How To Play Almost Any Mental Game Over The Net — Concurrent Composition via Super-Polynomial Simulation

Boaz Barak*
Department of Computer Science
Princeton University
Princeton, New-Jersey
boaz@cs.princeton.edu

Amit Sahai†
Department of Computer Science
University of California Los Angeles
Los Angeles, California
sahai@cs.ucla.edu

Abstract

We construct a secure protocol for any multi-party functionality that remains secure (under a relaxed definition of security introduced by Prabhakaran and Sahai (STOC '04)) when executed concurrently with multiple copies of itself and other protocols, without *any assumptions on existence of trusted parties, common reference string, honest majority or synchronicity of the network. The relaxation of security is obtained by allowing the ideal-model simulator to run in* quai-polynomial *(as opposed to polynomial) time. Quasi-polynomial simulation suffices to ensure security for most applications of multi-party computation. Furthermore, Lindell (FOCS '03, TCC' 04) recently showed that such a protocol is* impossible *to obtain under the more standard definition of* polynomial-time *simulation by an ideal adversary.*

Our construction is the first such protocol under reasonably standard cryptographic assumptions (i.e., existence of a hash function collection that is collision resistent with respect to circuits of subexponential size, and existence of trapdoor permutations which are secure with respect to circuits of quasi-polynomial size).

We introduce a new technique: "protocol condensing". That is, taking a protocol that has strong security properties but requires super-polynomial *communication and computation, and then transforming it into a protocol with* polynomial *communication and computation, that still inherits the strong security properties of the original protocol. Our result is obtained by combining this technique with previous techniques of Canetti, Lindell, Ostrovsky, and Sahai (STOC '02) and Pass (STOC '04).*

*Part of the work was done while in the Institute for Advanced Study and partially supported by NSF grants DMS-0111298 and CCR-0324906.

†This research was supported by generous grants from the NSF ITR and Cybertrust programs, an equipment grant from Intel, and an Alfred P. Sloan Foundation Fellowship.

1. Introduction

In the 1980's a sequence of groundbreaking papers [53, 52, 50, 9, 31, 32, 29, 54] led to the rather amazing result of Goldreich, Micali and Wigderson [28] (henceforth GMW) that it is possible in principle to obtain a secure protocol for essentially *every* cryptographic task one can think of, whether it is secure electronic elections, auctions, privacy-preserving data mining, or poker. GMW achieved this result by constructing a compiler that transformed a naive protocol that achieves some task with no security whatsoever (e.g., in the case of elections, a protocol where all parties send their votes to a party T which counts the votes and announces the results) into a protocol that seemed to obtain the highest level of security one can hope for. That is, their protocol guaranteed that every party or coalition of parties, (even if they cheat and do not follow the protocol), still cannot learn more information or have a larger effect on the outcome than they are entitled to obtain by simply following the rules (e.g., in the example of elections, no party or coalition of parties can vote more than their number or deduce about the other votes more than can be deduced from the publicly announced results).

Although it was always clear that the GMW protocol is far from being practical in terms of the overhead it incurred in computation and communication, it might have seemed initially that there is not much to improve on its *security*. However, with the advent of modern networks, it became clear that this is *not* the case. The reason is that although GMW's protocol (and also protocols for simpler tasks such as zero knowledge) guarantees security in the case of an isolated execution, it does not *not* guarantee sufficient security in the increasingly common situation in which parties run the protocol concurrently with other arbitrary network activity, which can include multiple executions of the same protocol and other cryptographic and non-cryptographic protocols. In fact, there are examples of instantiations of GMW and other stand-alone protocols with

Proceedings of the 2005 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)
0-7695-2468-0/05 $20.00 © 2005 IEEE

particular primitives, for which there are known successful *attacks* in the concurrent setting [27, 21].

Thus, in the 1990's, researchers began to work on definitions and protocols that are applicable for this "general network" setting. Although, as we elaborate below, this very extensive line of research had many successes, it still fell short of obtaining the corresponding stronger version of GMW's theorem: i.e., a general multi-party computation protocol (or even protocols for specific tasks such as commitment schemes or zero-knowledge proofs) that remains secure in this setting under standard cryptographic assumptions. Even more disturbingly, [12, 14, 16, 38, 39] gave increasingly stronger negative results, showing that it is actually *impossible* to obtain a protocol satisfying the natural strengthening of the stand-alone definition to the general-network setting.

As we discuss below (see Section 1.1) there have been many works suggesting approaches to bypass the negative results. Recently, Prabhakaran and Sahai [49] suggested a definition which seemed to bypass the impossibility results without changing the network model or making any setup assumptions. Their approach (which we follow here), is to allow the ideal-world simulator to run in *super-polynomial* time (a notion first explicitly suggested by Pass [45]). As discussed below, this relaxation still provides meaningful and strong security for the canonical application of multi-party computation.[1] However, the result of [49] was under a highly non-standard computational assumption (see below) and hence it was not clear whether their definition is in fact satisfiable.

Our results. In this work we obtain a protocol satisfying the [49] definition under reasonably standard cryptographic assumptions (namely, existence of subexponentially strong hash functions and quasipolynomially strong trapdoor permutations).[2] For *every* polynomial-time functionality $\mathcal{F}$ we construct a protocol that securely realizes $\mathcal{F}$ in the general network setting, without any setup assumptions, with security defined as existence of an ideal-model simulator that runs in *quasi-polynomial* time.[3] That is, if the adversary runs in time T, our simulator runs in time $2^{(\log T)^c}$ for some constant $c > 1$, and hence we can simulate a polynomial-time adversary in quasi-polynomial time, and a subexponential-time adversary (with a low enough exponent) in subexponential time.

At the heart of our construction is a fully *concurrent* and *non-malleable* zero-knowledge protocol using quasi-polynomial simulation. This protocol has a constant number of rounds and is based on the assumption that there exists a hash function collection that is collision-resistent with respect to 2^{k^ϵ}-sized circuits (where k is the security parameter and $\epsilon > 0$ is some constant). Plugging this protocol into the results of Canetti, Lindell, Ostrovsky, and Sahai [17], we obtain a *fully concurrent and non-malleable* protocol for computing any polynomial-time functionality under reasonably standard assumptions (i.e., existence of quasi-polynomially strong enhanced trapdoor permutations). Again, security of this protocol is demonstrated by a *quasi-polynomial* simulator. Furthermore, our protocol utilizes only a *constant* number of communication rounds and remains secure also with respect to *adaptive* adversaries (without using memory erasures). See Section 2 below for formal statements and more details on our results.

Why is quasi-polynomial simulation good enough? In the simulation paradigm, we simply *define* a protocol to be secure if its execution can be simulated in an ideal model where a polynomial-time adversary has only access to "ideal boxes" that implement the functionality. In our opinion, this standard definition is justified by two points:

1. It is the strongest possible, in the sense that it is *impossible* to prevent an attack that is feasible in this ideal model.
2. Intuitively, simulation-based security should imply the actual security concerns of the user such as privacy, integrity, input independence, etc. (although more often than not this implication is not explicitly spelled out).

In the definition we and [49] use, the ideal model is augmented to allow the adversary (some fixed) *super-polynomial* computation while accessing these "ideal boxes". This means that we no longer enjoy Property 1 of the standard definition. However, it seems that we still, in many cases, enjoy Property 2. The reason is that in most cases, if the security in the ideal model for *polynomial-time* adversaries indeed implies privacy, integrity, etc.., then this will actually hold for all adversaries with running time at most $T(n)$ for some explicit super-polynomial function $T(\cdot)$ that depends on the hardness assumptions used.[4] Thus, using quantitatively strong enough hardness assumptions and large enough security parameter, we can ensure that $T(n)$ is larger than the time we allow our simulator to run. Note that this is not *always* the case (and hence the "almost" in the title) – for some functionalities such as the game of

[1] By "canonical application" we mean using a multi-party computation protocol to obtain a protocol for a specific task satisfying task-specific security properties such as privacy, integrity, and input independence. That is, using simulation as *tool* to derive security and not an *end result*. Even though secure multi-party computation has other applications beyond this, we believe that the name "canonical application" is appropriate as this is the application that motivated both the constructions and the definitions of general secure computation protocols.

[2] Both these assumptions are implied by the assumption that there's a constant $\epsilon > 0$ such that the factoring problem is hard for 2^{n^ϵ}-sized circuits.

[3] This is opposed to the standard (impossible to achieve) definition of *polynomial-time* simulation.

[4] In fact, in many cases the ideal model is simple enough that this implication holds even if the adversary can run in *unbounded* time.

Proceedings of the 2005 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)
0-7695-2468-0/05 $20.00 © 2005 IEEE

Chess or proof-of-work schemes [19] it is not possible to make even the ideal model secure against super-polynomial time. Note however that such functionalities are also problematic for polynomial-time simulation. We also note that typically in polynomial-simulation protocols the simulation time is not just polynomial-time but is actually a fixed explicitly known polynomial in the adversary's running time. This property, which is lost in super-polynomial simulation, has been useful before in applications such as *deniable protocols* [18, 20, 13, 43] and hence our protocol fails to achieve such applications.[5]

General composition or "chosen protocol attack". Another requirement that was considered in the literature is that a concurrently-composable protocol should remain secure even if it is used concurrently with arbitrary other protocols, including even protocols that were maliciously designed to be insecure when interacting with the concurrently-composable protocol. This property was called "chosen protocol attack" by [34] and *general composition* by [38]. Although for this requirement to make sense the other protocols have to be secure (as otherwise composition is meaningless), in the case of super-polynomial simulation they have to be "strongly secure" (strong even against super-polynomial time) and hence our protocols cannot be said to fully satisfy this notion. Note however that similar restrictions hold also for protocols such as [17] in the common reference string model (where the other protocols are required not to use the reference string), and [33] (where the other protocols are required to introduce timeout and delay mechanisms), although such restrictions do not hold for protocols in the honest majority setting such as [8].

New technique – "condensed protocols"[6]. To achieve our result, we introduce a new technique that allows us to take a protocol Π that has super-polynomial communication and computation requirements (but polynomial-sized inputs), and "condense" it to obtain a protocol Π' with only polynomial communication and computation requirements, while ensuring that the condensed protocol Π' retains the strong security properties of the super-polynomial protocol Π. (This is useful since, using the techniques of Pass [46], it is possible to construct such a super-polynomial protocol Π with the attractive security properties we need.) Roughly speaking, the initial idea behind this "condensation" is to replace every super-polynomially long message m in Π with its short hash $h(m)$, and use Universal Arguments [6] to prove correctness of the hashed value. This by no means completes our task, as we have two fundamental problems: (1) the hashed messages now contain too little information to allow for the other party to compute a proper response; and (2) even if one had the long message to compute with, the computation time required to compute a response would still be super-polynomial. Solving Problem (1) involves a few technical tricks and is responsible for many of this work's technical complications. To solve Problem (2), we use the following approach: we "encrypt" all communication in the protocol, and then provide honest parties an "honest backdoor" that allows them to successfully complete the protocol using their private information. In the context of a zero-knowledge proof of the statement $x \in L$, this can be done by allowing the prover to prove that *either* the encryption of the super-polynomial protocol Π is accepting, *or* that $x \in L$ is true. Since the honest prover will have a witness to the truth of $x \in L$, it can use this knowledge to quickly (i.e. in polynomial time) prove the statement, without ever actually participating in the super-polynomial protocol. Remarkably, because an adversary can never be sure of which condition actually holds, we are able to argue that such a condensed protocol Π' retains the strong security properties of the super-polynomial protocol Π.

[5] Note however that deniability is a delicate property that is hard even to define in the general concurrent network setting, and some previous works in this area such as [17] also fail to achieve deniability, even when using setup assumptions (e.g., see [44]).

[6] We use quite a few known techniques, and introduce several new techniques as well. We discuss in detail our techniques in Section 3, and so in this paragraph we'll restrict ourselves to a terse summary of the main new technique introduced.

1.1. Related Works.

There has been a very large body of research on multi-party secure computation and on composition of cryptographic protocols. In this section we will briefly describe some of the works on concurrent composition of protocols for general multi-party functionalities. We discuss the works relevant to the techniques of this paper in Section 3. See the books by Goldreich [25, CHAP. 7] and Lindell [37], and the references therein for a more comprehensive review of the literature.

As already mentioned above, the natural strengthening of GMW's result to the concurrent setting was not obtained, and in fact some negative results [12, 14, 16, 38, 39] showed that this is inherent. Thus, previous works considered various *relaxation* which included either assumptions on existence of trusted parties, limits on the concurrency or asynchronicity of the network, or relaxed security. The standard definition of security under general concurrent composition was given by Canetti [12] (and is known as UC or ES security). Some weaker definitions were considered by [18] and [20] in the context of zero-knowledge proofs and other specific two-party functionalities.

The CRS model: Canetti, Lindell, Ostrovsky and Sahai [17] gave a UC-secure protocol for any functionality in *common reference string* (CRS) model [10], where the only

Proceedings of the 2005 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)
0-7695-2468-0/05 $20.00 © 2005 IEEE

assumption is that there is a publicly known string that was chosen once and for all by some *completely trusted* party. An approach to distributing some of the trust of choosing the reference string was recently taken by [5]. **Honest majority:** Canetti [12] showed that the protocol of Ben-Or, Goldwasser and Wigderson [8] is UC-secure under the assumption that a majority of the participating parties can be trusted. However, this assumption seems to be less reasonable in a general network setting such as the Internet, and in particular does not allow for 2-party protocols or subprotocols. **Timing assumptions:** Taumann-Kalai, Lindell and Prabhakaran gave a UC-secure multi-party protocol *timing model* [20], in which one assumes that all the parties have clocks with some bounds on the drift between the clocks and on the time to transmit a message across the network. The main problem with the protocol of [33] (and all other protocols in the timing model such as [20, 24]) is that they require that *every* message in *every* protocol running in the network will be delayed by amount of time that is larger than the latency of the slowest link in the network. Thus, such protocols do not seem suitable for a heterogenous network in which some parties have significantly faster connections than other parties. **Bounded concurrency:** Lindell [36], later improved by Pass and Rosen and Pass [47, 46, 48] gave *bounded-concurrent* [1] protocols for any functionality. Such protocols assume that there is a fixed known polynomial upper-bound M on length of *all* the communication throughout the entire network. However, these protocols use computation and communication that is *larger* than M, and this was shown to be necessary by [36]. Hence, while bounded-concurrent protocols can be sometimes very useful tools in other constructions (and indeed we use techniques from Pass [46] and Pass and Rosen [48] in this paper), they do not seem suitable as a solution for obtaining secure computation in the general network setting.

Relaxed security in the standard model. Another approach, which is the one taken in this work, is not to make stronger assumptions on the network or trust, but rather achieve a weaker notion of security. **Super-polynomial simulation:** The relaxation considered in this work is to allow the ideal-model simulator to run in time which not a polynomial in the running time of the adversary but rather some super-polynomial (e.g., quasi-polynomial) function in this time. This notion was implicit in some works (e.g., [15]) but was first explicitly put forward in [45], who suggested this notion could be used as a way to obtain concurrently composable protocols and in particular used this relaxation to obtain concurrent zero knowledge. **The PS paper:** Prabhakaran and Sahai [49] gave a construction of a concurrently-composable multi-party computation protocol in the general-network setting under the super-polynomial simulation definition of security and using non-standard computational assumptions. Since the previous negative results were often interpreted that one must either use setup assumptions, or give up on ideal-model simulation-based security, [49] offered the exciting possibility of obtaining secure protocols without giving up either. However, in our view the weak point of [49] was the computational assumption used, which essentially assumes that there exists a cryptographic hash function (not a collection of functions) that is a non-malleable commitment scheme. While unlike the Random Oracle Model [7], the assumptions of [49] are valid and well-defined mathematical assumptions, they are not well-studied, and seem to be difficult to analyze because of their complexity. On a more technical level, although [49] tackles some major technical difficulties such as getting UC composition to work in this setting, they essentially do not tackle non-malleability from a technical standpoint, and instead assume it to be present in the hash function. The current work can be seen as subsuming the result of [49] by obtaining it under standard assumptions[7].

2. Model and Results

The network model we consider is the same one as in [11, 12, 38] . There is a network of point-to-point channels between a set of parties. Each party has a string that uniquely identifies it (which we call the party's ID). The parties do not need to be aware of each other's existence. An adversary can do the following: (1) Control some of the parties (such parties are said to be "corrupted"), (2) create new parties dynamically, (3) view all messages submitted on the network, and (4) fully control the scheduling of these messages. We denote the strategy that an honest party P_i uses as π_i. This strategy models all the activity of P_i, including all protocols[8], cryptographic or non-cryptographic, that are executed sequentially or concurrently by P_i. We denote the collection of all these strategies for all parties by π. We note that in this model the adversary can control all scheduling of messages to honest parties, and hence can indefinitely postpone the delivery of messages to any honest party. Thus this work (as is the case with [17] and with [28] in the non honest-majority case) does not guarantee security against denial of service attacks or provide the related

[7] Prabhakaran and Sahai [49], aside from obtaining their result on secure multiparty computation, also put forward a new framework for security definitions. This is something we do not do in this paper. Our result can be seen as holding within the "Angel" definitional framework of [49]. However, for the sake of being as self-contained as possible, we instead prove our result directly in the context of the definitions of [12]. We also note that recently [40] gave a different construction in the [49] model, which is based on different non-standard assumptions of a more number-theoretic nature (they assume some non-malleability of the discrete log problem).

[8] Another equivalent way to model this, following [12], is to have a special adversarial entity called an *environment* that models all other protocols happening in the system, other than the one being analyzed. We follow this modeling in the detailed description of our protocol.

Proceedings of the 2005 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)
0-7695-2468-0/05 $20.00 © 2005 IEEE

guarantee of fairness [30].

Security definition. If $\mathcal{F}$ is some (possibly probabilistic, stateful) functionality, then the *$\mathcal{F}$-hybrid model* is the same model augmented by an additional trusted party that computes $\mathcal{F}$. We say that a protocol ρ *securely computes $\mathcal{F}$ with polynomial simulation* if the following holds: for every polynomial-sized adversary $\mathtt{Adv}$ there exists an polynomial-sized adversary $\mathtt{Adv}'$ in the $\mathcal{F}$-hybrid model such that if π is an honest parties strategy that includes calls to ρ as a subroutine, then the view of $\mathtt{Adv}$ when interacting with π is indistinguishable from the view of $\mathtt{Adv}'$ when interacting with π', where π' is obtained from π by replacing all calls to the ρ subroutine with calls to the ideal function $\mathcal{F}$. We say that ρ *securely computes $\mathcal{F}$ with quasi-polynomial simulation* if $\mathtt{Adv}'$ is allowed to be of quasi-polynomial (i.e. , $k^{\log^{O(1)} k}$) size.

2.1. Our Results.

We consider the zero-knowledge ideal functionality $\mathcal{F}_{\mathsf{ZK}}$ (for an NP-complete problem such as SAT) which gets as input from party P_i two strings y and w and the identity of a party P_j, and sends to P_j the tuple (ZK,P_i, P_j, y,) if w is a satisfying assignment for the formula y, and does nothing otherwise.

Our main result is a construction of a protocol for securely implementing the $\mathcal{F}_{\mathsf{ZK}}$ functionality under general composition. Namely, we prove the following theorem:

Theorem 2.1 (General-concurrent zero knowledge). *Suppose that there exists a hash function collection that is collision resistent for 2^{k^ϵ}-sized adversaries (where $\epsilon > 0$ is a constant and k denote the collection's security parameter). Then, there exists a protocol that securely realizes the $\mathcal{F}_{\mathsf{ZK}}$ functionality with quasi-polynomial simulation.*

Canetti *et al.* [17] showed how to securely compute *any* functionality in the $\mathcal{F}_{\mathsf{ZK}}$-hybrid model. Thus, by observing that their results "scale up" and hold in our model, and by plugging in Theorem 2.1, we obtain the following result:

Theorem 2.2 (General-concurrent secure function evaluation). *Suppose that there exists a hash function collection that is collision-resistent for 2^{k^ϵ}-sized adversaries (where $\epsilon > 0$ is a constant and k denote the collection's security parameter). Then, there exists $c = c(\epsilon)$ such that if there is a collection of enhanced trapdoor permutations which is secure for $k^{\log^c k}$-sized adversaries then for* every *(possibly probabilistic) polynomial-time functionality $\mathcal{F}$, there is a protocol $\rho_{\mathcal{F}}$ that securely realizes $\mathcal{F}$ with quasi-polynomial simulation.*

3. Overview of Our Techniques.

In this section we provide an rough overview of our approach to obtaining a zero-knowledge protocol that is secure under general concurrent composition. That is, we describe our approach to proving Theorem 2.1. We start by briefly describing some of the primitives and tools we use. We then present how one can obtain such a protocol by combining two different approaches that fail with some new techniques and tricks. We warn the reader that this description is missing a few important subtleties and issues that make the our actual construction and proof more complicated. Because of these subtleties, our actual construction (Protocol 3.1) does not exactly follow the approach illustrated in this section, but follows a more "low level" approach.

3.1. Preliminaries.

We will use the following primitives and sub protocols. Because this is an overview section, we describe the primitives in an informal way, and also present each primitive in its simplest variant, even if this variant requires stronger assumptions than the ones stated in Theorem 2.1. We will use the following primitives:

Commitment schemes. A non-interactive perfectly binding and computationally hiding commitment scheme Com [9, 42].

Zero-Knowledge proofs of knowledge. A constant-round zero-knowledge proof/argument of knowledge for $\mathbf{NP}$ [22, 26]. We will also sometimes use the weaker notion of a *witness indistinguishable* proof, which we denote by WIP [23]. We note that witness-indistinguishability, unlike zero-knowledge, is closed under concurrent composition.

Collision resistant hash functions. A collection Hash of functions that map arbitrarily long strings into polynomial-sized strings such that it is hard given a random $h \in \mathsf{Hash}$ to find x, y such that $h(x) = h(y)$. We note that by combining a hash function with a commitment scheme we can obtain a commitment scheme that allows us to commit to messages of unbounded size.[9]

Universal arguments. A constant-round public-coin argument of knowledge for $\mathbf{Ntime}(T)$ for a *super-polynomial* function $T(\cdot)$ (e.g., $T(k) = k^{\log k}$). Universal arguments were first constructed by [35], with improved analysis in [41] and [6] (with the latter work

[9] We ignore here the issue of who gets to choose the hash function – the sender or the receiver. Although intuitively it seems that the receiver should choose the hash function, it turns out that in some cases we actually want the sender to choose it. For the sake of this overview, the reader can assume that each party chooses its own hash function and then they use the function that on input x returns the concatenation of both functions applied to x. This function is guaranteed to be collision resistant if one of the parties is honest.

Proceedings of the 2005 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)
0-7695-2468-0/05 $20.00 © 2005 IEEE

showing they are a proof of knowledge). We'll also use constructions of universal arguments that are *zero knowledge* and *witness indistinguishable* [35, 6, 3]. Universal arguments have the property that the total communication and running time of the verifier is always polynomial, even if the statement proven is not in $\mathbf{NP}$. Furthermore, the running time of the prover is polynomial in the time to actually verify the instance being proven. For example, if $L \in \mathbf{NP}$ and $L' \in \mathbf{Ntime}(k^{\log k})$ and one is proving using universal arguments that $x \in L \cup L'$ then if x is in fact in L and the prover is given a witness to this fact, then the prover can execute the proof in polynomial time.

Knowledge commitments. We denote by KCom the protocol in which a sender commits to a string x using $\mathsf{Com}(\cdot)$ and then proves knowledge of the committed string using a zero-knowledge proof of knowledge. We denote by UAKCom the same protocol in which the sender commits to $h(x)$ and proves knowledge of x using a zero-knowledge universal argument.

Weak commitments. We denote by $\mathsf{Com}_{\mathsf{weak}}$ a commitment scheme that can be completely broken in time that is smaller than the time to violate the security of all the other primitives we use. Such a commitment scheme can be constructed under our assumptions using the complexity leveraging technique of [15].

Brute force breaking opportunity. We denote by BFOP the protocol in which a verifier sends $\mathsf{Com}_{\mathsf{weak}}(r)$ and then the prover sends $\mathsf{KCom}(r')$ for some string r'. We say that the prover *broke* this instance if $r' = r$. Note that this protocol can be broken by breaking $\mathsf{Com}_{\mathsf{weak}}$. Similar tricks were used in several previous works such as [15, 45].

3.2. First Attempt: The Brute Force Protocol

Recall that we're trying to prove Theorem 2.1 by constructing a general-concurrent secure zero-knowledge argument. Here's a naive attempt at such a protocol (that was used by Pass [45] in a similar context), which we denote by Π_{BF}: let L be an $\mathbf{NP}$-language with a corresponding relation R. To prove that $x \in L$, given w such that $(x, w) \in R$, the prover and verifier interacts as follows:

1. Prover sends $\mathsf{Com}_{\mathsf{weak}}(w)$ to the verifier.
2. Prover and verifier interact in a brute-force breaking opportunity BFOP.
3. Prover proves to verifier in WI that it either committed to the witness in the first step or that it broke the BFOP in the second step.

It is not hard to verify that this protocol satisfies completeness and soundness. In fact, in a *real* concurrent interaction, whenever the verifier is honest, the probability that it accepts a proof without the weak commitment actually containing a witness is negligible. There is a natural straight-line black-box simulator for Π_{BF} [45]: when simulating an interaction in which the adversary is a verifier, the simulator commits to 0^k instead of to the witness, and then breaks BFOP and uses this fact to run the WI proof of Step 3. It is not hard to prove that the simulator's output is indeed indistinguishable from a real execution[10].

When simulating an interaction in which the adversary is the prover, the simulator will attempt to extract a witness by breaking the weak commitment sent by the adversary. However, in this case, we are not sure that it will succeed. The property we're looking for, that even during the simulation the adversary's proof must contain a real witness, is called *simulation soundness* [51], and this property lies at the heart of constructing non-malleable zero-knowledge protocols. Unfortunately, it can be shown that protocol Π_{BF} does *not* satisfy this property (i.e., there is a known attacking strategy on instantiations of Π_{BF} with particular primitives).

3.3. Second Attempt: The Condensed Protocol

The problem with the first attempt was that that protocol did not satisfy simulation soundness / non-malleability (it is essentially the same property). There are very few simulation-sound zero-knowledge protocols without setup assumptions [18, 2, 46, 48] and most of these are only analyzed in the scenario where there are only two executions occurring concurrently: one in which the adversary is the verifier and another in which the adversary is the prover. Pass [46] constructed the first protocol which remained simulation sound even when the adversary interacts not just in two executions but in k (where k is the security parameter) executions – playing the role of prover in some, and playing the role of verifier in others. Here, k is the security parameter. However, that protocol used $O(k)$ rounds which will be problematic in this setting. Nonetheless, it was observed in [4] that using the ideas of Pass and Rosen [48], it is possible to convert a different protocol of Pass [46] to a constant-round protocol with this property.[11] We denote this protocol (which is essentially based on [46]) by bgcZK

[10] Thus, the protocol Π_{BF} is a concurrent zero knowledge protocol with quasi-polynomial simulation. But note that we need more than just concurrent zero knowledge, because the adversary can also play the role of prover.

[11] Pass [46] did give a also a *constant-round* protocol satisfying this property assuming the ID's of each party come from a polynomial-sized domain. Pass and Rosen [48] showed how one can convert this protocol to a standard simulation-sound protocol by having a party with ID $\alpha = \alpha_1, \ldots, \alpha_k$ run k parallel executions of Pass's protocol using the ID $\langle i, \alpha_i \rangle$ for the i^{th} execution. Barak *et al.*[4] observed that if one first encodes the ID using an error-correcting code with $\mathrm{poly}(k)$ alphabet-size and relative distance larger than $1 - 1/k$ then the [48] protocol actually handles k concurrent sessions. In the full protocol we actually use a different trick, based on signature schemes.

Proceedings of the 2005 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)
0-7695-2468-0/05 $20.00 © 2005 IEEE

(for bounded general-concurrent zero knowledge).

A strange idea. This leads us to the following strange idea - why don't we try to use Protocol bgcZK, but set the security parameter to *super-polynomial* size? Unfortunately there is a good reason cryptographers do not set the security parameter to super-polynomial values: because this yields a protocol with super-polynomial communication and computation even for the honest parties. Can we overcome this difficulty? We do have a way to compress at least the communication, using hash functions combined with universal arguments. That is, we define $\Pi_{\mathsf{condensed}}$ to be the protocol that is the result of executing bgcZK with security parameter $k^{\log k}$ (where k is our "true" security parameter), but replacing each message m in $\mathsf{bgcZK}(k^{\log k})$ which is of super-polynomial size with $h(m)$ followed by a universal argument proving knowledge of m. Now, it is not at all clear that this protocol makes sense, because if a party needs to change its action in $\mathsf{bgcZK}(k^{\log k})$ according to the contents of a super-polynomially sized message m, then during an interaction in $\Pi_{\mathsf{condensed}}$, this party won't be able to recover m regardless of its computation powers (indeed, the polynomial-sized transcript simply does not contain enough information about m).

Thus we are left with two problems: **(1)** $\Pi_{\mathsf{condensed}}$ is not a valid protocol since the parties needs to run in super-polynomial time, if they can work at all *and* **(2)** Even though $k^{\log k}$ concurrent sessions of $\mathsf{bgcZK}(k^{\log k})$ can be simulated, that does not mean that the same holds for $\Pi_{\mathsf{condensed}}$, since now the simulator needs to *rewind* to extract the long messages sent and rewinding in a concurrent setting is notoriously problematic. Both problems are rather serious but can be resolved by moving to a third protocol that tries to combine the good properties of Π_{BF} and $\Pi_{\mathsf{condensed}}$.

3.4. The Combined Protocol: Two Protocols with Two Simulators.

We now present our third protocol, which will actually be (almost) a concurrently simulation-sound zero-knowledge protocol.[12] The idea is the following: we will run both Π_{BF} and $\Pi_{\mathsf{condensed}}$, but we'll run $\Pi_{\mathsf{condensed}}$ in an "encrypted" form, that is replacing every message m of bgcZK by $\mathsf{KCom}(m)$ if m is of polynomial size and $\mathsf{UAKCom}(m)$ if m is of super-polynomial size. At the end, we will prove in a witness indistinguishable way that one of these protocols succeeded. That is, our combined protocol, which we denote by Π_{combined} will operate as follows, when proving $x \in L$ with w a witness for x:

1. Prover sends to verifier $\mathsf{Com}_{\mathsf{weak}}(w)$.
2. Prover and verifier engage in a brute-force breaking opportunity BFOP.
3. Prover and verifier engage in "encrypted and condensed" version of $\mathsf{bgcZK}(k^{\log k})$: any message m is replaced with $\mathsf{KCom}(m)$ if m is polynomial size and $\mathsf{UAKCom}(m)$ if m is of super-polynomial size.
4. Prover and verifier engage in a witness indistinguishable universal argument that *either*: **(a)** the commitment in Step 1 is indeed a witness *or* **(b)** prover broke BFOP *or* **(c)** there exists a transcript for $\mathsf{bgcZK}(k^{\log k})$ that the honest verifier of that protocol accepts, and this transcript is consistent with the "encrypted condensed" transcript of Step 2.

What is this good for? First of all note that, unlike $\Pi_{\mathsf{condensed}}$, in Π_{combined} both parties can be implemented using only polynomial time computation, and so at least we got rid of one of our problems. Like Π_{BF}, Protocol Π_{combined} has a simple straight-line black-box simulator. However, our intention is that unlike in the case Π_{BF} this simulator will enjoy the simulation soundness property and furthermore that we will be able to prove that this is the case. Our idea is to prove simulation soundness using what we call a *virtual simulator*. The virtual simulator will have two properties: **(1)** it will satisfy the simulation-soundness property and **(2)** it will be *strongly indistinguishable* from the output of the straight-line simulator, in the sense that it will be indistinguishable *even for algorithms with enough running time to break* $\mathsf{Com}_{\mathsf{weak}}$. These two properties together will imply that our straight-line simulator must also satisfy the simulation soundness requirement.

Why do we need the straight-line simulator? If the virtual simulator already satisfies the simulation soundness condition, why do we need to use the straight-line simulator at all? The reason is that the virtual simulator will actually use the *witness* as part of its input. This is OK since the virtual simulator is not the "real" simulator and is only used as part of the security proof. Note that it is not at all clear that using the witness helps the virtual simulator as we can't commit to the witness in Step 1 without destroying the strong indistinguishability property.

The operation of the virtual simulator. The virtual simulator will try to run the simulator of the protocol $\mathsf{bgcZK}(k^{\log k})$, which does enjoy the simulation-soundness property. The question is how do we solve our second problem above – namely, how do can we use the simulator of $\mathsf{bgcZK}(k^{\log k})$ when we are unable to rewind in a concurrent setting. The trick is that we *are* able to rewind using the *witnesses*. That is, in order to produce the auxiliary sessions we need for rewinding we actually use the witness to perform a straight-line simulation. The reason we can get away with using the witness in these auxiliary sessions is

[12] The qualifier "almost" is because there are still some subtleties that we ignore here. Some of these are discussed below, while others are only handled in the full proof presented in the full version of this work.

that the auxiliary sessions don't need to be strongly indistinguishable from the main simulation, but rather only need to be indistinguishable "enough" to ensure successful extraction. The reason we can't use the breaking opportunity is that in order to ensure the simulation soundness we need to make sure that the running time of the virtual simulator is less even than the time to break $\mathsf{Com}_{\mathsf{weak}}$.

3.5. Some issues and subtleties.

Witness-based continuation: To actually implement this idea, we need to make sure that regardless at which point we are in the simulation, we can always continue in a straight-line fashion using the witness alone, without requiring the internal state of any of the parties. Toward this end, we use a compiler, which we call a *witness-based-continuation (WBC) compiler* that transforms the protocol to a protocol that satisfies this property. Loosely speaking, we first make sure that the only prover messages that unavoidably depend on internal state are the last messages sent in some proof system used as a sub-protocol. We then change the prover to have these messages not sent in the clear but rather in a weak commitment, along with a weak commitment to a string w' and a WI proof that either the committed message causes the verifier to accept or w' is a witness.[13]

"Forcing" scheduling constraints on adversary: Another point is that when we transformed bgcZK into its condensed version, we converted each message into an interactive universal argument, thus ruining the "atomicity" of individual messages. The security proof of bgcZK actually relies on this atomicity and hence we need to do something to restore it. Our solution is to use brute force breaking opportunities as "buffers" between individual messages. It turns out that if during a session in which the adversary is a verifier, it schedules the universal arguments for two messages during the same time as it schedules the universal argument for a single message in the session where it is a prover, then in this case it is actually "safe" for the virtual simulator to break the BFOP (even though this requires more running time than the virtual simulator is officially "allowed"). Thus, we can use the straight-line simulator in the cases where the adversary's scheduling violates the atomicity condition.

3.6. Guide to the actual protocol and proof.

Our general-concurrent zero knowledge argument scheme is Protocol 3.1. This protocol follows broadly the approach sketched above, but its analysis and design are more "low level". That is, instead of combining "generic" components such as Π_{BF} and bgcZK and proving something about the composition of any two such components, we use the ideas behind Π_{BF} and bgcZK to construct our protocol which we then analyze. The reason is that there are some subtleties, especially involving the ability of the adversary to dynamically schedule messages and choose the statements to be proven, that make the low level approach preferable. Some points in which we deviate from the description above include using more complexity levels than just two, and using verification keys of digital signatures to avoid issues with dynamically chosen statements.

Under our assumptions we have for every constant i a super-polynomial function $T_i(k)$ such that, by using an appropriately scaled security parameter, we can get a variant of each of the primitives above that is secure against $T_i(k)$-sized adversaries but completely broken by $T_{i+0.1}(k)$-sized adversaries.[14] We use a subscript i to denote such primitives. A full description of this protocol and its analysis can be found in the full version of this work.

4. Conclusions and future directions.

We presented a general feasibility result for secure multiparty computation in the general-concurrent setting, under well-studied assumptions. In some sense, this work brings provable security closer to practice, since the security properties, which are proven under standard assumptions, are strong enough to model what happens in realistic networks. However, in terms of efficiency our constructions leaves much room for improvement. Even though polynomial simulation is impossible, there is also room for improvement on our protocol in terms of the simulation overhead. We hope that the ideas presented here will prove useful in obtaining more practical protocols, which still can be proven secure in the general concurrent setting under well-understood assumptions. An example for such a problem is obtaining a practical fully concurrent and non-malleable commitment scheme under such well-known number-theoretic assumptions such as the hardness of factoring or the discrete logarithm problem.

On a technical level, we introduced a new technique for "condensing" protocols to achieve stronger security. We believe this technique may have many other applications. In particular, we believe there is hope for using such techniques to obtain a concurrent zero-knowledge protocol using a constant number of communication rounds, with polynomial simulation overhead. Such a protocol is known if we allow super-polynomial simulation, but it would be nice to obtain it using polynomial simulation, since, unlike the case of general computation, super-polynomial simulation does not seem necessary in this case.

[13] We use a variant of this compiler with trapdoor commitments a la [22, 17] to obtain security with respect to *adaptive* adversaries.

[14] For example, if we have hardness against 2^{k^ϵ}-sized circuits, we can use $T_i(k) = 2^{\log(T_0(k))^{(1/\epsilon)^{30i}}}$.

Proceedings of the 2005 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)
0-7695-2468-0/05 $20.00 © 2005 IEEE

Public input: 1^k: security parameter , $x \in \{0,1\}^{\ell_{\mathsf{stmt}}}$ (statement to be proved is "$x \in L$") **Prover's auxiliary input:** $w \in \{0,1\}^{\ell_{\mathsf{wit}}}$ (a witness that $x \in L$)	w $\quad$ $1^k, x$ $\downarrow$ $\quad$ $\downarrow$ $\boxed{P}$ $\quad$ $\boxed{V}$
Step V1.1 (Verifier's hash): Verifier chooses a random hash function $h \leftarrow_{\mathrm{R}} \mathsf{Hash}_6$ and sends h.	$\xleftarrow{h \leftarrow_{\mathrm{R}} \mathsf{Hash}_6}$
Steps P,V1.2 (Prover's "verification key"): Prover chooses $VK = 0^{\ell_{\mathsf{VK}}}$ and sends $c_{\mathsf{VK}} = \mathsf{Com}_1(VK)$ to the verifier.	$\xrightarrow{c_{\mathsf{VK}} = \mathsf{Com}_1(VK\ [=0^{\ell_{\mathsf{VK}}}]\ )}$
Steps V,P2.x (Verifier's first challenge): Verifier chooses $r_1 = 0^k$, computes $c_{r_1} = \mathsf{Com}_5(h(r_1))$ and proves knowledge of r_1 using a ZKUA.	*Slot 1* $\xleftarrow{c_{r_1} = \mathsf{UAKCom}_5^h(r_1\ [=0^k]\ )}$
Steps P,V3.x (Breaking opportunities): Prover and verifier engage in a $T_2(k)$, and $T_3(k)$-secure brute force breaking opportunities.	*"Unsafe" period* $B_{\mathsf{easy}} = \mathsf{BFOP}_2; B_{\mathsf{hard}} = \mathsf{BFOP}_3$
Steps V,P4.x (Verifier's second challenge): Verifier chooses $r_2 = 0^k$, computes $c_{r_2} = \mathsf{Com}_5(h(r_2))$ and proves knowledge of r_2 using a ZKUA.	*Slot 2* $\xleftarrow{c_{r_2} = \mathsf{UAKCom}_5^h(r_2\ [=0^k]\ )}$
Step P,V5.x (Commitment to "Signature"): Prover lets $\sigma = 0^{\ell_{\mathsf{sig}}}$ and sends $c_{\mathsf{sig}} = \mathsf{KCom}_5(\sigma)$ to the verifier.	$\xrightarrow{c_{\mathsf{sig}} = \mathsf{Com}_5(\sigma\ [=0^{\ell_{\mathsf{sig}}}]\ )}$
Steps P,V6.x ("committed" universal argument): Prover and verifier run $T_6(k)$-sound universal argument UA for **[KOLM]** where prover sends T_5-strong commitments to its messages . Honest prover uses commitments to "junk" (i.e. 0^k) in this stage. **Statement [KOLM]:** Let $M = 2T_{0.1}(k)$. For $j \in [\ell_{\mathsf{VK}}]$ let $\ell_j = (VK_j) \cdot \ell_{\mathsf{VK}} + j$ (i.e., $\ell_j \in [2\ell_{\mathsf{VK}}]$) and let $\ell_j^1 = \ell_j \cdot M$ and $\ell_j^2 = (4\ell_{\mathsf{VK}} + 1 - \ell_j)M$. **Then,** for every $j \in [\ell_{\mathsf{VK}}]$ there exist $s \in \{1,2\}$, a TM Π_s of description size $\leq \ell_j^s - k$ and a string r_s such that: **(a)** Π_s outputs r_s within $\leq T_{1.4}(k)$ steps *and* **(b)** r_s is consistent with c_{r_s}. That is, $h(r_s) \in \mathsf{Com}^{-1}(c_{r_s})$.	$\xrightarrow{c_{\mathsf{UA}} = \mathsf{Com}_5\mathsf{UA}_6 \text{ of } \textbf{[KOLM]}}$
Step P.7.1 (Commitment to Witness): Prover sends $c_{\mathsf{wit}} = \mathsf{Com}_4(w)$ to the verifier.	$\xrightarrow{c_{\mathsf{wit}} = \mathsf{Com}_4(w)}$
Steps P,V7.2.x (WI proof): Prover proves to verifier using a $T_5(k)$-WI proof that one of the following holds: **either** **[WIT]** $\mathsf{Com}^{-1}(c_{\mathsf{wit}})$ is a witness for x **or** **[BFOP]** Broke B_{hard} **or** **[UA]** $\mathsf{Com}^{-1}(c_{\mathsf{UA}})$ is accepting transcript. **or** **[SIG]** Broke B_{easy} and c_{sig} is commit to sig on x.	$\longrightarrow$ $\boxed{\mathsf{WIP}_5 \text{ that } \textbf{[WIT] / [BFOP] / [UA] / [SIG]}}$

(The WBC compiler changes a last prover message m of a sub-proof systems (i.e., B_{easy},B_{hard} and the final WIP) to $\mathsf{Com}_4(m)$, $\mathsf{Com}_4(w'\ [=0^{\ell_{\mathsf{wit}}}]\ )$ and WI-proof that either m convinces the verifier or w' is a witness for x.)

Protocol 3.1. Non-Malleable Concurrent Zero Knowledge (before WBC-compiler)

Proceedings of the 2005 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)
0-7695-2468-0/05 $20.00 © 2005 IEEE

Acknowledgements Both authors' understanding of the issues surrounding multi-party computation was shaped in discussions with many colleagues. We are especially grateful to Ran Canetti, Oded Goldreich, Shafi Goldwasser, Yehuda Lindell, Silvio Micali, Moni Naor, Rafael Pass, Manoj Prabhakaran, Omer Reingold, Alon Rosen and Salil Vadhan.

References

[1] B. Barak. How to go beyond the black-box simulation barrier. In *Proc. 42nd FOCS*, pages 106–115, 2001.

[2] B. Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In *Proc. 43rd FOCS*. IEEE, 2002.

[3] B. Barak. *Non-Black-Box Techniques in Cryptography*. PhD thesis, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, 2004.

[4] B. Barak, R. Canetti, Y. Lindell, R. Pass, and T. Rabin. Secure computation without authentication. Crypto '05, 2005.

[5] B. Barak, R. Canetti, J. B. Nielsen, and R. Pass. Universally composable protocols with relaxed set-up assumptions. In *Proc. 45th FOCS*, pages 186–195. IEEE, 2004.

[6] B. Barak and O. Goldreich. Universal arguments and their applications. In *Annual IEEE Conference on Computational Complexity (CCC)*, volume 17, 2002.

[7] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the First Annual Conference on Computer and Communications Security*, pages 62–73. ACM, November 1993.

[8] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. 20th STOC*, pages 1–10, 1988.

[9] M. Blum. Coin flipping by phone. In *Proc. 24th IEEE Computer Conference (CompCon)*, pages 133–137, 1982. See also *SIGACT News*, Vol. 15, No. 1, 1983.

[10] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *Proc. 20th STOC*, pages 103–112, 1988.

[11] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[12] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. 42nd FOCS*, pages 136–147, 2001. Preliminary full version available as Cryptology ePrint Archive Report 2000/067.

[13] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable encryption. In *Crypto '97*, pages 90–104, 1997. LNCS No. 1294.

[14] R. Canetti and M. Fischlin. Universally composable commitments. Report 2001/055, Cryptology ePrint Archive, July 2001. Extended abstract appeared in CRYPTO 2001.

[15] R. Canetti, O. Goldreich, S. Goldwasser, and S. Micali. Resettable zero-knowledge. In *Proc. 32th STOC*, pages 235–244. ACM, 2000.

[16] R. Canetti, E. Kushilevitz, and Y. Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In *Eurocrypt '03*, 2003 LNCS No. 2656.

[17] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party computation. In *Proc. 34th STOC*, pages 494–503, 2002.

[18] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437 (electronic), 2000. Preliminary version in STOC 1991.

[19] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Crypto '92*, pages 139–147, 1992. LNCS No. 740.

[20] C. Dwork, M. Naor, and A. Sahai. Concurrent zero knowledge. In *Proc. 30th STOC*, pages 409–418. ACM, 1998.

[21] U. Feige. *Alternative Models for Zero Knowledge Interactive Proofs*. PhD thesis, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, 1990.

[22] U. Feige and A. Shamir. Zero knowledge proofs of knowledge in two rounds. In *Crypto '89*, pages 526–545, 1989. LNCS No. 435.

[23] U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *Proc. 22nd STOC*, pages 416–426. ACM, 1990.

[24] O. Goldreich. Concurrent zero-knowledge with timing, revisited. In *Proc. 34th STOC*, pages 332–340, 2002.

[25] O. Goldreich. *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004.

[26] O. Goldreich and A. Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–189, Summer 1996.

[27] O. Goldreich and H. Krawczyk. On the composition of zero-knowledge proof systems. *SIAM J. Comput.*, 25(1):169–192, Feb. 1996. Preliminary version appeared in ICALP' 90.

[28] O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In ACM, editor, *Proc. 19th STOC*, pages 218–229. ACM, 1987. See [25, Chap. 7] for more details.

[29] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, July 1991. Preliminary version in FOCS' 86.

[30] S. Goldwasser and L. Levin. Fair computation of general functions in presence of immoral majority. In *Crypto '90*, pages 77–93, 1990. LNCS No. 537.

[31] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, Apr. 1984. Preliminary version appeared in STOC' 82.

[32] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989. Preliminary version in STOC' 85.

[33] Y. T. Kalai, Y. Lindell, and M. Prabhakaran. Concurrent general composition of secure protocols in the timing model. In *Proc. 37th STOC*, pages 644–653, 2005.

[34] J. Kelsey, B. Schneier, and D. Wagner. Protocol interactions and the chosen protocol attack. In *Proc. 1997 Security Protocols Workshop*, pages 91–104, 1997. Appeared in LNCS vol. 1361.

[35] J. Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proc. 24th STOC*, pages 723–732. ACM, 1992.

[36] Y. Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *Proc. 35th STOC*, pages 683–692. ACM, 2003.

[37] Y. Lindell. *Composition of Secure Multi-Party Protocols: a comprehensive study*, volume 2815 of *Lecture Notes in Computer Science*. Springer-Verlag Inc., New York, NY, USA, 2003.

[38] Y. Lindell. General composition and universal composability in secure multi-party computation. In *Proc. 44th FOCS*, pages 394–403, 2003.

[39] Y. Lindell. Lower bounds for concurrent self composition. In *Theory of Cryptography Conference (TCC)*, volume 1, pages 203–222, 2004.

[40] T. Malkin, R. Moriarty, and N. Yakovenko. Generalized environmental security from number theoretic assumptions, 2005. In preparation.

[41] S. Micali. CS proofs. In *Proc. 35th FOCS*, pages 436–453. IEEE, 1994.

[42] M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991. Preliminary version in CRYPTO' 89.

[43] M. Naor. Deniable ring authentication. In *Crypto '02*, 2002. LNCS No. 2442.

[44] R. Pass. On deniability in the common reference string and random oracle model. In *Crypto '03*, 2003.

[45] R. Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In *Eurocrypt '03*, 2003. LNCS No. 2656.

[46] R. Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *Proc. 36th STOC*, pages 232–241, 2004.

[47] R. Pass and A. Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. In *Proc. 44th FOCS*, 2003.

[48] R. Pass and A. Rosen. New and improved constructions of non-malleable cryptographic protocols. In *Proc. 37th STOC*, 2005.

[49] M. Prabhakaran and A. Sahai. New notions of security: achieving universal composability without trusted setup. In *Proc. 36th STOC*, pages 242–251, 2004.

[50] M. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.

[51] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *Proc. 40th FOCS*, pages 543–553. IEEE, 1999.

[52] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11), Nov. 1979.

[53] A. Shamir, R. L. Rivest, and L. M. Adleman. Mental poker. In D. Klarner, editor, *The Mathematical Gardner*, pages 37–43. Wadsworth, Belmont, California, 1981. Preliminary version as MIT TM-125, 1978.

[54] A. C. Yao. How to generate and exchange secrets. In *Proc. 27th FOCS*, pages 162–167. IEEE, 1986.