Black-Box Constructions of Composable Protocols without Set-Up

Huijia Lin[1,⋆] and Rafael Pass[2,⋆⋆]

[1] MIT and Boston University
huijia@csail.mit.edu
[2] Cornell University
rafael@cs.cornell.edu

Abstract. We present the first *black-box* construction of a secure multi-party computation protocol that satisfies a meaningful notion of *concurrent security* in the plain model (without any set-up, and without assuming an honest majority). Moreover, our protocol relies on the minimal assumption of the existence of a semi-honest OT protocol, and our security notion "UC with super-polynomial helpers" (Canetti et al, STOC'10) is closed under universal composition, and implies super-polynomial-time simulation security.

1 Introduction

The notion of *secure multi-party computation* allows m mutually distrustful parties to securely compute (or, *realize*) a functionality $f(\bar{x})$ of their corresponding private inputs $\bar{x} = x_1, ..., x_m$, such that party P_i receives the i^{th} component of $f(\bar{x})$. Loosely speaking, the security requirements are that the output of each party is distributed according to the prescribed functionality—this is called *correctness*—and that even malicious parties learn nothing more from the protocol than their prescribed output—this is called *privacy.* These properties should hold even in case that an arbitrary subset of the parties maliciously deviates from the protocol.

Soon after the concept was proposed [47], general constructions were developed that appeared to satisfy the intuitive correctness and secrecy for practically any multi-party functionality [47,19]. These constructions require only authenticated communication and can use any enhanced trapdoor permutation. However, definitions that capture the security properties of secure multi-party computation protocols (and, in fact, of secure cryptographic protocols in general) took more time to develop. Here, the *simulation paradigm* emerged as a natural approach: Originally developed for capturing the security of encryption and then extended to Zero-Knowledge [21,22]. The idea is to say that a protocol π securely realizes f if running π *"emulates"* an idealized process where all parties secretly provide inputs to an imaginary trusted party that computes f and returns the outputs to the parties; more precisely, any "harm" done by

⋆ Supported in part by a DARPA Grant FA8750-11-2-0225, NSF Grant CCF-1018064.
⋆⋆ Supported in part by a Microsoft New Faculty Fellowship, NSF CAREER Award CCF-0746990, AFOSR Award FA9550-08-1-0197 and BSF Grant 2006317.

a polynomial-time adversary in the real execution of π, could have been done even by a polynomial-time adversary (called a *simulator*) in the ideal process. The simulation paradigm provides strong security guarantees: It ensures that running the protocols is "as good as" having a trusted third party computing the functionality for the players, and an adversary participating in the real execution of the protocols does not gain any "computational advantage" over the simulator in the ideal process (except from polynomial time advantage). We call this definition *basic security.*

The original setting in which secure multi-party protocols were investigated, however, only allowed the execution of a single instance of the protocol at a time; this is the so called stand-alone setting. A more realistic setting, is one which allows the concurrent execution of protocols. In the concurrent setting, many protocols are executed at the same time. This setting presents a new risk of a "coordinated attack" in which an adversary interleaves many different executions of a protocol and chooses its messages in each instance based on other partial executions of the protocol. To prevent coordinated attacks, we require the following basic security guarantee:

> **Concurrent Security:** The security properties, correctness and privacy, of the *analyzed protocol* should remain valid even when multiple instance of the protocol are concurrently executed in a potentially unknown environment.

Another natural desideratum is the capability of supporting modular design of secure protocols.

> **Modular Analysis:** The notion of security should support designing composite protocols in a modular way, while preserving security. That is, there should be a way to deduce security properties of the overall protocol from security properties of its components. This is essential for asserting security of complex protocols.

Unfortunately, these properties are not implied by the basic security. In the literature, the strongest and also the most realistic formalization of concurrent security is the notion of Universal Composability (UC) [5]: It considers the concurrent execution of an unbounded number of instances of the analyzed protocol, in an arbitrary, and adversarially controlled, network environment. It also supports modular analysis of protocols. But, these strong properties come at a price: Many natural functionalities cannot be realized with UC security in the *plain model,* where players only have access to authenticated communication channels; some additional trusted set-up is necessary [7,8]; furthermore, the need for additional trusted set up extends to any protocol that only guarantees a concurrent extension of basic security [35]. A large body of works (e.g. [10,1,28,11,24,29,6]) have shown that indeed, with the appropriate trusted set-ups, UC-security becomes feasible. However, in many situations, trusted set-up is hard to come by (or at least expensive). It is thus important to have a notion of concurrent security that can be achieved in the plain model. Several notions of concurrent security have since been proposed.

Concurrent Security in the Plain Model. *Security with super-polynomial simulators (SPS)* [39] is a relaxation of UC security that allows the adversary in the ideal execution to run in super-polynomial time. Informally, this corresponds to guaranteeing that "any polytime attack that can be mounted against the protocol can also be mounted in the ideal execution—albeit with super-polynomial resources." Although SPS security is sometimes weaker than basic security, it often provides an adequate level of security. In constrast to basic security, however, SPS directly considers security in the concurrent setting. Protocols that realize practically any functionality with SPS security in the plain model were shown based on sub-exponential hardness assumptions [39,2,33]. Very recently, improved constructions are presented [9,17,34] that are based on only standard polynomial-time hardness assumptions. Another notion of security that is closely related to SPS security is input indistinguishability. It is shown in [38] that input indistinguishable protocols for general functionalities can be constructed from standard polynomial time hardness assumptions.

One drawback of SPS security that it is not closed under composition; thus it is not a convenient basis for modular analysis of protocols. *Angel-based UC security* [43] is a framework for notions of security that provides similar security guarantees as SPS and at the same time supports modular analysis. Specifically, angel-based security considers a model where both the adversary and the simulator have access to an oracle (an "angel") that allows some judicious use of super-polynomial resources. Since the angels can be implemented in super-polynomial time, for any angel, angel-based security implies SPS security. Furthermore, akin to UC security, angel-based UC security, with any angel, can be used as a basis for modular analysis. Prabhakaran and Sahai [43] exhibited an angle with respect to which practically all functionalities can be securely realized; later another angle is given by [37]; both constructions, however, rely on some non-standard hardness assumptions.

Recently, Canetti, Lin and Pass [9] proposed a new notion of security, called *UC with super-polynomial time helpers.* This notion is very similar to the angel-based security where both the adversary and the simulator have access to a helper that provides some super-polynomial time help through a limited interface. Like angel-based security, UC security with super-polynomial time helpers implies SPS security. But, unlike angel-based security where angels are non-interactive and stateless, the helpers are *highly interactive and stateful.* Canetti, Lin and Pass [9] then constructed protocols that realize practically all functionalities with respect to a particular super-polynomial-time interactive helper, based on the existence of enhanced trapdoor permutations.

Summarizing the state-of-the-art, there are constructions [9,17,34] of protocols satisfying a meaningful notion of concurrent security—SPS security—in the plain model based on standard polynomial time hardness assumptions. Furthermore, the construction of [9] also supports modular analysis. (The constructions of [17,34] are better in terms of round-complexity—they only require a constant number of communication rounds—but they only acheive "non-composable" SPS security).

However, all these constructions are *non-black-box*, that is, the constructed protocols make non-black-box use of the underlying primitives. In fact, these constructions all follow the "Feige-Shamir" paradigm [16]: The protocols contain "trapdoors" embedded into the messages of the protocol, allowing a super-polynomial time simulator to extract the trapdoor and simulate messages in the protocol by proving that "it knows the trapdoor". In general, protocols following this approach seem hard to turn into a "practical" protocol for secure computations; as such, there results should only be viewed as "feasibility results" regarding concurrent secure computation without set-ups, but not candidates for practical purposes.

In contrast, black-box constructions that only use the underlying primitives through their input/output interfaces, are often much more efficient and are more suitable for implementation. Therefore, a series of recent works [14,26,27,36,46,23] have focused on constructing *black-box construction of secure computation protocols*, as an important step towards bringing secure multi-party computation closer to the practice. However, their constructions are all in either the stand-alone setting or rely on strong trusted set-ups (e.g., trusted hardware). This leaves open the following basic questions:

> *Can we obtain a* black-box *construction of concurrently secure protocols in the plain model (preferrably based only standard polynomial-time assumptions)?*
>
> *Can we have such a black-box construction that also satisfies a notion of security supporting composability?*

1.1 Our Results

We present a black-box construction of protocols that satisfy UC security with super-polynomial time helper for a specific helper, based on the existence of a stand-alone semi-honest oblivious transfer (OT) protocols. The framework of UC with super-polynomial time helper of [9] is formalized through the extended UC (EUC) framework of [6]; it is identical to the standard UC model [4] except that the corrupted parties (and the environement) have access to an super-polynomial time entity $\mathcal{H}$, called a helper functionality.

Main Theorem (Informally Stated): *Assume the existence of stand-alone semi-honest oblivious transfer protocols. Then there exists a sub-exponential-time computable interactive machine $\mathcal{H}$ such that for any "well-formed"*[1] *polynomial-time functionality $\mathcal{F}$, there exists a protocol that realizes $\mathcal{F}$ with $\mathcal{H}$-EUC security, in the plain model. Furthermore, the protocol makes only black-box calls to the underlying oblivious transfer protocol.*

As far as we know, this is the first black-box construction of secure multi-party computation protocols that achieve any non-trivial notion of concurrent security in the plain model (without any trusted-set up, and without assuming an honest majority).

[1] See [10] for a definition of well-formed functionalities.

The main technical tool used in our construction is a new notion of a commitment that is secure against adaptive Chosen Commitment Attack (CCA security). The notion of CCA secure commitments was previously introduced in [9]. Roughly speaking, a tag-based commitment scheme (i.e., commitment scheme that take an identifier—called the tag—as an additional input) is said to be *CCA-secure* if the value committed to using the tag id remains hidden even if the receiver has access to a (super-polynomial time) oracle that "breaks" commitments using any tag $\mathsf{id}' \neq \mathsf{id}$, where by breaking, it means the oracle returns a decommitment of the commitment. Thus the oracle is called a decommitment oracle. In [9], a commitment scheme that is CCA-secure w.r.t. a decommimient oracle is constructed based on the minimal assumption of one-way functions. However, their construction is non-black-box. In this work, to obtain black-box secure computation protocols, we need a new *black-box* construction of a CCA-secure commitment scheme. Towards this, we weaken the notion of CCA security w.r.t. decommitment oracle to instead consider an oracle that "breaks" commitments by returning only the unique committed value[2] (instead of the the decommitment information); we call this the committed-value oracle. We then provide a black-box construction of a commitment scheme that is CCA-secure w.r.t. the committed-value oracle.

Theorem (Informally Stated): *Assume the existence of one-way functions. Then, for every $\epsilon > 0$, there exists an $O(n^\epsilon)$-round commitment scheme that is CCA-secure w.r.t. the committed-value oracle and only relies on black-box access to one-way functions (where n is the security parameter).*

1.2 Outline

In Section 2, we define the notion of CCA-security w.r.t. the committed-value oracle. In Section 3, we first reduce the task of achieving UC security with super-polynomial time helpers to the task of constructing a UC-OT (with super-polynomial time helpers); we then sketch our construction of the UC-OT protocol, using CCA-secure commitments. Finally, in Section 4, we present our black-box robust CCA-secure commitment scheme.

2 Definition of CCA-Secure Commitments

We assume familiarity with the definition of commitment schemes and the statistically/computational binding and statistically/computational hiding properties. Unless specified otherwise, by a commitment scheme, we mean one that is statistically binding and computationally hiding. A *tag-based commitment schemes* with $l(n)$-bit identities [40,15] is a commitment scheme where, in addition to the security parameter 1^n, the committer and the receiver also receive a "tag"—a.k.a. the identity—id of length $l(n)$ as common input.

[2] The oracle returns $\perp$ if there is no unique committed value.

2.1 CCA-Security w.r.t. Committed Value Oracle

Let $\langle C, R\rangle$ be a tag-based commitment scheme with $l(n)$-bit identities. A committed-value oracle $\mathcal{O}$ of $\langle C, R\rangle$ acts as follows in interaction with an adversary A: It participates with A in many sessions of the commit phase of $\langle C, R\rangle$ as an honest receiver, using identities of length $l(n)$, chosen adaptively by A. At the end of each session, if the session is *valid*, it reveals the unique committed value of that session to A; otherwise, it sends $\perp$. (If a session has multiple committed values, the committed-value oracle also returns $\perp$. The statistically binding property guarantees that this happens with only negligible probability.) Loosely speaking, a tag-based commitment scheme $\langle C, R\rangle$ is said to be CCA-secure w.r.t. the committed-value oracle, if the hiding property of the commitment holds even with respect to adversaries with access to the committed-value oracle $\mathcal{O}$. More precisely, denote by $A^{\mathcal{O}}$ the adversary A with access to the committed-value oracle $\mathcal{O}$. Let $\mathsf{IND}_b(\langle C, R\rangle, A, n, z)$, where $b \in \{0,1\}$, denote the output of the following probabilistic experiment: on common input 1^n and auxiliary input z, $A^{\mathcal{O}}$ (adaptively) chooses a pair of challenge values $(v_0, v_1) \in \{0,1\}^n$—the values to be committed to—and an identity $\mathsf{id} \in \{0,1\}^{l(n)}$, and receives a commitment to v_b using identity id. Finally, the experiment outputs the output y of $A^{\mathcal{O}}$; the output y is replaced by $\perp$ if during the execution A sends $\mathcal{O}$ any commitment using identity id (that is, any execution where the adversary queries the decommitment oracle on a commitment using the same identity as the commitment it receives, is considered invalid).

Definition 1 (CCA-secure Commitments) . *Let $\langle C, R\rangle$ be a tag-based commitment scheme with $l(n)$-bit identities. We say that $\langle C, R\rangle$ is* CCA-secure w.r.t. the committed-value oracle*, if for every $\mathcal{PPT}$ ITM A, the following ensembles are computationally indistinguishable:*

- $\{\mathsf{IND}_0(\langle C, R\rangle, A, n, z)\}_{n\in N, z\in\{0,1\}^*}$
- $\{\mathsf{IND}_1(\langle C, R\rangle, A, n, z)\}_{n\in N, z\in\{0,1\}^*}$

2.2 *k*-Robustness w.r.t. Committed-Value Oracle

Consider a man-in-the-middle adversary that participates in an *arbitrary* left interaction with a *limited number of rounds*, while having access to a committed-value oracle. Roughly speaking, $\langle C, R\rangle$ is k-robust if the (joint) output of every k-round interaction with an adversary having access to the oracle $\mathcal{O}$, can be simulated without the oracle. In other words, having access to the oracle does not help the adversary in participating in any k-round protocols much.

Definition 2 . *Let $\langle C, R\rangle$ be a tag-based commitment scheme with $l(n)$-bit identities. We say that $\langle C, R\rangle$ is* k-robust w.r.t. the committed-value oracle*, if there exists a simulator S, such that, for every $\mathcal{PPT}$ adversary A, the following two conditions hold.*

Simulation: *For every $\mathcal{PPT}$ k-round ITM B, the following two ensembles are computationally indistinguishable.*

- $\left\{\mathsf{output}_{B,A^{\mathcal{O}}}[\langle B(y), A^{\mathcal{O}}(z)\rangle(1^n, x)]\right\}_{n\in N, x,y,z\in(\{0,1\}^*)^3}$
- $\left\{\mathsf{output}_{B,S^{A}}[\langle B(y), S^{A(z)}\rangle(1^n, x)]\right\}_{n\in N, x,y,z\in(\{0,1\}^*)^3}$

where $\mathsf{output}_{A,B}[\langle B(y), A(z)\rangle(x)]$ *denote the joint output of* A *and* B *in an interaction between them, on common input* x *and private inputs* z *to* A *and* y *to* B *respectively, with uniformly and independently chosen random inputs to each machine.*

Efficiency: *There exists a polynomial* t *and a negligible function* μ*, such that, for every* $n \in N$*,* $z \in \{0,1\}^*$ *and* $x \in \{0,1\}^*$*, and every polynomial* T*, the probability that* S *with oracle access to* $A(z)$ *and on input* $1^n, x$*, runs for more than* $T(n)$ *steps is smaller than* $\frac{t(n)}{T(n)} + \mu(n)$*.*

The following proposition shows that to construct a k-robust CCA-secure commitment scheme for identities of length n, it suffices to construct one for identities of length $\ell(n) = n^{\varepsilon}$. The same proposition is established in [9] for robust CCA-security w.r.t. decommitment oracles, and the proof there also applies to CCA-security w.r.t. committed-value oracles; we omit the proof here.

Proposition 1. *Let* ε *be any constant such that* $0 < \varepsilon < 1$*,* ℓ *a polynomial such that* $\ell(n) = n^{\varepsilon}$*, and* $\langle C, R\rangle$ *a* γ*-round* k*-robust CCA-secure commitment scheme (w.r.t. the committed-value oracle) with* ℓ*-bit identities. Then assuming the existence of one-way functions, there exists a* $\gamma + 1$*-round* k*-robust CCA-secure commitment scheme* $\langle \hat{C}, \hat{R}\rangle$ *(w.r.t. the committed-value oracle) with* n*-bit identities.*

3 BB UC-Secure Protocols with Super-Poly Helpers

We consider the model of UC with super-polynomial helper introduced in [9]. At a very high-level, this model is essentially the same as the UC-model introduced by [4], except that both the adversary and the environment in the real and ideal worlds have access to a super-polynomial time functionality that acts as a helper. See [9] for a formal definition of the model. In this section, we show:

Theorem 1. *Let* δ *be any positive constant. Assume the existence of a* T'_{OT}*-round stand-alone semi-honest oblivious transfer protocol. Then there exists a super-polynomial time helper functionality* $\mathcal{H}$*, such that, for every well-formed functionality* $\mathcal{F}$*, there exists a* $O(\max(n^{\delta}, T'_{OT}))$*-round protocol* Π *that* $\mathcal{H}$*-EUC-emulates* $\mathcal{F}$*. Furthermore, the protocol* Π *only uses the underlying oblivious transfer protocol in a black-box way.*

3.1 Overview of Our Construction

Towards Theorem 1, we need to first exhibit a super-polynomial time helper functionality $\mathcal{H}$. Roughly speaking, $\mathcal{H}$ simply acts as the committed-value oracle of a CCA secure commitment scheme. More precisely, consider the following two building blocks: First, given any $T'_{OT}(n)$-round stand-alone semi-honest OT protocol, it follows from previous works [26,25] that there exists an $T_{OT}(n)$-round OT protocol $\langle S, R\rangle$ that is secure against a malicious sender and a semi-honest receiver—called mS-OT protocol for short—that only relies on black-box access to the semi-honest OT protocol; furthermore $T_{OT} = O(T'_{OT}(n))$. Second, we need a $T_{OT}(n)$-robust CCA-secure commitment scheme $\langle C, R\rangle$, whose committed-value oracle $\mathcal{O}$ can be computed in sub-exponential time.[3] As we will show in the next section such a protocol exists with $O(\max(T_{OT}, n^{\delta})) = O(\max(T'_{OT}, n^{\delta}))$ rounds, relying on the underlying OWF in a black-box way. Since OWFs can be constructed from a semi-honest OT protocol in a black-box way. Therefore, we have that the second building block can also be based on the semi-honest OT protocols in a black-box way.

Consider a helper functionality $\mathcal{H}$ that "breaks" commitments of $\langle C, R\rangle$ in the same way as its committed-value oracle $\mathcal{O}$ does, subject to the condition that player P_i can only query the functionality on commitments that uses identity P_i. More precisely, every party P_i in a secure computation can simultaneously engage with $\mathcal{H}$ in multiple sessions of the commit phase of $\langle C, R\rangle$ as a committer using identity P_i, where the functionality simply forwards all the messages internally to the committed-value oracle $\mathcal{O}$, and forwards P_i the committed value returned from $\mathcal{O}$ at the end of each session. Since the committed-value oracle $\mathcal{O}$ can be computed in sub-exponential time, this functionality can also be implemented in sub-exponential time.

We show that Theorem 1 holds w.r.t. the helper functionality defined above in two steps. First, note that to realize any well-formed functionality in a black-box way, it suffices to realize the *ideal oblivious transfer functionality* $\mathcal{F}_{\mathsf{OT}}$. This is because it follows from previous works [31,3,20,27] that every functionality can be UC securely implemented in the $\mathcal{F}_{OT}$-hybrid model, even w.r.t. super-polynomial time environments. Based on previous works, [9] further shows that by considering only dummy adversaries and treating environments with access to a super-polynomial functionality $\mathcal{H}$ as sub-exponential time machines, we have that every functionality can be $\mathcal{H}$-EUC securely implemented in the $\mathcal{F}_{\mathsf{OT}}$ model. Formally, we have the following lemma from [9].

Lemma 2. *Fix any super-polynomial time functionality $\mathcal{H}$. For every well-formed functionality $\mathcal{F}$, there exists a constant-round $\mathcal{F}_{OT}$-hybrid protocol that $\mathcal{H}$-EUC-emulates $\mathcal{F}$.*

Next we show how to implement the $\mathcal{F}_{OT}$ functionality in the $\mathcal{H}$-EUC model. Then combining with Lemma 2, we conclude Theorem 1.

Lemma 3. *Let δ be any positive constant. Assume the existence of a T'_{OT}-round semi-honest oblivious transfer protocol. Then there exists a $O(\max(n^{\delta}, T'_{OT}))$-round protocol Π_{OT} that $\mathcal{H}$-EUC-emulates $\mathcal{F}_{\mathsf{OT}}$. Furthermore, the protocol Π_{OT} only uses the underlying oblivious transfer protocol in a black-box way.*

[3] This can be instantiated by simply using a normal T_{OT}-robust CCA secure commitment that has an exponential time committed value $\mathcal{O}$, with a "scaled-down" security parameter.

3.2 Overview of the OT Protocol Π_{OT}

In this section we provide an overview of our black-box construction of $\mathcal{H}$-EUC secure OT protocol Π_{OT}. Our construction is based on the black-box construction of an OT protocol secure against malicious players from a mS-OT protocol of [26,25]. Roughly speaking, the protocol of [26,25], relying on a stand-alone mS-OT protocol $\langle S, R\rangle$, proceeds in the following four stages:

Stage 1 (Receiver's Random Tape Generation). The sender and the receiver jointly decide the receiver's inputs and random tapes in Stage 2 using $2n$ parallel "coin tossing in the well" executions.

Stage 2 (OT with Random Inputs). The sender and the receiver perform $2n$ parallel OT executions of $\langle S, R\rangle$ using *random* inputs (s_j^0, s_j^1) and r_j respectively, where the receiver's inputs r_j's (and its random tapes) are decided in Stage 1.

Stage 3 (Cut-and-Choose). A random subset $Q \subset [2n]$ of n locations is chosen using a 3-round coin-tossing protocol where the sender commits to a random value first. (Thus the receiver knowing that random value can bias the coin-tossing output.) The receiver is then required to reveal its randomness in Stage 1 and 2 at these locations, which allows the sender to check whether the receiver behaved honestly in the corresponding OT executions. The randomness of the receiver at the rest of locations remains hidden.

Stage 4 (OT Combiner). Finally, for these locations $j \notin Q$ that are not open, the receiver sends $\alpha_j = u \oplus c_j$ where u is the receiver's true input. The sender replies with $\beta_0 = v_0 \oplus (\bigoplus_{j \notin Q} s_j^{\alpha_j})$ and $\beta_1 = v_1 \oplus (\bigoplus_{j \notin Q} s_j^{1-\alpha_j})$. The honest receiver obtains $s_j^{c_j}$'s through the OT execution, and thus can always recover v_u.

At a very high-level, the protocol of [26,25] augments security of the mS-OT protocol $\langle S, R\rangle$ to handle malicious receivers, by adding the cut-and-choose (as well as the random tape generation) stage to enforce the adversary behaving honestly in *most* (Stage 2) OT executions. (This is in a similar spirit as the non-black-box approach of requiring the receiver to prove that it has behaved honestly.) Then the security against malicious receivers can be based on that against semi-honest receivers of $\langle S, R\rangle$.

Wee [46] further augmented the stand-alone security of the protocol of [26,25] to achieve parallel security, that is, obtaining a protocol that is secure against man-in-the-middle adversaries that simultaneously acts as sender and receiver in many parallel executions. Towards this, Wee instantiated the commitments used in coin-tossing in Stage 3 of the above protocol, with ones that are satisfy a notion of "non-malleability w.r.t. extraction". Roughly speaking, non-malleability w.r.t. extraction [46] is a weaker notion than non-malleability of [15,32]; it guarantees that no matter what values the adversary is receiving commitments to, the committed values extracted out of the commitments from the adversary (with over-extraction) are indistinguishable. This guarantees that a simulator can bias the coin-tossing output by extracting the committed values from the adversary while the adversary cannot, as otherwise, by non-malleability w.r.t. extraction, it could do so even if the honest player sends a commitment to 0 instead of its true random challenge q. However, this is impossible as in this case no information of q is revealed. In other words, the coin-tossing protocol when instantiated with a non-malleable w.r.t. extraction commitment becomes parallel secure; Wee then relies on the parallel security of the coin-tossing protocol to show the parallel security of the OT protocol.

Towards $\mathcal{H}$-EUC-Secure OT Protocols, we need to further overcome two problems. First, we need to go from parallel security to concurrent security. In other words, we need a coin-tossing protocol that is concurrently secure. Informally speaking, non-malleability w.r.t. extraction guarantees that the simulator can extract the committed values of commitments from the adversary (to bias the output of the coin-tossing) while keeping the commitment to the adversary hiding amid rewindings (to ensure that the adversary cannot bias the output). However, this only holds in the parallel setting, as non-malleability only guarantees hiding of a commitment when values of the commitments from the adversary are extracted in parallel at the end of the execution. But, in the concurrent setting, the simulator needs to extract the committed values from the adversary in an on-line manner, that is, whenever the adversary successfully completes a commitment the committed value needs to be extracted. To resolve this problem, we resort to CCA-secure commitments, which guarantees hiding of a commitment even when the committed values are extracted (via the committed-value oracle) concurrently and immediately after each commitment. Now, instantiating the commitment scheme in the coin-tossing protocols with a CCA-secure commitment yields a coin-tossing protocol that is concurrently secure.

The second problem is that to achieve $\mathcal{H}$-EUC-security (similar to UC-security), we need to design a protocol that admits straight-line simulation. The simulator of a OT protocol has three tasks: It needs to simulate the messages of the honest senders and receivers, extract a choice from the adversary when it is acting as a receiver, and extract two inputs when it is acting as a sender. To achieve the first two tasks, the original simulation strategy in [26,25,46] relies on the capability of breaking the non-malleable commitments from the adversary using rewindings. When using CCA-secure commitments, the simulator can extract the committed values in a straight-line, by forwarding the commitment from the adversary to the helper functionality $\mathcal{H}$ that breaks the CCA commitments using brute force. For the last task, the original simulation strategy uses the simulator of the mS-OT protocol $\langle S, R \rangle$ against malicious senders to extract the adversary's inputs s_j^b's in all the Stage 3 OT executions, which then allows extraction of the real inputs v_0 and v_1 from the last message. However, the simulator of the mS-OT protocol may use rewindings. To solve this, one way is to simply assume a mS-OT protocol that has a straight-line simulator. We here however, present a different solution.

In our protocol, the sender and the receiver participate in parallel "coin tossing in the well" executions to decide the sender's random inputs s_j^b (and random tapes) in the parallel OT executions (besides the receiver's inputs and random tapes). Since the simulator can bias the coin-tossing in a straight line, it can determine the sender's inputs s_j^b's, which allows extraction of the sender's true inputs. For this to work, we need to make sure that a malicious sender would indeed uses the outputs of coin-tossing as inputs in the OT executions. Towards this, we again use the cut-and-choose technique: After the OT execution, the sender is required to reveal its randomness in the coin-tossing and OT execution at a randomly chosen subset of locations. The cut-and-choose technique guarantees that a malicious sender will behave consistently in most OT executions. Therefore the simulator extracts. the inputs s_j^b's correctly at *most* locations. However, in the protocol of [26,25,46], to recover the real inputs v_0 and v_1, the simulator needs to obtain *all* s_j^b's correctly. To bridge the gap, we modify the protocol to have the sender compute a random secret-sharing $\left\{a_j^b\right\}$ of each input v_b and hide each share using the appropriate s_j^b, that is, it sends $a_j^b \oplus s_j^{b \oplus \alpha}$ for every j (that is not open in the cut-and-choose procedures). Then, the simulator, able to extract most s_j^b's correctly, can recover enough shares to decode to the real inputs correctly. In contrast, a malicious receiver that is enforced to behave honestly in most OT executions by the cut-and-choose procedure, cannot obtain enough shares for both inputs and thus can only recover one of them. Finally, we remark that as in [46], to avoid over-extraction from the secret shares, we use the technique used in [12,13], which adds another cut-and-choose procedure. We defer the formal description of our OT protocol and its security proof (i.e., proof of Lemma 3) to the full version.

4 Black-Box Robust CCA-Secure Commitments

In this section, we present a *black-box* construction of a robust CCA-secure commitment scheme w.r.t. committed-value oracle based on one-way functions. For simplicity of exposition, the presentation below relies on a non-interactive statistically binding commitment scheme com; this can be replaced with a standard 2-round statistically binding commitment scheme using standard techniques[4].

4.1 Building Blocks

Our construction makes use of previous black-box constructions of extractable commitments and trapdoor commitment scheme. So let's start by reviewing them.

Extractable Commitments. Intuitively, an extractable commitment is one such that for any machine C^* sending a commitment, a committed value can be extracted from C^* if the commitment it sends is valid; otherwise, if the commitment is invalid, then no guarantee is provided, that is, an arbitrary garbage value may be extracted. This is known as the "over-extraction" problem. As shown

[4] This can be done by sending the first message of a 2-round commitment scheme at the beginning of the protocol, and using the second message of the 2-round commitment scheme w.r.t. that first message as a non-interactive commitment in the rest of the protocol.

in [41], the following protocol used in the works of [15,42,45] (also [30]) yields a black-box extractable commitment scheme ExtCom: To commit to a value $v \in \{0,1\}^m$, the committer and receiver on common input a security parameter 1^n, proceed as follows:

Commit: The committer finds n pairs of random shares $\{v_0^i, v_1^i\}_{i\in[n]}$ that sum up to v, (i.e., $v_0^i \oplus v_1^i = v$ for all $i \in [n]$) and commits to them in parallel using the non-interactive statistically binding commitment scheme com. Let c_b^i be the commitment to v_b^i.

Challenge: The receiver sends a n-bit string $ch \in \{0,1\}^n$ sampled at random.

Reply: The committer opens commitments $c_{ch_i}^i$ for every $i \in [n]$.

To decommit, the sender sends v and opens the commitments to all n pairs of strings. The receiver checks whether all the openings are valid and also $v = v_0^i \oplus v_1^i$ for all i.

It is proved in [41] that ExtCom is extractable. Furthermore, the commitment scheme has the property that from any two accepting transcripts of the commit stage that has the same commit message but different challenge messages, the committed value can be extracted. This property is similar to the notion of special-soundness for interactive proof/argument systems; here we overload this notion, and refer to this special extractability property of ExtCom as special-soundness.

In our construction, we will actually need an extractable commitment scheme to a string $\sigma \in \{0,1\}^m$ for which we can open any subset of the bits in σ without compromising the security (i.e. hiding) of the remaining bits. As shown in [41], we may obtain such a scheme $\mathsf{PExtCom}$ by running ExtCom to commit to each bit of σ in parallel. It is easy to see that $\mathsf{PExtCom}$ is also special-sound in the sense that, given two accepting transcripts of $\mathsf{PExtCom}$ that have the same commit message and two challenge messages that contain a pair of different challenges for every ExtCom commitment, the committed string σ can be extracted. We call such two transcripts a pair of $\mathsf{admissible\ transcripts}$ for $\mathsf{PExtCom}$.

Trapdoor Commitments. Roughly speaking, a trapdoor commitment scheme is a computationally biding and computationally hiding commitment scheme, such that, there exists a simulator that can generate a simulated commitment, and later open it to any value. (See [41] for a formal definition.) Pass and Wee [41] presented a black-box trapdoor bit commitment scheme $\mathsf{TrapCom}$. To commit to a bit σ, the committer and the receiver on common input 1^n do:

Stage 1: The receiver picks a random string challenge $e = (e_1, \ldots, e_n)$ and commits to e using the non-interactive statistically binding commitment scheme com.

Stage 2: The committer prepares $v_1, \ldots, v_n$. Each v_i is a 2×2 0,1-matrix given by $v_i = \left[v_i^{b_1,b_2}\right]_{2\times 2}$ where $v_i^{0,b} = \eta_i$, $v_i^{1,b} = \sigma \oplus \eta_i$, with η_i is a random bit. The sender commits to $v_1, \ldots, v_n$ using $\mathsf{PExtCom}$. In addition, the sender prepares $(a_1^0, a_1^1), \ldots, (a_n^0, a_n^1)$ where a_i^β is the opening to $v_i^{\beta 0}, v_i^{\beta 1}$ (i.e., either the top or bottom row of v_i).

Stage 3: The receiver opens to the challenge $e = (e_1, \ldots, e_n)$; the sender responds with $a_1^{e_1}, \ldots, a_n^{e_n}$.

To decommit, the sender sends σ. In addition, it chooses a random $\gamma \in \{0, 1\}$, and sends the openings to values $v_i^{0\gamma}, v_i^{1\gamma}$ for $i = 1, 2, \ldots, n$ (i.e., either the left columns or the right columns of all the matrices). The receiver checks that all the openings are valid, and also that $\sigma = v_1^{0\gamma} \oplus v_1^{1\gamma} = \cdots = v_n^{0\gamma} \oplus v_n^{1\gamma}$.

As shown in [41], the protocol $\mathsf{TrapCom}$ is trapdoor, following a Goldreich-Kahan [18] style proof; moreover, by running $\mathsf{TrapCom}$ in parallel, we obtain a trapdoor commitment scheme $\mathsf{PTrapCom}$ for multiple bits. Furthermore, since Stage 2 of the protocol $\mathsf{TrapCom}$ is simply an execution of $\mathsf{PExtCom}$, given any two *admissible transcripts* of Stage 2, the matrices $v_1, \ldots, v_n$ prepared in Stage 2 can be extracted; it is easy to see that from these matrices, the actual bit committed in the $\mathsf{TrapCom}$ commitment can be extracted, provided that the commitment is valid and has a unique committed value. We call this, again, the special-soundness of $\mathsf{TrapCom}$. Again, the notion of special soundness (and admissible transcripts) can be easily extended for $\mathsf{PTrapCom}$.

4.2 Overview of Our Construction

Towards a black-box construction of robust CCA secure commitment scheme, we start with the non-black-box construction of [9] (CLP), and tries to replace the non-black-box components in the CLP construction with "equivalent" black-box ones.

The CLP Construction: At a very high level, the CLP construction proceeds by having the committer first commit to the value v using a normal statistically binding commitment com, followed by a sequence of *poly*(n) $\mathcal{WISSP}$ proofs of the committed value. The $\mathcal{WISSP}$ proofs are the non-black-box component of the CLP construction, but are crucial for achieving CCA-security. Recall that proving CCA-security w.r.t. $\mathcal{O}$ amounts to showing that the views of A in experiments IND_0 and IND_1 are indistinguishable (when A has oracle access to $\mathcal{O}$). Let us refer to the adversary's interaction with C as the *left interaction*, and its interactions with $\mathcal{O}$ as the *right interactions*. The main hurdle in showing the indistinguishability of IND_0 and IND_1 is that the oracle $\mathcal{O}$ is not efficiently computable; if it were, indistinguishability would directly follow from the hiding property of the left interaction. The main idea of the security proof of [9] is then to implement the oracle $\mathcal{O}$ by extracting the committed values from the adversary, via "rewinding" the special-sound proofs in the right interactions. The following tow main technical challenges arise in simulating oracle $\mathcal{O}$.

First, once the simulation starts rewinding the right interactions, A might send new messages also in the left interaction. So, if done naively, this would rewind the left interaction, which could violate its hiding property. To solve this problem, the CLP protocol schedules messages in the special-sound proofs using a special message scheduling (according to the identity of the commitment), called the CLP scheduling, which is a variant of the message scheduling technique of [15,32]. The special message scheduling ensures that for every accepting right interaction with an identity that is different from the left interaction, there exists many points—called $\mathsf{safe\text{-}points}$—in the interaction, from which one can rewind the right interaction without requesting any *new* message in the left interaction.

Second, in the experiment IND_b, the adversary A expects to receive the committed value at the very moment it completes a commitment to its oracle. If the adversary "nests" its oracle calls, these rewindings become recursive and the running-time of the extraction quickly becomes exponential. To avoid the extraction time from exploding, the simulation strategy in CLP rewinds from $\mathsf{safe\text{-}points}$ using a concurrent extraction strategy that is similar to that used in the context of concurrent $\mathcal{ZK}$ by Richardson and Killian [44].

New Approach: To obtain a black-box construction, our main goal is to replace the $\mathcal{WISSP}$ proofs with an "equivalent" black-box component. The key property that the CLP proof relies on is that the protocol contains many *3-round* constructs satisfying that rewinding the last two messages reveals the committed value, but rewinding three messages reveals nothing. It seems that the 3-round commitment scheme $\mathsf{PExtCom}$ is a good replacement of $\mathcal{WISSP}$ proofs as one such 3-round construct: The special-soundness property of $\mathsf{PExtCom}$ ensures that rewinding the last two messages reveals the committed value, and the hiding property ensures that rewinding three messages reveals nothings. It is thus tempting to consider a commitment scheme in which the committer commits to value v using $\text{poly}(n)$ invocations of $\mathsf{PExtCom}$, arranged according to the CLP scheduling; the CLP extraction strategy guarantees that for every accepting right interaction, (the last two messages of) one $\mathsf{PExtCom}$ commitment is rewound and a committed value is extracted. Indeed, if a commitment of this scheme is valid, meaning that all the $\mathsf{PExtCom}$ commitments contained in it are valid commitments to the same value, the CLP extraction strategy returns the unique committed value. However, if the commitment is invalid, there arises the over-extraction problem: The CLP extraction strategy may extract a garbage value from an invalid $\mathsf{PExtCom}$ commitment or from a valid commitment that is inconsistent with the other commitments.

To solve the over-extraction problem, we use the cut-and-choose technique to enforce the committer to give valid and consistent $\mathsf{PExtCom}$ commitments. Instead of having the committer commit to v directly, let it commit to a $(n+1)$-*out-of-*$10n$ *Shamir's secret sharing* $s_1, \ldots, s_{10n}$ *of* v using many $\mathsf{PExtCom}$ invocations, still arranged according to the CLP scheduling; we refer to all the commitments to the j^{th} share s_j the j^{th} column. After all the $\mathsf{PExtCom}$ commitments, the receiver requests the committer to open all the commitments in n randomly chosen columns; the receiver accepts only if each column contains valid commitments to the same value. It follows from the cut-and-choose technique that except with negligible probability, at most n columns may contain invalid or inconsistent commitments. Therefore, when applying the CLP extraction strategy on a commitment of this scheme, it guarantees to extract out a secret-sharing that is .9-close to all the secret-sharing committed to in this commitment. Then by relying on the error-correcting property of the secret sharing,

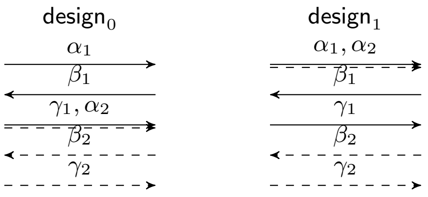

Fig. 1. Description of the schedules used in Stage 2 of the protocol. $(\alpha_1, \beta_1, \gamma_1)$ and $(\alpha_2, \beta_2, \gamma_2)$ are respectively the transcripts of a pair of rows in Stage 2 of $\langle C, R \rangle$.

The robust CCA-secure protocol $\langle C, R \rangle$

Let κ be an arbitrary polynomial, ℓ, η two polynomials such that $\ell(n) = n^{\nu}$ and $\eta(n) = n^{\varepsilon}$ for $\nu, \varepsilon > 0$, and L a polynomial such that $L(n) = \max(\kappa(n) + \eta(n), 4\ell(n)\eta(n))$. To commit to a value v, the committer C and the receiver R, on common input 1^n and the identity $\mathsf{id} \in \{0,1\}^{\ell(n)}$ of the committer C do:

Stage 1: The receiver sends the Stage 1 message of a commitment of PTrapCom. That is, a commitment of com to a randomly chosen string challenge $e = (e_1, \ldots, e_n)$.

Stage 2: The committer C prepares a $(n+1)$-out-of-$10n$ Shamir's secret sharing $s_1, \ldots, s_{10n}$ of the value v, and commits to these shares using Stage 2 of the protocol PTrapCom in parallel, *for $L(n)$ times*; we call the i^{th} parallel commitment the i^{th} *row*, and all the commitments to the i^{th} share s_i the i^{th} *column*.
Messages in the first $4\ell(n)\eta(n)$ rows are scheduled based on the identity id and relies on scheduling pairs of rows according to schedules design_0 and design_1 depicted in Figure 1. More precisely, Stage 2 consist of $\ell(n)$ phases. In phase i, the committer provides $\eta(n)$ sequential $\mathsf{design}_{\mathsf{id}_i}$ pairs of rows, followed by $\eta(n)$ sequential $\mathsf{design}_{1-\mathsf{id}_i}$ pairs of rows. Messages in the rest of the rows are simply arranged sequentially.

Stage 3: The receiver opens the Stage 1 commitment to the challenge e. The committer completes the $10nL(n)$ executions of PTrapCom w.r.t. challenge e in parallel.

Stage 4 (cut-and-choose): The receiver sends a randomly chosen subset $\Gamma \in [10n]$ of size n. For every $j \in \Gamma$, the committer opens all the commitments in the j^{th} column of Stage 3. The receiver checks that all the openings are valid, and reveal the same committed values s_j.

Decommitment Message: To decommit, the committer sends v, and opens all the commitments in the first row of Stage 2 to $s_1, \ldots, s_{10n}$. The receiver checks all the openings to $s_1, \ldots, s_{10n}$ are valid; furthermore, it checks that $s_1, \ldots, s_{10n}$ is 0.9-close to a valid codeword $w = (w_1, \cdots, w_{10n})$, and for every $j \in \Gamma$, w_j equals to the share s_j revealed in Stage 4.
In other words, a commitment of $\langle C, R \rangle$ is valid if and only if the first row in Stage 2 of the commitment contains valid commitments to shares $s_1, \ldots, s_{10n}$, such that, $s_1, \ldots, s_{10n}$ is 0.9 close to a valid codeword w, and w agrees with all the shares revealed in Stage 4 (i.e., for every $j \in \Gamma$, $w_j = s_j$).

Fig. 2. The formal description of the $\kappa(n)$-robust CCA-secure protocol $\langle C, R \rangle$

a valid committed value can be reconstructed. The formal analysis is actually more subtle; to avoid over-extraction, we employ the technique used in [12,13,46], which involves setting the validity condition of the commitment scheme carefully so that invalid commitment can be identified.

Unfortunately, our use of the cut-and-choose technique brings another problem: The above commitment scheme may not be hiding. This is because, in the last stage, the receiver may request the committer to open an adaptively chosen subset of commitments of PExtCom, and thus the remaining unopened commitments may not be hiding, unless PExtCom were secure against selective opening attack. To resolve this problem, we use the trapdoor commitment scheme PTrapCom to replace PExtCom. Since PTrapCom is trapdoor, it is secure against selective opening attack, and thus the hiding property holds. Furthermore, since Stage 2 of PTrapCom is simply a commitment of PExtCom, we can use Stage 2 of PTrapCom as an implementation of the 3-round construct needed for the CLP scheduling and extraction strategy. More precisely, the commitment scheme proceeds as follow: The committer commits to a $(n+1)$-out-of-$10n$ secret sharing of the value v using many invocations of PTrapCom, where all the invocations share the same Stage 1 message sent at the beginning, followed by all the 3-round Stage 2 executions arranged according to the CLP scheduling, and then all the Stage 3 executions performed in parallel; finally, the committer and the receiver conducts a cut-and-choose consistency check as described above. A formal description of our CCA secure protocol $\langle C, R\rangle$ in Figure 2.

It seems that the security proof of our CCA-secure commitment should follow from that of the non-black-box construction of [9]. Unfortunately, due to the fact that the "rewinding slots" of our protocol, that is the commitment of ExtCom, may have over-extraction, whereas the $\mathcal{WISSP}$ proofs in the CLP protocol never has this problem, the technical proof of [9] does not go through. In the full version, we rely on a different analysis to show the security of our protocol.

References

1. Barak, B., Canetti, R., Nielsen, J.B., Pass, R.: Universally composable protocols with relaxed set-up assumptions. In: FOCS, pp. 186–195 (2004)
2. Barak, B., Sahai, A.: How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In: FOCS, pp. 543–552 (2005)
3. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: STOC, pp. 1–10 (1988)
4. Canetti, R.: Security and composition of multiparty cryptographic protocols. Journal of Cryptology, 143–202 (2000)
5. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS, pp. 136–145 (2001)
6. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally Composable Security with Global Setup. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer, Heidelberg (2007)
7. Canetti, R., Fischlin, M.: Universally Composable Commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)

8. Canetti, R., Kushilevitz, E., Lindell, Y.: On the Limitations of Universally Composable Two-Party Computation Without Set-Up Assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 68–86. Springer, Heidelberg (2003)
9. Canetti, R., Lin, H., Pass, R.: Adaptive hardness and composable security in the plain model from standard assumptions. In: FOCS, pp. 541–550 (2010)
10. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC, pp. 494–503 (2002)
11. Canetti, R., Pass, R., Shelat, A.: Cryptography from sunspots: How to use an imperfect reference string. In: FOCS, pp. 249–259 (2007)
12. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Black-Box Construction of a Non-malleable Encryption Scheme from Any Semantically Secure One. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 427–444. Springer, Heidelberg (2008)
13. 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)
14. Damgård, I., 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)
15. Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. SIAM Journal on Computing 30(2), 391–437 (2000)
16. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: STOC, pp. 416–426 (1990)
17. Garg, S., Goyal, V., Jain, A., Sahai, A.: Concurrently Secure Computation in Constant Rounds. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 99–116. Springer, Heidelberg (2012)
18. Goldreich, O., Kahan, A.: How to construct constant-round zero-knowledge proof systems for NP. Journal of Cryptology 9(3), 167–190 (1996)
19. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: STOC, pp. 218–229 (1987)
20. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. J. ACM 38(3), 690–728 (1991)
21. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. 28(2), 270–299 (1984)
22. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on Computing 18(1), 186–208 (1989)
23. Goyal, V.: Constant round non-malleable protocols using one way functions. In: STOC, pp. 695–704 (2011)
24. Groth, J., Ostrovsky, R.: Cryptography in the Multi-string Model. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 323–341. Springer, Heidelberg (2007)
25. 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)
26. Ishai, Y., Kushilevitz, E., Lindell, Y., Petrank, E.: Black-box constructions for secure computation. In: STOC, pp. 99–108 (2006)
27. 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)
28. Kalai, Y.T., Lindell, Y., Prabhakaran, M.: Concurrent general composition of secure protocols in the timing model. In: STOC, pp. 644–653 (2005)

29. Katz, J.: Universally Composable Multi-party Computation Using Tamper-Proof Hardware. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 115–128. Springer, Heidelberg (2007)
30. Kilian, J.: Founding cryptography on oblivious transfer. In: STOC, pp. 20–31 (1988)
31. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: STOC, pp. 723–732 (1992)
32. Lin, H., Pass, R., Venkitasubramaniam, M.: Concurrent Non-malleable Commitments from Any One-Way Function. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 571–588. Springer, Heidelberg (2008)
33. Lin, H., Pass, R., Venkitasubramaniam, M.: A unified framework for concurrent security: universal composability from stand-alone non-malleability. In: STOC, pp. 179–188 (2009)
34. Lin, H., Pass, R., Venkitasubramaniam, M.: UC from semi-honest OT (2012) (manuscript)
35. Lindell, Y.: Lower Bounds for Concurrent Self Composition. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 203–222. Springer, Heidelberg (2004)
36. Lindell, Y., Pinkas, B.: An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg (2007)
37. Malkin, T., Moriarty, R., Yakovenko, N.: Generalized Environmental Security from Number Theoretic Assumptions. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 343–359. Springer, Heidelberg (2006)
38. Micali, S., Pass, R., Rosen, A.: Input-indistinguishable computation. In: FOCS, pp. 367–378 (2006)
39. Pass, R.: Simulation in Quasi-Polynomial Time, and its Application to Protocol Composition. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 160–176. Springer, Heidelberg (2003)
40. Pass, R., Rosen, A.: Concurrent non-malleable commitments. In: FOCS, pp. 563–572 (2005)
41. 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)
42. Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero knowledge with logarithmic round-complexity. In: FOCS, pp. 366–375 (2002)
43. Prabhakaran, M., Sahai, A.: New notions of security: achieving universal composability without trusted setup. In: STOC, pp. 242–251 (2004)
44. Richardson, R., Kilian, J.: On the Concurrent Composition of Zero-Knowledge Proofs. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 415–432. Springer, Heidelberg (1999)
45. Rosen, A.: A Note on Constant-Round Zero-Knowledge Proofs for NP. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 191–202. Springer, Heidelberg (2004)
46. Wee, H.: Black-box, round-efficient secure computation via non-malleability amplification. In: FOCS, pp. 531–540 (2010)
47. Yao, A.C.-C.: How to generate and exchange secrets (extended abstract). In: FOCS, pp. 162–167 (1986)