Round-Optimal Secure Two-Party Computation

Jonathan Katz [1] ⋆ Rafail Ostrovsky [2] ⋆⋆

[1] Dept. of Computer Science, University of Maryland
jkatz@cs.umd.edu
[2] Dept. of Computer Science, U.C.L.A.
rafail@cs.ucla.edu

Abstract. We consider the central cryptographic task of *secure two-party computation*: two parties wish to compute some function of their private inputs (each receiving possibly different outputs) where security should hold with respect to arbitrarily-malicious behavior of either of the participants. Despite extensive research in this area, the exact round-complexity of this fundamental problem (i.e., the number of rounds required to compute an *arbitrary* poly-time functionality) was not previously known.

Here, we establish the **exact** round complexity of secure two-party computation with respect to black-box proofs of security. We first show a lower bound establishing (unconditionally) that four rounds are *not* sufficient to securely compute the coin-tossing functionality for any super-logarithmic number of coins; this rules out 4-round protocols for other natural functionalities as well. Next, we construct protocols for securely computing *any* (randomized) functionality using only five rounds. Our protocols may be based either on certified trapdoor permutations or homomorphic encryption schemes satisfying certain additional properties. The former assumption is implied by, e.g., the RSA assumption for large public exponents, while the latter is implied by, e.g., the DDH assumption. Finally, we show how our protocols may be modified — without increasing their round complexity and without requiring erasures — to tolerate an *adaptive* malicious adversary.

1 Introduction

Round complexity measures the number of messages that parties need to exchange in order to perform some joint task. Round complexity is a central measure of efficiency for any interactive protocol, and much research has focused on improving bounds on the round complexity of various cryptographic tasks. As representative examples (this list is not exhaustive), we mention work on upper- and lower-bounds for zero-knowledge proofs and arguments [6, 7, 40, 19, 27, 28], concurrent zero-knowledge [42, 35, 13, 41, 17, 15], and secure two-party

⋆ Part of this work was supported by NSF Trusted Computing Grant #0310751.
⋆⋆ Part of this work was supported by a gift from the Teradata Corporation.

and multi-party computation [43, 4, 5, 31, 11, 21, 32, 34, 37, 14, 22, 23, 33, 10]. The study of secure two-party computation is fundamental in this regard: not only does it encompasses functionalities whose round-complexity is of independent interest (such as coin tossing or the zero-knowledge functionality), but it also serves as an important special case in the study of secure computation.

Yao [43] presented a constant-round protocol for secure two-party computation when the adversarial party is assumed to be *honest-but-curious* (or *passive*). Goldreich, Micali, and Wigderson [29, 25] extended Yao's result, and showed a protocol for secure multi-party computation (and two-party computation in particular) tolerating *malicious* (or *active*) adversaries. Unfortunately, their protocol does not run in a constant number of rounds. Recently, Lindell [37] gave the first constant-round protocol for secure two-party computation in the presence of malicious adversaries; he achieves this result by constructing the first constant-round coin-tossing protocol (for polynomially-many coins) and then applying the techniques of [29]. The number of rounds in the resulting protocol for secure two-party computation is not specified by Lindell, but is on the order of 20–30.

The above works all focus on the case of a non-adaptive adversary. A general methodology for constructing protocols secure against an *adaptive* adversary is known [12], and typically requires additional rounds of interaction.

Lower bounds on the round-complexity of secure two-party computation with respect to black-box[1] proofs of security have also been given. (We comment further on black-box bounds in Section 1.2.) Goldreich and Krawczyk [28] showed that, assuming $NP \not\subseteq BPP$, zero-knowledge (ZK) proofs or arguments for NP require 4 rounds. Since ZK proofs (of knowledge) are a particular example of a two-party functionality, this establishes a lower bound of 4 rounds for secure two-party computation. Under the same complexity assumption, Lindell [38] has shown that for some polynomial p, secure coin-tossing of $p(k)$ coins requires at least 4 rounds.

1.1 Our Results

Here, we *exactly characterize* the (black-box) round complexity of secure two-party computation by improving the known bounds. In particular:

Lower bound: We show that 5 rounds are necessary for securely tossing any super-logarithmic (in the security parameter) number of coins, with respect to black-box proofs of security. Thus implies a 5-round black-box lower bound for a number of other (deterministic) functionalities as well. Beyond the implications for the round complexity of secure computation, we believe the result is of independent interest due to the many applications of coin-tossing to other cryptographic tasks.

The result of Goldreich and Krawczyk [28] mentioned above implies a black-box lower bound of five rounds for the "symmetric" ZK functionality (where the

[1] Throughout this paper, "black-box" refers to black-box use *of an adversary's code/circuit* (and not black-box use *of a cryptographic primitive*, as in [30]). A definition of black-box proofs of security is given in Appendix A.

parties simultaneously prove statements to each other) — and hence the same lower bound on the black-box round complexity of secure two-party computation of general functionalities — assuming $NP \not\subseteq BPP$. In contrast, our lower bound holds *unconditionally*.

Matching upper bound: As our main result, we construct 5-round protocols for securely computing any (randomized) poly-time functionality in the presence of a *malicious* adversary. Our protocols may be based on various cryptographic assumptions, including certified, enhanced trapdoor permutations (see Definition 1 and Remark 1), or homomorphic encryption schemes satisfying certain additional properties. The former may be based on, for example, the RSA assumption for large public exponents, while the latter may be based on, for example, the decisional Diffie-Hellman (DDH) assumption in certain groups. Due to space limitations, we focus on the (more difficult) case of certified trapdoor permutations, and refer the reader to the full version for protocols based on alternate assumptions.

In Section 4.1, we sketch how our protocols can be extended — without increasing the round complexity and without requiring erasures — to tolerate an *adaptive* adversary. The necessary cryptographic assumptions are described in more detail there.

1.2 A Note on Black-Box Lower Bounds

Until the recent work of Barak [1, 2], a black-box impossibility result was generally viewed as strong evidence for the "true" impossibility of a given task. Barak showed, however, that non-black-box use of an adversary's code could, in fact, be used to circumvent certain black-box impossibility results [1]. Nevertheless, we believe *there is still an important place in cryptography for black-box impossibility results* for at least the following reasons:

1. A black-box impossibility result is useful insofar as it rules out a certain *class of techniques* for solving a given problem.
2. With respect to our current understanding, protocols constructed using non-black-box techniques, currently seem inherently less efficient than those constructed using black-box techniques.

It remains an interesting open question to beat the lower bound given in this paper using non-black-box techniques, or to prove that this is impossible.

1.3 Discussion

Yao's results [43] give a 4-round protocol secure against *honest-but-curious* adversaries, assuming the existence of enhanced [26, Sec. C.1] trapdoor permutations (an optimal 3-round protocol secure against honest-but-curious adversaries can be constructed based on the existence of homomorphic encryption schemes). Our lower bound shows that additional rounds are necessary to achieve security against the stronger class of *malicious* adversaries. Our upper bound, however, shows that (at least in the case of trapdoor permutations) a *single* (i.e., fifth) additional round suffices.

Our technique for achieving security against *adaptive* adversaries applies only to adversaries who corrupt at most one of the players. An interesting open question is to construct a *constant-round* protocol tolerating an adaptive adversary who can potentially corrupt *both* players.

2 Definitions and Cryptographic Preliminaries

We omit the (completely standard) definitions of security for two-party computation used in this work, which follow [39, 9, 25, 10]. However, we provide in Appendix A our definition of *black-box simulation* which is used to prove the lower bound of Section 3.

We assume the reader is familiar with the cryptographic tools we use and refer the reader elsewhere for definitions of non-interactive (perfectly binding) commitment schemes [24], 3-round witness-indistinguishable (WI) proofs of knowledge [20, 24], witness-extended emulation for proofs/arguments of knowledge [37], and the Feige-Shamir 4-round ZK argument of knowledge [19, 18]. We note that all the above may be constructed based on the existence of certified, enhanced trapdoor permutations.

To establish notation, we provide here our working definitions of *trapdoor permutations*, *hard-core bits*, and Yao's *garbled circuit technique*. We also discuss *equivocal commitment*, and show a new construction of this primitive.

Trapdoor permutations. For the purposes of the present abstract, we use the following simplified definition of trapdoor permutations (but see Remark 1):

Definition 1. *Let $\mathcal{F}$ be a triple of* PPT *algorithms* $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Invert})$ *such that if* $\mathsf{Gen}(1^k)$ *outputs a pair* (α, td)*, then* $\mathsf{Eval}(\alpha, \cdot)$ *is a permutation over* $\{0,1\}^k$ *and* $\mathsf{Invert}(\mathsf{td}, \cdot)$ *is its inverse.* $\mathcal{F}$ *is a* **trapdoor permutation family** *if the following is negligible in k for all poly-size circuit families $\{A_i\}$:*

$$\Pr[(\alpha, \mathsf{td}) \leftarrow \mathsf{Gen}(1^k); y \leftarrow \{0,1\}^k; x \leftarrow A_k(\alpha, y) : \mathsf{Eval}(\alpha, x) = y].$$

*We additionally assume that $\mathcal{F}$ satisfies (a weak variant of) "***certifiability***": namely, given some α it is possible to decide in polynomial time whether* $\mathsf{Eval}(\alpha, \cdot)$ *is a permutation over* $\{0,1\}^k$*.*

For notational convenience, we let (α, td) be implicit and will simply let $\mathsf{f}(\cdot)$ denote $\mathsf{Eval}(\alpha, \cdot)$, and $\mathsf{f}^{-1}(\cdot)$ denote $\mathsf{Invert}(\mathsf{td}, \cdot)$ (where α, td are understood from the context). Of course, f^{-1} can only be efficiently evaluated if td is known.

Remark 1. The above definition is somewhat less general than others that have been considered (e.g., that of [24, Def. 2.4.5]); in particular, the present definition assumes a domain of $\{0,1\}^k$ and therefore no "domain sampling" algorithm is necessary. Furthermore, the protocol of Section 4 does *not* immediately generalize for trapdoor permutations requiring such domain sampling. Nevertheless, by introducing additional machinery it is possible to modify our protocol so that it may be based on any family of enhanced trapdoor permutations (cf. [26, Sec. C.1]) satisfying the certifiability condition noted above. For simplicity, however, we use the above definition in proving our results and defer the more complicated protocol (and proof) to the full version.

Hard-core bits. We assume the reader is familiar with the notion of hard-core bits for any trapdoor permutation family (see [24]), and thus we merely describe the notation we use. Let $H = \{h_k : \{0,1\}^k \to \{0,1\}\}$ be a hard-core bit for some trapdoor permutation family $\mathcal{F}$ (we will let k be implicit, and set $h = h_k$); thus (informally), $h(z)$ is "hard" to predict given $\mathsf{f}(z)$. We extend this notation to a vector of k hard-core bits in the following way:

$$\boldsymbol{h}(z) \stackrel{\text{def}}{=} h(z)|h(f(z))|\cdots|h(\mathsf{f}^{k-1}(z)).$$

Now (informally), $\boldsymbol{h}(z)$ "looks pseudorandom" given $\mathsf{f}^k(z)$.

Yao's "garbled circuit". Our secure computation protocol uses as a building block the "garbled circuit" technique of Yao [43] which enables constant-round secure computation for *honest-but-curious adversaries.* We abstract Yao's technique, and only consider those aspects of it which are necessary for our proof of security. In what follows, F is a description of a two-input/single-output circuit whose inputs and output have the same length k (yet the technique may be generalized for inputs and output of arbitrary polynomial lengths). Yao's results give PPT algorithms $\mathsf{Yao}_1, \mathsf{Yao}_2$ for which:

- Yao_1 is a randomized algorithm which takes as input a security parameter 1^k, a circuit F, and a string $y \in \{0,1\}^k$. It outputs a "garbled circuit" $\mathsf{circuit}$ and *input-wire labels* $Z_{1,0}, Z_{1,1}, \ldots, Z_{k,0}, Z_{k,1} \in \{0,1\}^k$. The "garbled circuit" may be viewed as representing the function $F(\cdot, y)$.
- Yao_2 is a deterministic algorithm which takes as input 1^k, a "garbled circuit" $\mathsf{circuit}$, and k values $Z_1, \ldots, Z_k \in \{0,1\}^k$. It outputs either an invalid symbol $\perp$, or a value $v \in \{0,1\}^k$.

(When k is clear from the context, we omit it.)

We briefly describe how the above algorithms may be used for secure computation in the honest-but-curious setting. Let player 1 (resp., 2) hold input x (resp., y), and assume that player 1 is to obtain the output $F(x,y)$. First, player 2 computes $(\mathsf{circuit}, \{Z_{i,b}\}) \leftarrow \mathsf{Yao}_1(F, y)$ and sends $\mathsf{circuit}$ to player 1. Then, the two players engage in k instances of oblivious transfer: in the i^{th} instance, player 1 enters with "input" x_i, player 2 enters with "input" $(Z_{i,0}, Z_{i,1})$, and player 1 obtains the "output" $Z_i \stackrel{\text{def}}{=} Z_{i,x_i}$. Player 1 then computes $v = \mathsf{Yao}_2(\mathsf{circuit}, Z_1, \ldots, Z_k)$ and outputs v.

A 3-round protocol for oblivious transfer (OT) based on trapdoor permutations may be constructed as follows (we remark that using number-theoretic assumptions, 2-round OT is possible): Let player 1 have input b and player 2 have input strings $Z_0, Z_1 \in \{0,1\}^k$ (the goal is for player 1 to obtain Z_b). Player 2 begins by generating trapdoor permutation $(\mathsf{f}, \mathsf{f}^{-1})$ and sending f to player 1. Next, player 1 chooses random $z'_0, z'_1 \in \{0,1\}^k$, sets $z_b = \mathsf{f}^k(z'_b)$ and $z_{\bar{b}} = z'_{\bar{b}}$, and sends

z_0, z_1 to player 2. Finally, player 2 computes $W_0 = Z_0 \oplus \boldsymbol{h}(\mathsf{f}^{-k}(z_0))$, computes W_1 analogously, and sends W_0, W_1 to player 1. Player 1 can then easily recover Z_b. (A proof of security for essentially the above protocol appears in [25].) Note that in the honest-but-curious setting it is secure to run polynomially-many executions of the above in parallel.

Putting everything together, we obtain the following 3-round protocol for secure computation of any single-output functionality in the honest-but-curious setting:

Round 1 Player 2 runs Yao_1 to generate $(\mathsf{circuit}, \{Z_{i,b}\})$. He then sends $\mathsf{circuit}$ and the f's for oblivious transfer.
Round 2 Player 1 sends k pairs (z_0, z_1).
Round 3 Player 2 sends k pairs (W_0, W_1).
Output computation Player 1 can now recover the appropriate $\{Z_i\}$ and thus compute the output value v using Yao_2, as discussed above.

Finally, any protocol for secure computation of single-output functionalities can be used for secure computation of two-output functionalities using only one additional round [25, Prop. 7.2.11]. Furthermore, any protocol for secure computation of deterministic functionalities may be used for secure computation of randomized ones (with the same round complexity) [25, Prop. 7.4.4].

With the above in mind, we describe the properties required of $\mathsf{Yao}_1, \mathsf{Yao}_2$. We first require *correctness*: for any F, y, any output $(\mathsf{circuit}, \{Z_i\})$ of $\mathsf{Yao}_1(F, y)$, and any x we have $F(x, y) = \mathsf{Yao}_2(\mathsf{circuit}, Z_{1,x_1}, \ldots, Z_{k,x_k})$. The algorithms also satisfy the following notion of *security*: there exists a simulator $\mathsf{Yao\text{-}Sim}$ which takes x, v as input, and which outputs $\mathsf{circuit}$ and a set of k input-wire labels $\{Z_i\}$; furthermore, the following distributions are computationally indistinguishable (by poly-size circuit families):

$$1.\ \left\{(\mathsf{circuit}, \{Z_{i,b}\}) \leftarrow \mathsf{Yao}_1(F, y) : (\mathsf{circuit}, \{Z_{i,x_i}\})\right\}_{x,y}$$

$$2.\ \left\{v = F(x, y) : \mathsf{Yao\text{-}Sim}(x, v)\right\}_{x,y}.$$

Algorithms $(\mathsf{Yao}_1, \mathsf{Yao}_2)$ satisfying the above definitions may be constructed assuming the existence of one-way functions.

Equivocal commitment. Although various notions of equivocal commitment have appeared previously, we present here a definition and construction specific to our application. Informally, an equivocal commitment scheme is an interactive protocol between a sender and a receiver which is computationally hiding and computationally binding in a real execution of the protocol. However, in a *simulated* execution of the protocol (where the simulator interacts with the receiver), the simulator is not bound to any particular value but can instead open the commitment to any desired value. Furthermore, for any (non-uniform) PPT receiver R and any string x, the view of R when the real sender commits/decommits to x is computationally indistinguishable from the view of R when the simulator "commits" in an equivocal way and later opens this commitment as x. We defer a formal definition, especially since one follows easily from the construction we now provide.

We construct an equivocal commitment scheme for a single bit in the following way: let Com be a non-interactive (perfectly binding) commitment scheme. To commit to a bit x, the sender chooses coins ω_1, ω_2 and computes $C = \mathsf{Equiv}(x; \omega_1, \omega_2) \stackrel{\text{def}}{=} \mathsf{Com}(x; \omega_1) || \mathsf{Com}(x; \omega_2)$. It sends C to the receiver and performs a zero-knowledge proof/argument that C was constructed correctly (i.e., that there exist x, ω_1, ω_1 such that $C = \mathsf{Equiv}(x; \omega_1, \omega_2)$). The receiver rejects in case the proof/argument fails. To decommit, the sender chooses a bit b at random and reveals x, ω_b. Note that a simulator can "equivocate" the commitment by setting $C = \mathsf{Com}(x; \omega_1) || \mathsf{Com}(\bar{x}; \omega_2)$ (where x is chosen at random in $\{0, 1\}$), simulating the zero-knowledge step, and then revealing ω_1 or ω_2 depending on x and the bit to be revealed. By committing bit-by-bit, the above extends easily to yield an equivocal commitment scheme for polynomial-length strings.

3 The Round Complexity of Coin Tossing

We show that any protocol for securely flipping a super-logarithmic number of coins (which is proven secure via black-box simulation) requires at least 5 rounds. (The reader is referred to Appendix A for a definition of black-box simulation.) More formally:

Theorem 1. *Let $p(k) = \omega(\log k)$, where k is the security parameter. Then there does not exist a 4-round protocol for tossing $p(k)$ coins which can be proven secure via black-box simulation.*

The above theorem refers to the case where *both* parties are supposed to receive the resulting coin as output.

Before starting our proof, we note that the above theorem is "tight" in the following two regards: first, for any $p(k) = O(\log k)$, 3-round protocols (proven secure using black-box simulation) for tossing $p(k)$ coins are known [8, 29, 25], assuming the existence of a non-interactive commitment scheme. Furthermore, our results of Section 4 imply a 5-round protocol (based on the existence of trapdoor permutations) for tossing any polynomial number of coins. In fact, we can also construct a 5-round protocol for tossing any polynomial number of coins based on the existence of a non-interactive commitment scheme; details will appear in the final version.

Proof (sketch). We assume (toward a contradiction) some 4-round protocol Π for tossing $p = p(k)$ coins. Without loss of generality, we may assume that player 1 sends the final message of Π (since in the ideal model, only player 1 has the ability to abort the trusted party); hence, player 2 must send the first message of Π. Consider a real-model adversary $\widetilde{A}_1$, corrupting player 1, who acts as follows: Let $\mathsf{Good} \subset \{0, 1\}^{p(k)}$ be some set of "small" but noticeable size, whose exact size we will fix later. $\widetilde{A}_1$ runs protocol Π honestly until it receives the third message, and then computes the value c of the tossed coin. If $c \in \mathsf{Good}$, then $\widetilde{A}_1$

completes execution of the protocol honestly and outputs some function of its view; otherwise, $\widetilde{A}_1$ aborts with output $\perp$.

Black-box security of Π implies the existence of a black-box ideal-model adversary $\widetilde{B}_1$ satisfying the following property (informally): conditioned upon receiving a coin $c \in \mathsf{Good}$ from the trusted party, with all but negligible probability $\widetilde{B}_1$ "forces" an execution with $\widetilde{A}_1$ in which $\widetilde{A}_1$ does not abort and hence $\widetilde{A}_1$'s view is consistent with some coin $c' \in \mathsf{Good}$ (for our proof, it does not matter whether $c' = c$ or not).

We next define a real-model adversary $\widetilde{A}_2$, corrupting player 2, acting as follows: $\widetilde{A}_2$ incorporates the code of $\widetilde{B}_1$ and — simulating the trusted party for $\widetilde{B}_1$ — feeds $\widetilde{B}_1$ a coin c randomly chosen from Good. By the above, $\widetilde{B}_1$ can with overwhelming probability "force" an execution with $\widetilde{A}_1$ in which $\widetilde{A}_1$ sees a view consistent with some $c' \in \mathsf{Good}$. We show that we can use $\widetilde{B}_1$ to "force" an execution with (the honest) A_1 in which A_1 outputs some $c' \in \mathsf{Good}$ with sufficiently high probability. Of course, $\widetilde{A}_2$ (and hence $\widetilde{B}_1$) interacts with the honest A_1, and not with adversarial $\widetilde{A}_1$; thus, in particular, $\widetilde{A}_2$ (and hence $\widetilde{B}_1$) cannot rewind A_1. However, since $\widetilde{A}_1$ acts "essentially" like the honest A_1 (with the only difference being due to aborts), we can show that $\widetilde{A}_2$ "forces" A_1 to output a coin $c' \in \mathsf{Good}$ with at least some inverse polynomial probability $1/q(k)$, where $q(k)$ relates to the number of queries $\widetilde{B}_1$ makes to its oracle for $\widetilde{A}_1$.

Choosing Good such that $|\mathsf{Good}|/2^k \leq 1/2q(k)$, we derive a contradiction: in any ideal-model execution, an honest player 1 outputs a coin in Good with probability at most $1/2q(k)$; in the real world, however, $\widetilde{A}_2$ forces an honest A_1 to output a coin in Good with probability at least $1/q(k)$. This implies a simple, poly-time distinguisher with non-negligible advantage at least $1/2q(k)$.

Remark 2. Theorem 1 immediately extends to rule out 4-round, black-box protocols for other functionalities (when both parties are supposed to receive output), and in particular some natural, deterministic ones. For example, the theorem implies that 4 rounds are not sufficient for computing the "xor" functionality (i.e., $F(x, y) = x \oplus y$) on inputs of super-logarithmic length, since any such protocol could be used to toss a super-logarithmic number of coins (in the same number of rounds). This can be generalized in the obvious way.

4 A 5-Round Protocol for Secure Computation

Here, we prove the existence of a 5-round protocol for secure computation of general functionalities based on the existence of (certified) trapdoor permutations (see Definition 1 and Remark 1). To simplify matters, we describe a 4-round protocol for secure computation of *deterministic* functionalities in which *only the first party receives output*; this suffices for our main result since any such protocol can be used for secure computation of randomized functionalities in which both parties receive (possibly different) outputs, at the cost of one more (i.e., fifth) additional round [25, Propositions 7.2.11 and 7.4.4].

Before describing our protocol, we provide some intuition about the "high-level" structure of our protocol and highlight some techniques developed in the course of its construction. We stress that our protocol does *not* merely involve "collapsing" rounds by running things in parallel — new techniques are needed to obtain a round-optimal protocol. At the core of our protocol is Yao's 3-round protocol tolerating *honest-but-curious* adversaries (it will be helpful in what follows to refer to the description of Yao's "basic" protocol in Section 2). The standard way of adding robustness against *malicious* adversaries (see [25]) is to "compile" this protocol by having the parties (1) commit to their inputs; (2) run (modified) coin-tossing protocols, so each party ends up with a random tape and the other party receives a commitment to this tape; and (3) run the basic Yao protocol with ZK proofs/arguments of correct behavior (given the committed values of the input and random tape) at each round. We may immediately note this approach will not suffice to obtain a 4-round protocol, since a ZK proof/argument for the first round of Yao's protocol alone will already require 4 rounds. Instead, we briefly (and informally) summarize some of the techniques we use to achieve a 4-round protocol. In the following (but not in the more formal description that follows), we number the rounds from 0–3, where round 0 corresponds to an "initialization" round, and rounds 1–3 correspond to rounds 1–3 of Yao's basic protocol.

- We first observe that in Yao's protocol *a malicious player 2 gains nothing by using a non-random tape* and thus coin-tossing for this party is not needed.
- It is essential, however, that player 1 is unable to choose his coins in round two. However, full-blown coin-tossing is unnecessary, and *we instead use a 3-round sub-protocol which "forces" player 1 to use an appropriate set of coins.* (This sub-protocol is run in rounds 0–2.) This component and its analysis are based loosely on earlier work of Barak and Lindell [3].
- When compiling Yao's protocol, *player 1 may send his round-two message before the proof of correctness for round one (being given by player 2) is complete* (here, we use the fact that the trapdoor permutation family being used is "certifiable"). We thus construct our protocol so the proof of correctness for round one completes in round three. To obtain a proof of security, we require player 2 to delay revealing $\mathsf{circuit}$ until round three. Yet, a proof of security also requires player 2 to be committed to a circuit at the end of the round one. We resolve this dilemma by having player 2 commit to $\mathsf{circuit}$ in round one using an *equivocal commitment scheme.*
- Finally, use a *specific* WI proof of knowledge (from [36]; see also [18]) with the property that *the statement to be proved (and, by implication, a witness) need not be known until the last round of the protocol,* yet soundness, completeness, and witness-indistinguishability still hold. (The proof *of knowledge* aspect must be dealt with more carefully; see Appendix B.) Furthermore, this proof system has the property that the first message from the prover is computed *independently* of the statement being proved (as well as its witness); we use this fact when constructing an adaptively-secure protocol in Section 4.1.

We also construct a novel 4-round ZK argument of knowledge with similar properties (see Appendix B), by modifying the Feige-Shamir ZK argument of knowledge [19]. Our new protocol may be of independent interest.

Let $\mathcal{F} = \{F_k\}_{k\in\mathbb{N}}$ be a polynomial-size (deterministic) circuit family representing the functionality of interest, where F_k takes two k-bit inputs and returns a k-bit output to player 1. (Clearly, the protocol extends for arbitrary input/output lengths. We have also mentioned earlier how the protocol may be extended for randomized, two-output functionalities.) When k is understood, we write F instead of F_k. Let $x = x_1 \cdots x_k \in \{0,1\}^k$ represent the input of player 1, let $y = y_1 \cdots y_k \in \{0,1\}^k$ represent the input of player 2, and let $v = F(x, y)$. In the following, i always ranges from 1 to k, and b ranges from 0 to 1.

First round. The protocol begins with first player proceeding as follows:

1. Player 1 chooses $2k$ values $\{r_{i,b}\} \stackrel{\text{def}}{=} \{r_{1,0}, r_{1,1}, \ldots, r_{k,0}, r_{k,1}\}$ at random from $\{0,1\}^k$. It then chooses $2k$ random coins $\{\omega_{i,b}\}$ and computes $\mathsf{Com}_{i,b} = \mathsf{Com}(r_{i,b}; \omega_{i,b})$, where Com is any perfectly-binding commitment scheme.
2. Player 1 also prepares the first message (which we call PoK_1) of a 3-round witness indistinguishable proof of knowledge (for a statement which will be fully determined in the third round; see the earlier remarks). For later reference, define $\mathsf{statement}_1$ as the following:
$$\exists\ \{(r_i, \omega_i)\}_{1\le i\le k} \text{ s.t. } \forall i : (\mathsf{Com}_{i,0} = \mathsf{Com}(r_i; \omega_i) \vee \mathsf{Com}_{i,1} = \mathsf{Com}(r_i; \omega_i))\,.$$
(Informally, $\mathsf{statement}_1$ represents the fact that player 1 "knows" either the decommitment of $\mathsf{Com}_{i,0}$ or the decommitment of $\mathsf{Com}_{i,1}$ for each i.)
3. Player 1 also prepares the first message (acting as the verifier) of the modified Feige-Shamir ZK argument of knowledge (see Appendix B). We denote this message by FS'_1.
4. The message sent by player 1 contains $\{\mathsf{Com}_{i,b}\}$, PoK_1, and FS'_1.

Second round. Player 2 proceeds as follows:

1. Player 2 generates $2k$ trapdoor permutations (denoted $\{(\mathsf{f}_{i,b}, \mathsf{f}_{i,b}^{-1})\}$) using $2k$ invocations of $\mathsf{Gen}(1^k)$, chooses $2k$ values $\{r'_{i,b}\}$ at random from $\{0,1\}^k$, and prepares the second message (denoted PoK_2) for the WI proof of knowledge initiated by player 1 in the previous round.
2. Next, player 2 generates a "garbled circuit" (cf. Section 2) for the functionality F, based on its own input y. This involves choosing random coins Ω and computing $(\mathsf{circuit}, \{Z_{i,b}\}) = \mathsf{Yao}_1(F, y; \Omega)$. Player 2 also computes commitments to the $\{Z_{i,b}\}$: that is, it chooses coins $\{\omega'_{i,b}\}$ and computes $\overline{\mathsf{Com}}_{i,b} = \mathsf{Com}(Z_{i,b}; \omega'_{i,b})$.
3. Player 2 next chooses random coins ζ and generates an equivocal commitment $\mathsf{Equiv} = \mathsf{Equiv}(\mathsf{circuit}; \zeta)$.
4. Next, player 2 prepares the second message (denoted FS'_2) for the modified Feige-Shamir ZK argument of knowledge (for a statement which will be fully determined in the fourth round; cf. Appendix B). For future reference, let $\mathsf{statement}_2$ be the following: there exist $\Big(y, \Omega, \mathsf{circuit}, \{Z_{i,b}\}, \{\omega'_{i,b}\}, \zeta\Big)$ s.t.:

(a) $(\mathsf{circuit}, \{Z_{i,b}\}) = \mathsf{Yao}_1(F, y; \Omega)$;
(b) $\forall i, b$: $\overline{\mathsf{Com}}_{i,b} = \mathsf{Com}(Z_{i,b}; \omega'_{i,b})$; and
(c) $\mathsf{Equiv} = \mathsf{Equiv}(\mathsf{circuit}; \zeta)$.

(Informally, $\mathsf{statement}_2$ states that player 2 performed the preceding two steps correctly.)

5. The message includes $\{\mathsf{f}_{i,b}\}$, $\{r'_{i,b}\}$, $\{\overline{\mathsf{Com}}_{i,b}\}$, Equiv, PoK_2, and FS'_2.

Third round. Player 1 proceeds as follows:

1. If any of the $\{\mathsf{f}_{i,b}\}$ are not valid[2], player 1 aborts. Otherwise, player 1 will use k parallel invocations of oblivious transfer to obtain the input-wire labels corresponding to its input x. Formally, for each i player 1 prepares values $(z_{i,0}, z_{i,1})$ in the following way:
   - If $x_i = 0$, choose random $z'_{i,0} \in \{0,1\}^k$ and set $z_{i,0} = \mathsf{f}^k_{i,0}(z'_{i,0})$. Also, set $z_{i,1} = r_{i,1} \oplus r'_{i,1}$ (recall, $r_{i,1}$ was committed to by player 1 in the first round, and $r'_{i,1}$ was obtained from player 2 in the second round).
   - If $x_1 = 1$, choose random $z'_{i,1} \in \{0,1\}^k$, set $z_{i,1} = \mathsf{f}^k_{i,1}(z'_{i,1})$, and set $z_{i,0} = r_{i,0} \oplus r'_{i,0}$.
2. Define $\mathsf{statement}_3$ as follows:
   $\exists \, \{(r_i, \omega_i)\}_{1 \le i \le k}$ s.t. $\forall i$:
   - $\big(\mathsf{Com}_{i,0} = \mathsf{Com}(r_i; \omega_i) \wedge z_{i,0} = r_i \oplus r'_{i,0}\big)$ **or**
   - $\big(\mathsf{Com}_{i,1} = \mathsf{Com}(r_i; \omega_i) \wedge z_{i,1} = r_i \oplus r'_{i,1}\big)$.

   Informally, this says that player 1 correctly constructed the $\{z_{i,b}\}$ values.
3. Player 1 then prepares the final message (denoted PoK_3) for the proof of knowledge begun in round 1. The statement[3] to be proved is: $\mathsf{statement}_1 \wedge \mathsf{statement}_3$. Player 1 also prepares the third message for the modified Feige-Shamir ZK protocol (denoted FS'_3).
4. The message includes $\{z_{i,b}\}$, PoK_3, and FS'_3.

Fourth round. The second player proceeds as follows:

1. If either PoK_3 or FS'_3 would cause rejection, player 2 aborts. Otherwise, player 2 completes the oblivious transfer in the standard way. Namely, for each $z_{i,b}$ sent in the previous round, player 2 computes $z'_{i,b} \stackrel{\text{def}}{=} \mathsf{f}^{-k}_{i,b}(z_{i,b})$ and xor's the k resulting hard-core bits with the corresponding input-wire labels thusly: $W_{i,b} = \boldsymbol{h}(z'_{i,b}) \oplus Z_{i,b}$.
2. Define $\mathsf{statement}_4$ as follows:
   $\exists \, \{(Z_{i,b}, \omega'_{i,b}, z'_{i,b})\}_{1 \le i \le k; b \in \{0,1\}}$ s.t. $\forall i, b$:

$$\big(\overline{\mathsf{Com}}_{i,b} = \mathsf{Com}(Z_{i,b}; \omega'_{i,b})\big) \bigwedge \big(\mathsf{f}^k_{i,b}(z'_{i,b}) = z_{i,b}\big) \bigwedge \big(W_{i,b} = \boldsymbol{h}(z'_{i,b}) \oplus Z_{i,b}\big) \, .$$

   Informally, this says that player 2 performed the oblivious transfer correctly.

[2] Recall (cf. Definition 1) that the trapdoor permutation family is certifiable.

[3] An honest player 1 actually knows multiple witnesses for $\mathsf{statement}_1$. For concreteness, we have the player choose one of these at random to complete the proof.

3. Player 2 prepares the final messages (denoted FS_4') for the modified Feige-Shamir protocol. The statement to be proved is: $\mathsf{statement}_2 \wedge \mathsf{statement}_4$.
4. Finally, player 2 decommits Equiv as $\mathsf{circuit}$ (recall from Section 2 how decommitment is done for equivocal commitments).
5. The message includes the $\{W_{i,b}\}$, $\mathsf{circuit}$ (and the corresponding decommitment), and FS_4'.

Output computation. The first player concludes the protocol as follows: If FS_4' or the decommitment of $\mathsf{circuit}$ would cause rejection, player 1 aborts. Otherwise, by completing the oblivious transfer (in the standard way) player 1 obtains $Z_i \stackrel{\text{def}}{=} Z_{i,x_i}$ (recall, x is the input of player 1) and computes $v = \mathsf{Yao}_2(\mathsf{circuit}, Z_1, \ldots, Z_k)$. If $v \neq \perp$, it outputs v. Otherwise, it aborts.

Sufficient assumptions. As noted in Section 2, every component of the above protocol may be based on the existence of a trapdoor permutation family (the certifiability property is only needed for the verification performed by player 1 at the beginning of the third round). Furthermore, as noted in Remark 1, although the description of the protocol (and its proof of security) use the definition of a trapdoor permutation family given by Definition 1, it is possible to adapt the protocol so that its security may be based on any family of (certified) enhanced trapdoor permutations, as per the definitions of [24, 26].

Theorem 2. *Assuming the existence of a trapdoor permutation family, the above protocol Π securely computes functionality F.*

Proof. We separately prove two lemmas dealing with possible malicious behavior of each of the parties; the theorem follows. We first consider the case when player 2 is malicious:

Lemma 1. *Let (A_1, A_2) be a pair of (non-uniform)* PPT *machines in which A_1 is honest. There exist a pair of (non-uniform) expected polynomial-time machines (B_1, B_2) such that*

$$\left\{\mathrm{REAL}_{\Pi,\overline{A}(z)}(x,y)\right\}_{x,y,z} \stackrel{\mathrm{c}}{\equiv} \left\{\mathrm{IDEAL}_{F,\overline{B}(z)}(x,y)\right\}_{x,y,z}. \tag{1}$$

Proof (sketch). Clearly, we may take B_1 to be honest. We assume that A_2 is deterministic, and construct B_2 using black-box access to A_2 as follows:

1. B_2 runs a copy of A_2 internally, passing to it any auxiliary information z. To emulate the first round of the protocol, B_2 acts exactly as an honest player 1, generates a first-round message, and passes this message to A_2. In return, B_2 receives a second-round message which includes, in particular, $\{r'_{i,b}\}$. If an honest player 1 would abort after receiving this second-round message, B_2 aborts (without sending any input to the trusted party) and outputs whatever A_2 outputs.

2. Otherwise B_2 generates a third-round message exactly as an honest player 1 would, with the following exception: for *all* i, b, it sets $z_{i,b} = r_{i,b} \oplus r'_{i,b}$. Note in particular that B_2 can easily compute PoK_3, since both $\mathsf{statement}_1$ and $\mathsf{statement}_3$ are true. It passes the third-round message to A_2, and receives in return a fourth-round message.
3. If an honest player 1 would abort after receiving the fourth-round message, B_2 aborts (without sending any input to the trusted party) and outputs whatever A_2 outputs. Otherwise, B_2 attempts to extract[4] from A_2 an input value y (cf. step 4 of the second round in the description of the protocol). If extraction fails, B_2 aborts and outputs fail.
4. Otherwise, B_2 sends y to the trusted party. It then stops and outputs whatever A_2 outputs.

We may note the following differences between the ideal world and the real world: (1) in the second round, B_2 sets $z_{i,b} = r_{i,b} \oplus r'_{i,b}$ for *all* i, b, whereas an honest player 1 does this only for i, b such that $x_i \neq b$; also (2) B_2 passes the input value y to the trusted party (and hence player 1 will receive the value $F(x, y)$ from this party), whereas in the real world player 1 will compute an output value based on the circuit and other values it receives from A_2 in the fourth round. Nevertheless, we claim that Equation (1) holds based on (1) the hiding property of the commitment scheme used in the first round and (2) the argument of knowledge (and hence soundness) property of the modified Feige-Shamir protocol (cf. Appendix B), as well as the correctness of the Yao "garbled circuit" construction. A complete proof appears in the full version.

Lemma 2. *Let (A_1, A_2) be a pair of (non-uniform) PPT machines in which A_2 is honest. There exist a pair of (non-uniform) expected polynomial-time machines (B_1, B_2) such that*

$$\left\{\mathrm{REAL}_{\Pi,\overline{A}(z)}(x,y)\right\}_{x,y,z} \stackrel{c}{\equiv} \left\{\mathrm{IDEAL}_{F,\overline{B}(z)}(x,y)\right\}_{x,y,z}. \tag{2}$$

Proof (sketch). Clearly, we may take B_2 to be honest. We assume that A_1 is deterministic, and construct B_1 using black-box access to A_1 as follows:

1. B_1 runs a copy of A_1 internally, passing to it any auxiliary information z and receiving a first-round message from A_1. Next, B_1 emulates the second round of the protocol as follows: it generates $\{\mathsf{f}_{i,b}\}$, $\{r'_{i,b}\}$, and PoK_2 exactly as an honest player 2. All the commitments $\overline{\mathsf{Com}}_{i,b}$, however, are random commitments to 0^k. Furthermore, commitment Equiv is set up in an "equivocal" way (cf. Section 2) so that B_1 will later be able to open this commitment to any value of its choice. B_1 prepares FS'_2 using the ZK simulator for the modified Feige-Shamir protocol (cf. Appendix B). B_1 passes the second-round message thus constructed to A_1, and receives in return a

[4] Technically, B_2 runs a *witness-extended emulator* [37] for the modified Feige-Shamir proof system, which results in a transcript t and a witness w. This is what we mean when we informally say that B_2 "attempts to extract".

third-round message. If an honest player 2 would abort after receiving this message, B_1 aborts (without sending any input to the trusted party) and outputs whatever A_1 outputs.
2. Otherwise, B_1 attempts to extract (cf. footnote 4) values $\{(r_i, \omega_i)\}_{1 \le i \le k}$ corresponding to (half) the commitments $\{\mathsf{Com}_{i,b}\}$ sent by A_1 in the first round. If extraction fails, B_1 outputs fail. Otherwise, let $b_i \in \{0,1\}$ be such that $\mathsf{Com}_{i,b_i} = \mathsf{Com}(r_i; \omega_i)$. B_1 then defines a string $x = x_1 \cdots x_k$ as follows:

$$\text{If } z_{i,b_i} = r_i \oplus r'_{i,b_i}, \text{ then } x_i = \bar{b}_i; \text{ otherwise, } x_i = b_i.$$

(x_i is B_1's "guess" as to which input-wire label A_1 is "interested in".) B_1 sends the string x thus defined to the trusted party, and receives a value v in return. It then runs $\mathsf{Yao\text{-}Sim}(x, v)$ to generate a garbled circuit $\mathsf{circuit}$ along with input-wire labels $\{Z_i\}$ (cf. Section 2). B_1 then prepares the "answers" $\{W_{i,b}\}$ to the oblivious transfer as follows, for each i: it correctly sets $W_{i,x_i} = h(\mathsf{f}_{i,x_i}^{-k}(z_{i,x_i})) \oplus Z_i$, but chooses $W_{i,\bar{x}_i}$ at random.
3. B_1 emulates the fourth round of the protocol as follows: it sends the $\{W_{i,b}\}$ as computed above, sends $\mathsf{circuit}$ as computed above (note that the corresponding decommitment can be given since Equiv was constructed in an "equivocal" way), and uses the simulator for the modified Feige-Shamir protocol to compute FS'_4 (cf. Appendix B). B_1 passes the final message thus constructed to A_1, and outputs whatever A_1 outputs.

We note (informally) the following differences between the ideal world and the real world: (1) $\{\overline{\mathsf{Com}}_{i,b}\}$ are commitments to 0^k rather than to "real" input-wire labels; (2) Equiv is set up so that B_1 can "equivocate" and later open this as any value it chooses; (3) the modified Feige-Shamir ZK argument is simulated rather than real; (4) the answers $\{W_{i,\bar{x}_i}\}$ are "garbage" (where x is B_1's guess as to the "input" of A_1); and (5) the garbled circuit is constructed using $\mathsf{Yao\text{-}Sim}$ rather than Yao_1. Nevertheless, we claim that Equation (2) holds. Due to lack of space, a complete proof appears in the full version.

4.1 Handling Adaptive Adversaries

We briefly sketch how the protocol above can be modified — without increasing the round complexity — to provide security against an *adaptive* adversary who can monitor communication between the parties and decide whom to corrupt at any point during the protocol based on this information. (We consider only an adversary who can corrupt at most *one* of the parties.) In brief, we modify the protocol by using a (public-key) adaptively-secure encryption scheme [12] to encrypt the communication between the two parties. Two issues arise:

1. The encryption scheme of [12] requires a key-generation phase which would necessitate additional rounds. We avoid this extra phase using the assumption of *simulatable public-key cryptosystems* [16] (see below). The existence of such cryptosystems is implied in particular by the DDH assumption [16]; see there for constructions based on alternate assumptions.

2. Regardless of the encryption scheme used, one additional round seems necessary just to exchange public keys. To avoid this, we do *not* encrypt the first message from player 1 to player 2. Nevertheless, the modified protocol is adaptively-secure: the proof uses the fact that the first round (as well as the internal state after the first round) is identical in both the real execution and the simulation for a malicious player 2 (cf. the proof of Lemma 1).

The modified protocol. Before describing our construction of an adaptively-secure encryption scheme, we outline how it will be used to achieve adaptive security for our protocol. Let Π denote the protocol given in the previous section. Our adaptively-secure protocol Π' proceeds as follows: in the first round of Π', player 1 sends a message just as in the first round of Π, but also sends sufficiently-many public keys (for an adaptively-secure encryption scheme) to enable player 2 to encrypt the messages of rounds two and four. (The adaptively-secure encryption scheme we use only allows encryption of a single bit; therefore, the number of public keys sent by player 1 is equal to the bit-length of messages two and four in Π.) In the second round of Π', player 2 constructs a message as in Π, encrypts this message using the corresponding public keys sent in round one, and additionally sends sufficiently-many public keys (for an adaptively-secure encryption scheme) to enable player 1 to encrypt the messages of rounds three and five. Π' proceeds by having the players construct a message just as in the corresponding round of Π and then having them encrypt these messages using the appropriate public keys sent by the other player.

We defer a proof of security for this construction to the final version.

An adaptively-secure encryption scheme. Informally, a public-key cryptosystem for single-bit plaintexts is *simulatable* if (1) it is possible to obliviously generate a public key without learning the corresponding secret key, and also (2) given a public key, it is possible to obliviously sample a random (valid) ciphertext without learning the corresponding message. We assume further that if a ciphertext is obliviously sampled in this way, then the probability that the corresponding plaintext will be a 0 or 1 is equal (or statistically close). See [16, Def. 2] for a formal definition.

Given such a cryptosystem, our construction of an adaptively-secure encryption scheme for a single-bit is as follows: the receiver generates k pairs $(pk_{i,0}, pk_{i,1})$ of public keys by generating one key of each pair (selected at random) using the key-generation algorithm, and the other key using the oblivious sampling algorithm. This also results in a set of k secret keys (one for each pair of public keys). To encrypt a bit m, the sender proceeds as follows: for each index i, choose a random bit b_i, set $C_{b_i} \leftarrow \mathcal{E}_{pk_{i,b_i}}(m)$ and choose $C_{\bar{b}_i}$ using the oblivious sampling algorithm. Then send the k ciphertext pairs (C_0, C_1).

To decrypt, the receiver decrypts one ciphertext out of each pair using the secret key it knows, and sets the decrypted message equal to the majority of the recovered bits. Note that correctness holds with all but negligible probability since (on average) 3/4 of the $2k$ ciphertexts constructed by the sender decrypt to the desired message m (namely, the k ciphertexts encrypted using the legitimate encryption algorithm, along with (on average) 1/2 of the remaining k ciphertexts chosen via the oblivious sampling algorithm).

We defer a proof that this scheme is adaptively secure to the full version.

Acknowledgments

We thank Bill Aiello for a helpful discussion, and the anonymous referees for their useful remarks. Thanks also to Yuval Ishai for his useful suggestions towards improving the presentation of our results.

References

1. B. Barak. How to Go Beyond the Black-Box Simulation Barrier. *42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 106–115, 2001.
2. B. Barak. Constant-Round Coin-Tossing with a Man-in-the-Middle or Realizing the Shared Random String Model. *43rd IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 345–355, 2002.
3. B. Barak and Y. Lindell. Strict Polynomial Time in Simulation and Extraction. *34th ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 484–493, 2002.
4. J. Bar-Ilan and D. Beaver. Non-Cryptographic Fault-Tolerant Computing in Constant Number of Rounds of Interaction. *Principles of Distributed Computing*, ACM, pp. 201–209, 1989.
5. D. Beaver, S. Micali, and P. Rogaway. The Round Complexity of Secure Protocols. *22nd ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 503–513, 1990.
6. M. Bellare, S. Micali, R. Ostrovsky. Perfect Zero-Knowledge in Constant Rounds STOC 1990: 482-493.
7. M. Bellare, S. Micali, R. Ostrovsky. The (True) Complexity of Statistical Zero Knowledge STOC 1990: 494-502
8. M. Blum. Coin Flipping by Phone. *IEEE COMPCOM*, pp. 133–137, 1982.
9. R. Canetti. Security and Composition of Multi-Party Cryptographic Protocols. *J. Cryptology* 13(1): 143–202, 2000.
10. R. Canetti, Y. Lindell, R. Ostrovsky, A. Sahai. Universally composable two-party and multi-party secure computation. STOC 2002: 494-503
11. R. Canetti, E. Kushilevitz, R. Ostrovsky, A. Rosen: Randomness versus Fault-Tolerance. J. Cryptology 13(1): 107-142 (2000)
12. R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively-Secure Multiparty Computation. *28th ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 639–648, 1996.
13. R. Canetti, J. Kilian, E. Petrank, and A. Rosen. Concurrent Zero-Knowledge Requires $\widetilde{\Omega}(\log n)$ Rounds. *33rd ACM Symp. on Theory of Comp. (STOC)*, ACM, pp. 570–579, 2001.
14. R. Cramer and I. Damgård. Secure Distributed Linear Algebra in a Constant Number of Rounds. *Adv. in Cryptology — Crypto '01*, LNCS 2139, Springer-Verlag, pp. 119–136, 2001.
15. G. Di Crescenzo and R. Ostrovsky. On Concurrent Zero-Knowledge with Preprocessing. In CRYPTO 1999: pp. 485-502.

16. I. Damgård and J.B. Nielsen. Improved Non-Committing Encryption Schemes. *Adv. in Cryptology — Crypto 2000*, LNCS vol. 1880, Springer-Verlag, pp. 432–450, 2000.
17. A. De Santis, G. DiCrescenzo, R. Ostrovsky, G. Persiano, A. Sahai: Robust Non-interactive Zero Knowledge. CRYPTO 2001: 566-598
18. U. Feige. Alternative Models for Zero Knowledge Interactive Proofs. PhD thesis, Weizmann Institute of Science, 1990.
19. U. Feige and A. Shamir. Zero Knowledge Proofs of Knowledge in Two Rounds. *Adv. in Cryptology — Crypto 1989*, LNCS vol. 435, Springer-Verlag, pp. 526–544, 1989.
20. U. Feige and A. Shamir. Witness Indistinguishability and Witness Hiding Protocols. *22nd ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 416–426, 1990.
21. M. Fitzi, J. Garay, U. Maurer, R. Ostrovsky: Minimal Complete Primitives for Secure Multi-party Computation. CRYPTO 2001: 80-100
22. R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. The Round Complexity of Verifiable Secret Sharing and Secure Multicast. *33rd ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 580–589, 2001.
23. R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. On 2-Round Secure Multi-party Computation. *Adv. in Cryptology — Crypto 2002*, LNCS vol. 2442, Springer-Verlag, pp. 178–193, 2002.
24. O. Goldreich. *Foundations of Cryptography, vol. 1: Basic Tools.* Cambridge University Press, Cambridge, UK, 2001.
25. O. Goldreich. Draft of a Chapter on Cryptographic Protocols, June 2003. Available at http://www.wisdom.weizmann.ac.il/~oded/foc-vol2.html.
26. O. Goldreich. Draft of an Appendix Regarding Corrections and Additions, June 2003. Available at http://www.wisdom.weizmann.ac.il/~oded/foc-vol2.html.
27. O. Goldreich and A. Kahan. How to Construct Constant-Round Zero-Knowledge Proof Systems for NP. *J. Cryptology* 9(3): 167–190, 1996.
28. O. Goldreich and H. Krawczyk. On the Composition of Zero-Knowledge Proof Systems. *SIAM J. Computing* 25(1): 169–192, 1996.
29. O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game: a Completeness Theorem for Protocols with Honest Majority. *19th ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 218–229, 1987.
30. R. Impagliazzo and S. Rudich. Limits on the Provable Consequences of One-Way Permutations. *21st ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 44–61, 1989.
31. J.. Kilian, E. Kushilevitz, S. Micali, R. Ostrovsky: Reducibility and Completeness in Private Computations. SIAM J. Comput. 29(4): 1189-1208 (2000)
32. Y. Ishai and E. Kushilevitz. Randomizing Polynomials: A New Representation with Applications to Round-Efficient Secure Computation. *41st IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 294–304, 2000.
33. J. Katz, R. Ostrovsky, and A. Smith. Round Efficiency of Multi-Party Computation with a Dishonest Majority. *Adv. in Cryptology — Eurocrypt 2003*, LNCS vol. 2656, Springer-Verlag, pp. 578–595, 2003.
34. E. Kushilevitz, R. Ostrovsky, A.Rosn: Amortizing Randomness in Private Multi-party Computations. SIAM J. Discrete Math. 16(4): 533-544 (2003)
35. J. Kilian and E. Petrank. Concurrent and Resettable Zero-Knowledge in Poly-logarithmic Rounds. *31st ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 560–569, 2001.

36. D. Lapidot and A. Shamir. Publicly-Verifiable Non-Interactive Zero-Knowledge Proofs. *Adv. in Cryptology — Crypto 1990*, LNCS vol. 537, Springer-Verlag, pp. 353–365, 1991.
37. Y. Lindell. Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation. *Adv. in Cryptology — Crypto 2001*, LNCS vol. 2139, Springer-Verlag, pp. 171–189, 2001.
38. Y. Lindell. Personal communication, 2001.
39. S. Micali and P. Rogaway. Secure Computation. *Adv. in Cryptology — Crypto 1991*, LNCS vol. 576, Springer-Verlag, pp. 392–404, 1991.
40. M. Naor, R. Ostrovsky, R. Venkatesan, M.Yung. Perfect Zero-Knowledge Arguments for NP Can Be Based on General Complexity Assumptions. CRYPTO 1992: 196-214
41. M. Prabhakaran, A. Rosen, and A. Sahai. Concurrent Zero Knowledge with Logarithmic Round-Complexity. *43rd IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 366–375, 2002.
42. R. Richardson and J. Kilian. On the Concurrent Composition of Zero-Knowledge Proofs. *Adv. in Cryptology — Eurocrypt 1999*, LNCS vol. 1592, Springer-Verlag, pp. 415–431, 1999.
43. A.C. Yao. How to Generate and Exchange Secrets. *27th IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 162–167, 1986.

A Black-Box Simulation

Typical definitions of security for two-party computation only require that for every pair of admissible real-world adversaries $\overline{A}$ there exists a pair of ideal-world adversaries $\overline{B}$ satisfying some relevant criterion (namely, indistinguishability of the resulting output distributions). Most work in this area, however, (and especially prior to the work of Barak [1]) proves the existence of such a $\overline{B}$ via what is known as a *black-box simulation*; this means that the ideal-model adversary B_i corresponding to the dishonest real-model adversary A_i is constructed using only oracle access to A_i.

More formally, a *black-box simulation for party 1* (with a completely analogous definition for black-box simulation for party 2) implies the existence of a simulator S_1 for which the following holds: For any real-model adversary A_1, let $B_1(x, z; r_A, r_S)$ (where x, z are the inputs of B_1 and $r = (r_A, r_S)$ are the random coins of B_1) be defined by $S_1^{A_1(x,z,\cdot\,;r_A)}(x; r_S)$, where $A_1(x, z, \cdot\,; r_A)$ denotes the next-message function of A_1 on the given inputs and random coins (we stress that S_1 is not explicitly given the auxiliary input z nor the random coins r_A). Then $\overline{A} = (A_1, A_2)$ and $\overline{B} = (B_1, B_2)$ (where A_2, B_2 are just the honest algorithms) satisfy the relevant criterion. Furthermore, S_1 runs in expected polynomial-time, where each oracle call to $A_1(x, z, \cdot\,; r_A)$ is counted as a single step. Finally (although this is not essential to our results), it is typical to assume that S_1 is a *uniform* algorithm. Note that if A_1 runs in strict polynomial time, the above implies that the entire algorithm B_1 runs in expected polynomial time; furthermore, if A_1 is uniform then so is B_1 (on the other hand, if A_1 is a non-uniform machine, then B_1 will be too). We say that a protocol is *proven secure via black-box simulation* if the simulations for both parties are black-box.

We stress a crucial point about the above: when we say S_1 runs in expected polynomial-time, we mean that there is a *fixed* polynomial $q(\cdot)$ such that the expected running time of S_1 on input x, when interacting with *any* A_1 (and counting queries to A_1 as a single step), is $q(|x|)$. On the other hand, the expected running time of B_1 (including the steps of A_1, and no longer counting each query to A_1 as a single step) cannot be bounded *a priori* by any fixed polynomial, as the running time of B_1 will of course depend on the running time of A_1. Of course, as noted above, if A_1 runs in strict polynomial time $q'(\cdot)$ then B_1 runs in expected time (at most) $q'(\cdot)q(\cdot)$, which is polynomial. (Note that this definition of black-box simulation avoids the technical problem of, e.g., [28] regarding the need for B_1 to feed A_1 coins r_A whose length depends on A_1 and is not bounded *a priori* by any polynomial.)

B Proof Systems Used in this Work

We provide here a laconic sketch of the proof systems claimed in Section 4; further details and proofs will appear in the full version. We first describe the WI proof of knowledge of [36] (as described in [18]).

We will be working with the NP-complete language HC of graph Hamiltonicity, and thus assume statements to be proved take the form of graphs, while witnesses correspond to Hamilton cycles. If thm is a graph, we abuse notation and also let thm denote the statement "$\mathsf{thm} \in HC$". We show how the proof system can be used to prove the following statement: $\mathsf{thm} \wedge \mathsf{thm}'$, where thm will be included as part of the first message, while thm' is only included in the last round (indeed, it will not be fixed until the third round begins). The proof system runs k parallel executions of the following 3-round protocol:

1. The prover commits to two adjacency matrices for two randomly-chosen cycle graphs C, C'. The commitment is done bit-by-bit using a perfectly-binding commitment scheme.
2. The verifier responds with a single bit b, chosen at random.
3. If $b = 0$, the prover opens all commitments. If $b = 1$, the prover sends two permutations mapping the cycle in thm (resp., thm') to C (resp., C'). For each non-edge in thm (resp., thm'), the prover opens the commitment at the corresponding position in C (resp., C').
4. The verifier checks that all commitments were opened correctly. If $b = 0$, the verifier additionally checks whether both decommitted graphs are indeed cycle graphs. If $b = 1$, the verifier checks whether each non-edge in thm (resp., thm') corresponds to a non-edge in C (resp., C').

Note that the prover does not need to know either thm or thm' (or the corresponding witnesses) until the beginning of the third round. However, we assume thm is fixed as part of the first-round message because this will enable us to claim stronger properties about the above proof system.

Very informally, we claim that the proof system above satisfies the following:

- It is complete and sound. In particular, the probability that an all-powerful prover can cause a verifier to accept when either thm or thm' are not true is at most 2^{-k}. We stress that this holds even if the prover can adaptively choose thm' after viewing the second-round message of the verifier.
- It is witness indistinguishable.
- It is a proof of knowledge for thm. (More formally, we can achieve a notion similar to that of witness-extended emulation [37] for thm.) We do not know whether such a claim holds for thm'.

Note also that the first round of the above proof system (as well as the internal state of the prover immediately following this round) is independent of thm or the associated witness. We rely on this fact in Section 4.1.

Next, we informally describe our modification of the Feige-Shamir ZK argument of knowledge [19] which will allow the prover to prove $\mathsf{thm} \wedge \mathsf{thm}'$, where thm is sent as part of the second round yet thm' is only sent as part of the last round (indeed, it need not be known until the beginning of that round). *We use the notation used in the description of the Feige-Shamir protocol in [18, Prot. 8.2.62].* Our modified protocol proceeds as follows:

1. The first round is as in the original protocol, and includes values x_1, x_2.
2. The prover chooses a random $R \in \{0,1\}^{2k}$ and computes $\mathsf{Equiv} = \mathsf{Equiv}(R, \zeta)$ (cf. Section 2). Let ok denote the statement that Equiv was formed correctly.
3. Let $\widetilde{\mathsf{thm}}$ denote the statement: $(\mathsf{thm} \wedge \mathsf{ok}) \vee (f(w') = x_1) \vee (f(w') = x_2)$ (this statement is reduced to a single graph $\widetilde{\mathsf{thm}}$). The prover sends Equiv and also the first message of the WI proof system described above.
4. The verifier's third message is as in the original protocol, except that the verifier additionally chooses and sends a random $R' \in \{0,1\}^{2k}$.
5. The prover decommits (as in Sec. 2) to R. Let prg be the statement that $r = R \oplus R'$ is pseudorandom (i.e., $\exists s$ s.t. $G(s) = r$, for G a PRG). Let $\widetilde{\mathsf{thm}}'$ be the statement $\mathsf{thm}' \vee \mathsf{prg}$ (reduced to a single graph $\widetilde{\mathsf{thm}}'$). The prover completes the WI proof system, as above, for the statement $\widetilde{\mathsf{thm}} \wedge \widetilde{\mathsf{thm}}'$.
6. The verifier checks the decommitment of R, and verifies the proof as before.

We claim the following about the above proof system:

- It is complete and sound (for a poly-time prover) for thm and thm'. (As argued earlier, rounds 2–4 constitute a proof of knowledge for $\widetilde{\mathsf{thm}}$. As in [18] — relying on the one-wayness of f — this implies that if a poly-time prover can cause a verifier to accept with "high" probability, then a witness for $\mathsf{thm} \wedge \mathsf{ok}$ can be extracted with essentially the same probability. If ok is true, then with all but negligible probability prg will *not* be true. Soundness of the proof of knowledge sub-protocol then implies that $\widetilde{\mathsf{thm}}'$ *is* true. But this means that thm' is true.)
- It is zero-knowledge. (In addition to simulating for $\widetilde{\mathsf{thm}}$ as in [18], the simulator also uses the equivocal commitment property to decommit to an R such that prg is true.)
- It is an argument of knowledge for thm (we have already argued as much above).