Concurrently Secure Computation in Constant Rounds

Sanjam Garg[1], Vipul Goyal[2], Abhishek Jain[1], and Amit Sahai[1]

[1] UCLA
[2] MSR India

Abstract. We study the problem of constructing concurrently secure computation protocols in the plain model, where no trust is required in any party or setup. While the well established UC framework for concurrent security is impossible to achieve in this setting, meaningful relaxed notions of concurrent security have been achieved.
The main contribution of our work is a new technique useful for designing protocols in the concurrent setting (in the plain model). The core of our technique is a new rewinding-based extraction procedure which only requires the protocol to have a constant number of rounds. We show two main applications of our technique.
We obtain the *first* concurrently secure computation protocol in the plain model with super-polynomial simulation (SPS) security that uses only a constant number of rounds and requires only standard assumptions. In contrast, the only previously known result (Canetti et al., FOCS'10) achieving SPS security based on standard assumptions requires polynomial number of rounds. Our second contribution is a new definition of input indistinguishable computation (IIC) and a constant round protocols satisfying that definition. Our definition of input indistinguishable computation is a simplification and strengthening of the definition of Micali et al. (FOCS'06) in various directions. Most notably, our definition provides meaningful security guarantees even for randomized functionalities.

1 Introduction

The notion of *secure computation* is central to cryptography. Introduced in the seminal works of [49, 19], secure multi-party computation allows a group of (mutually) distrustful parties $P_1, \ldots, P_n$, with private inputs $x_1, \ldots, x_n$, to jointly compute any functionality f in such a manner that the honest parties obtain correct outputs and no group of malicious parties learn anything beyond their inputs and prescribed outputs. The original definition of secure computation, although very useful and fundamental to cryptography, is only relevant to the *stand-alone* setting where security holds only if a single protocol session is executed in isolation. As it has become increasingly evident over the last two decades, stand-alone security does not suffice in real-world scenarios where several protocol sessions may be executed *concurrently* – a typical example being protocols executed over modern networked environments such as the Internet.

Concurrent Security. Towards that end, the last decade has seen a push towards obtaining protocols that have strong concurrent *composability* properties. For example, we could require concurrent self-composability: the protocol should remain secure even when there are multiple copies executing concurrently. The framework of *universal composability* (UC) was introduced by Canetti [10] to capture the more general security requirements when a protocol may be executed concurrently with not only several copies of itself but also with other protocols in an arbitrary manner.

Unfortunately, strong impossibility results have been shown ruling out the existence of secure protocols in the concurrent setting. UC secure protocols for most functionalities of interest have been ruled out in [11, 8]. These results were further generalized [35] to rule out the existence of protocols providing even concurrent self-composability. Protocols in even less demanding settings (where all honest party inputs are fixed in advance) were ruled out in [4]. All these impossibility results refer to the "plain model," where parties do not trust any external entity or setup. We stress that, in fact, some of these impossibility results provide an *explicit attack* in the concurrent setting using which the adversary may even fully recover the input of an honest party (see, e.g., the chosen protocol attack in [4]). Hence, designing secure protocols in the concurrent setting is a question of great theoretical as well as practical interest. Unfortunately, the only known positive results for concurrent composition in the plain model are for the zero-knowledge functionality [46, 30, 44].

To overcome these impossibility results, UC secure protocols were proposed based on various "trusted setup assumptions" such as a common random string that is published by a *trusted party* [11, 9, 1, 14, 28, 15]. Nevertheless, a driving goal in cryptographic research is to eliminate the need to trust other parties. In the context of UC secure protocols based on setup assumptions, while there has been some recent effort [26, 24, 18] towards reducing the extent of trust in any single party (or entity), obviously this approach cannot completely eliminate trust in other parties (since that is the very premise of a trusted setup assumption). Ideally, we would like to obtain concurrently-secure protocols in the *plain model* (which is the main focus of this paper).

Relaxing the Security Notion. To address the problem of concurrent security for secure computation in the plain model, a few candidate definitions have been proposed, including input-indistinguishable security [37] and super-polynomial simulation [40, 45, 5]. We discuss the each of these notions (and the state of the art) separately.

Super-polynomial Simulation. The notion of security with *super-polynomial simulators* (SPS) is one where the adversary in the ideal world is allowed to run in (fixed) super-polynomial time. Very informally, SPS security guarantees that any polynomial-time attack in the real execution can also be mounted in the ideal world execution, albeit in super-polynomial time. This is directly applicable and meaningful in settings where ideal world security is guaranteed statistically or information-theoretically (which would be the case in most "end-user" functionalities that have been considered, from privacy-preserving data mining to electronic voting). SPS security for concurrently composable zero knowledge proofs was first studied by [40], and SPS security for concurrently composable secure computation protocols was first studied by [45, 5]. The SPS definition guarantees security with respect to concurrent self-composition of the secure computation protocol being studied, and guarantees security with respect to general concurrent composition with arbitrary other protocols in the context of super-polynomial adversaries.

In recent years, the design of secure computation protocols in the plain model with SPS security has been the subject of several works [45, 5, 34, 13]. Very recently, Canetti, Lin, and Pass [13] obtained the first secure computation protocol that achieves SPS security based on *standard assumptions*[3].

Unfortunately, however, the improvement in terms of assumptions comes at the cost of the round complexity of the protocol. Specifically, the protocol of [13] incurs *polynomial-round complexity.* The latency of sending messages back and forth has been shown to often be the dominating factor in the running time of cryptographic protocols [36, 6]. Indeed, round complexity has been the subject of a great deal of research in cryptography. For example, in the context of concurrent zero knowledge (ZK) proofs, round complexity was improved in a sequence of works [46, 30, 44] from polynomial to slightly super-logarithmic (that nearly matches the lower bound w.r.t. black-box simulation [12]). The round complexity of non-malleable commitments in the stand-alone and concurrent settings has also been studied in several works [17, 2, 43, 42, 31, 48, 22, 32], improving the round complexity from logarithmic rounds to constant rounds under minimal assumptions. We observe that for the setting of concurrently secure computation protocols with SPS security, the situation is much worse since the only known protocol that achieves SPS security based on standard assumptions incurs polynomial-round complexity [13].

Input-Indistinguishable Computation. The notion of input indistinguishable computation [37] is a relaxation of the standard notion of secure computation akin to how witness indistinguishability is a relaxation of the notion of zero-knowledge. In input indistinguishable computation (IIC), very roughly, given the output vector (consisting of outputs in all concurrent sessions), consider any two honest party input vectors $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ "consistent" with the output vector. The security guarantee requires the adversary to have only a negligible advantage in distinguishing which of these is the actual input vector. While SPS security definition is based on the ideal/real world paradigm, the security definition of IIC is a game based one where various required properties (such as input independence) are formalized separately. In IIC, no guarantees are provided for any two input vectors which don't lead to the identical output (e.g., the functionality may be randomized; furthermore, the outputs may only be computationally indistinguishable as opposed to coming from identical or statistically close distributions).

[3] In fact, the work of [13], together with [45, 5], considers the stronger "angel-based security model" of [45]. In this work, we focus only on SPS security.

1.1 Our Contributions

The main contribution of our work is a new technique useful for designing protocols in the concurrent setting (in the plain model). The core of our technique is a new rewinding-based extraction procedure which only requires the protocol to have a constant number of rounds. Overall, our technique allows us to improve upon the previous works in terms of round complexity, the security notion being achieved as well the assumptions. We show two main applications of our technique in this work.

Super Polynomial Simulation. We construct the first *constant-round* concurrently composable secure computation protocol that achieves SPS security based on only *standard assumptions*. In addition, our construction only uses black-box simulation techniques.

In contrast to prior works where several powerful tools were employed to obtain positive results, e.g., CCA-secure commitments [13], our new proof technique allows us to only use relatively less powerful primitives, such as standard non-malleable commitments. Our positive result relies on the nearly minimal assumptions that constant-round (semi-honest) oblivious transfer (OT) exists and collision-resistant hash functions (CRHFs) exist.[4]

Input Indistinguishable Computation. We introduce a new definition of input indistinguishable computation and prove that, in fact, the same protocol (as for constant round super-polynomial simulation) satisfies this notion as well. Our definition of input indistinguishable computation is a simplification and strengthening of the definition in [37] in various directions. In particular, our definition provides meaningful security guarantees even for randomized functionalities. Furthermore, the security guarantees hold even when the output distributions resulting from the two honest party inputs (among which the adversary is trying to distinguish) are computationally indistinguishable (as opposed to coming from identical distributions) [5]. We follow the real/ideal world paradigm for formalizing the security guarantees which leads to an arguably simpler definition. Additionally, we show that our definition *implies* the definition of [37].

The essence of our new definition can be understood as follows. Consider a real world adversary. For any two input vectors $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$, we require the existence of a (PPT) ideal world simulator such that the output distribution in the ideal and the real world are indistinguishable. Hence, the only relaxation compared to the standard ideal/real world definition is now the ideal world simulator could be different for different pairs $(\boldsymbol{x_1}, \boldsymbol{x_2})$. The key intuition behind such a guarantee is that for any two honest party input vectors $(\boldsymbol{x_1}, \boldsymbol{x_2})$ leading to the same output vector (on the input vector chosen by the adversary), the simulator in the ideal world has no advantage in distinguishing which of the two

[4] We believe that our assumption of CRHFs can be removed by employing techniques from the recent work of [33], leaving only the minimal assumption that constant-round OT exists. We leave this for the full version of this paper.

[5] This is comparable to the relationship between witness indistinguishability and *strong* witness indistinguishability.

was used. This implies that even to the real world adversary should only have a negligible distinguishing advantage. We stress that in our definition, this holds even if the functionality is randomized and the outputs are computationally indistinguishable (as opposed to being identical). In addition, as opposed to [37], our ideal world simulator is required to extract the input being used by the adversary (in PPT) and send it to the trusted party. This provides a form of "input-awareness" guarantee.

While the above simple definition already provides meaningful security guarantees, the guarantees are unsatisfactory if there exists a "splitting input" which the ideal world simulator uses even when the real world adversary is such that it does not use a splitting input. A more detailed discussion of such issues can be found in [37]. Towards that end, we propose an extension of our definition and finally show that it implies the definition in [37]. To see an example of a functionality for which our definition provides meaningful security guarantees which neither the definition in [37] nor the SPS definition provide, please refer to the full version.

1.2 The Main Technique

A ubiquitous technique for simulation-based proofs in cryptography is that of *rewinding* the adversary. In the concurrent setting (which is the setting we consider in this paper), where an adversary can interleave messages from different protocols in any arbitrary manner, rewinding an adversary (to correctly simulate each session) is often problematic. The rewinding becomes recursive because of which the protocols typically requires a large number of rounds (in a single protocol). For example, in the context of concurrent zero knowledge, the best known result [44] requires super-logarithmic round complexity, which nearly matches the lower bound w.r.t. black-box simulation [12].

To deal with the problem of concurrent rewinding, we develop a novel proof technique using which we can limit the depth of such recursion to at most 2. Such a significant relaxation of the properties we need from our rewinding technique allows us to obtain our result. In the following discussion, we give a more detailed intuition behind our techniques, where we assume somewhat greater familiarity with recent work in this area. The discussion is primarily for obtaining constant round providing with SPS security although similar intuition applies for IIC as well.

We first note that all prior works on obtaining secure computation protocols with SPS security crucially use the super-polynomial time simulator to "break" some cryptographic scheme and extract some "secret information". Then, to avoid any complexity-leveraging type technique (which would lead to non-standard assumptions), and yet argue security, the technique used in [13] was to replace the super-polynomial time simulator with a polynomial-time rewinding "hybrid experiment" via a hybrid argument in the security proof. Indeed, this is why their protocol incurs large round complexity (so as to facilitate concurrent-rewinding). We also make use of rewinding, but crucially, in a weaker way. The main insights behind our rewinding technique are explained as follows:

– We first note that (like other works) we will restrict our usage of rewinding only to the creation of "look-ahead threads". Very roughly, this means that a rewinding simulator never changes its actions on the "main thread" of execution; and as such, the rewinding is employed only to extract some information from the adversary. Here, we again stress that our final simulator does not perform any rewinding, and that we only perform rewindings in hybrid experiments to bridge the gap between the real and ideal world executions.
– Now that we use rewindings only to extract some information from the adversary, and only in hybrid experiments, we make the critical observation that, in fact, we can make use of the secret inputs of the honest parties in the look-ahead threads. Indeed, in *all* our intermediate hybrid experiments, we perform rewindings to create look-ahead threads where we make "judicious" use of the honest party's inputs. In this manner, we eventually end up with a rewinding (hybrid) simulator that simulates the *main thread* without the honest party's inputs, but still uses them in the look-ahead threads (in a manner that guarantees extraction). This is our main conceptual deviation from prior work, where, to the best of our knowledge, honest party's inputs were only used in *some* intermediary hybrids, with the main goal being to eventually remove their usage even from the look-ahead threads. We show that this is in fact unnecessary, since our final simulator does not perform any rewindings, but instead runs in super-polynomial time to extract the same information that was being earlier extracted via rewinding in the hybrid experiments. We only need to argue that the main thread output by the rewinding (hybrid) experiment and the main thread output by the final simulator be indistinguishable. Indeed, we are able to argue that there is only a small statistical distance between our final simulator (that corresponds to the ideal execution) and the previous rewinding-based hybrid experiment. This statistical distance corresponds to the probability that the rewinding-based extraction is unsuccessful, since the SPS extraction is always successful.
– We further note that since we use the honest party's inputs in the look-ahead threads, we can bypass complex *recursive* rewinding schedules used in previous works and simply use "local rewindings" that only require constant rounds (in fact, only "one slot").
– Finally, we observe that since we perform rewindings only in hybrid experiments, we do not need the rewinding to succeed with probability negligibly close to 1, as is needed for concurrent ZK. Instead, we only require rewinding to succeed with probability $1-\epsilon$, where ϵ is related to the success probability of the distinguisher that is assumed to exist for the sake of contradiction. This observation, yet again, allows us to use a simpler rewinding strategy.
– Our overall proof strategy only makes use of relatively well understood primitives like standard non-malleable commitments. This is a departure from [13] which introduces a new primitive called CCA-secure commitment schemes.

At this point, an informed reader may question the feasibility of a "sound implementation" of the above approach. Indeed, a-priori it is not immediately clear whether it is even possible for the simulator to "cheat" on the main thread, yet behave honestly in look-ahead threads *at the same time*. In a bit more detail, recall that any given look-ahead thread shares a prefix with the main thread of execution. Now consider any session i on a look-ahead thread. Note that since some part of session i may already be executed on the *shared prefix*, it is not clear how the simulator can continue simulating session i on the look-ahead thread *without ever performing any recursive rewindings* if it was already cheating in session i on the shared prefix.

We address the above issues by a careful protocol design that guarantees that a rewinding simulator can always extract some "trapdoor" information *before* it "commits" to cheating in any session. As a result, during the simulation, whenever a look-ahead thread is forked at any point from the main thread, the simulator can either always continue cheating, or simply behave honestly (without any conflict with the main thread) in any session.

In our overall proof, SPS is used only at the very last step to stop the look-ahead threads (which required knowledge of honest party inputs to execute). A modification of this step is required to prove that the protocols satisfies our new notion of IIC as well. Instead of stopping the look-ahead threads (which used honest party inputs), we will now run "two-sets" of look-ahead threads one for each input vector given to the ideal world simulator. Since of these two is the real honest party input vector, at least one of the sets of look-ahead threads is guaranteed to be successful.

1.3 Other Related Work

Here we discuss some additional prior work related to the work in this paper. We note that while the focus of this work is on the notions of SPS security and IIC as means to obtain concurrently-secure protocols in the plain model, some recent works have investigated alternative security models for the same. Very recently, [25, 23] considered a model where the ideal world adversary is allowed to make additional queries (as compared to a single query, as per the standard definition) to the ideal functionality per session. While our protocol bears much similarity to the construction in [23], our rewinding technique (and the overall proof) is quite different.

Independent of our work, a constant round protocol providing SPS security was recently obtained by Lin, Pass and Venkitasubramaniam [41]. Their technique are quite different from ours and make use of a non-uniform argument. An advantage of our work over that of Lin et. al. is that we provide a *uniform* reduction to the underlying hardness assumptions. Hence, our construction guarantees security against uniform adversaries assuming that the underlying primitives are only secure against uniform adversaries. Lin et. al. crucially require the underlying primitives to be secure against non-uniform adversaries to provide any meaningful security guarantees.

We note that their techniques seem not to apply to get a construction satisfying our IIC security notion. Since the IIC simulator has to extract the adversarial inputs in PPT, a rewinding technique in the concurrent setting is crucially required.

2 Our Definitions

2.1 UC Security and SPS

In this section we briefly review UC security. For full details see [10]. Following [21, 20], a protocol is represented as an interactive Turing machine (ITM), which represents the program to be run within each participant.

Security of protocols. Protocols that securely carry out a given task (or, protocol problem) are defined in three steps, as follows. First, the process of executing a protocol in an adversarial environment is formalized. Next, an "ideal process" for carrying out the task at hand is formalized. In the ideal process the parties do not communicate with each other. Instead they have access to an "ideal functionality," which is essentially an incorruptible "trusted party" that is programmed to capture the desired functionality of the task at hand. A protocol is said to securely realize an ideal functionality if the process of running the protocol amounts to "emulating" the ideal process for that ideal functionality.

Securely realizing an ideal functionality. We say that a protocol Π *emulates* protocol ϕ if for any adversary $\mathcal{A}$ there exists an adversary $\mathcal{S}$ such that no environment $\mathcal{Z}$, on any input, can tell with non-negligible probability whether it is interacting with $\mathcal{A}$ and parties running Π, or it is interacting with $\mathcal{S}$ and parties running ϕ. This means that, from the point of view of the environment, running protocol Π is 'just as good' as interacting with ϕ. We say that Π *securely realizes* an ideal functionality $\mathcal{F}$ if it emulates the ideal protocol $\Pi(\mathcal{F})$. More precise definitions follow. A distribution ensemble is called *binary* if it consists of distributions over $\{0, 1\}$.

Definition 1 *Let Π and ϕ be protocols. We say that Π UC-emulates ϕ if for any adversary $\mathcal{A}$ there exists an adversary $\mathcal{S}$ such that for any environment $\mathcal{Z}$ that obeys the rules of interaction for UC security we have* $\mathrm{EXEC}_{\phi,\mathcal{S},\mathcal{Z}} \approx \mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$.

Definition 2 *Let $\mathcal{F}$ be an ideal functionality and let Π be a protocol. We say that Π UC-realizes $\mathcal{F}$ if Π UC-emulates the ideal process $\Pi(\mathcal{F})$.*

UC Security with Super-polynomial Simulation We next provide a relaxed notion of UC security by giving the simulator access to super-poly computational resources. The universal composition theorem generalizes naturally to the case of UC-SPS, the details of which we skip.

Definition 3 *Let Π and ϕ be protocols. We say that Π UC-SPS-emulates ϕ if for any adversary $\mathcal{A}$ there exists a* super-polynomial time *adversary $\mathcal{S}$ such that for any environment $\mathcal{Z}$ that obeys the rules of interaction for UC security we have* $\mathrm{EXEC}_{\phi,\mathcal{S},\mathcal{Z}} \approx \mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$.

Definition 4 *Let $\mathcal{F}$ be an ideal functionality and let Π be a protocol. We say that Π* UC-SPS-realizes *$\mathcal{F}$ if Π UC-SPS-emulates the ideal process $\Pi(\mathcal{F})$.*

For simplicity of exposition, in the rest of this paper we assume authenticated communication; that is, the adversary may deliver only messages that were actually sent. (This is however not essential as shown previously [3].)

2.2 Input Indistinguishable Computation

Under our notion, very roughly, an adversaries' goal is to guess the input, among two pre-specified inputs, used by the honest party. We say that a protocol is *input indistinguishable* if an adversary can not guess the honest parties input in the protocol execution any better than what it could have done in the ideal scenario. We formalize this by saying that the adversary learns nothing more than the two pre-specified inputs (which it already knows) and the output it learns in the ideal world. This naturally implies that if the adversary can not guess the honest parties input in the ideal scenario then it can not do so in the protocol execution as well.

CONCURRENT EXECUTION IN THE IDEAL MODEL. In the ideal model, there is a trusted party $\mathcal{F}$ that computes the functionality f (described above) based on the inputs handed to it by the two parties – P_1, P_2 which are involved in $m = m(n)$ sessions (polynomial in the security parameter, n). An execution in the ideal model with an adversary that controls P_1 or P_2 proceeds as follows:

Inputs: The honest party and adversary each obtain a vector of m inputs each of length n; denote this vector by $\boldsymbol{w}$ (i.e., $\boldsymbol{w} = \boldsymbol{x}$ or $\boldsymbol{w} = \boldsymbol{y}$).

Honest parties send inputs to trusted party: The honest party sends its entire input vector $\boldsymbol{w}$ to the trusted party $\mathcal{F}$.

Adversary interacts with trusted party: For every $i = 1, \ldots, m$, the adversary can send (i, w'_i) to the trusted party, for any $w'_i \in \{0,1\}^*$ of its choice. Upon sending this pair, it receives back its output based on w'_i and the input sent by the honest party. (That is, if P_1 is corrupted, then the adversary receives $f_1(w'_i, y_i)$ and if P_2 is corrupted then it receives $f_2(x_i, w'_i)$.) The adversary can send the (i, w'_i) pairs in any order it wishes and can also send them *adaptively* (i.e., choosing inputs based on previous outputs). The only limitation is that for any i, *at most one pair* indexed by i can be sent to the trusted party.

Adversary answers honest party: Having received all of its own outputs, the adversary specifies which outputs the honest party receives. That is, the adversary sends the trusted party a set $I \subseteq \{1, \ldots, m\}$. Then, the trusted party supplies the honest party with a vector $\boldsymbol{v}$ of length m such that for every $i \notin I$, $v_i = \perp$ and for every $i \in I$, v_i is the party's output from the i^{th} execution. (That is, if P_1 is honest, then for every $i \in I$, $v_i = f_1(x_i, w'_i)$ and if P_2 is honest, then $v_i = f_2(w'_i, y_i)$.)

Outputs: The honest party always outputs the vector $\boldsymbol{v}$ that it obtained from the trusted party. The adversary may output an arbitrary (probabilistic polynomial-time computable) function of its initial-input and the messages obtained from the trusted party.

Let $\mathcal{S}$ be a non-uniform probabilistic polynomial-time ideal-model machine (representing the ideal-model adversary). Then, the ideal execution of f (on input vectors $(\boldsymbol{x}, \boldsymbol{y})$ of length m and auxiliary input z to $\mathcal{S}$) denoted by $\text{IDEAL}_{\mathcal{F},\mathcal{S}}(\boldsymbol{x}, \boldsymbol{y}, z)$, is defined as the output pair of the honest party and $\mathcal{S}$ from the above ideal execution.

Execution in the Real model. We next consider the real model in which a real two-party protocol is executed (and there exists no trusted third party). Let $m = m(n)$ be a polynomial, let f be as above and let Π be a two-party protocol for computing f. Furthermore, let $\mathcal{A}$ be a non-uniform probabilistic polynomial-time machine that controls either P_1 or P_2. Then, the real concurrent execution of Π (on input vectors $(\boldsymbol{x}, \boldsymbol{y})$ of length $m(n)$ and auxiliary input z to $\mathcal{A}$), denoted $\text{REAL}_{\Pi,\mathcal{A}}(\boldsymbol{x}, \boldsymbol{y}, z)$, is defined as the output pair of the honest party and $\mathcal{A}$, resulting from $m(n)$ executions of the protocol interaction, where the honest party always inputs its i^{th} input into the i^{th} execution. The scheduling of all messages throughout the executions is controlled by the adversary. That is, the execution proceeds as follows. The adversary sends a message of the form (i, α) to the honest party. The honest party then adds α to the view of its i^{th} execution of Π and replies according to the instructions of Π and this view. The adversary continues by sending another message (j, β), and so on. Adversary can schedule these the messages in any way it likes. (Formally, view the schedule as the ordered series of messages of the form $(index, message)$ that are sent by the adversary.)

Definition 5 (Input Indistinguishable Computation (IIC).) *Let $\mathcal{F}$ and Π be the ideal trusted parted and the protocol realizing functionality f, as defined above. Protocol Π is said to* input indistinguishably compute *(or, IIC) f* for P_1 *under concurrent composition if for every polynomial $m = m(n)$, for every inputs $\boldsymbol{x}_0, \boldsymbol{x}_1 \in (\{0,1\}^n)^m$ of the honest party P_1, for every real-model non-uniform probabilistic polynomial-time adversary $\mathcal{A}$ controlling party P_2, there exists an ideal-model non-uniform probabilistic polynomial-time adversary $\mathcal{S}$ controlling P_2 such that $\forall \boldsymbol{x} \in \{\boldsymbol{x}_0, \boldsymbol{x}_1\}$,*

$$\{\text{IDEAL}_{\mathcal{F},\mathcal{S}}(\boldsymbol{x}, \boldsymbol{y}, z)\}_{n\in\mathbb{N};z\in\{0,1\}^*} \overset{c}{\equiv} \{\text{REAL}_{\Pi,\mathcal{A}}(\boldsymbol{x}, \boldsymbol{y}, z)\}_{n\in\mathbb{N};z\in\{0,1\}^*}$$

Protocol Π is said to input indistinguishably compute *(or, IIC) f if it input indistinguishably computes f both for P_1 and P_2.*

The above definition has various shortcomings and can be seen as only a stepping stone to our final definition (which implies the one in [37]). We refer the reader to the full version for our extended definition and for the relationship between various notions.

3 Building Blocks

We now discuss the main cryptographic primitives that we use in our construction.

Statistically Binding String Commitments. In our protocol, we will use a (2-round) statistically binding string commitment scheme, e.g., a parallel version of Naor's bit commitment scheme [38] based on one-way functions. For simplicity of exposition, in the presentation of our results in this manuscript, we will actually use a non-interactive perfectly binding string commitment.[6] Such a scheme can be easily constructed based on a 1-to-1 one way function. Let $\textsc{com}(\cdot)$ denote the commitment function of the string commitment scheme. For simplicity of exposition, in the sequel, we will assume that random coins are an implicit input to the commitment function.

Extractable Commitment Scheme. We will also use a simple challenge-response based extractable statistically-binding string commitment scheme $\langle C, R\rangle$ that has been used in several prior works, most notably [44, 47]. We note that in contrast to [44] where a multi-slot protocol was used, here (similar to [47]), we only need a one-slot protocol.

Protocol $\langle C, R\rangle$*.* Let $\textsc{com}(\cdot)$ denote the commitment function of a non-interactive perfectly binding string commitment scheme (as described in Section 3). Let n denote the security parameter. The commitment scheme $\langle C, R\rangle$ is described as follows.

Commit Phase:

1. To commit to a string $\mathtt{str}$, C chooses $k = \omega(\log(n))$ independent random pairs $\{\alpha_i^0, \alpha_i^1\}_{i=1}^k$ of strings such that $\forall i \in [k]$, $\alpha_i^0 \oplus \alpha_i^1 = \mathtt{str}$; and commits to all of them to R using $\textsc{com}$. Let $B \leftarrow \textsc{com}(\mathtt{str})$, and $A_i^0 \leftarrow \textsc{com}(\alpha_i^0)$, $A_i^1 \leftarrow \textsc{com}(\alpha_i^1)$ for every $i \in [k]$.
2. R sends k uniformly random bits $v_1, \ldots, v_n$.
3. For every $i \in [k]$, if $v_i = 0$, C opens A_i^0, otherwise it opens A_i^1 to R by sending the appropriate decommitment information.

Open Phase: C opens all the commitments by sending the decommitment information for each one of them.

This completes the description of $\langle C, R\rangle$.

Modified Commitment Scheme. Due to technical reasons, we will also use a minor variant, denoted $\langle C', R'\rangle$, of the above commitment scheme. Protocol $\langle C', R'\rangle$ is the same as $\langle C, R\rangle$, except that for a given receiver challenge string, the committer does not "open" the commitments, but instead simply reveals the appropriate

[6] It is easy to see that the construction given in Section 4 does not necessarily require the commitment scheme to be non-interactive, and that a standard 2-round scheme works as well. As noted above, we choose to work with non-interactive schemes only for simplicity of exposition.

committed values (without revealing the randomness used to create the corresponding commitments). More specifically, in protocol $\langle C', R' \rangle$, on receiving a challenge string $v_1, \ldots, v_n$ from the receiver, the committer uses the following strategy: for every $i \in [k]$, if $v_i = 0$, C' sends α_i^0, otherwise it sends α_i^1 to R'. Note that C' does not reveal the decommitment values associated with the revealed shares.

When we use $\langle C', R' \rangle$ in our main construction, we will require the committer C' to prove the "correctness" of the values (i.e., the secret shares) it reveals in the last step of the commitment protocol. In fact, due to technical reasons, we will also require the the committer to prove that the commitments that it sent in the first step are "well-formed". Below we formalize both these properties in the form of a *validity* condition for the commit phase.

Proving Validity of the Commit Phase. We say that commit phase between C' and R' is *valid* with respect to a value $\hat{\mathtt{str}}$ if there exist values $\{\hat{\alpha}_i^0, \hat{\alpha}_i^1\}_{i=1}^k$ such that:

1. For all $i \in [k]$, $\hat{\alpha}_i^0 \oplus \hat{\alpha}_i^1 = \hat{\mathtt{str}}$, and
2. Commitments B, $\{A_i^0, A_i^1\}_{i=1}^k$ can be decommitted to $\hat{\mathtt{str}}$, $\{\hat{\alpha}_i^0, \hat{\alpha}_i^1\}_{i=1}^k$ respectively.
3. Let $\bar{\alpha}_1^{v_1}, \ldots, \bar{\alpha}_k^{v_k}$ denote the secret shares revealed by C in the commit phase. Then, for all $i \in [k]$, $\bar{\alpha}_i^{v_i} = \hat{\alpha}_i^{v_i}$.

We can define *validity* condition for the commitment protocol $\langle C, R \rangle$ in a similar manner.

Constant-Round Non-Malleable Zero Knowledge Argument. In our main construction, we will use a constant-round non-malleable zero knowledge (NMZK) argument for every language in **NP** with perfect completeness and negligible soundness error. In particular, we will use a specific (stand-alone) NMZK protocol, denoted $\langle P, V \rangle$, based on the concurrent-NMZK protocol of Barak et al [4]. Specifically, we make the following two changes to Barak et al's protocol: (a) Instead of using an $\omega(\log n)$-round PRS preamble [44], we simply use the one-slot commitment scheme $\langle C, R \rangle$ (described above). (b) Further, we require that the non-malleable commitment scheme being used in the protocol be constant-round and public-coin w.r.t. receiver. We note that such commitment schemes are known due to Pass, Rosen [43]. Further, in full version, we show how to adapt the scheme of Goyal [22] to incorporate the public-coin property.[7] We now describe the protocol $\langle P, V \rangle$.

Protocol $\langle P, V \rangle$. Let P and V denote the prover and the verifier respectively. Let L be an NP language with a witness relation R. The common input to P and V is a statement $\pi \in L$. P additionally has a private input w (witness for π). Protocol $\langle P, V \rangle$ consists of two main phases: (a) the *preamble phase*, where the

[7] We note that while the commitment scheme of [43] admits a non black-box security proof, the security proof of Goyal's scheme is black-box. As such, the resultant NMZK protocol has a black-box security proof as well.

verifier commits to a random secret (say) σ via an execution of $\langle C, R \rangle$ with the prover, and (b) the *post-preamble phase*, where the prover proves an NP statement. In more detail, protocol $\langle P, V \rangle$ proceeds as follows.

PREAMBLE PHASE.

1. P and V engage in the execution of $\langle C, R \rangle$ where V commits to a random string σ.

POST-PREAMBLE PHASE.

2. P commits to 0 using a statistically-hiding commitment scheme. Let c be the commitment string. Additionally, P proves the knowledge of a valid decommitment to c using a statistical zero-knowledge argument of knowledge (SZKAOK).
3. V now reveals σ and sends the decommitment information relevant to $\langle C, R \rangle$ that was executed in step 1.
4. P commits to the witness w using a constant-round public-coin extractable non-malleable commitment scheme.
5. P now proves the following statement to V using SZKAOK:
   (a) *either* the value committed to in step 4 is a valid witness to π (i.e., $R(\pi, w) = 1$, where w is the committed value), *or*
   (b) the value committed to in step 2 is the trapdoor secret σ.
   P uses the witness corresponding to the first part of the statement.

Decoupling the Preamble Phase from the Protocol. Note that the preamble phase in $\langle P, V \rangle$ is independent of the proof statement and can therefore be executed by P and V *before* the proof statement is fixed. Indeed, this is the case when we use $\langle P, V \rangle$ in our main construction in Section 4. Specifically, in our main construction, the parties first engage in multiple executions of $\langle C, R \rangle$ at the beginning of the protocol. Later, when a party (say) P_i wishes to prove the validity of a statement π to (say) P_j, then P_i and P_j engage in an execution of the post-preamble phase of $\langle P, V \rangle$ for statement π. The protocol specification fixes a particular instance of $\langle C, R \rangle$ that was executed earlier as the preamble phase of this instance of $\langle P, V \rangle$. In the description of our main construction, we will abuse notation and sometimes refer to the post-preamble phase as $\langle P, V \rangle$.

Straight-line Simulation of $\langle P, V \rangle$. A nice property of protocol $\langle P, V \rangle$ is that it allows *straight-line* simulation of the prover if the trapdoor secret σ is available to the simulator S. (Note that S can rewind V during the execution of $\langle C, R \rangle$ in order to extract σ.) See the full version for a description of the simulation strategy.

Constant-Round Statistically Witness Indistinguishable Arguments. In our construction, we shall use a constant-round statistically witness indistinguishable (SWI) argument $\langle P_{\mathsf{swi}}, V_{\mathsf{swi}} \rangle$ for proving membership in any **NP** language with perfect completeness and negligible soundness error. Such a protocol can be constructed by using $\omega(\log n)$ copies of Blum's Hamiltonicity protocol [7] in parallel, with the modification that the prover's commitments in the Hamiltonicity protocol are made using a constant-round statistically hiding commitment scheme [39, 27, 16].

Semi-Honest Two Party Computation. We will also use a constant-round semi-honest two party computation protocol $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ for any functionality $\mathcal{F}$ in the stand-alone setting. The existence of such a protocol follows from the existence of constant-round semi-honest 1-out-of-2 oblivious transfer [49, 19, 29].

4 Our Construction

Let $\mathcal{F}$ be any well-formed functionality[8] that admits a constant round two-party computation protocol in the semi-honest setting. In particular, $\mathcal{F}$ can be a universal functionality. In this section we will give a protocol Π that UC-SPS-realizes $\mathcal{F}$. Note that in the UC framework any two parties (say P_i and P_j) might interact as per the protocol Π on initiation by the environment for some session corresponding to a SID *sid*. For simplicity of notation, we will describe the protocol in terms of two parties P_1 and P_2, where these roles could be taken by any two parties in the system. Further we will skip mentioning the SID to keep the protocol specification simple.

In order to describe our construction, we first recall the notation associated with the primitives that we use in our protocol. Let $\textsc{com}(\cdot)$ denote the commitment function of a non-interactive perfectly binding commitment scheme, and let $\langle C, R \rangle$ denote the one-slot extractable commitment scheme, and $\langle C', R' \rangle$ be its modified version (see Section 3). Further, we will use a constant-round NMZK protocol $\langle P, V \rangle$ (see Section 3), a constant-round SWI argument $\langle P_{\mathsf{swi}}, V_{\mathsf{swi}} \rangle$, and a constant-round *semi-honest* two party computation protocol $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ that securely computes $\mathcal{F}$ as per the standard simulation-based definition of secure computation.

Let P_1 and P_2 be two parties with inputs x_1 and x_2 provided to them by the environment $\mathcal{Z}$. Let n be the security parameter. Protocol $\Pi = \langle P_1, P_2 \rangle$ proceeds as follows.

I. Trapdoor Creation Phase.

1. $P_1 \Rightarrow P_2$: P_1 samples a random string σ_1 (of appropriate length; see below) and engages in an execution of $\langle C, R \rangle$ with P_2, where P_1 commits to σ_1. We will denote this commitment protocol by $\langle C, R \rangle_{1 \to 2}$.
2. $P_2 \Rightarrow P_1$: P_2 now acts symmetrically. That is, P_2 samples a random string σ_2 and commits it via an execution of $\langle C, R \rangle$ (denoted as $\langle C, R \rangle_{2 \to 1}$) with P_1.
3. $P_1 \Rightarrow P_2$: P_1 creates a commitment $com_1 = \textsc{com}(0)$ to bit 0 and sends com_1 to P_2. P_1 and P_2 now engage in an execution of (the post-preamble phase of) $\langle P, V \rangle$, where P_1 proves that com_1 is a commitment to bit 0. The commitment protocol $\langle C, R \rangle_{2 \to 1}$ (executed earlier in step 2) is fixed as the preamble phase for this instance of $\langle P, V \rangle$ (see Section 3).

[8] See [9] for a definition of well-formed functionalities.

4. $P_2 \Rightarrow P_1$: P_2 now acts symmetrically.

Informally speaking, the purpose of this phase is to aid the simulator in obtaining a "trapdoor" to be used during the simulation of the protocol. As discussed earlier in Section 1.2, in order to bypass the need of recursive rewindings (even though we consider concurrent security), we want to ensure that a "hybrid" simulator (that performs rewindings) can always extract a "trapdoor" *before* it begins cheating in any protocol session. Here, we achieve this effect by decoupling the preamble phase of $\langle P, V\rangle$ from the post-preamble phase (see Section 3) and executing the preamble phase at the very beginning of our protocol.

II. Input Commitment Phase. In this phase, the parties commit to their inputs and random coins (to be used in the next phase) via the commitment protocol $\langle C', R'\rangle$.

1. $P_1 \Rightarrow P_2$: P_1 first samples a random string r_1 (of appropriate length, to be used as P_1's randomness in the execution of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}}\rangle$ in phase III) and engages in an execution of $\langle C', R'\rangle$ (denoted as $\langle C', R'\rangle_{1\to 2}$) with P_2, where P_1 commits to $x_1 \| r_1$. Next, P_1 and P_2 engage in an execution of $\langle P_{\mathsf{swi}}, V_{\mathsf{swi}}\rangle$ where P_1 proves the following statement to P_2: (a) *either* there exist values $\hat{x}_1$, $\hat{r}_1$ such that the commitment protocol $\langle C', R'\rangle_{1\to 2}$ is *valid* with respect to the value $\hat{x}_1 \| \hat{r}_1$ (see Section 3), *or* (b) com_1 is a commitment to bit 1.
2. $P_2 \Rightarrow P_1$: P_2 now acts symmetrically. Let r_2 (analogous to r_1 chosen by P_1) be the random string chosen by P_2 (to be used in the next phase).

Informally speaking, the purpose of this phase is aid the simulator in extracting the adversary's input and randomness.

III. Secure Computation Phase. In this phase, P_1 and P_2 engage in an execution of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}}\rangle$ where P_1 plays the role of P_1^{sh}, while P_2 plays the role of P_2^{sh}. Since $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}}\rangle$ is secure only against semi-honest adversaries, we first enforce that the coins of each party are truly random, and then execute $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}}\rangle$, where with every protocol message, a party gives a proof using $\langle P_{\mathsf{swi}}, V_{\mathsf{swi}}\rangle$ of its honest behavior "so far" in the protocol. We now describe the steps in this phase.

1. $P_1 \leftrightarrow P_2$: P_1 samples a random string r_2' (of appropriate length) and sends it to P_2. Similarly, P_2 samples a random string r_1' and sends it to P_1. Let $r_1'' = r_1 \oplus r_1'$ and $r_2'' = r_2 \oplus r_2'$. Now, r_1'' and r_2'' are the random coins that P_1 and P_2 will use during the execution of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}}\rangle$.
2. Let t be the number of rounds in $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}}\rangle$, where one round consists of a message from P_1^{sh} followed by a reply from P_2^{sh}. Let transcript $T_{1,j}$ (resp., $T_{2,j}$) be defined to contain all the messages exchanged between P_1^{sh} and P_2^{sh} before the point P_1^{sh} (resp., P_2^{sh}) is supposed to send a message in round j. For $j = 1, \ldots, t$:
   (a) $P_1 \Rightarrow P_2$: Compute $\beta_{1,j} = P_1^{\mathsf{sh}}(T_{1,j}, x_1, r_1'')$ and send it to P_2. P_1 and P_2 now engage in an execution of $\langle P_{\mathsf{swi}}, V_{\mathsf{swi}}\rangle$, where P_1 proves the following statement:

   i. *either* there exist values $\hat{x}_1$, $\hat{r}_1$ such that (a) the commitment protocol $\langle C', R' \rangle_{1 \to 2}$ is *valid* with respect to the value $\hat{x}_1 \| \hat{r}_1$ (see Section 3), and (b) $\beta_{1,j} = P_1^{\mathsf{sh}}(T_{1,j}, \hat{x}_1, \hat{r}_1 \oplus r'_1)$
   ii. *or*, com_1 is a commitment to bit 1.

(b) $P_2 \Rightarrow P_1$: P_2 now acts symmetrically.

This completes the description of protocol Π. We now claim the following.

Theorem 1 *Assume the existence of constant round semi-honest OT and collision resistant hash functions.Then for every well-formed functionality $\mathcal{F}$, there exists a constant-round protocol that UC-SPS-realizes $\mathcal{F}$.*

We prove the above claim by arguing that the protocol $\Pi = \langle P_1, P_2 \rangle$ described earlier UC-SPS-realizes $\mathcal{F}$. Note that our simulator will run in sub-exponential time, where the desired parameters can be obtained by using a "scaled-down" security parameter of the commitment scheme COM. See the full version for the proof.

5 Acknowledgements

Research supported in part from a DARPA/ONR PROCEED award, NSF grants 1136174, 1118096, 1065276, 0916574 and 0830803, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0389. 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. Barak, B., Canetti, R., Nielsen, J., Pass, R.: Universally composable protocols with relaxed set-up assumptions. In: FOCS. pp. 186–195 (2004)
2. Barak, B.: Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In: FOCS. pp. 345–355 (2002)
3. Barak, B., Canetti, R., Lindell, Y., Pass, R., Rabin, T.: Secure computation without authentication. In: CRYPTO. pp. 361–377 (2005)
4. Barak, B., Prabhakaran, M., Sahai, A.: Concurrent non-malleable zero knowledge. In: FOCS. pp. 345–354 (2006)
5. 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. IEEE Computer Society (2005)
6. Ben-David, A., Nisan, N., Pinkas, B.: Fairplaymp: a system for secure multi-party computation. In: ACM Conference on Computer and Communications Security. pp. 257–266 (2008)
7. Blum, M.: How to prove a theorem so no one else can claim it. In: International Congress of Mathematicians. pp. 1444–1451 (1987)

8. Canetti, R., Kushilevitz, E., Lindell, Y.: On the limitations of universally composable two-party computation without set-up assumptions. J. Cryptology 19(2), 135–167 (2006)
9. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC. pp. 494–503 (2002)
10. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS. pp. 136–145 (2001)
11. Canetti, R., Fischlin, M.: Universally composable commitments. In: CRYPTO. pp. 19–40. Lecture Notes in Computer Science, Springer (2001)
12. Canetti, R., Kilian, J., Petrank, E., Rosen, A.: Black-box concurrent zero-knowledge requires $\widetilde{\Omega}(\log n)$ rounds. In: STOC. pp. 570–579 (2001)
13. Canetti, R., Lin, H., Pass, R.: Adaptive hardness and composable security in the plain model from standard assumptions. In: FOCS. pp. 541–550 (2010)
14. Canetti, R., Pass, R., Shelat, A.: Cryptography from sunspots: How to use an imperfect reference string. In: FOCS. pp. 249–259 (2007)
15. Chandran, N., Goyal, V., Sahai, A.: New constructions for uc secure computation using tamper-proof hardware. EUROCRYPT (2008)
16. Damgård, I., Pedersen, T.P., Pfitzmann, B.: On the existence of statistically hiding bit commitment schemes and fail-stop signatures. J. Cryptology 10(3), 163–194 (1997)
17. Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. SIAM J. Comput. 30(2), 391–437 (2000)
18. Garg, S., Goyal, V., Jain, A., Sahai, A.: Bringing people of different beliefs together to do uc. In: TCC. pp. 311–328 (2011)
19. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: STOC '87: Proceedings of the 19th annual ACM conference on Theory of computing. pp. 218–229. ACM Press, New York, NY, USA (1987)
20. Goldreich, O.: Foundation of Cryptography - Basic Tools. Cambridge University Press (2001)
21. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. 18(1), 186–208 (1989)
22. Goyal, V.: Constant round non-malleable protocols using one-way functions. In: STOC (2011)
23. Goyal, V., Jain, A., Ostrovsky, R.: Password-authenticated session-key generation on the internet in the plain model. In: CRYPTO. pp. 277–294 (2010)
24. Goyal, V., Katz, J.: Universally composable multi-party computation with an unreliable common reference string. In: TCC. pp. 142–154 (2008)
25. Goyal, V., Sahai, A.: Resettably secure computation. In: EUROCRYPT. pp. 54–71 (2009)
26. Groth, J., Ostrovsky, R.: Cryptography in the multi-string model. In: CRYPTO. pp. 323–341 (2007)
27. Halevi, S., Micali, S.: Practical and provably-secure commitment schemes from collision-free hashing. In: CRYPTO. pp. 201–215 (1996)
28. Katz, J.: Universally composable multi-party computation using tamper-proof hardware. In: Eurocrypt (2007)
29. Kilian, J.: Founding cryptography on oblivious transfer. In: STOC. pp. 20–31 (1988)
30. Kilian, J., Petrank, E.: Concurrent and resettable zero-knowledge in poly-loalgorithm rounds. In: STOC. pp. 560–569 (2001)
31. Lin, H., Pass, R.: Non-malleability amplification. In: STOC. pp. 189–198 (2009)

32. Lin, H., Pass, R.: Constant-round non-malleable commitments from any one-way function. In: STOC (2011)
33. Lin, H., Pass, R., Tseng, W.L.D., Venkitasubramaniam, M.: Concurrent non-malleable zero knowledge proofs. In: CRYPTO. pp. 429–446 (2010)
34. 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)
35. Lindell, Y.: Lower bounds for concurrent self composition. In: Naor, M. (ed.) TCC. Lecture Notes in Computer Science, vol. 2951, pp. 203–222. Springer (2004)
36. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay - secure two-party computation system. In: USENIX Security Symposium. pp. 287–302 (2004)
37. Micali, S., Pass, R., Rosen, A.: Input-indistinguishable computation. In: FOCS. pp. 367–378. IEEE Computer Society (2006)
38. Naor, M.: Bit commitment using pseudorandomness. J. Cryptology 4(2), 151–158 (1991)
39. Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: STOC. pp. 33–43 (1989)
40. Pass, R.: Simulation in quasi-polynomial time, and its application to protocol composition. In: EUROCRYPT. pp. 160–176 (2003)
41. Pass, R.: Personal Communication (2011)
42. Pass, R., Rosen, A.: Concurrent non-malleable commitments. In: FOCS. pp. 563–572 (2005)
43. Pass, R., Rosen, A.: New and improved constructions of non-malleable cryptographic protocols. In: STOC. pp. 533–542 (2005)
44. Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero knowledge with logarithmic round-complexity. In: FOCS. pp. 366–375 (2002)
45. Prabhakaran, M., Sahai, A.: New notions of security: achieving universal composability without trusted setup. In: STOC. pp. 242–251 (2004)
46. Richardson, R., Kilian, J.: On the concurrent composition of zero-knowledge proofs. In: EUROCRYPT. pp. 415–431 (1999)
47. Rosen, A.: A note on constant-round zero-knowledge proofs for np. In: TCC. pp. 191–202 (2004)
48. Wee, H.: Black-box, round-efficient secure computation via non-malleability amplification. In: FOCS. pp. 531–540 (2010)
49. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: FOCS. pp. 162–167. IEEE (1986)