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

August 26, 2005

Abstract

We construct a secure protocol for any multi-party functionality that remains secure (under a relaxed definition of security) when executed concurrently with multiple copies of itself and other protocols. We stress that we do *not* use any assumptions on existence of trusted parties, common reference string, honest majority or synchronicity of the network. The relaxation of security, introduced by Prabhakaran and Sahai (STOC '04), 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. That is, existence of a hash function collection that is collision resistent with respect to circuits of subexponential size, and existence of trapdoor permutations that 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).

Keywords: Non-malleable protocols, concurrent composition, multi-party secure computation

*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.

Contents

List of Figures

List of Tables

1 Introduction

In the 1980's a sequence of groundbreaking papers [SRA78, Sha79, Rab81, Blu82, GM82, GMR85, GMW86, Yao86] led to the rather amazing result of Goldreich, Micali and Wigderson [GMW87] (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, the GMW 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 its computation and communication overhead, 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* 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 there are known successful *attacks* in the concurrent setting for instantiations of GMW and other stand-alone protocols with particular choices for the underlying components [GK90, Fei90].

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 remain secure in this setting under standard cryptographic assumptions. Even more disturbingly, [Can01, CF01, CKL03, Lin03c, Lin04] 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. Most of these involved making some assumptions on trusted setup or limits of the network's synchronicity. Recently, Prabhakaran and Sahai [PS04] 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 [Pas03b]). As discussed below, this relaxation still provides meaningful and strong security for the canonical application of multi-party computation.[1] However, the result of [PS04] was under a highly non-standard com-

[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 as 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.

putational 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 [PS04] 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 [CLOS02], 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 zero-knowledge protocol utilizes only a *constant*[4] 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 [PS04] 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

[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]Note however that the multi-party computation protocol of [CLOS02] uses a super-constant number of communication rounds when dealing with adaptive adversaries. So our final protocol for computing any functionality requires a super-constant number of rounds, as well. Constructing a constant-round protocol for multi-party computation, even without concurrent security, that is secure against adaptive adversaries without memory erasure, remains an interesting open problem.

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.[5] 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 Chess or proof-of-work schemes [DN92] 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* [DDN91, DNS98, CDNO97, Nao02] and hence our protocol fails to achieve such applications.[6]

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 [KSW97] and *general composition* by [Lin03c]. 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 [CLOS02] in the common reference string model (where the other protocols are required not to use the reference string), and [KLP05] (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 [BOGW88].

New technique – "condensed protocols"[7]. 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 [Pas04], 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 [BG02] 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

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

[6] 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 [CLOS02] also fail to achieve deniability, even when using setup assumptions (e.g., see [Pas03a]).

[7] 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.

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.

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 relevant to our results; we discuss the works relevant to the techniques of this paper in Section 3. See the books by Goldreich [Gol04, Chapter 7] and Lindell [Lin03b], and the references therein for a more comprehensive review of the literature.

Secure multi-party computations. Protocols for secure function evaluation in the stand-alone setting were given by [Yao86, GMW87]. The latter paper also introduced the paradigm of "forcing" honest but curious behavior using zero-knowledge proofs [GMR85, GMW87], which has been widely used in many subsequent papers in this area (including the current one). A satisfactory definition of security for such protocols (in the stand-alone setting) was given by [Can00], following [GL90, MR91, Bea91]. A constant round protocol was given in [KOS03], and a simpler and improved such protocol was given in [Pas04].

Concurrent setting. Security in the concurrent setting was first considered in the context of zero-knowledge protocols by [DNS98]. A construction was given in [RK99], and improvements in the number of rounds were made in [KP01, PRS02]. Some negative results were given in [KPR98, Ros00, CKPR01]. The more general setting where the adversary can play *different* roles in each execution (i.e., the person-in-the-middle attack) was first studied by [DDN91], who gave protocols for commitment and zero knowledge that withstand such an attack in *two* concurrent executions. Constant round protocols were given [Bar02], and simpler and improved such protocols were given by [PR05]. Composition with arbitrary other protocols was considered by [PW00, PSW00]. Security in the most general setting of an arbitrary polynomial number of concurrent executions, in which parties can play different roles and interact in different protocols, was considered by [Can01] who termed such protocols "universally composable" (UC). However, without some setup assumptions, very broad impossibility results were shown to hold for the definition of [Can01] and even significantly relaxed definitions, as long as they require *polynomial-time* simulation [Can01, CF01, CKL03, Lin03c, Lin04].

Security in relaxed models. Because of the failure to obtain secure protocols in a model where there are no trusted parties, and parties interact in a fully asynchronous way, there were several works considering more relaxed models. **The CRS model:** One such model is the *common* *reference string* (CRS) model, originally introduced in the context of non-interactive zero-knowledge [BFM88], where the only assumption is that there is a publicly known string that was chosen once and for all by some trusted party. In this model [CLOS02] gave a construction of multi-party computation protocol satisfying the UC definition [Can01] which implies that it remains secure under general concurrent composition. The main problem with the CRS model is that it places an enormous amount of trust on the party choosing the common string. Indeed, by cheating in choosing this string, this party can completely and undetectably break the security of the [CLOS02] protocol and of essentially all other protocols in this model. An approach to distributing some of this trust was recently taken by [BCNP04]. **Honest majority:** Another assumption which was used to construct such protocols is the existence of a majority of honest parties [BOGW88, RBO89, Can01]. 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:** Yet another assumption that was used is the *timing model* [DNS98], 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. [DNS98] gave a concurrent zero-knowledge proof system in this model. Recently, [KLP05] gave a multi-party computation protocol for this setting that remains secure under general concurrent composition. The main problem with the protocol of [KLP05] (and all other protocols in the timing model such as [DNS98, Gol02]) 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:** Yet another assumption, introduced in the context of zero knowledge in [Bar01] and extended to the general case in [Lin03a], is that there is a fixed known polynomial upper-bound M on length of *all* the communication throughout the entire network. [Lin03a], later improved by [PR03, Pas04, PR05], gave constructions for multi-party computation protocols that remain secure under general composition using this assumption. However, these protocols use computation and communication that is *larger* than M, and this was shown to be necessary by [Lin03a]. Hence, while bounded-concurrent protocols can be sometimes very useful tools in other constructions (and indeed we use techniques from [Pas04, PR05] 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:** One natural relaxation (which is the one 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., [CGGM00]) but was first explicitly put forward in [Pas03b], 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. As argued above and in [Pas03b], super-polynomial simulation provides sufficient security for almost all applications of multi-party computation. While allowing super-polynomial simulation makes constructing concurrent zero-knowledge protocols much easier it did not seem to be so helpful in constructing *non-malleable* protocols. Also, super-polynomial simulation seemed to ruin the most attractive feature of the UC framework of [Can01], namely the UC composition theorem. Thus, it might have seemed initially that it will be possible to generalize the extensive impossibility results of [Can01, CF01, CKL03, Lin03c, Lin04] to rule out even super-polynomial simulation. **The PS paper:** The first *positive* result in this direction was given by Prabhakaran and Sahai [PS04]. They gave a construction of a fully concurrent and non-malleable multi-party computation protocol in the general-network setting, but they required new (quite unstudied and 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, [PS04] offered the exciting possibility of obtaining secure protocols without giving up either. However, in our opinion the weak point of [PS04] is 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 [BR93], the assumptions of [PS04] are well-defined complexity-theoretic assumptions, they are not well-studied, and seem to be difficult to analyze because of their complexity. On a more technical level, although [PS04] 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 [PS04] by obtaining it under standard assumptions[8]. **Other relaxed security notions:** There are other security definitions for particular cryptographic tasks which are outside of the ideal-model simulation paradigm. However, to the best of our knowledge, under standard assumptions, all such definitions are weaker than the ideal-model simulation, and (assuming one uses a conservative enough security parameter), this holds even if the simulator runs in quasi-polynomial time.

1.2 Overview of this paper.

In Section 2 we discuss the definitions and model we use, state our results, and elaborate on why these results provide a meaningful notion of security. In Section 3 we give an overview of our techniques. The main component we construct — a fully concurrent and non-malleable zero-knowledge protocol — is Protocol 4.1 (outlined in Page 17). A detailed description of the protocol is given in Section 4, with the simulation soundness property proven in Section 5. The construction of a general multi-party protocol from the zero-knowledge protocol (using the results of [CLOS02]) is described in Section 6.

2 Model and Results

The network model we consider is the same one as in [Can00, Can01, Lin03c] . 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

[8]Prabhakaran and Sahai [PS04], 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 [PS04]. However, for the sake of being as self-contained as possible, we instead prove our result directly in the context of the definitions of [Can01]. We also note that recently [MMY05] gave a different construction in the [PS04] 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).

P_i uses as π_i. This strategy models all the activity of P_i, including all protocols[9], 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 [CLOS02] and with [GMW87] in the non honest-majority case) does not guarantee security against denial of service attacks or provide the related guarantee of fairness [GL90].

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.* [CLOS02] 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.*

[9]Another equivalent way to model this, following [Can01], 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.

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 our actual construction and proof more complicated. Because of these subtleties, our actual construction (Protocol 4.1) 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 [Blu82, Nao89].

Zero-Knowledge proofs of knowledge. A constant-round zero-knowledge proof/argument of knowledge for **NP** [FS89, GK96]. We will also sometimes use the weaker notion of a *witness indistinguishable* proof, which we denote by WIP [FS90]. We note that witness-indistinguishability, unlike zero-knowledge, is closed under concurrent composition. Indeed, under some strong but reasonable assumptions it is even possible to have two-message or even one-message WI proofs, which are trivially closed under concurrent composition [DN00, BOV03].

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.[10]

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 [Kil92], with improved analysis in [Mic94] and [BG02] (with the latter work showing they are a proof of knowledge). We'll also use constructions of universal arguments that are *zero knowledge* and *witness indistinguishable* [Kil92, BG02, Bar04]. 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 **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 $x \in L \cup L'$

[10] 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.

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 [CGGM00].

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 [CGGM00, Pas03b].

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 [Pas03b] in a similar context), which we denote by Π_{BF}: let L be an **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} [Pas03b]: 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.[11]

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* [Sah99], 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).

[11] Thus, as shown in [Pas03b], the protocol Π_{BF} is a concurrent zero knowledge (cZK) protocol with quasi-polynomial simulation. However note that we need stronger security than cZK, because in our case the adversary can play both the roles of prover and verifier during the attack.

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 [DDN91, Bar02, Pas04, PR05] 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 [Pas04] 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 [BCL+05] that using the ideas of Pass and Rosen [PR05], it is possible to convert a different protocol of Pass [Pas04] to a constant-round protocol with this property.[12] We denote this protocol (which is essentially based on [Pas04]) by bgcZK (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 $\Pi_{\mathsf{condensed}}$.

[12] Pass [Pas04] 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 [PR05] 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.*[BCL+05] 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 [PR05] protocol actually handles k concurrent sessions. We note that in our actual protocol we use a different trick, based on signature schemes, to achieve a similar goal.

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.[13] 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 of 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

[13] 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 later sections.

$\mathsf{bgcZK}(k^{\log k})$ 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 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.[14]

"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 4.1(Page 17). 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.

[14]We use a variant of this compiler with trapdoor commitments a la [FS89, CLOS02] to obtain security with respect to *adaptive* adversaries.

4 Construction of a Concurrent and Non-Malleable Zero-Knowledge Protocol

4.1 Preliminaries

Hardness assumptions. We will make use of a number of "complexity levels" in our protocol and its analysis. As the analysis is quite delicate, and for sake of understandability, we do *not* attempt to optimize the number of complexity levels (described below), but rather we choose a very conservative setting of parameters in order to simplify the presentation to the best extent possible. We assume we have primitives that with security parameter k' are secure against $2^{k'^{\epsilon}}$ sized circuits, (in the sense that no $2^{k'^{\epsilon}}$-size circuit can break them with success better than $2^{-k'^{\epsilon}}$) but can be completely broken in time $2^{k'}$. We assume that our adversary's running time is $T_0(k)$ (e.g., $T_0(k) = k^{\log k}$ or $T_0(k) = 2^{k^{\delta}}$). Define $T_i(k) = 2^{\log(T_0(k))^{(1/\epsilon)^{30i}}}$. By appropriate scaling, we can obtain for every constant i, a primitive that is secure against $T_i(k)$-sized adversaries but is completely broken in time much less than $T_{i+0.1}(k)$. We call such a primitive $T_i(k)$-*secure* and we sometimes use a subscript i to denote it (e.g., Com_i). We say that a probabilistic event is $T_i(k)$-*observable* if there is a $T_i(k)$-time computable predicate that decides whether or not the event holds. Note that we'll sometimes drop k when it can be inferred from the context. We say that $f(k) \ll g(k)$ if $2^{\log^{(1/\epsilon)} f(k)} = g(k)^{o(1)}$.

Throughout this paper *negligible* will mean probability that is less than $1/T_0(k)^c$ for any fixed $c > 0$. We say that two random variables X and Y are (s, ϵ)-computational indistinguishable if no $s^{O(1)}$-sized circuit can distinguish between X and Y with $\epsilon^{\Omega(1)}$ advantage. We say that they are s indistinguishable if they are $(s, 1/s)$-indistinguishable.

For sake of visual simplicity, we will often drop the dependence on the global security parameter k, and simply write T_i for $T_i(k)$.

Complexity levels. Throughout the protocol and analysis, we make an extensive use of various complexity levels. For convenience we list all these levels in Table 1 (Page 16) up to polynomial factors (e.g., identifying T_0 and $(T_0)^5$). We again stress that we have not tried to minimize the number of complexity levels used.

Size and time of adversary, number of sessions.	T_0
Value of M in definition of statement [**KOLM**].	$T_{0.1}$
Distinguishing advantage between VSim and Sim, between Sim and real execution.	$\frac{1}{T_{0.5}}$
Probability of simulation soundness failure in VSim, Sim.	$\frac{1}{T_{0.6}}$
Strength of commitment to verification key VK (c_{VK}).	T_1
Running time of VSim verifier (VHV).	$T_{1.1}$
Running time of VSim prover (VHP) in Case 3: $\mathtt{win} = $ '`UA`'.	$T_{1.5}$
Security of B_{easy}.	T_2
Running time of VSim prover (VHP) in Case 2: $\mathtt{win} = $ '`SIG`'.	$T_{2.5}$
Security of B_{hard}.	T_3
Running time of Sim prover algorithm. Running time of VSim prover (VHP) in Case 1: $\mathtt{win} = $ '`BFOP`'.	$T_{3.5}$
Security of commitment to witness c_{wit}, commitments to witness and response in WBC protocol.	T_4
Running time of Sim verifier (time to extract witness).	$T_{4.1}$
Security of all other commitments in protocol. Indistinguishability of WI and ZK protocol used.	T_5
Soundness and knowledge soundness of all proof system used. Strength of hash function and signature scheme.	T_6

Table 1: Complexity levels and quantities used in the protocol and proof (up to polynomial factors).

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 $\downarrow$ $\boxed{P}$ $\quad 1^k, x$ $\downarrow$ $\quad \boxed{V}$
Step V1.1 (Verifier's hash): Verifier chooses a random hash function $h \leftarrow_{\mathrm{R}} \mathsf{Hash}_6$ and sends h. **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.	$\xleftarrow{h \leftarrow_{\mathrm{R}} \mathsf{Hash}_6}$ $\xrightarrow{c_{\mathsf{VK}} = \mathsf{Com}_1(VK\ [= 0^{\ell_{\mathsf{VK}}}]\ )}$
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, denoted B_{easy} and B_{hard}.	*"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. **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.	$\xrightarrow{c_{\mathsf{wit}} = \mathsf{Com}_4(w)}$ $\longrightarrow$ WIP_5 that **[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 4.1. Non-Malleable Concurrent Zero Knowledge (before WBC-compiler)

4.2 The protocol.

Our general-concurrent zero knowledge argument scheme is Protocol 4.1 (Page 17). This protocol follows broadly the approach sketched in Section 3, 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 (and in particular the "two slot technique" of [Pas04]) 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.

Let L be an **NP** language, where the statement is of length ℓ_{stmt} and the witness is of length ℓ_{wit} (both polynomially related to k). We now describe a concurrent non-malleable $T_{4.1}(k)^{O(1)}$-time simulateable zero-knowledge protocol for L. A concurrent execution of the protocol involves $\mathrm{poly}(k) \leq T_0(k)$ concurrent session in which the adversary plays the verifier, and one session (also concurrent with the others) in which the adversary plays the prover. We will present a $T_{4.1}(k)^{O(1)}$-time simulator that outputs a string indistinguishable to the transcript of all these executions, along with a witness to the statement proven by the adversary (unless this statement is a copy of a statement proven in one of the other sessions).

4.2.1 Components.

We use the following components in our protocol:

- A T_6-secure signature scheme. We denote by ℓ_{VK} the length of the verification key and by ℓ_{sig} the length of a signature on messages of length ℓ_{stmt}.
- A T_6-secure hash function ensemble Hash_6 of functions mapping $\{0,1\}^*$ to $\{0,1\}^{\ell_{\mathsf{h}}}$.
- Non-interactive computationally-hiding, perfect binding commitment schemes at various strengths. Com_i indicates a commitment secure against $T_i(k)$-time adversaries and breakable in time $\ll T_{i+0.1}$.
- A T_6-sound constant-round public coin *universal argument*, which we denote by UA [BG02]. By combining this with a standard constant-round zero-knowledge argument of knowledge for **NP** (e.g., [FS90] or [Bar01][15]), we also have a $T_6(k)$-sound, zero-knowledge universal argument (ZKUA) where the zero-knowledge property holds with polynomial simulation overhead (for adversaries of size $\leq T_6$) and the output of the simulator is indistinguishable for $T_5(k)$-sized adversaries. See [Bar01, BG02, Bar04] for more details.
- We denote by KCom_i (short for "knowledge commitment") the interactive commitment scheme where the sender commits to a value x using Com_i and then proves knowledge of x using a constant-round zero-knowledge argument of knowledge, which is sound against $T_6(k)$-sized adversaries and indistinguishable for $T_i(k)$-sized adversaries. (Again, we'll use the protocol of [Bar01] as the zero-knowledge argument in this scheme.)

[15] We'll actually use the latter protocol, since it will be convenient for us to assume that the proof systems we use as sub-protocols are public-coin systems.

- We denote by UAKCom_i^h for the protocol where the sender uses the hash function h to first hash x and then commits to $h(x)$ using Com_i and proves knowledge of x using a zero-knowledge universal argument which is sound against $T_6(k)$-sized adversaries and indistinguishable for $T_i(k)$-sized adversaries.

- $T_i(k)$-secure one way functions for various levels i. We use the notation $OWF_i : \{0,1\}^{\ell_{\mathsf{OWF}_i}} \to \{0,1\}^{\ell_{\mathsf{OWF}_i}}$.

- We call the following sub-protocol a *brute force breaking oppurtunity of level i*: the verifier chooses $r \leftarrow_{\mathrm{R}} \{0,1\}^{\ell_{\mathsf{OWF}_i}}$ and sends $y = \mathsf{OWF}_i(r)$ to the prover. The prover responds with $\mathsf{Com}_5(r')$ (where the honest prover chooses $r' = 0^{\ell_{\mathsf{OWF}_i}}$) and a proof of knowledge of r' using a constant-round zero-knowledge argument of knowledge for **NP** (with $T_6(k)$ soundness and $T_5(k)$ indistinguishability). We denote this protocol by BFOP_i and we say the prover *broke* this instance of the protocol if $r' \in \mathsf{OWF}^{-1}(y)$. We assume that OWF is a permutation for simplicity (as otherwise the verifier may send an element y that is not in the range of OWF). This assumption is not necessary, as we can also have the verifier prove in ZK that its challenge y is in the range, or replace the one-way function with a commitment scheme. However, we avoid this complication in describing the protocol and its simulation.

Note: We assume that all the proof systems we use as components (WI,ZK) etc.. have the following properties:

- The verifier is *stateless*. By this we mean that each message of the verifier can be computed using fresh randomness and the previous public transcript. (In particular, this holds for public coins (a.k.a. Arthur-Merlin) protocols.)

- The prover's messages are composed of a sequence of unopened commitments and then at the end a message m. The verifier then decides whether to accept by applying a publicly known polynomial-time predicate on the entire transcript.

It is not hard to verify that such components exist under our assumptions.

Note also that for the proofs of knowledge we will use the property that given a prover algorithm P_* of size T that causes the verifier to accept the statement x with probability at least μ, the knowledge extractor can using $\mathrm{poly}(T/\mu)$ steps to output a witness for x with probability at least $1-\mu$.[16]

4.2.2 Operation of the protocol.

Our non-malleable zero-knowledge protocol is Protocol 4.1 (Page 17).[17] It consists of the following stages:

Initial phase (Steps V1.1 , P1.2): Verifer chooses a hash function h in Hash_6. Prover commits using Com_1 to a string VK. (Honest prover lets $VK = 0^{\ell_{\mathsf{VK}}}$.)

[16] This is actually a weaker property than standard proof of knowledge, which requires the extractor to run in time $\frac{1}{\mu}\mathrm{poly}(T)$. However, in our context of super-polynomial simulation, this weaker property will be sufficient.

[17] Actually, as noted below, Protocol 4.1 is not a complete description of the protocol as it ignores a "compiler" that we apply to it to get the final protocol. However, this compiler can and should be ignored in the first reading.

First "slot" (Steps V,P2.x): Verifier chooses a string r_1 and commits to prover to r_1 using UAKCom_5^h. (Honest verifier uses $r_1 = 0^k$.)

"Unsafe period" - brute force challenge (Steps P,V3.x): Prover and verifier engage in two breaking opportunities (in parallel, although this doesn't matter) one that is T_2-secure which we call B_{easy} and the other that is T_3-secure which we call B_{hard}.

Second "slot" (Steps V,P4.x): Verifier chooses a string r_2 and commits to prover to r_2 using UAKCom_5^h. (Honest verifier uses $r_2 = 0^k$.)

Commitment to "signature" (Step P5): Prover sends to verifier $\mathsf{Com}_5(\sigma)$ where $\sigma = 0^{\ell_{\mathsf{sig}}}$.

"Committed" universal argument (Steps P,V6.x): Prover and verifier run a T_6-sound universal argument for the statement **[KOLM]** (see below), but the prover does not send its messages in the clear but rather using T_5-secure commitments. Note that the universal argument is a public-coins/Arthur-Merlin protocol and hence the verifier does not need to view the prover's messages to compute its own. However, of course, the verifier cannot verify the correctness of the universal argument.

Commitment to witness (Step P7.1): Prover sends a T_4-secure commitment to the witness $c_{\mathsf{wit}} = \mathsf{Com}_4(w)$.

WI Proof (Steps P,V7.2.x): Prover proves to verifier using a statistically-sound (with soundness error 2^{-k} which we assume is $\ll 1/T_6$)[18], T_5-WI proof that one of the following conditions hold **[WIT]** or **[BFOP]** or **[UA]** or **[SIG]** (see below).

The statements proven. The statements used in the above proof systems are the following:

[KOLM] Let $M = 2T_{0.1}$. For $j \in [\ell_{\mathsf{VK}}]$ let $\ell_j = (VK_j) \cdot \ell_{\mathsf{VK}} + j$. In other words, ℓ_j maps $\langle j, VK_j \rangle$ to $[2\ell_{\mathsf{VK}}]$ in a one-to-one manner. Let $\ell_j^1 = \ell_j \cdot M$ and $\ell_j^2 = (4\ell_{\mathsf{VK}} + 1 - \ell_j)M$. **Then,** for every $j \in [\ell_{\mathsf{VK}}]$, for either $s = 1$ or $s = 2$, there exists 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, $h(r_s) \in \mathsf{Com}^{-1}(c_{r_s})$.

[WIT] The commitment c_{wit} contains a witness to the statement x.

[BFOP] B_{hard} is broken: Let y be the first message sent by the verifier in the B_{hard} protocol of Steps P,V3.x. Let c be the second message sent by the prover in this protocol. Then c contains a commitment to r' such that $y = \mathsf{OWF}(r')$.

[UA] The committed universal argument transcript is is an accepting transcript for the statement **[KOLM]**.

[SIG] The commitment c_{sig} is a valid signature on the statement x with respect to the public key VK that is committed to in c_{VK}, *and* B_{easy} is broken: let y be the first message sent by the verifier in the B_{easy} protocol of Steps P,V3.x. Let c be the second message sent by the prover in this protocol. Then c contains a commitment to r' such that $y = \mathsf{OWF}_2(r')$.

[18]We can use also a T_6-computationally sound argument here.

4.2.3 Witness-based continuation (WBC) compiler.

We will need to apply the following transformation to the prover strategy of this protocol. It may be better to ignore this transformation in the first reading. The crucial observation for this transformation is that all messages sent by the prover during the protocol fall into one of the following categories:

Commitments - messages that contain only unopened commitments.

Verification messages for UAKCom - messages that the prover sends when it is acting as a verifier in Steps V,P2.x and V,P4.x. We assume that these can be computed in a stateless way, without any need for internal state, just by looking at the transcript.

Final messages of a $\mathsf{WI/ZK}$ proof - the last message in a $\mathsf{WI/ZK}$ proof. We note that we can ensure that all of the prover messages in the zero-knowledge or WI proof are commitments except for the last message. We also note that given the proof transcript so far, it can be decided in polynomial-time whether or not this message causes the verifier of the proof to accept. These messages occur in the following places: the last messages of B_{easy} and B_{hard} of Steps V,P3.x and the last message of the final WIP of Steps P,V7.2.x.

Our compiler does not change the prover's behavior on the first two kinds of messages. However, instead of sending a message m of the last type, it will send $\mathsf{Com}_4(m), \mathsf{Com}_4(w)$ and a WI proof of knowledge that either m causes the verifier to accept this particular sub-proof or that w is a witness for x. The WI proof will be statistically sound and witness indistinguishable for T_5-sized adversaries. We can use a standard 3-round proof for this part (e.g. parallelized [Blu87]). Even the honest prover will use a commitment to $0^{\ell_{\mathsf{wit}}}$ instead of a commitment to the witness in this compiler. The final protocol that is obtained after the WBC compiler is applied to Protocol 4.1 is called **Protocol $\mathcal{X}$**.

Inner and outer prover algorithms. It is useful to separate the prover algorithm for Protocol $\mathcal{X}$ into two components: the *inner* prover and the *outer* prover. The inner prover is the prover strategy for the uncompiled protocol (i.e., Protocol 4.1). The outer prover stands between the inner prover and the verifier for Protocol $\mathcal{X}$, and adds the WBC layer to the behavior of the inner prover. We consider two strategies for the outer prover. **The relaying strategy:** in this strategy the outer provers simply relays the messages from the inner prover to the verifier, and when given the last message m of some WI/ZK proof, it uses $\mathsf{Com}_4(m), \mathsf{Com}_4(0^{\ell_{\mathsf{wit}}})$ as its message and then runs the WI proofs proving that m causes the verifier to accept. If m does *not* cause the verifier to accept then the outer provers does not continue in the execution (however, this will never happen if the inner prover is honest). Note that to use this strategy the outer prover does not need to have any private input, such as the witness for the statement being proven. **The witness-based strategy:** We will want to maintain the property that it is possible for the outer prover to switch over from using the relaying strategy to using the witness-based strategy at any point during the proof. In the witness-based strategy, the outer prover, knowing a witness to $x \in L$, acts as follows:

- If the prover reaches a point when it must send a commitment, it commits to "junk" (i.e., all zeros) messages of appropriate length.

- If the prover reaches a point when it must send a "stateless" message because it is playing a verifier within a proof, then it acts honestly.
- If the prover reaches a point when it must begin giving a WI proof of knowledge corresponding to the last message of an inner proof, then it replaces $\mathsf{Com}_4(m)$ with a commitment to "junk" (all zeros), and commits to the witness $\mathsf{Com}_4(w)$. It then uses the witness condition to complete the WI proof of knowledge.
- If the prover has already committed according to the relaying strategy $\mathsf{Com}_4(m), \mathsf{Com}_4(0^{\ell_{\mathsf{wit}}})$, then the prover finishes the WI proof of knowledge using knowledge of m and the rest of the inner transcript so far. This is possible because in order to prove that the verifier would accept m within the inner proof, one only needs to look at the (inner) transcript – this is because the verifier is stateless.

As noted above, the honest prover for the protocol will be using the relaying strategy (although this is not crucial, and will be changed for the adaptive case).

Properties of the WBC compiler. We note that the WBC compiler has the following effects on the WI or ZK proofs it is applied to:

1. It does not ruin the WI or ZK property.
2. The compiled proof is still sound as a proof system of the combined statement (i.e., that the original statement holds *or* that the commitment contains a witness).
3. It does not ruin the proof of knowledge property (if the original proof system had such a property) in the following sense: if with probability $\geq \mu$ it holds that the combined proof succesfully ends *and* the commitment of the WBC layer does not contain witness, then we can extract a witness for original statement in time $\mathrm{poly}(T/\mu)$ (where T is the prover's running time).
4. We can continue the proof at any point with a witness without access to the internal coins of the original WI/ZK prover.

4.3 The (actual) simulator Sim.

Our simulator for Protocol $\mathcal{X}$, which we denote by Sim, is a straight-line black-box simulator[19]. This simulator does *not* get a witness as input, and hence will break B_{hard} to facilitate the simulation. That is, when simulating the prover Sim will deviate from the honest prover strategy by:

- Choosing a verification key VK for the signature scheme, and use a commitment to this key (as opposed to $0^{\ell_{\mathsf{VK}}}$) in the commitment c_{VK} of Step P1.2. (Sim will use the same verification key in all honest-prover sessions).[20]

[19]This simulator Sim will actually be the simulator we use to prove that Protocol $\mathcal{X}$ UC-realizes the ideal zero-knowledge functionality with quasi-polynomial overhead, in the UC framework. However, for ease of understanding for those not intimately familiar with the UC framework, we describe Sim here informally. This is easily converted to a formal description in the UC framework (which we do in Section 6).

[20]We note that by some slight complication to the protocol, it is possible to have the simulator follow the honest strategy also in this step.

- Using a commitment to $0^{\ell_{\mathsf{wit}}}$ instead of the real witness in Step P7.1.
- Breaking B_{hard} and use that to facilitate the WIP of Steps P,V7.2.x.

Like the honest prover, the simulator Sim will use the *relaying* outer-prover strategy for the WBC layer.

When simulating the verifier, Sim will follow the honest verifier strategy. However, when the execution of such a session is completed successfully and the statement proven was not proven in some previous honest-prover session, , the simulator Sim will use $T_{4.1}$-time to break all Com_4 commitments used by the adversary that may contain a witness (i.e., the commitment c_{wit} and all commitments of the WBC-compiler). If it finds such a witness, Sim will output this witness as an auxiliary output. Otherwise, it will output `ss-failure`. A more detailed description of Sim appears in Section 6, in the language of the UC framework.

We'll prove that Sim satisfies two properties:

Indistinguishable output The simulator Sim's output is $(T_4(k), 1/T_0(k))$-computationally indistinguishable from the transcript of a real concurrent execution of `Adv` with the honest provers and verifier.

Simulation soundness The probability that in an honest verifier's session of the simulated transcript it holds that the verifier accepts a statement x that was not proved before in an honest prover session, and none of the T_4-strong commitments contains a witness for x is less than $1/T_{0.5}$.

We'll first prove the simulation soundness property. We then prove the indistinguishable output property in Section 6 (in the proof of Theorem 6.2).[21]

5 The Virtual Simulator VSim.

We now prove that Sim satisfies simulation soundness. That is, we prove the following lemma:

Lemma 5.1 (Simulation soundness pf Sim)**.** *The probability that in an honest verifier's session of the simulated transcript it holds that the verifier accepts a statement x that was not proved before in an honest prover session, and none of the T_4-strong commitments contains a witness for x is less than $1/T_{0.5}$.*

To prove Lemma 5.1, we will construct a"virtual simulator", denoted by VSim. This will not be a "real" simulator in the sense that, unlike Sim, VSim will get as an additional input all the honest parties' private inputs (and hence in particular it will have access to all the witnesses used by these parties when proving)[22]. Nevertheless this simulator will be useful to prove simulation soundness since we'll prove that **(a)** the virtual simulator's output is computationally indistinguishable from from the output of the real simulator even by poly(T_5)-sized distinguishers, and **(b)** for the virtual simulator, the probability that the "bad" event of Lemma 5.1 (namely the event that in an honest

[21] The proof of the indistinguishable output property uses the simulation soundness property.

[22] In the language of the UC framework, the environment $\mathcal{Z}$ will provide witnesses for honest prover sessions to VSim.

verifier's session there is an accepted proof for a new theorem without a commitment to the witness) only happens with roughly $1/T_{0.6}$ probability. Since this event is observable in time $T_{4.1} \ll T_5$ this would imply Lemma 5.1.[23]

The following observation will be very useful for us: it is enough to provide such a simulator for the case that there are some m ($\leq T_0(k)$) honest provers interacting with the adversary and only *one* honest verifier, with no other interaction going on. This is shown in detail in Section 6.

Organization of this section. We start by describing the virtual simulator VSim in Section 5.1. After we describe VSim we will prove that it satisfies the following three properties:

Completeness: (Section 5.2) The probability that VSim fails to simulate and aborts the computation is $\ll 1/T_{0.6}$.

Strong indistinguishability: (Section 5.3) The output of VSim is $(T_5, 1/T_0)$-indistinguishable from the output of the "real" simulator Sim.

Simulation soundness: (Section 5.4) The probability that in the transcript outputted by VSim, in the session where the adversary interacts with the honest verifier the verifier accepts a statement x but yet c_{wit} does *not* contain a commitment to a witness for x is $\ll 1/T_{0.6}$.

5.1 Description of the Virtual Simulator VSim

Similarly to Sim, the simulator VSim will be composed of $m+1$ separate interactive strategies for simulating the m honest provers and the honest verifier, where we denote these strategies by $\mathsf{VHP}_1, \ldots, \mathsf{VHP}_m$ and VHV. However, these strategies will not be completely independent, and will use some global variables as means of coordination. We assume that the execution happens in discrete time, where at time t the adversary adaptively decides in which session it wants to send its next message. The virtual simulator's strategy is sketched in Figure 1 (Page 25) although this figure would probably be easier to parse after at least skimming through the following subsections.[24]

5.1.1 Notations, inputs and global variables.

Notation. Since there is only one session in which the adversary interacts with the honest verifier, we will call this session the *honest verifier session*. Also, we will typically use primes to denote the messages sent in this session (e.g., use c'_{wit} for the commitment to the witness in this session). Whenever a computation of a particular step by the simulator uses super-polynomial time or memory, we will explicitly note the resources taken in square brackets.

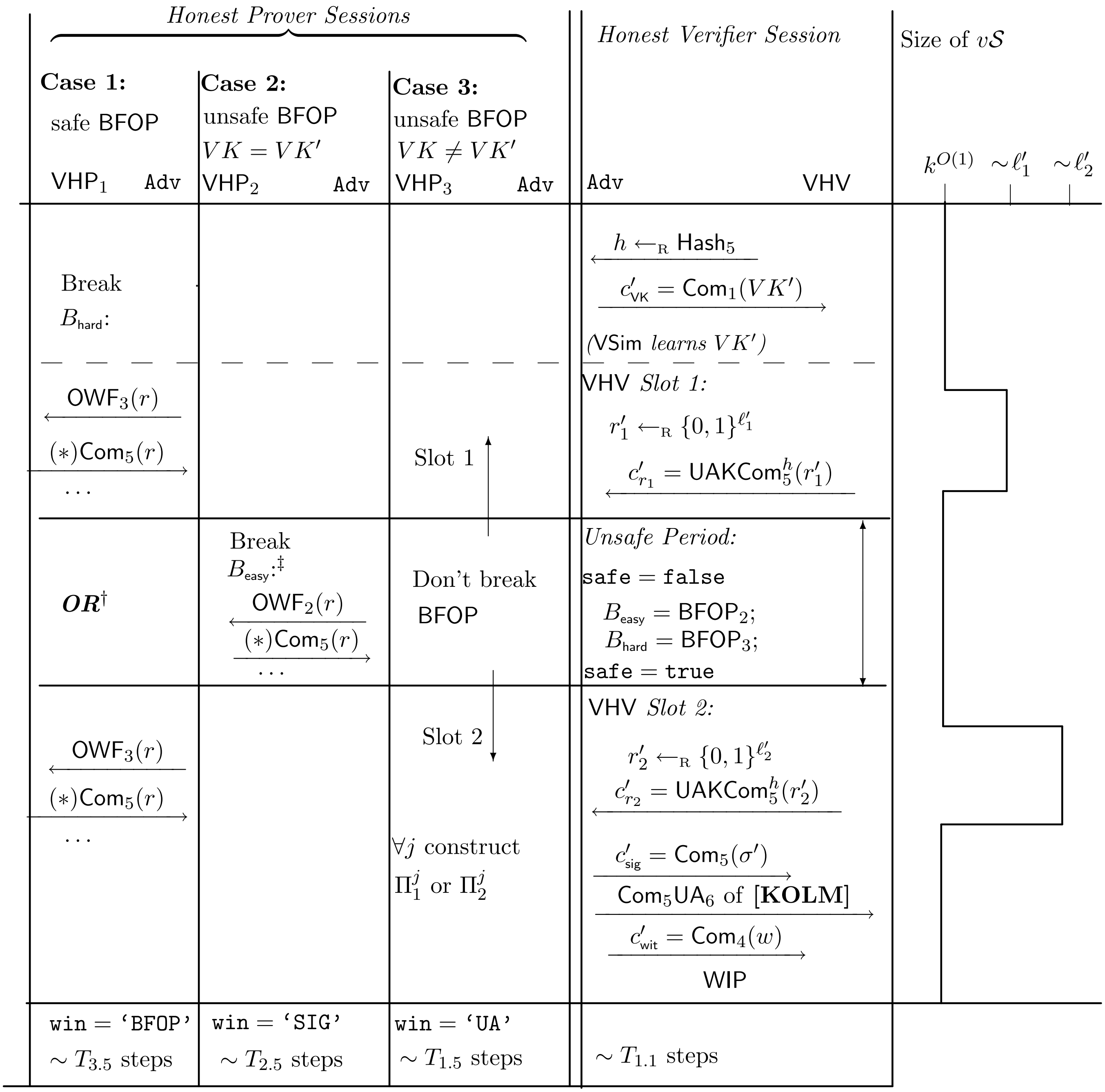
Inputs. VSim uses the following inputs (whose total size is bounded by $(T_0)^2$).

- The adversary's code[25] `Adv`.

[23]Note that this means that the virtual simulator can *not* commit to a real witness in the c_{wit} commitment in the view it outputs. Nor can it depart from the relaying strategy for its outer prover.

[24]For the benefit of the reader who is not intimately familiar with the UC framework, we present VSim in an intuitive manner here, not referring to the environment $\mathcal{Z}$ and ideal process. However, we will make use of VSim inside a hybrid experiment within the UC framework in Section 6. This will be done in a way that is obvious given our description of VSim.

[25]In the UC framework, this would include the code of the environment $\mathcal{Z}$ and the code of the adversary $\mathcal{A}$, as well as the code of any other aspects of the simulation outside of what VSim does.



†In Case 1, the message $(*)$ of B_{hard} is scheduled either before or after the unsafe period

‡Note that by the time this point is reached, VSim already knows whether or not $VK = VK'$.

Figure 1: Operation of the virtual simulator VSim

- *All* private and public inputs used by *all* the honest parties in the protocol[26].

Global Variables. VSim will use the following global variables:

- t - will always store the current time.
- (VK, SK) - initialized to be a verification and signing key pair chosen using the signature key-generating algorithm.
- VK' - represents the verification key that the adversary uses in the honest verifier session. It is initially empty.
- $v\mathcal{S}$ - the current internal state of the simulated honest verifier VHV. In addition, VSim maintains the *history* of all updates to this string. Thus, we will use the notation $v\mathcal{S}[t]$ to denote the contents of $v\mathcal{S}$ just after the step of time t.

  Note: When the verifier uses fresh randomness to compute its message at time t, this randomness is only added to the string $v\mathcal{S}$ at time t, and not before that. Also, we only add to $v\mathcal{S}$ the *internal* state of the verifier. Hence, if the verifier is running a public-coin sub-protocol (in which it is stateless) then we do not add anything to $v\mathcal{S}$ during this sub-protocol.
- `safe` - a Boolean flag, initially set to `true`. Intuitively, this flag tells the simulated honest provers when it is safe to break the B_{hard} challenge.
- $\mathsf{in\text{-}trans}$ - the transcript of the *inner prover* messages. That is, all the messages sent by the inner provers during the protocol (essentially containing aside from the public transcript the plain-texts m of the last messages in the various WI/ZK subproofs). The virtual simulator's goal is that $\mathsf{in\text{-}trans}$ will be $(T_5, 1/T_{0.6})$-indistinguishable from the corresponding "inner transcript" of Sim. Of all the global variables, only $\mathsf{in\text{-}trans}$ will be modified by the simulated prover.

The witness-assisted continuation procedure `Cont`. We denote the *residual strategy* of a simulated honest prover VHP_i at time t by $\mathsf{VHP}_{i,t}$. We will have a poly(k)-time procedure called `Cont` which will take a number $i \in [m]$ which identifies a simulated honest prover VHP_i and a time t. It will then return a *different* residual prover strategy $\widetilde{\mathsf{VHP}}_{i,t}$. This residual prover strategy will be a poly(k)-*time* interactive algorithm which we'll later prove to be $(T_4, 1/T_{0.6})$ indistinguishable from the "real" simulated prover residual strategy $\mathsf{VHP}_{i,t}$. We describe the procedure `Cont` in Section 5.1.4.

5.1.2 Simulation of Honest Verifier VHV.

We denote the "virtual honest verifier" algorithm used by VSim by VHV. We describe its operation by describing where it departs from the strategy used by the actual honest verifier.

- In all steps the verifier will add its internal state to the global variable $v\mathcal{S}$ as it proceeds. However, we note that in most parts of the proof, the verifier is *stateless* and hence has no internal state. The only place where it will need to maintain internal state is when executing

[26] In the UC framework, these would be provided by the environment $\mathcal{Z}$.

the two instances of UAKCom. We will explicitly describe below the data that the verifier is recording and deleting during these steps. **Important Note:** we assume that VHV does *not* choose all its random tape in advance, but selects the randomness required to compute a particular message when needed. Thus, the state variable $v\mathcal{S}$ at time t does *not* contain the randomness that will be used by VHV at time $t' > t$.

- After adversary commits to VK' (Step P1.2), the verifier breaks this commitment using $T_{1.1}$ time and records VK'. [Time needed for this step: $T_{1.1}$, Memory needed: $\mathrm{poly}(k)$]

- If $VK' \neq VK$, the verifier lets j_0 be the first index such that $VK'_{j_0} \neq VK_{j_0}$. Otherwise it lets $j_0 = 1$. It lets $\ell'(j_0) = VK'_{j_0} \cdot \ell_{\mathsf{VK}} + j_0$ and lets $\ell'_1 = \ell'(j_0) \cdot M$ and $\ell'_2 = (4\ell_{\mathsf{VK}} + 1 - \ell'(j_0)) \cdot M$. Note that ℓ'_1 and ℓ'_2 can be computed from the global variables VK and VK', and hence we may assume that they are global variables as well. Recall that $M = 2T_{0.1}$.

- When entering into Steps V,P2.x (the UAKCom of r_1) the verifier will choose r'_1 as a random string in $\{0,1\}^{\ell'_1}$. It will record r'_1 in $v\mathcal{S}$ (note that $|r'_1| = \ell'_1$). It will also record the (polynomial-sized) randomness used in computing the proof. At the end of the proof the verifier will *remove* the string r'_1 from $v\mathcal{S}$. [Time required for this step: $\mathrm{poly}(T_0)$, Memory required during the computation of step: $\ell'_1 + \mathrm{poly}(k)$]

- At the start of Steps P,V3.x (breaking opportunity), VHV will set the `safe` global variable to `false`. At the end of these steps it will set this variable back to `true`.

- The verifier will also perform the analogous computation in Steps V,P4.x, choosing r'_2 at random from $\{0,1\}^{\ell'_2}$. [Time required for this step: $\mathrm{poly}(T_0)$, Memory required during the computation of step: $\ell'_2 + \mathrm{poly}(k)$]

  Note: The only times $v\mathcal{S}$ will contain a super-polynomial sized string will be during the computation of these two steps. See Figure 1 for a graphic depiction of the verifier's operation and the size of $v\mathcal{S}$.

5.1.3 Simulation of the honest prover VHP

We now describe the operation of the simulated honest prover VHP_i. Note that, apart from using the global variables, VHP_i is a straight-line interactive algorithm. As before, we only describe the ways in which VHP_i deviates from the honest prover strategy. **Important note:** Like Sim, VHP_i will use the relaying outer prover strategy. Thus, when describing the simulated strategy, we only describe the strategy for the *inner prover*. Because the outer prover strategies of both VSim and Sim are identical, later it will be sufficient to prove this strategy is $(T_5, 1/T_{0.6})$ indistinguishable from the simulated inner strategy. Whenever, computing an output message of the inner strategy, VHP_i will add that message to the $\mathsf{in\text{-}trans}$ global variable.

- The prover has an internal variable `win` which can take a value in {`'none'`, `'BFOP'`, `'SIG'`, `'UA'`}. Initially it is set to `'none'`. Intuitively, this variable tells the simulated prover in which way it can "win" the session and convince the verifier in the final WI proof.

- The prover computes a signature σ on the statement proven in this session using the signing key SK.

- When the prover obtains the first message of the $\mathsf{UAKCom}_5^h(r_1)$ of Steps V,P2.x (i.e., the commitment $c_{r_1} = \mathsf{Com}_5\left(h(r_1)\right)$) it lets t_1 store the current time.

- When the prover obtains the first "challenge" message of the breaking opportunity in Steps V,P3.x, it checks whether the global `safe` flag is set to `true`. If so, then it breaks the B_{hard} challenge [using $T_{3.5}$ steps] and sets `win` = BFOP. Note this is depicted as Case 1 in Figure 1. Otherwise, we must already be finished with Slot 1 and in particular with $\mathsf{KCom}_5(VK')$ in the honest verifier session. Therefore, the variable VK' is already set. If $VK' = VK$ then the prover breaks B_{easy} and sets `win` = `'SIG'` [using $T_{2.1}$ steps]. Note that this is Case 2 of Figure 1. In both cases the prover now has information which will allow it to successfully run the WI proof of Steps P,V7.2.x.

- When the prover obtains the first message of the $\mathsf{UAKCom}_5^h(r_2)$ of Steps V,P4.x (i.e., the commitment $c_{r_2} = \mathsf{Com}_5(h(r_2)))$ it lets t_2 store the current time.

- When the simulated prover gets to the start of the "committed" universal argument (Steps P,V6.x), if `win` $\neq$ `'none'` then it just uses "junk" commitments in this part. It then uses its `win` strategy to continue and finish the WI proof. If `win` = `'none'` then the simulated prover does the following: (note that in this case we are in Case 3 of Figure 1)

  1. For $s = 1, 2$ do the following:
     (a) The prover obtains for each $j \in [m] \setminus \{i\}$, $\tilde{\mathsf{VHP}}_{j,t_s} = \mathtt{Cont}(j, t_s)$. Note that all of these are polynomial-time and polynomial-size algorithms. Thus, the advice needed for all of them together is less than $m \cdot \mathrm{poly}(k) \leq (T_0)^2$. Note that the procedure `Cont` is described below.
     (b) It then uses $v\mathcal{S}[t_s]$ to obtain the residual VHV algorithm at time t_s. Note that the residual verifier's running time is bounded by $\mathrm{poly}(T_{1.1})$ and its description is bounded by $\ell'_s + \mathrm{poly}(k)$.
     (c) Note that the advice needed to describe these residual algorithms is bounded by $\ell'_s + o(T_{0.1})$.
     (d) It combines all the algorithms described in (a) and (b) above with the adversary's algorithm to obtain a stand-alone prover algorithm P_* for the universal argument of knowledge of slot s.[27]
     (e) It lets $\tilde{\Pi}_s$ be the probabilistic program that on the empty input does the following:
       - Let $\mu = 1/(T_{0.6})^3$.
       - Use the knowledge extractor of the universal argument to obtain from P_* the string r^* that is compatible with the hash given at the start of the universal argument, with probability $1-\mu$, assuming that the probability that P_* convinces the universal argument verifier is at least μ.
     (f) Note that $\tilde{\Pi}_s$ can be described with $\ell'_s + O(T_{0.1})$ bits and its running time is $\mathrm{poly}(1/\mu) \cdot (m \cdot \mathrm{poly}(k) + \mathrm{poly}(T_{1.1}) \ll T_{1.2}$.
     (g) Let Π_s be the *deterministic* program that is obtained by derandomizing $\tilde{\Pi}_s$ using a pseudorandom generator for size $T_{1.2}$, going over all options for the pseudorandom

[27] The crucial point here is that none of these algorithm utilize the internal coins that VHP_i uses during the execution of the UA of slot s, in the session where VHP_i plays the role of the *verifier*.

generator's seed, outputting r_s and v such that $\mathsf{Com}(h(r_s); v) = c_{r_s}$ if such strings are found. Under our assumption there exists such a generator with seed size $< \log(T_{1.3}(k))$. Hence that Π_s has the same description size as $\tilde{\Pi}$ and its running time is $\mathrm{poly}(T_{1.3})$ steps.

(h) If Π_s does *not* output such a witness on the empty string then abort the simulation and output `ext-failure`.

2. For every $j \in [\ell_{\mathsf{VK}}]$, because $\langle j_0, VK'_{j_0}\rangle \neq \langle j, VK_j\rangle$, there is an $s \in \{1, 2\}$ such that $\ell'_s \leq \ell^s_j - M$. Thus, we can use Π_s (whose size is $\ell'_s + o(M)$) as a witness to run the universal argument. Note that running the universal argument on a statement verifiable in time $\mathrm{poly}(T_{1.3})$ will take us time $\ll T_{1.4}$.
3. After the universal argument is finished it sets `win` = `'UA'` and continues with the simulation.

- By the time we get to the WIP of Steps P,V7.2.x, `win` is already different from `'none'` and prover has information that allows to finish successfully this step.

5.1.4 Description of `Cont`.

The procedure `Cont` takes as input a number $i \in [m]$ which identifies a simulated honest prover VHP_i and a time t. It retrieves from the common inputs and global variables the inner transcript up to the point t, and the witness w for the statement proved by VHP_i. It then returns a $\mathrm{poly}(k)$-time residual prover strategy $\tilde{\mathsf{VHP}}_{i,t}$. This residual prover strategy is a true interactive algorithm in the sense that it will not access any of the global variables and only use what is hardwired into it. The procedure `Cont` runs in $\mathrm{poly}(k)$-time. We'll later prove that $\tilde{\mathsf{VHP}}_{i,t}$ is $(T_4, 1/T_{0.6})$ indistinguishable from the "real" simulated prover residual strategy $\mathsf{VHP}_{i,t}$ *even if the distinguisher gets access to the contents of all global variables at time t as an additional input.*

Operation of `Cont`. To describe the operation of `Cont`, we can simply describe the residual strategy $\tilde{\mathsf{VHP}}_{i,t}$. The residual strategy will be basically to use the *witness-based outer prover strategy.* That is, it will get as input the inner transcript of the i^{th} session up to point t and the witness for the statement proven in that session. Given the WBC-compiler we use, it is quite obvious what $\tilde{\mathsf{VHP}}_{i,t}$ will do: continue from this point using only "junk" commitments for commitment type messages, use the honest strategy for verification messages (which are stateless, and so can be computed from the public transcript.), and use the witness instead of the actual message to facilitate the WI proof.

5.2 Completeness of VSim

Recall that as we described VSim, there is a possibility that it will abort the computation and output `ext-failure`.[28] In this section we argue that this event only happens with negligible probability.

Lemma 5.2 (VSim is complete)**.** *The probability that* VSim *outputs* `ext-failure` *is at most* $1/T_{0.6}$.

[28]We note that the simulator does not abort in the case that a session ends because of the adversary's "fault" (i.e., the adversary fails to successfully complete some sub-protocol). In this case, the simulator simply outputs the partial transcript of the session.

Proof. The intuition behind this lemma is as follows: the only way that VSim outputs `ext-failure` is if it fails to extract r_1 or r_2 from a universal argument given to it by the adversary in some honest-prover session. Now, the only reason why the proof of knowledge of the universal argument doesn't immediately imply that this won't happen is that for the purposes of extraction VSim uses the witness-based strategy of the WBC compiler. However, because the success of the proof is a polynomial-time (and so in particular a T_4-time) observable event, the stand-alone prover constructed using witness-based continuation will have essentially the same success probability as the prover obtained from the actual VSim simulation, and hence this witness-based prover can be used just as well to extract r_1 or r_2. We now proceed with the formal proof.

Suppose, for the sake of contradiction, that VSim outputs `ext-failure` with probability at least $1/T_{0.6}$. Now, as mentioned above, the only places where VSim may abort is in the extraction of the verifier's challenges r_1 and r_2 by one of the VHP_i's.

Let us order all the UAKCom's done in the honest prover sessions by the timing of the *last* message in that sub-protocol. Under our assumption, there exists $i \in [m]$ and $s \in \{1, 2\}$ such that with probability at least $1/(T_0 \cdot T_{0.6}) \geq 1/(T_{0.6})^2$ the extraction of r_s in the i^{th} session is the *first* extraction that fails in the simulation (i.e., the universal argument ends successfully when simulating it for the first time, but extraction from it fails). This implies that there exists a prefix π of the simulation up until the time the universal argument of the Slot s in the i^{th} session starts, such that if we continue the simulation from the prefix π then with probability at least $1/(T_{0.6})^2$ the following will hold simultaneously:

1. The virtual simulator will *not* output `ext-failure` before the last message of this proof is sent.

2. The universal argument will finish successfully.

3. The virtual simulator will output `ext-failure` because of extraction failure in this universal argument.

(π contains both the transcript up to that point and the internal state of all parties up to that point.)

Indeed, let $\mathbf{H_A}$ denote the transcript of the simulation starting from π until the point that universal argument ends. Now let $\mathbf{H_B}$ denote this transcript where in all the cases the WBC compiler is used by the simulated honest provers, we use a commitment to the witness of the statement proven, instead of to a junk string. Note that both $\mathbf{H_A}$ and $\mathbf{H_B}$ are generated using $\ll T_4$ time and hence $\mathbf{H_A}$ and $\mathbf{H_B}$ are T_4-indistinguishable.

Now, we let $\mathbf{H_C}$ denote the transcript where whenever some virtual honest prover sends as part of the WBC compiler $\mathsf{Com}_4(w)\mathsf{Com}_4(m)$, it uses the first option (proving that w is a witness) rather than the second option (proving that m is valid) in the WI system.[29] Clearly $\mathbf{H_B}$ and $\mathbf{H_C}$ are T_4-indistinguishable by the WI property.

We now define $\mathbf{H_D}$ to be the hybrid where *all* commitments of type $\mathsf{Com}_4(m)$ sent by the virtual honest provers are commitments to "junk". Since we no longer use the coins used in generating these commitments, $\mathbf{H_D}$ is T_4-indistinguishable from $\mathbf{H_C}$.[30]

[29] Note that m is always valid in the simulation, since if the simulator can't come up with a valid m it simply outputs `ext-failure` and does not go through with the WBC compiler.

[30] Actually later commitments may depend on the coins used by the earlier commitments. Thus, to move from $\mathbf{H_C}$ to $\mathbf{H_D}$ we change these commitments to junk one by one, starting from the last commitment sent.

Now, by T_4 indistinguishability, we claim that with probability at least $1/(T_{0.6})^2 - 1/T_4 \geq 1/2(T_{0.6})^2$ the following will hold simultaneously in $\mathbf{H_D}$:

- The virtual simulator will *not* output `ext-failure` before the last message of this proof is sent.
- The universal argument will finish successfully.

However, the only difference between $\mathbf{H_D}$ and the transcript of a simulation where all virtual honest provers use a witness-based continuation is that in a witness-based continuation we never output `ext-failure`. Therefore, also in the latter case the universal argument will finish successfully with probability at least $1/2(T_{0.6})^2$. Now consider the prover algorithm P_* for the universal argument of this session that the virtual simulator obtains by combining all the witness based continuations from π and the residual honest verifier continued from π. This means that P_* will convince the verifier to accept with probability at least $1/2(T_{0.6})^2 \geq 1/(T_{0.6})^3$ and hence starting from π, the knowledge extractor will obtain a witness from this prover with probability at least $1-(T_{0.6})^3$, and hence will output `ext-failure` with probability less than $1/2(T_{0.6})^2$, contradicting Property 3 of the prefix π as stated above.

□

5.3 T_5-Indistinguishability of VSim

In this section, we will argue that the simulations produced by Sim and VSim are $(T_5, 1/T_{0.5})$-indistinguishable[31]. Note that it will be important to have indistinguishability against adversaries that are much stronger than T_4, because we will need to argue that even an adversary that can break the commitment $c_{\mathsf{wit}} = \mathsf{Com}_4(w)$, where w is the witness, cannot distinguish between simulations. We will use this fact later to argue that simulation-soundness for VSim implies simulation-soundness for Sim.

The indistinguishability argument will be through a series of hybrids. These hybrids will employ rewinding strategies; this means we will take care to ensure problems of efficiency in concurrent simulation do not arise.[32] Our overall strategy is to use zero-knowledge simulation only as an intermediary between straight-line simulation hybrids, so that the rewinding deals with only a single protocol execution. In this way, our hybrid simulators will never need to deal with the interaction between rewinding the adversary in one session and the rewinding of the adversary in another session. We call this the *technique of intermediate rewinding hybrids.*

Lemma 5.3 (T_5-indistinguishability of VSim and Sim)**.** *The outputs of* Sim *and* VSim *are* $(T_5, 1/T_{0.5})$*-indistinguishable.*

Proof. Recall that in our setting, the adversary is only involved in a single session in which the adversary plays the role of Prover, and the simulator plays the role of Verifier. All other history and state information is given as nonuniform advice to the adversary.

We will maintain the invariant that all our hybrids will run in time polynomial in $T_{4.5}$, which is sufficient time to break B_{hard} and extract the witness from the single session in which the adversary

[31] Again, the proof in this section can easily be translated to the language of the UC framework.

[32] Actually, we will use non-black-box simulation instead of rewinding-based simulation, because we use the protocol of [Bar01]. However, once the code of the adversary is given to the simulator, non-black-simulation is only easier than rewinding, and thus the intuitions we have on when rewinding work still hold.

plays the role of the prover. Therefore, for sake of arguing indistinguishability, if these hybrids are used as subroutines within a poly(T_5) procedure, this would yield another poly(T_5) procedure.

We note that because we need indistinguishability against strong adversaries, what we'll prove is the $(T_5, 1/T_{0.5})$ indistinguishability of the *inner* transcripts, which makes sense, since the outer prover strategy of VSim and Sim is identical (namely, the relaying strategy).

The inputs to all hybrids will be the same as the inputs to VSim. Let $\mathbf{H_A} = \mathsf{Sim}$.

Let $m < T_0$ be the maximum number of sessions in which the adversary plays the role of Verifier and the simulator plays the role of Prover.

5.3.1 Moving to simulation of VHV.

Using long r_1 and r_2. The only difference between the simulations of the honest verifier in VSim and Sim is that in VSim, the simulated honest verifier uses the UAKCom protocol to commit to random strings of super-polynomial length.

So we define a hybrid $\mathbf{H_{B/1}}$ that is identical to $\mathbf{H_A}$ except that the simulator uses the (rewinding/non-BB) ZK simulator for the ZK Universal Argument of Knowledge for r_1. The $(T_5, 1/T_5)$-indistinguishability of $\mathbf{H_A}$ from $\mathbf{H_B}$ follows from the ZK property as follows: We construct a T_5-time verifier V' to play the role of a verifier in the stand-alone ZK Universal Argument of Knowledge. This verifier V' internally runs the $\mathbf{H_A}$ simulation of all parties except for the prover's role in the ZK Universal Argument of Knowledge (recall that the simulated verifier plays the prover in this argument). It is crucial to note that this simulation does not need to record the internal state of the simulated verifier during this argument to carry out the rest of the simulation. Therefore V' is independent of the internal state of the prover in the universal argument, and so a valid stand-alone verifier. Hence, if there were a $T_5^{O(1)}$-time distinguisher for $\mathbf{H_B}$ and $\mathbf{H_A}$, it would yield a $T_5^{O(1)}$-time distinguisher for the ZK property of the universal argument with the same distinguishing probability.

Similarly, we may define a hybrid $\mathbf{H_C}$ that is identical to $\mathbf{H_B}$ except that the simulator uses the ZK simulator for the ZK Universal Argument of Knowledge for r_2. Note that the ZK simulation for r_2 is disjoint from the simulation for r_1. The $(T_5, 1/T_5)$-indistinguishability of $\mathbf{H_C}$ and $\mathbf{H_B}$ follows from an identical argument to the above.

Then we define hybrid $\mathbf{H_D}$ that is identical to $\mathbf{H_C}$ except that the simulator commits using Com_5 to the hash under h of random strings r_1 and r_2 of length $\ell^1_{j_0}$ and $\ell^2_{j_0}$ respectively, where these lengths are defined according to the rules of VSim. Note that the lengths of these strings are still polynomial in $T_{0.1}$, and therefore the overall running time of the hybrid simulator is still polynomial in T_5. Therefore the indistinguishability of Com_5 implies the $(T_5, 1/T_5)$-indistinguishability of $\mathbf{H_D}$ and $\mathbf{H_C}$.

We define hybrid $\mathbf{H_E}$ that is identical to $\mathbf{H_D}$ except that the simulator uses the honest prover strategy to prove knowledge of r_1 and r_2 in the UAKCom protocol. Again the $(T_5, 1/T_5)$-indistinguishability of $\mathbf{H_E}$ and $\mathbf{H_D}$ follows from the ZK property of the Universal Argument, by the argument given above for the indistinguishability of $\mathbf{H_A}$ and $\mathbf{H_B}$. We note that hybrid $\mathbf{H_E}$ is now a straight-line simulation.

This technique of intermediate rewinding hybrids is one that we will use repeatedly. In the sequel, we will not explicitly go through all these hybrids, but go immediately to the end result.

5.3.2 Moving to simulation of VHP

Committing to valid signatures. We define hybrid $\mathbf{H_F}$ to be identical to $\mathbf{H_E}$, except that all sessions set c_{sig} to be a commitment using Com_4 to σ_i, which is a signature using signing key SK to the theorem x_i being proved in the i'th session. By the security of the commitment scheme Com_5, it follows that hybrid $\mathbf{H_F}$ is $(T_5, 1/T_5)$-indistinguishable from $\mathbf{H_E}$. Note also that $\mathbf{H_F}$ is a straight-line simulation.

Breaking B_{easy} when needed. Recall that VSim, for certain sessions, breaks B_{easy}. We define hybrid $\mathbf{H_G}$ to be identical to $\mathbf{H_F}$ except that in certain sessions, B_{easy} is broken, according to the same criteria used by VSim. Again using the technique of intermediate rewinding hybrids above, we obtain that hybrid $\mathbf{H_G}$ is $(T_5, 1/T_5)$-indistinguishable from $\mathbf{H_F}$. The crucial observation that is needed to make this work is that no other party has access to the internal state a virtual prover VHP_i uses during the execution of the zero-knowledge argument of B_{easy}.

Satisfying `'UA'` when needed. We define hybrid $\mathbf{H_H}$ to be identical to $\mathbf{H_G}$ except that it follows the strategy given in the description of VSim to decide for which sessions to commit to a valid universal argument for the statement `'UA'`. Lemma 5.1 shows that with probability at least $1-(1/T_{0.6})$, and in time $\mathrm{poly}(T_{1.5})$, the hybrid simulator can produce a valid universal argument for the `'UA'` condition (otherwise it outputs `ext-failure` in which case $\mathbf{H_H}$ halts). This procedure uses rewinding, but it will be very important to us later that the rewinded sessions are simulated using witness-based continuation. Note that since every message of the committed Universal Argument is committed using Com_5, we obtain the $(T_5, 1/T_{0.5})$-indistinguishability of hybrids $\mathbf{H_H}$ and $\mathbf{H_G}$.

Using other success conditions in WIP. We are almost ready to compare our hybrid with VSim. Now the only difference between hybrid $\mathbf{H_H}$ and VSim is that $\mathbf{H_H}$ still always breaks B_{hard} and uses the `'BFOP'` condition to complete the final Witness-Indistinguishable Proof (WIP), whereas VSim sometimes does not break B_{hard} and it uses multiple conditions in the WIP. Note however that these same conditions are true in hybrid $\mathbf{H_H}$, though it just does not use them.

So, we first consider a hybrid $\mathbf{H_I}$ that is identical to $\mathbf{H_H}$, except that in the final WIP, for every session, hybrid $\mathbf{H_I}$ uses the same conditions for success that VSim would. Therefore, by the WI property of the WIP, we have that hybrid $\mathbf{H_I}$ is $(T_5, 1/T_5)$-indistinguishable from $\mathbf{H_H}$.

Eliminating unnecessary breakings of B_{hard} Finally, we construct our final hybrid $\mathbf{H_J} = \mathsf{VSim}$, in which B_{hard} is not broken in certain sessions, according to the rules of VSim. The $(T_5, 1/T_5)$-indistinguishability of $\mathbf{H_J}$ and $\mathbf{H_I}$ follows using the same arguments (the technique of intermediate rewinding hybrids) used to show the indistinguishability of hybrids $\mathbf{H_G}$ and $\mathbf{H_F}$. Here, when arguing indistinguishability based on ZK, in order to build a stand-alone verifier V', we must observe that even though certain B_{hard} sessions that need to be modified in this hybrid may overlap with Slot 2, which may need to be "rewound" in order to generate valid Committed Universal Arguments for use in other sessions, this rewinding is done using witness-based continuation, and therefore is independent of the stand-alone prover's internal state. Note that V' can incorporate knowledge of r_1 and r_2 and the randomness used to produce Universal Arguments of Knowledge of these strings, because all this information is only $T_{0.1}^{O(1)}$ in length, and our V' is a T_4-size circuit.

Thus, we obtain the result that the outputs of Sim and VSim are $(T_5, 1/T_{0.5})$-indistinguishable.

□

5.4 Simulation-Soundness of VSim

In this section we prove that VSim satisfies the simulation-soundness property. Namely, we prove the following lemma:

Lemma 5.4 (Simulation-soundness of VSim)**.** *The probability that in the transcript output by* VSim *all the following three conditions hold simultaneously:*

1. *The verifier of the honest-verifier session accepts the proof.*
2. *The statement x' proven in the honest-verifier session is distinct from all statements proven in the other sessions.*
3. *None of the T_4-secure commitment in the interaction contains a witness to the fact that $x' \in L$.*

is less than $1/T_{0.5}$.

We note that Lemma 5.4 also implies that Protocol $\mathcal{X}$ satisfies the standard (non-simulation) soundness property. This is because standard soundness follows from applying the simulation soundness property to an adversary that ignores everything that happens outside of the honest verifier session.

We prove Lemma 5.4 by proving the following sequence of propositions:

Proposition 5.5 (**[BFOP]** false in HV session)**.** *The probability that in the transcript output by* VSim *both following conditions hold is at most $1/T_3$:*

- *The simulated verifier* VHV *accepts the proof given by the adversary in the honest verifier session.*

  and

- *The condition **[BFOP]** of the honest verifier holds (i.e., if we let y be the first message and* $\mathsf{Com}(r)$ *be the second message of the sub-protocol B_{hard} of the honest verifier session, then* $y = \mathsf{OWF}(r)$*)*

  and

- *The witness-based continuation of B_{hard} in the honest verifier session does not contain a commitment to a witness.*

Proof. Suppose otherwise. Fix a "typical" partial transcript π of the history up to the point the simulated honest verifier VHV sends the first message of B_{hard} (i.e., $y = \mathsf{OWF}_3(r)$) to the prover, such that with at least $1/T_3$ probability if we continue the simulation from π then all three events mentioned above will occur. We let $s = s(\pi)$ be all the internal states of all simulated parties until that point. Note that $|\pi|, |s| \leq \text{poly}(T_0)$. Now, since during the entire execution of this instance of B_{hard}, (i.e., the "unsafe" period) the simulator VSim always uses $\ll T_{2.6}$ computational steps, we get that there exists a $T_{2.6}$-size adversary that with probability $\geq 1/T_3$ finishes this B_{hard} successfully with its second message containing $\mathsf{Com}(r')$ with $\mathsf{OWF}(r') = y$ and *without* the commitment in the

WBC part containing a witness. This means that conditioning on the event (that happens with at least $1/T_3$ probability) that the second message contains such a successful commitment, and we still have $1/T_3$ probability that the proof will finish successfully without containing a witness. This means that with probability at least $1/T_3$ if we continue the execution up to the point where $\mathsf{Com}_4(m), \mathsf{Com}_4$ are sent it will be the case that m is a valid message and there is still $1/T_3$ probability that the WIPOK of the WBC continuation will finish successfully. In such a case we can extract m in poly(T_3) time, and repeating this entire procedure poly(T_3) time we can obtain enough messages m to extract r' thus inverting OWF_3 with poly(T_3) time and contradicting its security. □

Proposition 5.6 ([**SIG**] false in HV session: part 1)**.** *The probability that in the transcript output by* VSim*, it holds that:*

1. $VK = VK'$
2. *The statement x' proven in the honest verifier session is distinct from all statements proven in honest provers session.*
3. *The condition [**SIG**] holds in the honest verifier session (in particular, the commitment c'_{sig} contains a valid signature for x').*

is at most $1/T_6$.

Proof. The entire simulation(even considering breaking of BFOP) takes less than T_5 steps, and since breaking c'_{wit} takes $\ll T_6$ steps, if the proposition was false we'd get a $\ll T_6$-size forging algorithm for the signature scheme with $\geq 1/T_6$ success probability. □

Proposition 5.7 ([**SIG**] false in HV session: part 2)**.** *The probability that in the transcript output by* VSim*, it holds that:*

1. $VK \neq VK'$
2. *The condition [**SIG**] holds in the honest verifier session (in particular, B_{easy} is broke.).*
3. *The witness-based continuation of B_{easy} in the honest verifier session does not contain a commitment to a witness.*

is at most $1/T_2$.

Proof. If $VK \neq VK'$ then we never break an "unsafe" B_{easy} or B_{hard}, and never use more than $T_{0.9}$-time during the unsafe period. This means that in the same way as in Proposition 5.5, B_{easy} cannot be broken in this case. □

Proposition 5.8 ([**UA**] false in HV session)**.** *The probability that in the transcript output by* VSim *the condition [**UA**] holds in the honest verifier session (i.e., the decommitted universal argument transcript is an accepting proof for [**KOLM**] is at most* $1/T_5$.

Proof. First note that for the particular r'_1 and r'_2 chosen by the VSim verifier, with very high probability (i.e., at least $1 - 2^{-k+2}$) their Kolmogorov complexity is more than $\ell'_1 - k$ and $\ell'_2 - k$ respectively. Now, suppose otherwise that the statement **[UA]** holds with probability at least $1/T_5$. Let π be a partial transcript of all simulation up to the point the universal argument starts, and let $sp = sp(\pi)$ be the internal state of the simulated *honest provers* (and not the verifier) up to this point. We claim that if the probability that **[UA]** holds in a continuation of π is at least $1/T_5$ then we have a time $T_{5.5}$ algorithm that with advice π,sp outputs a witness to the statement **[KOLM]**. We do that by simply considering a $T_{5.5}$-time standalone prover for the universal argument that with advice π,sp combines all the simulated honest provers and the adversary into one, and breaks the level 5 commitments we use to commit to the universal argument. The reason this adversary does not need to use the honest verifier's internal state is that the universal argument is a public-coin protocol and hence the honest verifier does not need any internal state to continue it. We then use the knowledge extractor of the universal argument to extract a witness from this standalone prover algorithm.

The witness for **[KOLM]** contains in particular two strings $\tilde{r}_1$ and $\tilde{r}_2$ such that for $s = 1, 2$, $\tilde{r}_s$ is consistent with the commitment c'_{r_s} and for every $j \in [\ell_{\mathsf{VK}}]$, either the Kolmogorov complexity of $\tilde{r}_1$ is at most $\ell^1_j - k$ or the Kolmogorov complexity of $\tilde{r}_2 - k$ is at most $\ell^2_j - k$. For $j = j_0$ (the index chosen by the verifier VSim) we have that because the commitment c'_{r_s} is perfectly binding, there is an $s \in \{1, 2\}$ such that the Kolmogorov complexity of $\tilde{r}_s$ is at most $\ell'_s - k$ but $h(\tilde{r}_s) = h(r'_s)$, where r'_s is the string the verifier chose.

However, with probability $\geq 1 - 2^{-k}$ the Kolmogorov complexity of r'_s, which was chosen at random in $\{0,1\}^{\ell'_s}$ is *larger* than $\ell'_s - k$. Hence we get that $r'_s \neq \tilde{r}_s$ but $h(r'_s) = h(\tilde{r}_s)$. Combining this and considering that the simulated honest verifier chooses the hash at random from the collection Hash_6, we obtain a $T_5^{O(1)}$ algorithm for obtaining collisions for h, contradicting its T_6-security. □

Proposition 5.9 (WIP is sound)**.** *The probability that in the honest verifier's session the verifier accepts the* WIP *of Steps P,V7.2.x but the statement proven is false is at most* 2^{-k}.

Proof. The WIP is statistically sound against computationally unbounded adversary. The simulator VSim does not rewind the honest verifier at this point, nor does it use the verifier's internal state at this point. (In fact, because WIP is public-coins, the verifier doesn't really have an internal state at this point.) □

It is not hard to verify that Propositions 5.5, 5.6, 5.7, 5.8 and 5.9 together imply Lemma 5.4. □

6 Construction of a Concurrently Secure Multi-Party Protocol

In this section, we describe how our protocol and the analysis thereof can be used to build a protocol for secure multi-party computation, with quasi-polynomial simulation. We build upon the work of [CLOS02]. In [CLOS02], it is shown how to UC-realize any polynomial-time functionality (based on standard hardness assumptions) in the UC framework of [Can01], but using a trusted setup assumption. However, this assumption is only used to UC-realize the $\mathcal{F}_{\mathsf{ZK}}$ functionality. The remainder of the construction does not rely on the trusted setup assumption, and instead builds on the $\mathcal{F}_{\mathsf{ZK}}$-hybrid model. The $\mathcal{F}_{\mathsf{ZK}}$-hybrid model is a model in which all parties have access to polynomially many ideal zero-knowledge functionalities. Equivalently, one can describe this model as one where all parties have access to a single instance of the ideal functionality $\hat{\mathcal{F}}_{\mathsf{ZK}}$ for an NP-complete language like SAT; $\hat{\mathcal{F}}_{\mathsf{ZK}}$ is a multi-session multiple-use version of the ideal zero knowledge functionality. Below, when we refer to the $\hat{\mathcal{F}}_{\mathsf{ZK}}$-hybrid model, we refer to the model with common access by all parties to a single instance of the $\hat{\mathcal{F}}_{\mathsf{ZK}}$ functionality.[33] The session ID we call *sid* is sufficient to identify different calls to the functionality.) The $\hat{\mathcal{F}}_{\mathsf{ZK}}$ functionality is shown in Figure 2 below.

Functionality $\hat{\mathcal{F}}_{\mathsf{ZK}}$

Functionality $\hat{\mathcal{F}}_{\mathsf{ZK}}$ proceeds as follows, interacting with parties $P_1, ..., P_n$ and an adversary $\mathcal{S}$:

- Upon receiving (ZK, sid, P_i, P_j, y, w) from P_i: If w is a satisfying assignment for the SAT formula y then send the message (ZK, sid, P_i, P_j, y) to P_j and $\mathcal{S}$ (unless such a message was already sent before). Otherwise, ignore the message.

Figure 2: The multiple-use multi-session version of $\mathcal{F}_{\mathsf{ZK}}$

We first note that the constructions of [CLOS02] can be applied to adversaries and environments that are stronger than polynomial-time simply by growing the security parameter and assuming that the hardness assumptions hold against stronger adversaries. In particular, we will instantiate these protocols to work against $T_4^{O(1)}$ adversaries. So, we use the following version of the theorem proven by [CLOS02]:

Theorem 6.1. [CLOS02] *Assume that (enhanced) trapdoor permutations secure against $T_5^{O(1)}$-sized circuits exist. Then, for any well-formed multi-party ideal functionality $\mathcal{F}$, there exists a non-trivial protocol that UC-realizes $\mathcal{F}$ in the $\hat{\mathcal{F}}_{\mathsf{ZK}}$-hybrid model in the presence of malicious, static $T_4^{O(1)}$-time adversaries and environments (with polynomial simulation overhead).*

Our aim is to show that Protocol $\mathcal{X}$ UC-realizes the $\hat{\mathcal{F}}_{\mathsf{ZK}}$ functionality with $T_{4.5}$ simulation overhead. Combined with Theorem 6.1, this will yield the result we desire. Note that we *do not* need to invoke the UC theorem on the $\hat{\mathcal{F}}_{\mathsf{ZK}}$ functionality, because only one instance of this functionality is needed.

We note that alternatively, we could show this result in the Angel-based model of [PS04], and thereby obtain a UC theorem for our protocol. But we choose to give a direct analysis that we obtain $\hat{\mathcal{F}}_{\mathsf{ZK}}$ in order for our analysis to remain as self-contained as possible.

Thus we will show:

Theorem 6.2. *Assume that collision-resistant hash function families secure against subexponential circuits exist. Then Protocol $\mathcal{X}$ (using as the statement x to be proven the tuple "(ZK,sid, P_i, P_j, y)" taken from the input to the $\hat{\mathcal{F}}_{\mathsf{ZK}}$ functionality) UC-realizes the $\hat{\mathcal{F}}_{\mathsf{ZK}}$ functionality against T_0-time static adversaries and environments, with $T_{4.5}$-time ideal adversaries.*

Combining Theorems 6.1 and 6.2, we obtain:

[33] Note that, because we have only a single instance of the $\hat{\mathcal{F}}_{\mathsf{ZK}}$ functionality, we have done away with the *ssid* session ID's that were present in the $\hat{\mathcal{F}}_{\mathsf{ZK}}$ functionality defined in [CLOS02], which were needed there for technical reasons.

Theorem 6.3. *Assume that there exist collision-resistant hash function families secure against subexponential (2^{k^ϵ}-sized for fixed $\epsilon > 0$) circuits exist. And that there exists $k^{\log^c k}$-strong (enhanced) trapdoor permutations (where $c = c(\epsilon)$ is some constant). Let $T_0(k) = k^{\log k}$ (and hence $T_i(k) = 2^{\log^{f(i)} k}$ for some function $f(\cdot)$). Then, for any well-formed multi-party ideal functionality $\mathcal{F}$, there exists a non-trivial protocol that UC-realizes $\mathcal{F}$ in the presence of malicious, static T_0-time adversaries and environments, with $T_{4.5}^{O(1)}$-time ideal adversaries.*

We now proceed to the proof of Theorem 6.2.

Proof. Let $\mathcal{A}$ be a malicious static[34] adversary running in time T_0. We construct an ideal process adversary $\mathcal{S}$ with access to $\hat{\mathcal{F}}_{\mathsf{ZK}}$, which simulates a real execution of Protocol $\mathcal{X}$with $\mathcal{A}$ such that no T_0-time environment $\mathcal{Z}$ can distinguish the ideal process with $\mathcal{S}$ and $\hat{\mathcal{F}}_{\mathsf{ZK}}$ from a real execution of Protocol $\mathcal{X}$with $\mathcal{A}$.

Recall that $\mathcal{S}$ interacts with the ideal functionality $\hat{\mathcal{F}}_{\mathsf{ZK}}$ and with the environment $\mathcal{Z}$. The ideal adversary $\mathcal{S}$ starts by invoking a copy of $\mathcal{A}$ and running a simulated interaction of $\mathcal{A}$ with the environment $\mathcal{Z}$ and parties running the protocol. (We refer to the interaction of $\mathcal{S}$ in the ideal process as *external interaction*. The interaction of $\mathcal{S}$ with the simulated $\mathcal{A}$ is called *internal interaction*.)

In the next section, we give a description of the simulator $\mathcal{S}$.

6.1 Description of $\mathcal{S}$

Informally, the simulator $\mathcal{S}$ proceeds by following the strategy for Sim described above – that is, breaking B_{hard} when simulating proofs; and breaking the Com_4 commitments to the witness in order to extract witnesses from adversarially given proofs. We describe this more formally below:

Initialization The simulator $\mathcal{S}$ initially runs the signature scheme key generation algorithm to obtain a pair (VK, SK). Note that the simulator $\mathcal{S}$ will actually never make use of the signing key SK. This is introduced here only for technical reasons. $\mathcal{S}$ uses the same set of corrupted parties as $\mathcal{A}$.

Simulating communication with $\mathcal{Z}$. Every input value that $\mathcal{S}$ receives from $\mathcal{Z}$ is written on the input tape of $\mathcal{A}$ (as if coming from $\mathcal{A}$'s environment). Likewise, every output value written by $\mathcal{A}$ on its own output tape is copied to $\mathcal{S}$'s output tape (to be read by the environment $\mathcal{Z}$).

Simulating "ZK" activations where the prover is not corrupted. In the ideal process, when an honest prover P_i receives an input (ZK, sid, P_i, P_j, y, w) from the environment $\mathcal{Z}$, then P_i writes this message on its outgoing communication tape for $\hat{\mathcal{F}}_{\mathsf{ZK}}$. Recall that by convention, the (ZK, sid, P_i, P_j, y) part of this message (i.e. everything but the witness) is public and can be read by $\mathcal{S}$. Now, upon seeing that P_i writes a "ZK" message for $\hat{\mathcal{F}}_{\mathsf{ZK}}$, the simulator $\mathcal{S}$ initiates a simulation (described below) of a real party P_i interacting with another real (possibly corrupt) party P_j executing Protocol $\mathcal{X}$, with the statement $x =$ "(ZK, sid, P_i, P_j, y)". We note that if P_j is not corrupted, $\mathcal{S}$ simulates the messages of P_j acting exactly as an honest verifier following Protocol $\mathcal{X}$. If P_j is corrupted, then its messages come from $\mathcal{A}$. Note that $\mathcal{S}$ will only allow delivery of P_i's

[34] Note that this proof is given for static adversaries, but we later sketch how to extend this analysis to adaptive adversaries.

message in the ideal process to $\hat{\mathcal{F}}_{\mathsf{ZK}}$, and the delivery of $\hat{\mathcal{F}}_{\mathsf{ZK}}$'s message to P_j, if the simulation ends with P_j accepting the simulated proof.

The simulator follows the honest prover protocol when simulating P_i, except in the inner protocol, it deviates from the honest prover as follows: **(1)** It computes and sends $c_{\mathsf{VK}} = \mathsf{Com}_4(VK)$ instead of using $0^{\ell_{\mathsf{VK}}}$. **(2)** It acts as the honest prover until it reaches the B_{hard} subprotocol. When P_i receives the challenge $y = \mathsf{OWF}_3(r)$, the simulator (running in $T_{3.5}$-time) inverts OWF (which we assume to be a permutation) to recover r. Then P_i sends $\mathsf{Com}_5(r)$ to P_j, and provides a proof of knowledge of r according to the honest prover strategy in the ZK argument of knowledge for B_{hard}. **(3)** The simulator then reverts to an honest simulation of P_i, until it reaches the commitment to the witness. At this point, P_i sends $\mathsf{Com}_4(0^{\ell_{\mathsf{wit}}})$ instead of the commitment to the witness (which $\mathcal{S}$ does not have). **(4)** Finally, in the final WI Proof, the simulated P_i uses the `'BFOP'` condition to complete the proof.

Simulating "ZK" activations when the prover is corrupted. When $\mathcal{A}$, controlling corrupted party P_i, delivers a ZK message x="(ZK, sid, P_i, P_j, y)" to an uncorrupted party P_j in the internal (simulated) interaction, then $\mathcal{S}$ works as follows. $\mathcal{S}$ simulates the verifier by exactly following the honest verifier strategy. If the protocol ends successfully, then $\mathcal{S}$ examines all places in the protocol transcript where the prover used Com_4 to commit to a string. Because $\mathcal{S}$ runs in time $T_{4.1}$, it can break each of these commitments. It checks to see if any of these strings is a valid witness for the statement y (i.e. a satisfying assignment to y). If any one (chosen arbitrarily) of these strings w is a valid witness, then $\mathcal{S}$ forces party P_i to send "(ZK, sid, P_i, P_j, y, w)" to $\hat{\mathcal{F}}_{\mathsf{ZK}}$, and delivers $\hat{\mathcal{F}}_{\mathsf{ZK}}$'s response to P_j. If none of these strings is a valid witness, and yet the adversary succeeds in convincing the honest verifier, then $\mathcal{S}$ halts and outputs `ss-failure`.

We note that the above simulation is a straight-line simulation that does not require rewinding any party's state. Therefore, if multiple sessions are interleaved, the simulation proceeds exactly as described above, independently for each session.

We next proceed to the indistinguishability proof.

6.2 Indistinguishability

We now prove that $\mathcal{Z}$ cannot distinguish an interaction of (multiple concurrent calls to) Protocol $\mathcal{X}$with $\mathcal{A}$, from an interaction in the ideal process with $\hat{\mathcal{F}}_{\mathsf{ZK}}$ and $\mathcal{S}$. In order to show this, we examine several hybrid experiments. Note that we assume without loss of generality that both $\mathcal{A}$ and $\mathcal{Z}$ are deterministic.

Let hybrid $\mathbf{H_A}$ be the simulated interaction described above. We define the output of this hybrid to be the transcript of the "internal" simulated interaction with $\mathcal{A}$. Note that this is enough to compute the output of the environment $\mathcal{Z}$. Note that the running time of $\mathbf{H_A}$ is less than $T_{4.1}$, where the most time-consuming step is the extraction of the witness.

Simulation Soundness: Correctness of extraction. We first make use of the *simulation soundness* condition to show that the simulated "witness extraction" succeeds with overwhelming probability:

Let $\mathbf{H_B}$ be the "simulated interaction" above, but with the following differences: In this hybrid, we replace the ideal functionality $\hat{\mathcal{F}}_{\mathsf{ZK}}$ with one that *does not require* witnesses to be provided by corrupted parties in the ideal execution. That is, the if the ideal functionality receives the message

(ZK, sid, P_i, P_j, y) from party P_i – and P_i is corrupted – then it simply forwards the message to P_j without verifying anything. Furthermore, in this hybrid, when the party P_i is corrupted and attempts to prove $x =$(ZK, sid, P_i, P_j, y) to P_j, then the simulator simply checks if the protocol succeeds, and if so, it forwards the message to the modified ideal functionality. Note that the running time of $\mathbf{H_B}$ is less than $T_{3.5}$, where the most time-consuming step is the breaking of B_{hard}. (That is, because $\mathbf{H_B}$ does not invoke the witness extracting procedure, its running time is significantly smaller than the running time of $\mathbf{H_A}$).

We now argue that $\mathbf{H_A}$ and $\mathbf{H_B}$ are *statistically indistinguishable*. Without loss of generality, we may assume that the adversary $\mathcal{A}$ and the environment $\mathcal{Z}$ are deterministic. Thus, the only use of randomness arises in the simulation. We also note, for use later, that simulated verifiers use independent randomness (because they behave according to the honest verifier protocol.). Thus, we may define $\mathbf{H_A}(r)$ and $\mathbf{H_B}(r)$ to be the outputs of the hybrids when randomness r is used in the simulation. We note that for any r, we have that $\mathbf{H_A}(r)$ is always a prefix of $\mathbf{H_B}(r)$, and they are not equal *only* if the simulator halts and outputs `ss-failure` in $\mathbf{H_A}(r)$. We want to show that this happens with probability less than $1/T_{0.5}$.

We will show that if the probability p that the simulator halts and outputs `ss-failure` is larger than $1/T_{0.5}$, then this will contradict Lemma 5.4.

Let $m \leq T_0$ be the maximum number of sessions in which the adversary plays the role of the prover.

We consider the following experiment $E(r)$: Using randomness r, both $\mathbf{H_A}(r)$ and $\mathbf{H_B}(r)$ are computed. If the outputs are equal, E outputs "none". Otherwise E outputs the number i corresponding to this first session in which $\mathbf{H_A}(r)$ fails and outputs `ss-failure` (because it fails to extract a witness).

We define p_i to the probability over r that $E(r)$ outputs i. Note that $\Sigma_i p_i = p$. Therefore there exists some number j such that $p_j \geq p/T_0$.

We define a new hybrid $\mathbf{H_{A'}}$ (which comes "in between" hybrids $\mathbf{H_A}$ and $\mathbf{H_B}$) as follows: It is the same as $\mathbf{H_B}$, except that in the j'th session where the adversary plays the role of prover, the simulator acts as $\mathcal{S}$ does in $\mathbf{H_A}$, namely it breaks the Com_4 commitments made during the j'th session in order to recover a witness for the statement being proven. If such a witness is not recovered, then the simulation halts with output `ss-failure`. Note that for all sessions $i \neq j$ where the adversary plays the role of prover, $\mathbf{H_{A'}}$ does *not* use the witness extraction procedure, but just uses the honest verifier strategy. The important property of $\mathbf{H_{A'}}$ is that if we let p' to be the probability that $\mathbf{H_{A'}}$ halts and outputs `ss-failure`, then we have $p' \geq p_j \geq p/T_0$. We note that the running time of $\mathbf{H_{A'}}$ is at most $T_{3.5}$ for sessions $i \neq j$ and at most $T_{4.5}$ for simulating the j^{th} session.

We next define a hybrid $\mathbf{H_{A''}}$ (used only for this proof) which is identical to $\mathbf{H_{A'}}$, except that is uses the VSim simulation strategies as follows: All sessions in which the adversary acts as verifier are simulated using VHP, but only the $j'th$ session where the adversary acts as the prover is simulated using VHV. In this session, the simulation also halts and outputs `ss-failure` if none of the Com_4 commitments are to a valid witness. All other sessions where the adversary acts as the prover are simulated by using the honest verifier strategy. We observe that all activity in $\mathbf{H_{A''}}$ outside of the VSim simulations can be computed in time polynomial in T_0. Like $\mathbf{H_{A'}}$, that the running time of $\mathbf{H_{A''}}$ is at most $T_{3.5}$ for sessions $i \neq j$ and at most $T_{4.5}$ for simulating the j^{th} session. Let p'' be the probability that $\mathbf{H_{A''}}$ halts with an output of `ss-failure`.

Then, by Lemma 5.3 (strong indistinguishability of Sim and VSim), we have that $p' \leq p'' +$

$1/T_{0.5}^2$. But by Lemma 5.4 (simulation soundness of VSim) and the naming conventions we use for statements, we know that $p'' \leq 1/T_{0.5}$.

Therefore, $p \leq 2T_0/T_{0.5}^2 \leq 1/T_{0.5}$, and we have that hybrids $\mathbf{H_A}$ and $\mathbf{H_B}$ are $1/T_{0.5}$-statistically indistinguishable.

Indistinguishability of prover simulation. We next move to a sequence of hybrids showing that the simulation of uncorrupted provers is good. We will first move from $\mathbf{H_B}$ to a situation where the witnesses are used in all uncorrupted prover sessions.

We first define hybrid $\mathbf{H_C}$ as identical to $\mathbf{H_B}$, except that the honest provers in simulation replace the commitment $c_{\mathsf{wit}} = \mathsf{Com}_4(0^{\ell_{\mathsf{wit}}})$ with a commitment $c_{\mathsf{wit}} = \mathsf{Com}_4(w)$, where w is the witness to the statement being proven in that session. Because both hybrids run in time $T_{3.5}$, a standard hybrid argument shows that $\mathbf{H_B}$ and $\mathbf{H_C}$ are $(T_4, 1/T_4)$-indistinguishable.

Let hybrid $\mathbf{H_D}$ be identical to $\mathbf{H_C}$, except that in all sessions with honest provers, the proving party switches to using the `'WIT'` condition to complete the final WI proof. A standard hybrid argument based on the WI property of the WI proof shows that $\mathbf{H_C}$ and $\mathbf{H_D}$ are $(T_5, 1/T_5)$-indistinguishable, since both hybrids run in time $T_{3.5}$.

We next define a series of hybrids that we will analyze using the technique of intermediate rewinding hybrids, introduced in Section 5.3. The goal of these hybrids is to switch the behavior of the honest provers in $\mathbf{H_D}$ to stop breaking B_{hard}. The problem with this switch is that we first need to switch to using the (rewinding) ZK simulator for the proof of knowledge within the B_{hard} protocol. If we did this for all sessions, we could end up interleaved rewindings that could cause an unacceptable increase in the running time of the hybrid experiment. Therefore, we only switch to one ZK simulator at a time, thus maintaining a good enough running time.

Let $m < T_0$ be the maximum number of sessions in which the adversary plays the role of Verifier and an honest party plays the role of Prover.

We consider a sequence of hybrids, for each $i \in [1, m]$, called $\mathbf{H_{E/i}}$, $\mathbf{H_{F/i}}$, and $\mathbf{H_{G/i}}$. The "order" of these hybrids will be $\mathbf{H_{E/1}}, \mathbf{H_{F/1}}, \mathbf{H_{G/1}}, \mathbf{H_{E/2}}, \mathbf{H_{F/2}}, \mathbf{H_{G/2}}, \mathbf{H_{E/3}}, \ldots, \mathbf{H_{G/m}}$. We will maintain the invariant that $\mathbf{H_{G/i}}$ will be a straight-line execution for all $i \in [1, m]$.

Hybrid $\mathbf{H_{E/i}}$ is identical to the previous hybrid (that is $\mathbf{H_D}$ in the case of $\mathbf{H_{E/1}}$), except that in the i'th session where an honest party P plays the prover when interacting in Protocol $\mathcal{X}$, the party P will switch from giving a proper ZK proof of knowledge for $\mathsf{Com}_4(r' = r)$ inside the B_{hard} subprotocol, to giving a (rewinding-based) ZK simulated proof instead. We argue that hybrid $\mathbf{H_{E/i}}$ is $(T_5, 1/T_5)$-indistinguishable from the previous hybrid as follows: We construct a $T_{3.5}$-time verifier V' to play the role of a verifier in the stand-alone ZK proof of knowledge. This verifier V' internally runs the previous hybrid execution for all parties except for the prover role of P in the ZK proof of knowledge. We observe that as such, V' is a valid stand-alone verifier. Hence, if there were a $T_4^{O(1)}$-time distinguisher for $\mathbf{H_{E/i}}$ and the previous hybrid, it would yield a $T_4^{O(1)}$-time distinguisher for the ZK property of the ZK proof with the same distinguishing probability.

Hybrid $\mathbf{H_{F/i}}$ is identical to $\mathbf{H_{E/i}}$, except that in the i'th session, in the B_{hard} subprotocol, the experiment stops breaking the verifier's challenge $y = \mathsf{OWF}_3(r)$, and the response commitment $\mathsf{Com}_5(r)$ will be replaced by $\mathsf{Com}_5(0^{\ell_{\mathsf{OWF}_3}})$. (Recall that we assume OWF is a permutation.) By the indistinguishability property of Com_4, and the fact that all hybrids run in time less than $T_{3.5}$ we have that $\mathbf{H_{F/i}}$ is $(T_5, 1/T_5)$-indistinguishable from $\mathbf{H_{E/i}}$.

Finally, $\mathbf{H_{G/i}}$ is identical to $\mathbf{H_{F/i}}$, except that in the i'th session, the simulation of ZK proof of knowledge for $\mathsf{Com}_5(0^{\ell_{\mathsf{OWF}_3}})$ is replaced with an honest ZK proof of knowledge. Again, we have that $\mathbf{H}_{\mathbf{G/i}}$ is $(T_5, 1/T_5)$-indistinguishable from $\mathbf{H}_{\mathbf{F/i}}$, by the ZK property, using the same argument as above. Note also that hybrid $\mathbf{H}_{\mathbf{G/i}}$ is a straight-line execution (i.e. it has no rewinding), as promised.

We note that hybrid $\mathbf{H}_{\mathbf{G/m}}$ can be implemented in time only polynomial in T_0. The only difference remaining between the environment's view of $\mathbf{H}_{\mathbf{G/m}}$ and the real world interaction is that in the real world, c_{VK} is $\mathsf{Com}_4(0^{\ell_{\mathsf{VK}}})$. Thus, the $(T_4, 1/T_4)$-indistinguishability of $\mathbf{H}_{\mathbf{G/m}}$ and the real world interaction follows from a standard hybrid argument based on the indistinguishability of the commitment scheme, and the fact that both hybrid $\mathbf{H}_{\mathbf{G/m}}$ and the real world interaction are implementable in time polynomial in T_0.

With this, the theorem is established. □

7 Security against Adaptive Adversaries

In this section, we sketch how to obtain security against adaptive adversaries for our zero knowledge protocol, which immediately implies such security for secure multi-party computation using the results of [CLOS02]. For adaptive security, we will assume the existence of one-way permutations secure against subexponential adversaries.

The high-level idea is that the Witness-Based Continuation (WBC) property almost gives us adaptive security automatically: When a proving party is corrupted, we could explain all previous messages using the witness in the witness-based continuation. (Note that simulated verifiers act honestly, so there is no need to "explain" their behavior.) The only problem with this approach is that the WI proof involved in WBC compiler is not secure against adaptive adversaries: If one gives a WI proof that either X is true or Y is true, using a witness for X, then there is no generic way to explain that proof using a witness for Y.

We alleviate this technical problem using ideas introduced in [CLOS02]. Instead of giving a standard WI proof inside the WBC compiler, we construct a specialized proof system. We first recall some concepts from [CLOS02]:

Underlying standard commitment. The basic underlying commitment scheme Com_5 is the standard non-interactive commitment scheme based on a one-way permutation f (that is T_5-secure) and a hard-core predicate b of f. That is, in order to commit to a bit σ, one computes $\mathsf{Com}_5(\sigma) = \langle f(U_k), b(U_k) \oplus \sigma \rangle$, where U_k is the uniform distribution over $\{0,1\}^k$. Note that Com_5 is computationally secret, and produces pseudorandom commitments: that is, the distributions $\mathsf{Com}_5(0)$, $\mathsf{Com}_5(1)$, and U_{k+1} are computationally indistinguishable.

The Feige-Shamir Commitment Scheme. We briefly describe the Feige-Shamir trapdoor commitment scheme [FS89], which is based on the zero-knowledge proof for Hamiltonicity of Blum [Blu87]. First, we fix a graph G (with q nodes) with a Hamiltonian cycle. (We will specify the graph to be used later.) Then, in order to commit to 0, the committer commits to a random permutation of G using the underlying commitment scheme Com_5 (and decommits by revealing the entire graph and the permutation). In order to commit to 1, the committer commits to a graph containing a randomly labeled q-cycle only (and decommits by opening this cycle only). Note that the ability to decommit to both 0 and 1 implies that the committer knows a Hamiltonian cycle in G. On the other hand, given a Hamiltonian cycle in G, it is possible to generate commitments that are indistinguishable from legal ones, and yet have the property that one can decommit to both a 0 and a 1. Note that if the graph G is not hamiltonian, then this commitment scheme is a perfectly-binding computationally-hiding scheme.

The modified graph-based commitment Com^G. Our graph-based scheme, introduced in [CLOS02], which we denote Com^G, differs from the [FS89] scheme above in the following way:

To commit to a 0, the sender picks a random permutation π of the nodes of G, and commits to the entries of the adjacency matrix of the permuted graph one by one, using Com_5. The sender also commits (using Com_5) to the permutation π. These values are sent to the receiver as $c = \mathsf{Com}^G(0)$. To decommit, the sender decommits to π and decommits to every entry of the adjacency matrix. The receiver verifies that the graph it received is $\pi(G)$.

To commit to a 1, the sender chooses a randomly labeled q-cycle, and for all the entries in the adjacency matrix corresponding to edges on the q-cycle, it uses Com_5 to commit to 1 values. For all the other entries, including the commitment to the permutation π, it simply produces random values from U_{k+1} (for which it does not know the decommitment!) These values are sent to the received as $c = \mathsf{Com}^G(1)$. To decommit, the sender opens only the entries corresponding to the randomly chosen q-cycle in the adjacency matrix.

This commitment scheme has the property of being computationally secret, *i.e.* the distributions $\mathsf{Com}^G(0)$ and $\mathsf{Com}^G(1)$ are computationally indistinguishable for any graph G. Also, given the opening of any commitment to both a 0 and 1, one can extract a Hamiltonian cycle in G. Finally, as with the scheme of [FS89], given a Hamiltonian cycle in G, one can generate commitments to 0 and then open those commitments to both 0 and 1.

Furthermore, here if the simulator has knowledge of a Hamiltonion cycle in G, it can also produce a random tape for the sender explaining $c = \mathsf{Com}^G(0)$ as a commitment to both 0 and 1. If, upon corruption of the sender, the simulator has to demonstrate that c is a commitment to 0 then all randomness is revealed. To demonstrate that c was generated as a commitment to 1, the simulator opens the commitments to the edges in the q-cycle and claims that all the unopened commitments are merely uniformly chosen strings (rather than commitments to the rest of G). This can be done since commitments produced by the underlying commitment scheme Com_5 are pseudorandom.

Modified Witness-Based Continuation Compiler. Recall that in the WBC Compiler, the prover uses a weak commitment Com_4 to commit to its "inner" response $c_m = \mathsf{Com}_4(m)$. We will change how the prover commits to the witness c_w (see below). It then proves a WI proof that either the message m from c_m is a valid response in the inner protocol, or that the witness w from c_w is a good witness for x.

We will change the protocol as follows: We will still use $c_m = \mathsf{Com}_4(m)$. We then consider the statement, that the message m from c_m is a valid response in the inner protocol, as an NP statement, and use a canonical reduction to construct a graph G_m, such that any witness to the truth of this statement can be mapped to some Hamiltonion cycle in G_m. We then use $c_w = \mathsf{Com}_4^{G_m}(w)$.

We also canonically construct a graph G_w corresponding to the statement that there is a valid opening message w for c_w that is a valid witness for x. We now use our graph based commitment scheme Com^{G_m} to provide a parallelized Blum proof of the Hamiltonicity of G_w. Namely, the following is done k times in parallel:

1. The prover uses Com^{G_m} to commit to a randomly permuted adjacency matrix for G_w.
2. The verifier responds with a single challenge bit b.
3. If $b = 0$, the prover provides the permutation and opens all commitments. If $b = 1$, the prover opens only the entries corresponding to a Hamiltonian cycle in G_w.

Note that this is still a statistically sound and $(T_5, 1/T_5)$-indistinguishable WI-system proving that either $\mathsf{Com}_4(w)$ contains a witness or $\mathsf{Com}_4(m)$ contains a valid message.

We note that we have changed the honest prover's strategy to always use the witness-based outer prover strategy (although instead of sending commitments to "junk" messages it will just send a random string of the appropriate length). The key observation is that a simulator can use knowledge of a Hamiltonian cycle in G_m (if m is a well-formed response in the inner protocol) to provide responses to all queries in this protocol without knowing the witness (and therefore without knowing a Hamiltonian cycle in G_w) – this would be by always using $\mathsf{Com}^{G_m}(0)$, and then opening it to whatever is necessary. But by the explainability property of Com^{G_m}, such a simulator could also explain its actions by providing honest-looking randomness in the protocol above.

We omit the details here, but this suffices to establish security against adaptive adversaries, without relying on erasures by honest parties.

8 Conclusions and future directions.

We presented a general feasibility result for secure multi-party 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.

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

[Bar01] B. Barak. How to go beyond the black-box simulation barrier. In *Proc. 42nd FOCS*, pages 106–115. IEEE, 2001. Preliminary full version available on `http://www.math.`

ias.edu/~boaz.

[BAR02] B. Barak. Constant-Round Coin-Tossing With a Man in the Middle or Realizing the Shared Random String Model. In *Proc.* 43*rd FOCS*. IEEE, 2002. Preliminary full version available on http://www.math.ias.edu/~boaz.

[BAR04] B. Barak. *Non-Black-Box Techniques in Cryptography*. PhD thesis, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, 2004.

[BCL+05] B. Barak, R. Canetti, Y. Lindell, R. Pass, and T. Rabin. Secure Computation Without Authentication. Submitted for publication., 2005.

[BCNP04] B. Barak, R. Canetti, J. B. Nielsen, and R. Pass. Universally Composable Protocols with Relaxed Set-Up Assumptions. In *Proc.* 45*th FOCS*, pages 186–195. IEEE, 2004.

[BG02] B. Barak and O. Goldreich. Universal Arguments and Their Applications. In *Annual IEEE Conference on Computational Complexity (CCC)*, volume 17, 2002. Preliminary full version available as Cryptology ePrint Archive, Report 2001/105.

[BOV03] B. Barak, S. J. Ong, and S. Vadhan. Derandomization in Cryptography, 2003.

[BEA91] D. Beaver. Foundations of Secure Interactive Computing. In *Crypto '91*, pages 377–391, 1991. LNCS No. 576.

[BR93] 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.

[BOGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In *Proc.* 20*th STOC*, pages 1–10. ACM, 1988.

[BLU82] M. Blum. Coin Flipping by Phone. In *Proc.* 24*th IEEE Computer Conference (CompCon)*, pages 133–137, 1982. See also *SIGACT News*, Vol. 15, No. 1, 1983.

[BLU87] M. Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, pages 1444–1451, 1987.

[BFM88] M. Blum, P. Feldman, and S. Micali. Non-Interactive Zero-Knowledge and Its Applications. In *Proc.* 20*th STOC*, pages 103–112. ACM, 1988.

[CAN00] R. Canetti. Security and Composition of Multiparty Cryptographic Protocols. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 13(1):143–202, 2000.

[CAN01] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In B. Werner, editor, *Proc.* 42*nd FOCS*, pages 136–147. IEEE, 2001. Preliminary full version available as Cryptology ePrint Archive Report 2000/067.

[CDNO97] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable Encryption. In *Crypto '97*, pages 90–104, 1997. LNCS No. 1294.

[CF01] R. Canetti and M. Fischlin. Universally Composable Commitments. Report 2001/055, Cryptology ePrint Archive, July 2001. Extended abstract appeared in CRYPTO 2001.

[CGGM00] R. Canetti, O. Goldreich, S. Goldwasser, and S. Micali. Resettable Zero-Knowledge. In *Proc.* 32*th STOC*, pages 235–244. ACM, 2000.

[CKPR01] R. Canetti, J. Kilian, E. Petrank, and A. Rosen. Black-Box Concurrent Zero-Knowledge Requires (Almost) Logarithmically Many Rounds. *SIAM Journal on Computing*, 32(1):1–47, Feb. 2003. Preliminary version in STOC '01.

[CKL03] 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.

[CLOS02] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally Composable Two-party Computation. In *Proc.* 34*th STOC*, pages 494–503. ACM, 2002.

[DDN91] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437 (electronic), 2000. Preliminary version in STOC 1991.

[DN92] C. Dwork and M. Naor. Pricing via Processing or Combatting Junk Mail. In *Crypto '92*, pages 139–147, 1992. LNCS No. 740.

[DN00] C. Dwork and M. Naor. Zaps and Their Applications. In *Proc.* 41*st FOCS*, pages 283–293. IEEE, 2000.

[DNS98] C. Dwork, M. Naor, and A. Sahai. Concurrent Zero Knowledge. In *Proc.* 30*th STOC*, pages 409–418. ACM, 1998.

[Fei90] 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.

[FS89] U. Feige and A. Shamir. Zero Knowledge Proofs of Knowledge in Two Rounds. In *Crypto '89*, pages 526–545, 1989. LNCS No. 435.

[FS90] U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *Proc.* 22*nd STOC*, pages 416–426. ACM, 1990.

[Gol02] O. Goldreich. Concurrent Zero-Knowledge With Timing, Revisited. In *Proc.* 34*th STOC*, pages 332–340. ACM, 2002.

[Gol04] O. Goldreich. *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004.

[GK96] 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.

[GK90] 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.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In ACM, editor, *Proc.* 19*th STOC*, pages 218–229. ACM, 1987. See [Gol04, Chap. 7] for more details.

[GMW86] 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.

[GL90] 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.

[GM82] S. Goldwasser and S. Micali. Probabilistic Encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, Apr. 1984. Preliminary version appeared in STOC' 82.

[GMR85] 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.

[KLP05] Y. T. Kalai, Y. Lindell, and M. Prabhakaran. Concurrent General Composition of Secure Protocols in the Timing Model. In *Proc.* 37*th STOC*, pages 644–653. ACM, 2005.

[KOS03] J. Katz, R. Ostrovsky, and A. Smith. Round Efficiency of Multi-party Computation with a Dishonest Majority. In *Eurocrypt '03*, 2003. LNCS No. 2656.

[KSW97] 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.

[Kil92] J. Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proc.* 24*th STOC*, pages 723–732. ACM, 1992.

[KP01] J. Kilian and E. Petrank. Concurrent and resettable zero-knowledge in poly-logarithm rounds. In *Proc.* 33*th STOC*, pages 560–569. ACM, 2001. Preliminary full version published as cryptology ePrint report 2000/013.

[KPR98] J. Kilian, E. Petrank, and C. Rackoff. Lower bounds for zero knowledge on the Internet. In *Proc.* 39*th FOCS*, pages 484–492. IEEE, 1998.

[Lin03a] Y. Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *Proc.* 35*th STOC*, pages 683–692. ACM, 2003.

[Lin03b] 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.

[Lin03c] Y. Lindell. General Composition and Universal Composability in Secure Multi-Party Computation. In *Proc. 44th FOCS*, pages 394–403. IEEE, 2003.

[Lin04] Y. Lindell. Lower Bounds for Concurrent Self Composition. In *Theory of Cryptography Conference (TCC)*, volume 1, pages 203–222, 2004.

[MMY05] T. Malkin, R. Moriarty, and N. Yakovenko. Generalized Environmental Security from Number Theoretic Assumptions, 2005. In preparation.

[Mic94] S. Micali. CS proofs. In *Proc. 35th FOCS*, pages 436–453. IEEE, 1994.

[MR91] S. Micali and P. Rogaway. Secure Computation. In *Crypto '91*, pages 392–404, 1991. LNCS No. 576.

[Nao89] M. Naor. Bit Commitment Using Pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991. Preliminary version in CRYPTO' 89.

[Nao02] M. Naor. Deniable Ring Authentication. In *Crypto '02*, 2002. LNCS No. 2442.

[Pas03a] R. Pass. On Deniability in the Common Reference String and Random Oracle Model. In *Crypto '03*, 2003.

[Pas03b] R. Pass. Simulation in Quasi-Polynomial Time, and Its Application to Protocol Composition. In *Eurocrypt '03*, 2003. LNCS No. 2656.

[Pas04] R. Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *Proc. 36th STOC*, pages 232–241. ACM, 2004.

[PR03] R. Pass and A. Rosen. Bounded-Concurrent Secure Two-Party Computation in a Constant Number of Rounds. In *Proc. 44th FOCS*. IEEE, 2003.

[PR05] R. Pass and A. Rosen. New and Improved Constructions of Non-Malleable Cryptographic Protocols. In *Proc. 37th STOC*. ACM, 2005.

[PSW00] B. Pfitzmann, M. Schunter, and M. Waidner. Cryptographic Security of Reactive Systems. *Electronic Notes in Theoretical Computer Science (ENTCS)*, 32, 2000. Workshop on Secure Architectures and Information Flow, Royal Holloway, University of London, December 1 - 3, 1999.

[PW00] B. Pfitzmann and M. Waidner. Composition and Integrity Preservation of Secure Reactive Systems. In S. Jajodia, editor, *Proceedings of the 7th ACM Conference on Computer and Communications Security*, pages 245–254, Athens, Greece, Nov. 2000. ACM Press.

[PRS02] M. Prabhakaran, A. Rosen, and A. Sahai. Concurrent Zero Knowledge with Logarithmic Round-Complexity. In *Proc. 43rd FOCS*. IEEE, 2002.

[PS04] M. Prabhakaran and A. Sahai. New notions of security: achieving universal composability without trusted setup. In *Proc. 36th STOC*, pages 242–251. ACM, 2004.

[Rab81] M. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.

[RBO89] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proc. 21st STOC*, pages 73–85. ACM, 1989.

[RK99] R. Richardson and J. Kilian. On the Concurrent Composition of Zero-Knowledge Proofs. In *Eurocrypt '99*, pages 415–432, 1999. LNCS No. 1592.

[Ros00] A. Rosen. A Note on the Round-Complexity of Concurrent Zero-Knowledge. In *Crypto '00*, pages 451–468, 2000. LNCS No. 1880.

[Sah99] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *Proc. 40th FOCS*, pages 543–553. IEEE, 1999.

[Sha79] A. Shamir. How to Share a Secret. *Communications of the ACM*, 22(11), Nov. 1979.

[SRA78] 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.

[Yao86] A. C. Yao. How to Generate and Exchange Secrets. In *Proc. 27th FOCS*, pages 162–167. IEEE, 1986.