2012 IEEE 53rd Annual Symposium on Foundations of Computer Science

Positive Results for Concurrently Secure Computation in the Plain Model

Vipul Goyal
Microsoft Research, India
Email: vipul@microsoft.com

***Abstract*—We consider the question of designing concurrently self-composable protocols in the plain model. We first focus on the minimal setting where there is a party P_1 which might interact with several other parties in any unbounded (polynomial) number of concurrent sessions. P_1 holds a single input x which it uses in all the concurrent sessions. An analogy is a server interacting with various clients at the same time. In this "single input" setting, we show that many (or even most) functionalities can be securely realized in the plain model. More precisely, we are able to realize all ideal functionalities except ones which are a (weak form of) cryptographic pseudorandom functions. We complement our positive result by showing an impossibility result in this setting for a functionality which evaluates a pseudorandom function.**

Our security definition follows the standard ideal/real world simulation paradigm (with no super polynomial simulation etc). There is no apriori bound on the number of concurrent executions.

We also show interesting extensions of our positive results to the more general setting where the honest parties may choose different inputs in different session (even adaptively), the roles that the parties assume in the protocol may be interchangeable, etc.

Prior to our work, the only positive results known in the plain model in the fully concurrent setting were for zero-knowledge.

I. Introduction

General positive results for secure computation were obtained more than two decades ago [Yao86], [GMW87]. These results were for the setting where each protocol execution is done in isolation. With the proliferation of the network setting (and especially internet), an ambitious effort to generalize these results was started. The study of concurrent zero-knowledge (ZK) was initiated by Dwork, Naor and Sahai [DNS98] with a protocol soon proposed in the plain model by Richardson and Kilian [RK99]. A sequence of works studied the round complexity of concurrent ZK [CKPR01], [KP01], [PRS02] (see also [Bar01]). In addition, a protocol for the "interchangeable role" setting (where the same party might play prover in one session and verifier in another) was proposed by Barak, Prabhakaran and Sahai [BPS06] (see also [LPTV10]).

However other than the above encouraging results for the zero-knowledge functionality, there are no known positive results in the setting where there could be any unbounded polynomial number of concurrent sessions (referred to as the fully concurrent setting). In fact, far reaching impossibility results were shown in a series of works [CKL06], [Lin03b], [Lin08], [BPS06]. These results refer to the "plain model" where the participating parties are not required to trust any external entity, they have no prior communication among themselves, etc.

To circumvent these results and obtain protocols secure in the setting of concurrent executions, one line of work has studied various "setup assumptions" where, for example, a trusted party publishes a uniformly chosen string or the participating parties may exchange physical tamper proof hardware tokens etc (see for example [CLOS02], [BCNP04], [Kat]). Another interesting line of works has studied weaker security definitions while still remaining in the plain model [Pas03], [PS04], [BS05], [MPR06], [CLP10], [GGJS12], [GM00].

In this paper, we focus on obtaining standard security guarantees in the plain model. Relevant to our paper is the line of works on obtaining concurrent self-composition in the so called bounded concurrent setting [Lin03a], [PR03], [Pas04]. In this setting, there is an apriori fixed bound on the total number of concurrent sessions in the system (and the protocol in turn might be dependent on this bound). This state of affairs raises the following natural question: *"Can we obtain interesting positive results for functionalities other than zero-knowledge in the fully concurrent setting?"*

Our Results: We first discuss our results for what we call the "single input setting" and then discuss a generalization.

Concurrently Secure Computation with a Single Input. We consider the setting where there is a single party P_1 which might interact with several other parties in any unbounded (polynomial) number of concurrent sessions. The party P_1 holds a single input x which it uses in all the concurrent sessions (if honest). However if P_1 is dishonest, there are no restrictions on how it behaves (except that it has to be PPT). An example is a server (e.g., holding a password file) interacting with several clients concurrently (to authenticate them). We refer to this as the "single input" setting (a more precise definition is given in section II-B).

In this setting, we show that many (or even most) functionalities can be securely realized in the plain model. More precisely, we are able to realize all functionalities except which are what we call a worst-case hard pseudoentropy function (WC-PEF). Very roughly, a WC-PEF is capable of generating an output with pseudoentropy much larger than

0272-5428/12 $26.00 © 2012 IEEE
DOI 10.1109/FOCS.2012.13

allowed information theoretically on at least some inputs (see section II-A for a more formal definition).[1]

We complement this positive result by showing an (unconditional) impossibility result in our setting for a functionality which evaluates a pseudorandom function on a committed key. In more detail, let COM be a non-interactive statistically binding commitment scheme and f be a (keyed) pseudorandom function. Our functionality is parameterized by a string σ. It takes as input k, r from P_1 and x from P_2. It first checks if $\sigma = COM(k, r)$, if not, it outputs $\perp$ to P_2. Else, it outputs $f(k, x)$ to P_2. In fact, it suffices to use a notion of worst case hard pseudorandom function (thus matching our positive results more closely). The impossibility holds both w.r.t. black-box as well as non-black-box simulation and represents the first negative result for an arguably natural functionality in the setting of fixed inputs. The only previous negative result known [BPS06] was for a rather contrived functionality (which allowed two modes; one for execution of zero-knowledge and another for oblivious transfer).

To prove our main negative result, the key technical tool is a new garbled circuit construction where the garbled circuit is executed by the receiver by a single k-out-of-2k OT (as opposed to k execution of 1-out-of-2 OT). We believe our construction is of independent interest since, to our knowledge, all previous constructions of protocols based on garbled circuit involved parties executing k (1-out-of-2) OTs. To be more precise, we actually provide a construction of *one time programs* [GKR08] based on a *single k-time-memory hardware token* (as opposed to requiring several one-time-memory hardware tokens per program). Using a single k-time-memory hardware token may be more compact and efficient than using k separate one-time hardware tokens.

`Generalization of our Positive Result.` We show that the positive results discussed above can be significantly generalized. Our construction only requires that the ideal world satisfy what we call the *key technical property*. The rough intuition behind the key technical property is as follows. In the ideal world execution, we require the existence of a *predictor* which, given information about (adversary's) input/output tuples for sufficiently many (ideal world) sessions, starts to be able to "predict" the output of the ideal functionality in some sessions with a noticeable probability (the exact definition is slightly technical and is discussed in Section II-C).[2] The positive results for the single input setting were then obtained by simply showing that this ideal world condition is satisfied for all functionalities except for those that behave as a WC-PEF.

However, the key technical property (KTP) is quite general and is naturally satisfied in many other settings. We consider the very general setting where the the honest parties may have different (possibly *adaptively chosen*) inputs in different sessions, there may be multiple parties in a protocol session with all of them getting different outputs, the adversary may choose to corrupt parties with different roles in different sessions (i.e., the interchangeable role setting [Lin08]), etc. We prove that the ideal world would still satisfy KTP as long the size of the total "state" of the honest parties in the ideal world is bounded and the ideal world is "hardness-free". In more detail, first we require the existence of an apriori bounded length string S which describes the state of all the honest parties at the beginning of the ideal world execution. This condition is naturally satisfied when, e.g., the total number of honest parties (with each party participating in any unbounded polynomial number of sessions) is apriori bounded. Secondly, a hardness-free ideal world requires that the code "consisting" of the ideal world functionality and the (ideal world) honest parties does not behave as a WC-PEF (see the full version [Goy11b] for more formal details). Thus, this gives us a *general positive result* for all bounded-size hardness-free ideal worlds. We conjecture that, in fact, a more general and cleaner statement is true (see the *bounded pseudoentropy conjecture* in the full version).

Interpretations and Applications of Our Positive Results: A simple example of a setting where KTP is satisfied is the well studied setting of concurrent self-composition in the bounded concurrent setting [Lin03a], [PR03], [Pas04]. In fact our techniques only make use of black-box simulation while all previously reported protocols in the bounded concurrent setting use non-black-box simulation techniques introduced by [Bar01]. On the flip side we note that the protocols in [PR03], [Pas04] are constant round while ours might require a large polynomial number of rounds. Another well studied setting where KTP is satisfied is that of concurrent zero-knowledge.

We believe the conceptual insight in this paper improves our basic understanding of when concurrently secure computation is possible. Our results not only subsume the known positive results on fully concurrent zero-knowledge and bounded concurrent secure computation, but rather *present a unified explanation* of why it might be possible to obtain such results. Prior to our work, the intuition behind why these two tasks are possible might have looked very different.

The above is, of course, in addition to our main contribution which is obtaining a host of new positive results in the fully concurrent setting. To start with, we remark that our results imply the first construction of a concurrent password based key exchange (PAKE) protocol in the plain model with standard ideal/real world security guarantees.

[1] Jumping ahead, the reason why we refer to worst case hard primitives (rather than average case) stems from the fact that the standard definition of secure computation requires the ideal world simulator to work for *every* honest party input (as opposed to just a random one).

[2] Note that being able to predict the output of a function is very different from requiring that the function is learnable. A simple example is a point function for which it is easy to predict the output on almost every input. Also, note that the existence of such a predictor does not mean that the output has to come from a polynomial-size domain; see section II-C.

The only previous construction of (fully) concurrent PAKE in the plain model was given recently by Goyal, Jain and Ostrovsky [GJO10]. However the construction in [GJO10] was according to the original definition of Goldreich and Lindell [GL01] which is a **weaker** definition (in comparison to the standard ideal/real world definition). Prior to the work of [GJO10], obtaining a fully concurrently secure protocol in the plain model according to *any* reasonable definition was an open problem (despite a number of works studying PAKE in the concurrent setting, c.f., [KOY01], [CHK$^+$05], [BCL$^+$05], [GL03]). We note that the setting in ours as well as in [GJO10] is that of a single fixed input.

We also get positive results for a number of other functionalities studied previously. One example is that of *private database search* where a party holds a database and another party wants to search and get the matching entries without revealing its search criteria (such as a keyword). Problems such as private information retrieval [CGKS95], [KO97], pattern matching [HL08b], oblivious document and database search [HL08a], etc are special instances of the general problem of private database search. Other examples of well studied problems are secure set intersection [FNP04], private matching [FNP04], securely computing the k-th ranked element [AMP04], etc. For all of these functionalities, we get concurrently secure protocols in the single input setting. We refer the reader to section II-C for an understanding of why these functionalities might satisfy the KTP.

In general, in the (large) body of published literature studying specific functionalities of interest, we found that almost all of them indeed have hardness-free ideal worlds (i.e., in the ideal world, the trusted party is not required to perform any cryptographic operations, etc). Some functionalities are naturally seen as an interaction between a client and a server where only the server accepts concurrent sessions. For such functionalities, the single input setting is already very realistic. For example, we get positive results for concurrent private database search where there is a server holding the database interacting with multiple clients each of which is holding a search criteria (in particular, this also implies a similar positive result for concurrent private information retrieval, concurrent pattern matching, etc).

Some functionalities however are more symmetric (such as secure set intersection). Hence, there is motivation to also study the setting where there may be multiple honest parties holding different inputs and accepting concurrent sessions. Towards that end, we remark that a positive result may be obtained even in this setting if the bounded-size ideal world requirement is satisfied (it would be, e.g., if the total number of honest parties and the size of their initial states is bounded).

Overview of our Construction: A well established approach to constructing secure computation protocols in the standalone setting is to use the GMW compiler: take a semi-honest secure computation protocol and "compile" it with zero-knowledge arguments. The natural start point of our construction is to follow the same principles in the concurrent setting: somehow compile a semi-honest secure computation protocol with a concurrent zero-knowledge protocol (for security in more demanding settings, compilation with concurrent non-malleable zero-knowledge [BPS06] may be required). Does such an approach (or minor variants) already give us protocols secure according to the standard ideal/real world definition in the plain model?

The fundamental problem with this approach is the following. We note that the known concurrent zero-knowledge simulators (in the fully concurrent setting) work by rewinding the adversarial parties. In the concurrent setting, the adversary is allowed to control the scheduling of the messages of different sessions. Then the following scenario might occur:

- Between two messages of a session s_1, there might exist another entire session s_2
- When the simulator rewinds the session s_1, it may rewind past the beginning of session s_2
- Hence throughout the simulation, the session s_2 may be executed multiple times from the beginning
- Every time the session s_2 is executed, the adversary may choose a different input (e.g., the adversary may choose his input in session s_2 based on the entire transcript of interaction so far).
- In such a case, the simulator would have to query the ideal functionality for session s_2 more than once. However note that for every session, simulator gets to query the ideal functionality only once!

Indeed, some such problem is rather inherent as indicated by various impossibility results [Lin08], [BPS06]. This is where the fact that our ideal world satisfies the key technical property comes in. Very roughly, KTP requires the existence of a *predictor* which is successful in predicting the output of the ideal functionality with a noticeable probability (under certain conditions). The basic idea is as follows. The rewinding strategy of the simulator would lead to a "main thread" and several "look ahead threads" (following the terminology of [PRS02]). Whenever the simulator needs to make a call to the ideal functionality in a look-ahead thread, it uses the predictor instead. This would help the simulator achieve the goal of querying the ideal functionality only once per session.[3]

Note that the output of the predictor is not guaranteed to be correct in all cases. Furthermore, the adversary might have complete auxiliary information about the input of the honest parties (and hence may distinguish the incorrect output from correct ones). In such a scenario, adversary might change its behavior in the look-ahead thread (or might

[3]Such an approach of giving a "made-up possibly incorrect answer" in look-ahead threads has also proven to be important in constructing non-malleable commitments [Goy11a], [GLOV12].

simply abort). Hence, several look-ahead threads might now fail. In general, the property of indistinguishability between the main thread and the look-ahead threads (which all previous rewinding strategies rely on) does not hold any more. To solve this problem, we rely on and analyze a specific rewinding strategy by Deng, Goyal and Sahai [DGS09]. We choose the number of rewinding opportunities based on the key parameters of the predictor (guaranteed by KTP).

Our initial protocol is based on compilation with concurrent zero-knowledge. However it only satisfies a weaker notion of security which provides concurrent security for first party while only guaranteeing security for the other party in the standalone setting. Our final protocol is based on compilation with concurrent non-malleable zero-knowledge [BPS06]. There are several problems that arise with such a compilation. First, the security of the BPS construction is analyzed only for the setting where all the statements being proven by honest parties are fixed in advance. Secondly, the extractor of BPS-CNMZK is unsuitable for extracting inputs of the adversary since it works after the entire execution is complete on a *session-by-session* basis. Fortunately, these challenges were tackled in a recent work on password based key exchange by Goyal, Jain and Ostrovsky [GJO10]. Goyal et. al. presented an approach which can be viewed as a technique to correctly compile a semi-honest secure protocol with BPS-CNMZK. In our final protocol, we borrow a significant part of the construction presented in [GJO10].

Open Problem: The number of rounds in our protocol depend on the parameters of the predictor associated with the functionality and may be quite large (although still polynomial). We leave it as an open problem to construct more round efficient protocols. Known lower bounds on the round complexity of protocols proven secure using black-box simulation [Lin08] imply that to get round efficient protocol (e.g., to get a protocol with round complexity dependent only on the security parameter), advancements in our understanding of non-black-box simulation techniques [Bar01] will be required. In particular, it seems that we need a construction for zero-knowledge with a non-rewinding simulator in the fully concurrent setting. *However obtaining such a construction is currently an important open problem.*

Organization of the Paper: What follows is a warmup construction which is meant to highlight the new ideas behind our positive result. This construction satisfies a weaker notion of security where the first party is guaranteed concurrent security (while interacting with multiple adversarial parties) while the other one is guaranteed security only in the standalone setting. This construction illustrates the main ideas in our work and is kept at an informal level. Our final construction is an extension of the warmup construction using known techniques [GJO10], [BPS06]. A complete description of the final construction along with a full self-contained proof can be found in the full version of this paper [Goy11b]. The details of our negative result can also be found in the full version.

II. Model and Definitions

A. Preliminaries

In this section, we first define a very minimal cryptographic primitive called worst-case hard one-way functions (WC-OWF). The existence of WC-OWF is implied by the assumption **NP** $\not\subseteq$ **BPP**.

Definition 1 (Worst-Case Hard One Way Functions): A function $f : \{0,1\}^n \rightarrow \{0,1\}^*$ is a worst-case hard one-way function if it is polynomial time computable but for all uniform probabilistic polynomial time algorithms A, there exists $x \in \{0,1\}^n$ such that $\Pr[f(A(f(x))) = f(x)] = \text{negl}(n)$.

We also define worst case hard pseudoentropy functions (WC-PEF) which imply the existence of WC-OWF. Very roughly, WC-PEF are functions which are capable of generating an output with "much higher pseudo-entropy" than the size of the input.

Definition 2 (Worst-Case Hard Pseudoentropy Function): A function $g : \{0,1\}^n \times \{0,1\}^* \rightarrow \{0,1\}^*$ is a worst-case hard pseudoentropy function if for all uniform probabilistic polynomial time algorithms A, there exists x, t and at least $100nt$ inputs $r_1, \ldots, r_{100nt}$ such that for all $i \in [100nt]$,

$$Pr\big[A(r_1, \ldots, r_i, g(x, r_1), \ldots, g(x, r_{i-1})) \neq g(x, r_i)\big] \geq \tfrac{1}{t}$$

That is, there exists at least $100nt$ input strings for which the probability of failing to predict the correct output is at least $\frac{1}{t}$ given all previous outputs in the input set (at least for one secret key x). Thus, observe that the expected number of incorrectly predicted output bits is at least $100n$, while, the entropy of the input is at most n. Then, it can be shown using standard techniques that WC-PEGs imply the existence of WC-OWFs. The basic idea would be to use the output strings seen so far to "sample" the secret key x and use that to predict the next output [IR89], [BGP00]. Similarly, one can also define a worst-case hard pseudorandom function (WC-PRF) where we require the output to be indistinguishable from random as opposed to just "slightly unpredictable". Please see the full version for details.

B. Concurrently Secure Computation with a Single Input

We now discuss the model for concurrently secure computation a single input. For simplicity, we keep the model to be a minimal (yet natural and fundamental) model of concurrently secure computation and discuss generalizations later on. For lack of space, what follows is a sketch and the details are given in the full version (building upon the model in [Lin08]). In the ideal world, there are two parties P_1 and P_2. In addition, we have a deterministic and non-reactive trusted functionality $\mathcal{F} : \{0,1\}^r \times \{0,1\}^s \rightarrow \perp \times \{0,1\}^*$ (i.e., only the second party gets an output). There could be any unbounded (polynomial) number sessions between P_1 and P_2. The party P_1 holds a fixed input $x_1 \in \{0,1\}^r$ and (if honest) sends it to $\mathcal{F}$ in every session. Party P_2 may

choose an input x_2 adaptively depending upon its view in the ideal world so far. The adversary may (statically) corrupt either P_1 or P_2. If a party is corrupted, it is free to behave in any way it wants (in particular, by starting any number of sessions and *adaptively* choosing an input in each session depending upon the entire view of the adversary so far). Each session in the ideal world is assigned a session index in the order of execution (i.e., session ℓ is completed before session $\ell + 1$ completes).

We note that our model and results can be significantly generalized to consider different (adaptively chosen) inputs in different sessions, interchangeable roles, multi-party protocols with multiple parties getting output etc. These generalizations are considered in the full version.

Relaxed Security Notion: Our warm up construction (based on compilation with concurrent zero-knowledge) only satisfies a weaker security notion where P_1 is guaranteed concurrent security while security for the party P_2 is guaranteed only in the standalone setting. We very informally outline the security notion in the following. The ideal and the real world remain as discussed above. However we place conditions on when we require the existence of an ideal world simulator. We wish to provide concurrent security for the party P_1. In particular, consider the case when P_1 is honest and the party P_2 is corrupt (and interacts with P_1 in several concurrent session). We require the existence of a simulator in the ideal world such that the ideal and the real world distribution are indistinguishable. Furthermore, consider the case where there is only a single session between P_1 and P_2 such that P_2 is honest but P_1 is corrupted. We again require the existence of a simulator in the ideal world such that the ideal and the real world distribution are indistinguishable. In all other cases (in particular when P_1 is corrupt and interacts with P_2 concurrently in several sessions), we do not require any guarantees.

C. Key Technical Property

In this section, we will define a property of the ideal world experiment (as defined in section II-B) called the key technical property (KTP). We first introduce some notation. Recall that the adversary may corrupt any subset of parties (or any one party in the two-party setting). Let M denote the subset of corrupted parties. Let there be k sessions. The input of the adversarial party in session ℓ is denoted by $I[\ell]$ while the corresponding output is denoted by $O[\ell]$. We first describe a warmup version of the key technical property (which is easier to understand) and then generalize it later.

Definition 3 (Warmup Key Technical Property): The warmup key technical property of an ideal world experiment requires the existence of a PPT predictor $\mathcal{P}$ satisfying the following conditions. For all sufficiently large n, there exists a bound D such that for all adversaries and honest party inputs,

$$\left|\left\{j : \mathcal{P}\big(\{I[\ell]\}_{\ell \le j}, \{O[\ell]\}_{\ell < j}\big) \neq O[j]\right\}\right| < D$$

In other words, the number of adversary inputs for which the predictor fails to predict the correct output (given all previous inputs/outputs) is at most $D - 1$. And hence given D adversary inputs, the predictor is guaranteed to succeed for at least one.

We now give some examples to illustrate this property. These examples provide important intuition about why the key technical property is so general.

Password based key exchange: Consider the following password checking functionality. Say there is a single honest party which might interact with several adversarial parties in an unbounded (polynomial) number of concurrent sessions. The honest party holds a password p and the ideal functionality is such that if both parties input the same password, it outputs 1 to the adversarial party (and $\perp$ otherwise). We note that in this case, the ideal world satisfies the KTP in fact for even $D = 2$. The predictor works as follows. It needs to guess the output in a session given the (actual) inputs/outputs in all previous sessions.

- If the output in a previous session is 1, then by looking at the *input of the adversary* in that session, the predictor can find out the actual password p. This is because the input of the adversary in that session must be p itself. Hence it uses that to correctly compute the output in the next session.
- However, if the output in all previous sessions is $\perp$, the predictor sets the output in the next session to be $\perp$ as well. It is easy to see that the predictor makes an error in at most a single session. (This is because as soon as it makes an error in a session, it can learns the correct password by looking at the input of the adversary in that session as explained in the previous bullet point).

A variant of this functionality is the password based key exchange functionality where, if the passwords match, the ideal functionality outputs a secret key to both parties (and $\perp$ otherwise). This functionality similarly satisfies the generalized KTP (see full version) for $D = 2$. Similar argument can be shown to hold for more general access control functionalities. For example, say that the first party has a list of k valid passwords (or say a list of k valid user-ID and password tuples). If the password supplied by the second party is in the list, both the parties get a shared secret key and $\perp$ otherwise. In this case, setting $D = k + 1$ would allow the construction of a predictor using similar ideas.

Private database search: In this scenario, the first (honest) party holds a database consisting of k entries. The second (adversarial) party has a predicate $g(\cdot)$ as input and gets as output all database entries on which this predicate evaluates to 1. In this case, the ideal world satisfies the KTP

for $D = k + 1$. The predicator is supplied with all previous inputs and outputs and has to produce an answer for a new input g. The predictor looks at the previous outputs (each of which would consist of some entries of the database). It now constructs a partial database consisting of these entries and then answers g according to this database. Indeed, the output of the predictor maybe incorrect since it might be missing an entry which is in the real database but not in the partial one. However every time the predictor makes a mistake, it learns a new database entry (to be added to the partial database) by looking at the *correct output* having that missing entry. Hence, the predictor can produce an incorrect output at most k times.

Several problems of interest (such as private information retrieval, pattern matching, etc) are instances of the general problem of private database search discussed above. Arguments similar to those used for private database search can be used to construct predictors for other problems such as secure set intersection, computing k-th ranked element, etc (see the introduction for more details).

Bounded concurrent multi-party computation: We now consider the setting where there is an apriori fixed bound k on the total number of concurrent sessions in the ideal world (see, e.g., [Lin03b], [PR03], [Pas04]). If we set $D > k$, then there is at least one set S_i which will not have any sessions (by the pigeonhole principle). Hence, the predictor trivially succeeds for this set with probability 1.

The final version of our key technical property is weaker (and thus more general) than the warmup version. Details regarding that can be found in the full version.

III. A Warmup Construction

A. KTP and Concurrently Secure Computation with a Single Input

Consider an ideal world in the model of concurrently secure computation with a single input (defined in Section II-B). We now show that the ideal world for all "non-cryptographic" functionalities $\mathcal{F}$ satisfies the key technical property. To be more precise, we claim the following (see the full version for the proof).

Lemma 1: Consider an ideal world for a functionality $\mathcal{F}$ in the single input setting (section II-B). If the ideal world does not satisfy the key technical property (definition 3), then the functionality $\mathcal{F}$ must be a WC-PEF (definition 2).

B. Overview of Our Construction and Simulator

In this section, we describe our protocol Σ for a given two-party functionality $\mathcal{F}$ (extension to the case of multi-party is discussed later). Let P_1 and P_2 be two parties with private inputs x_1 and x_2 respectively. Let COM denote a non-interactive statistically binding commitment scheme. By WIAOK, we will refer to a witness indistinguishable argument of knowledge. Let Π_{SH} be any *semi-honest* secure two party computation protocol that emulates the functionality $\mathcal{F}$ in question in the stand-alone setting (as per the standard Ideal/Real world definition of secure computation). Let U_η denote the uniform distribution over $\{0,1\}^\eta$, where η is a function of the security parameter.

We shall make use of the DGS preamble [DGS09]. The DGS preamble allows a party to commit to the desired value in a way the simulator can extract that value by rewinding in the concurrent setting (similar PRS [PRS02] and RK [RK99] preambles). The total number of "slots" in the DGS preamble will be $m = \frac{2n^3D^2}{p}$ (where D and p come from definition 3). We now describe our protocol Σ.

I. Input Commitment Phase:

1) $P_2 \leftrightarrow P_1$: Party P_2 does the following. Generate a string $r_2 \stackrel{\$}{\leftarrow} U_\eta$ and let $X_2 = \{x_2, r_2\}$. Here r_2 is the randomness to be used (after coin-flipping with P_1) by P_2 in the execution of the protocol Π_{SH}. Using the commitment scheme COM, P_2 commits to the string X_2. Denote the commitment by B_2 and the decommitment information by β_2 (β_2 consists of the string X_2 and the randomness used to create B_2). Now P_2 additionally prepares mn pairs of secret shares $\{\alpha^0_{i,j}, \alpha^1_{i,j}\}_{i\in[n],j\in[m]}$ such that $\alpha^0_{i,j} \oplus \alpha^1_{i,j} = \beta_2$ for all i, j. Using the commitment scheme COM, P_2 also commits to all its secret shares. Denote these commitments by $\{A^0_{i,j}, A^1_{i,j}\}_{i\in[n],j\in[m]}$. Note that the string β_2 will also be called the "preamble secret".
   The party P_2 now engages in the execution of a (standalone) computational zero-knowledge argument with P_1 in order to prove that the above commit phase is "consistent" (i.e., all pairs of the secret shares sum up to the same value which is the preamble secret).
   P_2 and P_1 now execute a challenge-response phase. For $j \in [m]$:
   a) $P_1 \rightarrow P_2$: Send challenge bits $z_{1,j}, \ldots, z_{n,j} \stackrel{\$}{\leftarrow} \{0,1\}^n$.
   b) $P_2 \rightarrow P_1$: Send $\alpha^{z_{1,j}}_{1,j}, \ldots, \alpha^{z_{n,j}}_{n,j}$ along with the relevant decommitment information.

   At the end of this step, party P_2 is committed to its input and randomness. Informally speaking, the purpose of this phase is aid the simulator in extracting the adversary's input and randomness in the concurrent setting by rewinding the DGS preamble.
2) $P_1 \leftrightarrow P_2$: Party P_1 does the following. Generate a string $r_1 \stackrel{\$}{\leftarrow} U_\eta$ and let $X_1 = \{x_1, r_1\}$. Here r_1 is the randomness to be used (after coin-flipping with P_2) by P_1 in the execution of the protocol Π_{SH}. Using the commitment scheme COM, P_1 commits to X_1; denote this commitments by B_1.
   In addition, P_1 now engages in the execution of a WIAOK with P_2 in order to prove that it knows either: (a) a decommitment to the commitment B_1, or, (b) a decommitment to the commitment B_2.

II. Secure Computation Phase: In this phase, the parties run an execution of the semi-honest two party protocol Π_{SH}. Since Π_{SH} is secure only against semi-honest adversaries, parties first run a coin-flipping protocol to force the coins of each party to be unbiased and then "compile" Π_{SH} to enforce honest behavior. More details follow.

Coin Flipping. P_1 and P_2 first engage in a coin-flipping protocol. More specifically,

1) $P_1 \rightarrow P_2$: P_1 generates $r_2' \xleftarrow{\$} U_\eta$ and sends it to P_2. Define $r_2'' = r_2 \oplus r_2'$.
2) $P_2 \rightarrow P_1$: Similarly, P_2 generates $r_1' \xleftarrow{\$} U_\eta$ and sends it to P_1. Define $r_1'' = r_1 \oplus r_1'$.

Now r_1'' and r_2'' are the random coins that P_1 and P_2 will use in the execution of protocol Π_{SH}.

Protocol Π_{SH}. The parties P_1 and P_2 now execute the protocol Π_{SH}. Along with each outgoing message of Π_{SH},

(a) The party P_1 proves using WIAOK that either it has behaved honestly in Π_{SH} so far (based on the input x_1 and randomness r_1'' as defined earlier) or it knows a decommitment to the commitment B_2.

(b) The party P_2 proves using a (standalone) computational zero-knowledge argument that it behaved honestly (based on the input x_2 and randomness r_2'' as defined earlier).

: The above protocol satisfies the relaxed security notion for the single input setting as outlined in section II-B. To prove security, we need to consider the following two arguments. In the standalone setting, we need to exhibit security against a corrupt P_1. Secondly, we need to exhibit concurrent security for an honest P_1 (who may interact with various parties P_2's all of whom may be corrupted). This is the more interesting case and we first give a proof of security for this case.

The Simulator: First we provide only a high level overview of the simulator $\mathcal{S}$. The first task of $\mathcal{S}$ is to rewind the DGS preambles (in all concurrent sessions) and extract the preamble secret for each from the adversary. Such rewinding gives rise to a "main thread" and as well as several "look-ahead" threads of execution.

- In the input commitment phase, $\mathcal{S}$ can extract the decommitment of B_2 which includes the input/randomness of the adversary (to be used in the protocol Π_{SH}) from the DGS preamble.
- Using this decommitment information, $\mathcal{S}$ can cheat throughout by using it to complete all the WIAOK executions.
- In the secure computation phase, $\mathcal{S}$, by invoking the simulator of the protocol Π_{SH}, can complete the secure computation phase *given the output from the trusted functionality*.
- Note that for any given session, the input of the adversary may be different in different threads (since it might choose its input based on the transcript of the interaction so far). Now to complete the secure computation phase, $\mathcal{S}$ would need to get the corresponding output. To complete the secure computation phases *in the main thread*, $\mathcal{S}$ gets the output simply by querying the trusted party. For all the look-ahead threads, $\mathcal{S}$ gets the output by running the predictor guaranteed by the KTP in the ideal world.

We now give more details of our simulator $\mathcal{S}$.

Rewinding strategy of the simulator: We assume there are a total of k sessions with each session having one DGS preamble. Each of these preambles has $m = \frac{2n^3D^2}{p}$ "slots" with a slot representing a rewinding opportunity. The beginning of a slot is when the simulator gives the challenge, the end of the slot being when it receives the response. In between these two messages, there might be messages of other sessions.

As with the strategy in [RK99] (and [PV08]), the DGS rewinding schedule is "adaptive". At a very high level, whenever a slot s completes, the simulator may rewind s by calling itself recursively on s. That is, the simulator chooses another challenge for s and recursively executes until either it receives the response (and hence "solves" the preamble) or it observes that the adversary has executed "too many" new slots in between or has aborted. By the time the simulator completes the preamble in any thread, our choice of the number of slots guarantees that there would exist at least one recursive level which will have at least $\frac{2n^2D^2}{p}$ slots of that preamble. Whenever the simulator observes $\frac{2n^2D^2}{p}$ slots in one level, it would rewind each of those slots exactly once and will try to solve that preamble. The formal description of our simulator rewinding strategy CEC-Sim is given below. Some of the text is borrowed from [DGS09].

- $d_{max} = \lceil \log_n(2k \cdot m) \rceil$ will denote the maximum depth of recursion. Note that d_{max} is a constant since the number of preambles $2k$ and number of slots per preamble m is polynomial in the security parameter n. It would be helpful to keep in mind that $2k \cdot m$ is the total number of slots at depth 0 (i.e., the main thread).
- slot(i, j) will denote slot j of preamble i. d denotes the current depth of the recursion.

SOLVE$(x, d, h_{initial}, s)$*:*

Let $h \leftarrow h_{initial}$. Repeat forever and update h after each step:

1) If the adversary aborts or the number of slots in h started after $h_{initial}$ (which we will call new slots) exceed $\frac{2k \cdot m}{n^d}$, return h;
2) If the next message is a simulator challenge for the beginning of a slot s', choose the challenge randomly and send it to the adversary.
3) If the next message is the end message (adversary's reply) of a slot $s' =$ slot(i', j'), proceed as follows:
   a) If $s = s'$, we have succeeded in solving the target slot and hence the preamble. Return h;

b) Otherwise if the preamble i' has already been solved or the number of new slots (including s') of preamble i' in h started after $h_{initial}$ is less than $\frac{2n^2D^2}{p}$, the simulator need not rewind this slot. Go to the condition in Step 5;

c) Otherwise, we have an unsolved preamble i' such that $\frac{2n^2D^2}{p}$ of its slots (from slot$(i', j' + 1 - \frac{2n^2D^2}{p})$ to slot(i', j')) have appeared at the current level. The $\mathcal{S}$ will rewind each of these slots once and will try to solve preamble i'. Observe that the depth d_{max} of the recursion is a constant and the total number of slots in a preamble is $\frac{2n^3D^2}{p}$. This means just by the pigeonhole principle, for every preamble i', we would have this case at some level before the preamble is concluded. For each slot s" in this list of $\frac{2n^2D^2}{p}$ slots:

i) Let h" be the prefix of h which contains all messages up to but excluding the simulator challenge for s". Set $h^* \leftarrow$ SOLVE$(x, d + 1, h", s")$.

ii) If h^* contains an accepting execution for slot s", the simulator has succeeded in solving s" and hence preamble i'.

4) If the next message is the last message of a preamble and the preamble has not been solved yet, the current set of look-ahead threads have failed. Abort and output Ext_Fail if we are in the main thread. If we are in a look-ahead thread, return.

5) If the next message is a message which belongs to the secure computation phase, the simulator strategy is given next (for main as well as look ahead threads). If the next message belongs to neither the DGS preamble nor the secure computation phase, the simulator executes in a straightline manner as described above (i.e., using the decommitment to B_2 to complete WIAOK, playing honestly during coin flipping and while acting as the zero-knowledge verifier).

$CEC - Sim(x, z)$:

Run SOLVE$(x, 0, \perp, \perp)$ and output the view returned by SOLVE, with the following exception. When the simulator generates random challenge for a slot and it becomes equal to the another challenge generated previously in a different thread for the same slot, the simulator aborts and outputs $\perp$.

Simulator strategy in the secure computation phase: By the time $\mathcal{S}$ begins the secure computation phase in any session, it would have already extracted the input/randomness X_2 to be used by the adversary in the semi-honest two-party protocol. It invokes the simulator of the protocol Π_{SH} who makes a call to the trusted functionality. Now we have two possible cases:

1: $\mathcal{S}$ is currently executing the main thread. In this case, $\mathcal{S}$ goes ahead and queries the ideal functionality. It uses the received output to complete the secure computation phase.

2: $\mathcal{S}$ is currently executing some look ahead thread. This is the more interesting case since $\mathcal{S}$ cannot simply query the ideal functionality. $\mathcal{S}$ would now use the predictor $\mathcal{P}$ guaranteed by the KTP of the ideal world. More precisely, $\mathcal{S}$ simply invokes $\mathcal{P}$ on x_2 and all previously seen inputs/outputs in the current thread. Note that these previous outputs might have been answered by $\mathcal{S}$ either by making a call to the ideal functionality or using the predictor itself (in which case, correctness of the output is not guaranteed). Indeed assuming the adversary has full auxiliary information about the inputs of the honest parties, it can distinguish a correct output from an incorrect one (generated using the predictor). The core of the analysis of this prediction strategy (along with how it fits with our rewinding strategy) can be found in lemma 2.

C. Indistinguishability of the Outputs

We now consider a series of hybrid experiments and show that the views of $\mathcal{A}$ in successive hybrids are indistinguishable from each other.

Experiment H_0: The simulator $\mathcal{S}$ is given all the inputs of the honest parties. By running honest programs for the honest parties, it generates their outputs along with $\mathcal{A}$'s view. This is the execution in the real world in protocol Σ.

Experiment H_1: This experiment is exactly the same as the final simulated experiment except for the following. $\mathcal{S}$ still has all the honest party inputs and uses that to compute the outputs in the look ahead threads (instead of using the predictor).

The indistinguishability of the output distributions in H_0 and H_1 follows from standard techniques. By the property of the DGS preamble [DGS09] (also see the full version), the simulator is successful in extracting the preamble secret β_2 (which is a decommitment to B_2). In H_1, the simulator switches to using the witness β_2 to complete all WIAOK executions. Here, the indistinguishability argument relies on the witness indistinguishability of the WIAOK system. Furthermore, the secure computation phase is completed by using the simulator of the Π_{SH} protocol. Here, the indistinguishability argument relies on the security of the Π_{SH} protocol. The formal proof is given in the full version for our final construction.

Experiment H_2: This experiment is exactly the same as the previous one except for the following. $\mathcal{S}$ starts using the predictor in the look ahead threads. However it still has all the honest party inputs. $\mathcal{S}$ *aborts the execution of a thread* as soon as the predictor returns an incorrect output value in the execution of that threads. That is, $\mathcal{S}$ returns the view so far in the current recursive call without continuing it any further thus returning the execution to the thread one level below.

We now prove the indistinguishability of the output distributions in H_1 and H_2. This is the core of our rewinding analysis. First observe that actually the main threads in H_1 and H_2 are identical conditioned on the event that the simulator does not fail to solve a preamble (and output Ext_Fail). We shall now prove that the probability of $\mathcal{S}$ outputting Ext_Fail in this experiment is negligible. This is also where we use the special properties of the DGS rewinding strategy and our key technical property.

Lemma 2: The probability of the simulator outputting Ext_Fail in experiment H_2 is negligible.

The proof of the above lemma constitutes the core of our analysis of the rewinding strategy. For lack of space, the details of the proof can be found in the full version [Goy11b].

Experiment H_3: This experiment corresponds to the final simulated experiment. That is, $\mathcal{S}$ no longer has the honest party inputs and hence does not abort the look ahead threads where the predictor makes an error. Observe that in this hybrid, the probability of $\mathcal{S}$ outputting Ext_Fail can only go down compared to that in H_2. Hence, indistinguishability of the output distributions in H_2 and H_3 immediately follows thus completing the proof.

Running Time of $\mathcal{S}$: To bound the number of queries $\mathcal{S}$ makes to $\mathcal{A}$, we consider the recursive execution tree (of constant depth) resulting out of $\mathcal{S}$ rewinding $\mathcal{A}$. Each call to the function SOLVE$(\cdot,\cdot,\cdot,\cdot)$ will represent one node in the execution tree. The nodes resulting from all further recursive calls to SOLVE will be treated as children of this node. Thus, the root node (at depth 0) is the call SOLVE$(x,0,\cdot,\cdot)$ made by $CEC - Sim(x,z)$. This call results in the main thread while recursive calls give rise to the look ahead threads.

Now consider the transcript generated by a function call representing a node at depth d (excluding the transcripts generated by any further recursive calls). The number of new slots in this transcript is bounded by $2k \cdot \frac{2n^3D^2}{p}$ (in fact $\frac{2k \cdot 2n^3D^2}{pn^d}$). Now, each of these slots may have up to one look ahead thread resulting in a total of up to $4k\frac{2n^3D^2}{p}$ children for this node. Hence, the execution tree is a tree of depth up to d_{max} and degree up to $4k\frac{2n^3D^2}{p}$. Hence, the total number of nodes is bounded by $(4k\frac{2n^3D^2}{p})^{d_{max}+1}$. The transcript of each node contains up to $O(4k\frac{2n^3D^2}{p})$ queries. Hence, the total number of queries $\mathcal{S}$ makes to $\mathcal{A}$ is $O((4k\frac{2n^3D^2}{p})^{d_{max}+2})$ which is a polynomial (since d_{max} is a constant). Also, its easy to see that each query to $\mathcal{A}$ takes only PPT assuming $\mathcal{A}$ is a PPT machine. This concludes our analysis.

Final Construction: The above construction only provides concurrent security for one party. This is because in case P_1 is corrupted and interacts with *multiple* honest parties, its input in one session may somehow be dependent upon the input of an honest parties in another session. To overcome this problem, we use techniques from concurrent non-malleable zero-knowledge protocols [BPS06]. Our final construction (relying on BPS concurrent non-malleable ZK) can be found in the full version.

ACKNOWLEDGMENTS

We thank Rafael Pass for useful suggestions and in particular for suggesting to present a simpler protocol first based on compilation only with concurrent zero-knowledge. We thank Amit Sahai for many useful suggestions. Thanks also to Abhishek Jain and Rafail Ostrovsky for useful discussions. Finally, we thank Yuval Ishai to urging us to explore connections with other published works more deeply. This led to a substantial strengthening of our results.

REFERENCES

[AMP04] Gagan Aggarwal, Nina Mishra, and Benny Pinkas. Secure computation of the k th-ranked element. In Cachin and Camenisch [CC04], pages 40–55.

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *FOCS*, pages 106–115, 2001.

[BCL+05] Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure computation without authentication. In *CRYPTO*, pages 361–377, 2005.

[BCNP04] B. Barak, R. Canetti, J.B. Nielsen, and R. Pass. Universally composable protocols with relaxed set-up assumptions. In *FOCS*, pages 186–195, 2004.

[BGP00] Mihir Bellare, Oded Goldreich, and Erez Petrank. Uniform generation of np-witnesses using an np-oracle. *Inf. Comput.*, 163(2):510–526, 2000.

[BPS06] Boaz Barak, Manoj Prabhakaran, and Amit Sahai. Concurrent non-malleable zero knowledge. In *FOCS*, pages 345–354, 2006.

[BS05] Boaz Barak and Amit Sahai. How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In *FOCS*, pages 543–552. IEEE Computer Society, 2005.

[Can08] Ran Canetti, editor. *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, volume 4948 of *Lecture Notes in Computer Science*. Springer, 2008.

[CC04] Christian Cachin and Jan Camenisch, editors. *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*. Springer, 2004.

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *FOCS*, pages 41–50, 1995.

[CHK+05] Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In *EUROCRYPT*, pages 404–421, 2005.

[CKL06] R. Canetti, E. Kushilevitz, and Y. Lindell. On the limitations of universally composable two-party computation without set-up assumptions. *J. Cryptology*, 19(2):135–167, 2006.

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

[CLOS02] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *STOC*, pages 494–503, 2002.

[CLP10] Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive hardness and composable security from standard assumptions. In *FOCS*, 2010.

[DGS09] Yi Deng, Vipul Goyal, and Amit Sahai. Resolving the simultaneous resettability conjecture and a new non-black-box simulation strategy. In *FOCS*, pages 251–260. IEEE Computer Society, 2009.

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

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Cachin and Camenisch [CC04], pages 1–19.

[GGJS12] Sanjam Garg, Vipul Goyal, Abhishek Jain, and Amit Sahai. Bringing people of different beliefs together to do uc. In *Eurocrypt (To appear)*, 2012.

[GJO10] Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Password-authenticated session-key generation on the internet in the plain model. In Rabin [Rab10], pages 277–294.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2008.

[GL01] Oded Goldreich and Yehuda Lindell. Session-key generation using human passwords only. In *CRYPTO*, pages 408–432, 2001.

[GL03] Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In *EUROCRYPT*, pages 524–543, 2003.

[GLOV12] Vipul Goyal, Chen Lee, Rafail Ostrovsky, and Ivan Visconti. Constructing non-malleable commitments: A black-box approach. In *FOCS*, 2012.

[GM00] Juan A. Garay and Philip D. MacKenzie. Concurrent oblivious transfer. In *FOCS*, pages 314–324. IEEE Computer Society, 2000.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC '87: Proceedings of the 19th annual ACM conference on Theory of computing*, pages 218–229, New York, NY, USA, 1987. ACM Press.

[Goy11a] Vipul Goyal. Constant round non-malleable protocols using one way functions. In *STOC*, 2011.

[Goy11b] Vipul Goyal. Positive results for concurrently secure computation in the plain model. *IACR Cryptology ePrint Archive*, 2011:602, 2011.

[HL08a] Carmit Hazay and Yehuda Lindell. Constructions of truly practical secure protocols using standards-smartcards. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM Conference on Computer and Communications Security*, pages 491–500. ACM, 2008.

[HL08b] Carmit Hazay and Yehuda Lindell. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In Canetti [Can08], pages 155–175.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *STOC*, pages 44–61. ACM, 1989.

[Kat] J. Katz. Universally composable multi-party computation using tamper-proof hardware. In *Eurocrypt 2007*.

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *FOCS*, pages 364–373, 1997.

[KOY01] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient password-authenticated key exchange using human-memorable passwords. In *EUROCRYPT*, pages 475–494, 2001.

[KP01] Joe Kilian and Erez Petrank. Concurrent and resettable zero-knowledge in poly-loalgorithm rounds. In *STOC*, pages 560–569, 2001.

[Lin03a] Yehuda Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *STOC*, pages 683–692. ACM, 2003.

[Lin03b] Yehuda Lindell. General composition and universal composability in secure multi-party computation. In *FOCS*, pages 394–403. IEEE Computer Society, 2003.

[Lin08] Yehuda Lindell. Lower bounds and impossibility results for concurrent self composition. *J. Cryptology*, 21(2):200–249, 2008.

[LPTV10] Huijia Lin, Rafael Pass, Wei-Lung Dustin Tseng, and Muthuramakrishnan Venkitasubramaniam. Concurrent non-malleable zero knowledge proofs. In Rabin [Rab10], pages 429–446.

[MPR06] Silvio Micali, Rafael Pass, and Alon Rosen. Input-indistinguishable computation. In *FOCS*, pages 367–378. IEEE Computer Society, 2006.

[Pas03] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 160–176. Springer, 2003.

[Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. pages 232–241, 2004.

[PR03] Rafael Pass and Alon Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. 2003.

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

[PS04] Manoj Prabhakaran and Amit Sahai. New notions of security: achieving universal composability without trusted setup. In László Babai, editor, *STOC*, pages 242–251. ACM, 2004.

[PV08] Rafael Pass and Muthuramakrishnan Venkitasubramaniam. On constant-round concurrent zero-knowledge. In Canetti [Can08], pages 553–570.

[Rab10] Tal Rabin, editor. *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*. Springer, 2010.

[RK99] Ransom Richardson and Joe Kilian. On the concurrent composition of zero-knowledge proofs. In *EUROCRYPT*, pages 415–431, 1999.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167. IEEE, 1986.