $\mathbf{x}_1^{a,i}$ *be the bottom row of* $\mathbf{M}^{a,i}$. *Also let* $\psi : \mathbb{Z}_q^q \to \mathbb{Z}_q^{q-1}$ *be the linear map* $\psi : \mathbf{x} = (x_0, \ldots, x_{q-1}) \mapsto (x_1 - x_0, \ldots, x_{q-1} - x_0)$. R *sends vectors* $\{\mathbf{v}^{0,i}, \mathbf{v}^{1,i}\}_{i=1,\ldots,3\kappa}$ *where each* $\mathbf{v}^{a,i} \in \mathbb{Z}_q^{q-1}$ *is generated as follows:*
- $\mathbf{v}^{\bar{b},i} = \psi(\mathbf{x}_0^{\bar{b},i})$;
- *draw* $c_b \xleftarrow{R} \{0,1\}^{3\kappa}$ *and set* $\mathbf{v}^{b,i} = \psi(\mathbf{x}_0^{b,i})$ *if* $c_{b,i} = 0$, $\mathbf{v}^{b,i} = -\psi(\mathbf{x}_1^{b,i})$ *if* $c_{b,i} = 1$.

**2.** $(\mathrm{S} \longrightarrow \mathrm{R})$**:** S *chooses random* $c \xleftarrow{R} \{0,1\}^{3\kappa}$, $\mathbf{r}' \xleftarrow{R} \mathbb{Z}_q^\kappa$, *and sends $c$ and $\mathbf{r}'$. Additionally,* S *chooses a trapdoor permutation* $(f, f^{-1}) \xleftarrow{R} \mathrm{Gen}(\mathbb{Z}_q^\kappa)$ *and sends $f$ to* R.

**3.** $\big(\mathrm{R} \longrightarrow \mathrm{S}\big)$**:** R *parses* $c$ *into* $(c_0, c_1)$ *such that* $c_0 \oplus c_1 = c$ *where* $c_b$ *is as in step 1. For both* $a \in \{0,1\}$, R *decommits to every coordinate of* $\mathbf{x}^{a,i}_{c_{a,i}}$ *as well as to one coordinate,* $[x^{a,i}_{\overline{c}_{a,i}}]_j$, *of* $\mathbf{x}^{a,i}_{\overline{c}_{a,i}}$. *When* $a = \overline{b}$, *this coordinate* $j$ *is chosen randomly, completing a decommitment to* $\hat{\mathbf{r}}^{\overline{b}}$ *(defined in step 1). When* $a = b$, R *draws a random* $\mathbf{y} \xleftarrow{R} \mathbb{Z}_q^\kappa$ *and sets* $\hat{\mathbf{r}}^b = \mathbf{A}\big(f^\kappa(\mathbf{y}) - \mathbf{r}'\big) \in \mathbb{Z}_q^{3\kappa}$. *Finally,* R *decommits to* $x^{b,i}_{\overline{c}_{b,i},j}$ $\forall\ i$, *where* $j$ *is such that* $[\hat{\mathbf{r}}^b]_i = x^{b,i}_{0,j} + x^{b,i}_{1,j}$. *Note that* R *has decommitted to* $(\hat{\mathbf{r}}^0, \hat{\mathbf{r}}^1)$.

**4.** $\big(\mathrm{S} \longrightarrow \mathrm{R}\big)$**:** *For all* $(a,i)$, S *has received decommitments to all of the coordinates of exactly one of* $\mathbf{x}_0^{a,i}$ *and* $\mathbf{x}_1^{a,i}$. S *checks either that* $\mathbf{v}^{i,a} = \psi(\mathbf{x}_0^{a,i})$ *or that* $\mathbf{v}^{a,i} + \psi(\mathbf{x}_1^{a,i}) = 0$. *If any of these checks fails,* S *aborts. Otherwise,* S *computes vectors* $(\mathbf{z}_0, \mathbf{z}_1)$ *where* $\mathbf{z}_a \in \mathbb{Z}_q^\kappa$ *is the unique vector such that* $\mathbf{A}\mathbf{z}_a = \hat{\mathbf{r}}^a + \mathbf{A}\mathbf{r}'$ *(such a value exists by linearity). If no such* $\mathbf{z}_a$ *exists for some* $a$ *then* S *aborts.* S *sends* $(\mathbf{w}_0, \mathbf{w}_1)$ *to* R *where* $\mathbf{w}_a = \mathbf{s}_a - \mathbf{h}\big(f^{-\kappa}(\mathbf{z}_a)\big)$.

***Output:*** R *outputs* $\mathbf{s}_b = \mathbf{w}_b + \mathbf{h}(\mathbf{y})$.

### 3.2 Four-Round Fully Simulatable Oblivious Transfer from $\Pi^{\mathrm{R}}_{\mathsf{OT}}$

We transform the one-sided simulatable $\Pi^{\mathrm{R}}_{\mathsf{OT}}$ into an OT which is simulatable for both the sender and the receiver using the following ingredients. We use a $(\kappa + 1, 2\kappa)$-secure Shamir Secret sharing scheme. Let $A \in \mathbb{Z}_q^{2\kappa \times \kappa}$ be the Vandermonde matrix and let $\phi$ be a linear map such that $\phi(A) = 0$.

First, the sender picks two *random* keys $x_0, x_1$, and computes their correspondent vectors of $2\kappa$ shares $\mathbf{v}_0, \mathbf{v}_1$ according to Shamir secret sharing. Then, the sender commits to each coordinate of vectors $\mathbf{v}_0, \mathbf{v}_1$ and proves that they are valid shares, in a black-box way. We build this proof using the observation that $\mathbf{v}$ is a valid vector of shares for a $(\kappa + 1, 2\kappa)$-secure Shamir secret sharing, iff $\phi(\mathbf{v}) = 0$, and that for any pair of vectors $\mathbf{a}, \mathbf{b}$ it holds that if $\mathbf{a} + \mathbf{b} = \mathbf{v}$ then also $\phi(\mathbf{a}) + \phi(\mathbf{b}) = 0$. Thus, to prove that a vector $\mathbf{v}$ is a vector of valid shares, the sender will commit to $\kappa$ pairs of vectors $\mathbf{a}_j, \mathbf{b}_j$ such that $\mathbf{v} = \mathbf{a}_j + \mathbf{b}_j$, and prove that there exists at least a $j$ such that the predicate $\phi(\mathbf{a}) + \phi(\mathbf{b}) = 0$ holds. This proof is easily implemented by having the sender commit to $\mathbf{a}_j, \mathbf{b}_j$ and $\mathbf{z}_j = \phi(\mathbf{a}_j)$ and having the receiver ask to either open $\mathbf{a}_j$ and check that $\mathbf{z}_j = \phi(\mathbf{a}_j)$, or to open $\mathbf{b}_j$ and check that $\phi(\mathbf{b}_j) + \mathbf{z}_j = 0$. Note that this proof only guarantees that there exists at least one $j$ for which the condition $\phi(\mathbf{a}) + \phi(\mathbf{b}) = 0$ is true. Summing up, S will commit to vectors $\mathbf{a}_{0,j}\mathbf{b}_{0,j}$ and $\mathbf{a}_{1,j}\mathbf{b}_{1,j}$ (for shares $\mathbf{v}_0, \mathbf{v}_1$) with an extractable commitment scheme, and run a proof of validity for each such pair. In the last round S will send the encryptions $x_0 + s_0, x_1 + s_1$ of his actual secret inputs. Now we need a way for R to retrieve the decommitments of the shares for the secret he is interested in, without the server knowing which decommitments are revealed. We accomplish this by using the OT protocol $\Pi^{\mathrm{R}}_{\mathsf{OT}}$ implemented above. Therefore, in parallel to such extractable commitments and proofs, the sender and the receiver will engage in $2\kappa$ parallel executions of $\Pi^{\mathrm{R}}_{\mathsf{OT}}$: in the $i$-th OT execution S plays with inputs the opening of the $i$-th coordinate

of $(\mathbf{a}_{0,j}, \mathbf{b}_{0,j})$ and $(\mathbf{a}_{1,j}, \mathbf{b}_{1,j})$ for *all* $j$, and R plays with bit $b_i$. Note that, opening to the $i$-th coordinate of all $j$ vectors allows the receiver to check that all $j$ vectors agree on the *same* coordinate $[v_{b_i}]_i$. This check, together with the proof of consistency provided above, will guarantee that most of the shares received via OT (and extracted by the simulator via the extractable commitments) are valid.

Before attempting to reconstruct the secret, R will test the consistency of $\kappa/2$ coordinates for vector $\mathbf{v}_0$ and $\mathbf{v}_1$ by playing with bit 0 and 1 accordingly, in the correspondent OTs, while he plays with the his secret bit $b$ for the remaining $\kappa$ executions. R will attempt to reconstruct the vector $\mathbf{v}_b$ only if the consistency test passes. We provide a formal description of such steps in Protocol 3.

**Protocol 3** *($\Pi_{\mathsf{OT}}$).* Sub-protocols. *Let* $\Pi^{\mathrm{R}}_{\mathsf{OT}} = \{\mathsf{OT1}, \mathsf{OT2}, \mathsf{OT3}, \mathsf{OT4}\}$ *denote the 4 messages exchanged in protocol* $\Pi^{\mathrm{R}}_{\mathsf{OT}}$ *(Prot. 2). Let* $\mathsf{OT}[\mathrm{i}]$ *denote the* $i$*-th parallel execution of* $\Pi^{\mathrm{R}}_{\mathsf{OT}}$*. Let* $\mathsf{ExtCom} = (\mathsf{ExtCom1}, \mathsf{ExtCom2}, \mathsf{ExtCom3})$ *be a 3-round statistically binding extractable commitment scheme with non-interactive decommitment* $\mathsf{ExtDec}$*. Let* $\mathsf{Share}, \mathsf{Recon}$ *be a* $(\kappa+1)$*-out-of-*$2\kappa$ *Shamir secret sharing scheme over* $\mathbb{Z}_p$*, together with a linear map* $\psi : \mathbb{Z}_p^{2\kappa} \to \mathbb{Z}_p^{\kappa-1}$ *such that* $\psi(\mathbf{v}) = 0$ *iff* $\mathbf{v}$ *is a valid sharing of some secret.*

**Public Input:** *A prime* $p$ *and* $\ell = \lfloor \log q \rfloor$ *st* $2^\ell/p = 1 - \mathbf{negl}(\kappa)$*, a Vandermonde matrix* $A \in \mathbb{Z}_p^{2\kappa \times \kappa}$*, linear map* $\phi$*.*
**Sender's Input:** $s_0, s_1 \in \mathbb{Z}_p$. **Receiver's Input:** $b \in \{0,1\}$.

1. $(\mathrm{R} \longrightarrow \mathrm{S})$**:** R *randomly chooses a set* $\mathsf{T}_{1-b} \in [2\kappa]$ *of* $\kappa/2$ *coordinates.* R *plays the* $i$*-th execution of* $\Pi^{\mathrm{R}}_{\mathsf{OT}}$ *with input* $b_i = (1-b)$*. For the remaining* $i \notin \mathsf{T}_{1-b}$ *set* $b_i = b$ R *sends* $(\mathsf{OT1}[1], \ldots, \mathsf{OT1}[2\kappa])$ *to* S*, where* $\mathsf{OT1}[\mathrm{i}]$ *is computed on input* $b_i$*.*
2. $(\mathrm{S} \longrightarrow \mathrm{R})$**:** *Upon receiving a correct first message,* S *proceeds as follows.*
   - *Pick random strings* $x_0, x_1 \in \mathbb{Z}_p$ *and secret share each string: Compute shares* $\mathbf{v}_b = ([v_b]_1, \ldots, [v_b]_{2\kappa}) \leftarrow \mathsf{Share}(x_b)$ *for* $b \in \{0,1\}$*.*
   - *To commit to shares* $\mathbf{v}_0, \mathbf{v}_1$ *and prove that they are valid shares of a* $\kappa$*-degree polynomial* S *proceeds as follows.*
     - *For* $j = 1, \ldots, \kappa$*, pick random* $\mathbf{a}_{0,j}, \mathbf{b}_{0,j} \in \mathbb{Z}_p^{2\kappa}$ *such that* $\mathbf{a}_{0,j} + \mathbf{b}_{0,j} = \mathbf{v}_0$ *and compute* $\mathbf{z}_{0,j} = \phi(\mathbf{a}_{0,j})$ *for all* $j$*. Resp., compute* $\mathbf{a}_{1,j}, \mathbf{b}_{1,j} = \mathbf{v}_1$
     - *Commit to each coordinate of* $\mathbf{a}_{b,j}$ *and* $\mathbf{b}_{b,j}$ *using* $\mathsf{ExtCom}$*, namely send* $\mathsf{acom}_{b,j,i} = \mathsf{ExtCom1}([a_{b,j}]_i)$*,* $\mathsf{bcom}_{b,j,i} = \mathsf{ExtCom1}([b_{b,j}]_i)$*.*

   S *sends to* R *the messages* $(\mathsf{OT2}[1], \ldots, \mathsf{OT2}[2\kappa])$*,* $\{\mathsf{ExtCom1}([a_{b,j}]_i)$*,* $\{\mathsf{ExtCom1}([b_{b,j}]_i)\}_{i \in 2\kappa}$*, and* $\mathbf{z}_{b,j}$ *for* $b = 0, 1$ *and* $j \in [\kappa]$*.*
3. $(\mathrm{R} \longrightarrow \mathrm{S})$**:** R *sends* $(\mathsf{OT3}[1], \ldots, \mathsf{OT3}[2\kappa])$*, the second message* $\mathsf{ExtCom2}$ *for the extractable commitment, and a random challenge* $c_1, \ldots, c_\kappa \in \{0,1\}^\kappa$*.*
4. $(\mathrm{S} \longrightarrow \mathrm{R})$**:** S *computes OT message* $\mathsf{OT4}[\mathrm{i}]$ *using as inputs the* $i$*-th coordinate of all* $j$ *vectors committed before. Specifically, in the* $i$*-th OT it uses decommitment to values* $(a_{[0,j]}]_i, [b_{0,j}]_i)\, \forall j$*;* $(a_{[1,j]}]_i, b_{1,j}]_i)\, \forall j$*. Additionally, for each* $j$*,* S *reveals vector* $\mathbf{a}_{0,j}, \mathbf{a}_{1,j}$ *if* $c_j = 0$*; or vectors* $\mathbf{b}_{0,j}, \mathbf{b}_{1,j}$ *if* $c_j = 1$*; and the messages for the third round of the extractable commitments, namely* $\{\mathsf{ExtCom3}([a_{b,j}]_i), \{\mathsf{ExtCom3}([b_{b,j}]_i)\}_{i \in 2\kappa, j \in \kappa}$*. Finally,* S *sends* $C_0 = s_0 \oplus x_0$ *and* $C_1 = s_1 \oplus x_1$*.*

**Verification and Output:** *If the extractable commitments are all successfully completed, proceeds as follows.*

- **Check Validity of Shares.** *For* $j = 1, \ldots, \kappa$, *if* $c_j = 0$ *check that* $\mathbf{z}_{0,j} = \phi(\mathbf{a}_{0,j})$ *and* $\mathbf{z}_{1,j} = \phi(\mathbf{a}_{1,j})$. *Else, if* $c_j = 1$ *check that* $\phi(\mathbf{b}_{0,j}) + \mathbf{z}_{0,j} = 0$ *and* $\phi(\mathbf{b}_{1,j}) + \mathbf{z}_{0,j} = 1$.
- **Test Phase.** R *randomly chooses a set* $\mathsf{T}_b$ *of* $\kappa/2$ *coordinates in* $\{[2\kappa]/\mathsf{T}_{1-b}\}$. *For each* $i \in \mathsf{T}_\sigma$, *with* $\sigma \in \{0,1\}$; *let* $[a_{\sigma,j}]_i$, $[b_{\sigma,j}]_i$ *be the coordinates obtained from the* $i$*-th OT.* R *checkes that, for all* $j$, *there exists a unique* $[v_\sigma]_i$ *such that* $[a_{\sigma,j}]_i + [b_{\sigma,j}]_i = [v_\sigma]_i$. *If so,* $[v_\sigma]_i$ *is then marked as* **consistent**. *If all shares obtained in this phase are consistent,* R *proceeds to the reconstruction phase. Else abort.*
- **Reconstruction Phase.** *For* $i \in \{[2\kappa]/\mathsf{T}_{1-b}\}$, *if there exists a unique* $[v_b]_i$ *such that* $[a_{b,j}]_i + [b_{b,j}]_i = [v_b]_i$, *mark share* $[v_b]_i$ *as* **consistent**. *If* R *obtains less than* $\kappa + 1$ *consistent shares, he aborts. Else, let* $[v_b]_{j_1}, \ldots, [v_b]_{j_{\kappa+1}}$ *be any set of* $\kappa + 1$ *consistent shares.* R *computes* $x_b \leftarrow \mathsf{Recon}([v_b]_{j_1}, \ldots, [v_b]_{j_{\kappa+1}})$ *and outputs* $s_b = C_b \oplus x_b$.

### 3.3 Proof of Security

In this section we provide the intuition behind the security of our constructions. The reader is referred to the full version for the complete proof.

**Security of $\Pi^{\mathbf{R}}_{\mathsf{OT}}$.** We start by proving that $\Pi^{\mathrm{R}}_{\mathsf{OT}}$ is one-sided simulatable.

*Indistinguishability Against a Malicious Sender.* It follows from the hiding of the commitment scheme used by R to commit to the secret vectors $\mathbf{r}^0, \mathbf{r}^1$. Indeed, the only difference between the transcript of a completed execution of $\Pi^{\mathrm{R}}_{\mathsf{OT}}$ when R uses input bit $b = 0$ and when $R$ uses bit 1 is in the matrices that are computed equivocally. In turn, the equivocal matrix differs from a binding matrix in that the sum of the rows of an equivocal $\mathbf{M}^{i,b}$ leads to the vector of all permuted values in $\mathbb{Z}_q$ while in a binding matrix the sum of the row of the $i$-th matrix corresponds to the vector $\hat{\mathbf{r}}^b_i$.

*Simulatability Against a Malicious Receiver.* For the case of a malicious receiver, we build a simulator who rewinds S and extracts R's input bit from the coin-flipping protocol. Note that when R's input bit is $b$, R commits equivocally to the matrices $\mathbf{M}^{b,i}$, and therefore cannot commit equivocally to the $\mathbf{M}^{\bar{b},i}$. It follows that when R is rewound and asked a new query, his decommitment from the $\bar{b}$ matrices will be the same. In this way, our simulator can figure out R's input bit. It remains to show that a malicious R cannot gain some advantage by committing equivocally to some of the $\mathbf{M}^{0,i}$ and some of the $\mathbf{M}^{1,i}$. This follows from the error-correction property guaranteed by the choice of the matrix $\mathbf{A}$. A more detailed proof is provided in the full version.

**Security of $\Pi_{\mathsf{OT}}$.** We now sketch the main ideas behind the security of $\Pi_{\mathsf{OT}}$. Correctness follows from the correctness of the underlying $\Pi^{\mathrm{R}}_{\mathsf{OT}}$ protocol, the correctness of the statistically binding commitment scheme and the Shamir secret sharing scheme: the receiver will be able to retrieve more than $\kappa + 1$ shares and reconstruct the key $x_b$ that allows to decrypt $s_b$. We now analyze the security of the protocol in case either of the parties is corrupted.

*Simulatability Against Malicious Receiver.* We show a PPT simulator that simulates the attack of the receiver in the ideal world as follows. $\mathsf{Sim}$ computes the messages of protocol $\Pi_{\mathsf{OT}}$ honestly till the third round, by committing to randomly selected $x_0, x_1$. In parallel, $\mathsf{Sim}$ extracts the bits played by $\mathrm{R}^*$ in protocol $\Pi^{\mathrm{R}}_{\mathsf{OT}}$ by running the simulator $\mathsf{Sim}^{\mathsf{OT}}_{\mathsf{R}}$ guaranteed by the one-sided simulatability property of $\Pi^{\mathrm{R}}_{\mathsf{OT}}$. $\mathsf{Sim}^{\mathsf{OT}}_{\mathsf{R}}$ outputs the bits $b_1, \ldots, b_{2\kappa}$ which are the selections made by $\mathrm{R}^*$ in the first 3 rounds of protocol $\Pi^{\mathrm{R}}_{\mathsf{OT}}$. (Note that in $\Pi^{\mathrm{R}}_{\mathsf{OT}}$ the server commits to its input only in the fourth round, when the selection has already beed committed. However, this will not be a problem because in $\Pi_{\mathsf{OT}}$ the sender is still using its secret input only in the last round). If there are more than $\kappa + 1$ bits pointing to the same bit $b$ then $\mathsf{Sim}$ sends this bit to $\mathcal{F}_{\mathsf{OT}}$ and receives the string $s_b$. Otherwise it will just send a random bit and continue the simulation of the protocol with random values. In the last round the simulator uses $\mathsf{Sim}^{\mathsf{OT}}_{\mathsf{R}}$ to complete the OT using in input the shares that were dictated by the bits $b_1, \ldots, b_{2\kappa}$ and it obtains messages $\mathsf{OT4}[\mathrm{i}]$ for $i \in [2\kappa]$. Finally, $\mathsf{Sim}$ completes the protocol by honestly computing message $\mathsf{CPmsg3}$, but it prepares $c_b = x_b \oplus s_b, c_{1-b} = r$, where $r$ is a randomly chosen string.

The indistinguishability of the simulation follows from the simulatability of the underlying OT, the security of Shamir secret sharing and the hiding of the underlying commitment scheme. We stress that in the proof we need to argue about the hiding of the unopened shares. Namely, we require to prove that the protocol satisfies a form of hiding in presence of selective opening attack. This is not a problem as our protocol is interactive and the positions that the receiver is choosing to open are fixed in advance before observing any commitment. This property allows us to prove indistinguishability by relying on standard hiding definition.

*Simulatability Against Malicious Sender.* We show a simulator that, having oracle access to the malicious sender $\mathrm{S}^*$, extracts both inputs $s_0, s_1$. $\mathsf{Sim}$ runs as receiver in the $\Pi_{\mathsf{OT}}$ protocol by choosing sets $\mathsf{T}_0$ and $\mathsf{T}_1$, and playing with a random bit in the remaining OT executions. Then, if the $\mathsf{Test\ phase}$ passes, $\mathsf{Sim}$ rewinds $\mathrm{S}^*$ to extract the vectors $(\mathbf{a}_{0,j}, \mathbf{b}_{0,j})$ and $(\mathbf{a}_{01,j}, \mathbf{b}_{1,j})$ from the extractable commitments. Due to the indistinguishability property of the underlying $\Pi^{\mathrm{R}}_{\mathsf{OT}}$ we have that any malicious sender cannot detect on which coordinates he will be tested. Therefore, if the test phase passes, then it holds that, for each bit, at least $\kappa/2 + 1$ of the remaining OT were computed correctly for that bit. Due to the binding of the commitment scheme, to the correctness of Shamir's secret sharing, and the correctness of the proof of consistency of the shares, the values reconstructed from the shares extracted by the simulator in the extractable

commitments correspond to the unique value that a honest receiver would have obtained from the shares retrieved via $\Pi_{\mathsf{OT}}^{\mathrm{R}}$.

### 3.4 Parallel OT

Protocol $\Pi_{\mathsf{OT}}$ can be used as a building block for constructing a protocol implementing the $\mathcal{F}_{\mathsf{OT}}^{m}$ functionality. The idea is to have the Sender S and the receiver R compute $m$ executions of $\Pi_{\mathsf{OT}}$ in parallel, and accepting a round of communication if and only if all the $m$ executions are computed correctly.

### 3.5 Round-Optimal Secure Two-Party Computation

The non-interactive secure two-party protocol proposed in [12] it is based on Yao [29] garbled circuits and works in the OT-hybrid model. The main contribution of [12] is to show an (asymptotically) more efficient black-box cut-and-choose for proving that a garbled circuit is computed correctly. The cut-and-choose is non-interactive in the OT-hybrid model. We can cast their construction to the simpler setting of stand-alone two-party computation and replace the ideal calls to the OT with our parallel OT $\Pi_{\mathsf{OT}}^{m}$.

**Acknowledgments.** We thank the anonymous reviewers for helpful comments. Work supported in part by NSF grants 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 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.

## References

1. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Simple, Black-box constructions of adaptively secure protocols. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 387–402. Springer, Heidelberg (2009)
2. Cramer, R., Damgård, I.B., Schoenmakers, B.: Proof of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994). http://dx.doi.org/10.1007/3-540-48658-5_19
3. Damgård, I.B., Ishai, Y.: Constant-round multiparty computation using a black-box pseudorandom generator. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 378–394. Springer, Heidelberg (2005). http://dx.doi.org/10.1007/11535218_23
4. Damgård, I., Scafuro, A.: Unconditionally secure and universally composable commitments from physical assumptions. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 100–119. Springer, Heidelberg (2013). http://dx.doi.org/10.1007/978-3-642-42045-0_6

5. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A.V. (ed.) Proceedings of the 19th Annual ACM Symposium on Theory of Computing, pp. 218–229. ACM, New York (1987). http://doi.acm.org/10.1145/28395.28420
6. Goyal, V.: Constant round non-malleable protocols using one-way functions. In: Proceedings of the 43rd Annual ACM Symposium on Theory of Computing, STOC 2011, pp. 695–704. ACM (2011)
7. Goyal, V., Lee, C.K., Ostrovsky, R., Visconti, I.: Constructing non-malleable commitments: A black-box approach. In: FOCS, pp. 51–60. IEEE Computer Society (2012)
8. Goyal, V., Ostrovsky, R., Scafuro, A., Visconti, I.: Black-box non-black-box zero knowledge. In: Symposium on Theory of Computing, STOC 2014, pp. 515–524 (2014)
9. Haitner, I.: Semi-honest to malicious oblivious transfer—the black-box way. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 412–426. Springer, Heidelberg (2008)
10. Hazay, C., Lindell, Y.: Efficient secure two-party protocols - techniques and constructions. In: Information Security and Cryptography. Springer (2010). http://dx.doi.org/10.1007/978-3-642-14303-8
11. Ishai, Y., Kushilevitz, E., Lindell, Y., Petrank, E.: Black-box constructions for secure computation. In: Proceedings of the 38th Annual ACM Symposium on Theory of Computing, STOC 2006, pp. 99–108 (2006)
12. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Prabhakaran, M., Sahai, A.: Efficient non-interactive secure computation. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 406–425. Springer, Heidelberg (2011)
13. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Proceedings of the 39th Annual ACM Symposium on Theory of Computing, STOC 2007, pp. 21–30 (2007)
14. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer – efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008). http://dx.doi.org/10.1007/978-3-540-85174-5_32
15. Katz, J., Ostrovsky, R.: Round-optimal secure two-party computation. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 335–354. Springer, Heidelberg (2004)
16. Kilian, J.: Founding cryptography on oblivious transfer. In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2–4, 1988, Chicago, Illinois, USA, pp. 20–31. ACM (1988)
17. Kilian, J.: A note on efficient zero-knowledge proofs and arguments. In: STOC, pp. 723–732 (1992)
18. Kiyoshima, S., Manabe, Y., Okamoto, T.: Constant-round black-box construction of composable multi-party computation protocol. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 343–367. Springer, Heidelberg (2014)
19. Lapidot, D., Shamir, A.: Publicly verifiable non-interactive zero-knowledge proofs. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 353–365. Springer, Heidelberg (1991)
20. Lin, H., Pass, R.: Black-box constructions of composable protocols without set-up. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 461–478. Springer, Heidelberg (2012)
21. Micciancio, D., Ong, S.J., Sahai, A., Vadhan, S.P.: Concurrent zero knowledge without complexity assumptions. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 1–20. Springer, Heidelberg (2006)

22. Ostrovsky, R., Rao, V., Scafuro, A., Visconti, I.: Revisiting lower and upper bounds for selective decommitments. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 559–578. Springer, Heidelberg (2013)
23. Ostrovsky, R., Scafuro, A., Venkitasubramanian, M.: Resettably sound zero-knowledge arguments from OWFs - the (semi) black-box way. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part I. LNCS, vol. 9014, pp. 345–374. Springer, Heidelberg (2015)
24. Pass, R., Wee, H.: Black-box constructions of two-party protocols from one-way functions. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 403–418. Springer, Heidelberg (2009)
25. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008)
26. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
27. Wee, H.: Black-box, round-efficient secure computation via non-malleability amplification. In: Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science, pp. 531–540 (2010)
28. Xiao, D.: (Nearly) round-optimal black-box constructions of commitments secure against selective opening attacks. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 541–558. Springer, Heidelberg (2011)
29. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: FOCS, pp. 162–167 (1986)