Witness Indistinguishable Proofs and Constant Round Zero Knowledge.

Boaz Barak

February 14, 2006

1 Constant-Round Witness Indistinguishable Proofs

We obtained constant-round witness indistinguishable proofs by the following sequence of results.

Theorem 1.1 (Blum 87)**.** *There exists a three-round zero knowledge proof system with soundness error* $1/2$ *for Hamiltonian Cycle.*

Blum's Hamiltonicity protocol. We used Protocol 1.2 to prove this. We let HAM denote the **NP**-complete language of all Hamiltonian graphs (i.e., n vertex graphs that contain the n-cycle as a subgraph). Blum's basic protocol for proving membership in HAM is Protocol 1.2. It is a 3-round public-coin proof for HAM with soundness error equal to $\frac{1}{2}$.

Theorem 1.3 (Feige Shamir)**.** *If a protocol is zero-knowledge then it is also witness indistinguishable.*

Theorem 1.4 (Feige Shamir)**.** *If a protocol is witness indistinguishable then its* k*-time parallel version is also witness indistinguishable.*

Theorem 1.5 (Bellare, Impagliazzo, Naor ?)**.** *If a* 3*-round protocol has soundness error* $1/2$ *then its* k*-time parallel version has soundness error* 2^{-k}*.*

(In class we showed this specifically for the Hamiltonian cycle protocol but it holds for general 3-round protocols, although there are some subtle problems for protocols with ≥ 4 rounds.)

To get the corollary

Corollary 1.6. *There exists a constant (three) round witness indistinguishable proof of knowledge (WIPOK) for every language in* **NP**.

2 Construction of Constant Round Zero Knowledge from Constant Round WI

Protocol 2.4 is our constant-round zero-knowledge protocol.

Theorem 2.2. *Protocol 2.4 is a zero-knowledge argument*[1] *system for* L*.*

[1] We call this an argument system and not a proof system since soundness holds for efficient cheating provers and not all cheating provers.

Public input: $x = (x_{i,j})$, the adjacency matrix of an undirected graph on n vertices. **Prover's auxiliary input:** w, a hamiltonian cycle in the graph x	$\begin{array}{ccc} w & x & \\ \downarrow & \downarrow & \\ \boxed{P} & & \boxed{V} \end{array}$
Step P1 (Commitment to permuted graph): Prover selects a random permutation φ on the vertices, and computes a commitment to the permuted graph. That is, prover sends the matrix of commitments $C = (C_{i,j})$ where $C_{i,j} = \mathsf{Com}(x_{\varphi(i),\varphi(j)})$.	$\xrightarrow{C = \left(\mathsf{Com}(x_{\varphi(i),\varphi(j)})\right)_{i,j\in[n]}}$
Step V2 (Send random query): The verifier selects a random bit $b \leftarrow_{\mathrm{R}} \{0,1\}$ and sends it.	$\xleftarrow{b \leftarrow_{\mathrm{R}} \{0,1\}}$
Step P3 (Open commitments): If $b = 0$ then the prover sends decommitments for all the commitments sent in Step P1 and in addition sends the permutation φ chosen in that step. Otherwise (if $b = 1$) the prover sends decommitments only for the commitments that correspond to the edges of $\varphi(w)$ (i.e., the edges of the permuted Hamiltonian cycle).	$\xrightarrow{\textit{decommitments}}$
If $b = 0$ then the verifier accepts iff all decommitments are valid and form a graph x' such that $x' = \varphi(x)$ (i.e., $x'_{i,j} = x_{\varphi(i),\varphi(j)}$ for all $i, j \in [n]$). If $b = 1$ then the verifier accepts iff the decommitments sent are for a Hamiltonian cycle and all the opened commitments are to the value 1.	

Protocol 1.2. Blum's basic protocol

Public input: 1^n: security parameter, $x \in \{0,1\}^n$ (statement to be proved is "$x \in L$") **Prover's auxiliary input:** w (a witness that $x \in L$)	$\begin{array}{cc} \overset{w}{\downarrow} & \overset{x}{\downarrow} \\ \boxed{P} & \boxed{V} \end{array}$
Step V1 (Verifier commits): Verifier chooses $r \leftarrow_{\mathrm{R}} \{0,1\}^n$ and sends $\mathsf{Com}(r)$ to prover. Denote by c the coins the verifier uses in this commitment. **Step P2 (Prover commits):** Prover chooses $r' = 0^n$, $r'' = 0^n$ and sends $\mathsf{Com}(r'), \mathsf{Com}(r'')$. Denote by c', c'' the coins the prover uses in these commitments. **Steps P,V3.x (WIPOK of commitments):** Prover proves to verifier that *either* the commitment to r' *or* the commitment to r'' was formed properly using a WIPOK. For concreteness let's say the honest prover proves the first case (and hence uses c' as a witness). Verifier accepts if proof is completed successfully.	$\xleftarrow{y = \mathsf{Com}(r;c)}$ $\xrightarrow{y' = \mathsf{Com}(r';c'), y'' = \mathsf{Com}(r'';c'')}$ $\overset{c'}{\downarrow} \quad \overset{y',y''}{\downarrow}$ $\boxed{\begin{array}{l} WIPOK \\ y' = \mathsf{Com}(r';c') \\ \textbf{or} \\ y'' = \mathsf{Com}(r'';c'') \end{array}}$ $\downarrow$ $0/1$
Step V4 (Verifier decommits): Verifier decommits by sending r,c to prover..	$\xleftarrow{r, c}$
Steps P,V5.x (Final WI-proof): Prover proves to verifier that *either* $x \in L$ *or* both $r' = r$ and $r'' = r$. Note that for the honest prover it is highly likely that r will not be equal to 0^n and hence the second case will be false. In any case the prover will use the witness for the fact that $x \in L$. Verifier accepts if proofs passes verification.	$\overset{w}{\downarrow} \quad \overset{x,r,y',y''}{\downarrow}$ $\boxed{\begin{array}{l} WIP \\ x \in L \ \textbf{or} \\ r = \mathsf{Com}^{-1}(y') = \mathsf{Com}^{-1}(y'') \end{array}}$ $\downarrow$ $0/1$

Protocol 2.1. Constant-Round Zero Knowledge

Showing that this protocol satisfies completeness is almost trivial, so to prove the theorem we need to prove the following two lemmas:

Lemma 2.3. *Let P^* be any polynomial-time cheating prover. Let V be the honest verifier. Then, the event that in an execution of Protocol 2.4 between V and P^* it holds that V accepts the first WI proof and $r = r' = r''$ happens with negligible probability.*

Once we prove Lemma 2.3 we'll get the Protocol 2.4 is sound from the soundness of the final WI system.

Proof. The proof follows by simply using the knowledge extractor of the WIPOK to recover r from $y = \mathsf{Com}(r)$ and hence break the commitment scheme. □

Lemma 2.4. *is zero knowledge.*

Proof. To prove zero knowledge, we first need to construct a simulator and then prove that the output of this simulator is indistinguishable. Our simulator will work as follows:
Simulator S:
Inputs:

- A string x (the statement we're simulating the proof of is "$x \in L$" but the simulator does not get the witness for that statement).
- Black-box access to the next-message function of the cheating verifier V^*.

Operation:

1. Ask V^* for its first message y.
2. Compute $y' = \mathsf{Com}(0^n), y'' = \mathsf{Com}(0^n)$ and feed y', y'' to V^*. Interact with V^* proving that y' or y'' are well formed by using the witness for y'.
3. If V^* does not send the decommitment to y (i.e., string r and coins c such that $y = \mathsf{Com}(r; c)$) then output the truncated interaction up to this point and halt.[2]
4. Rewind V^* to the point just after it sent y and this time repeat the process with y' and y'' two independent commitments to r (where r is the string learned in Step 3). That is, set $r' = r$ and $r'' = r$ and let $y' = \mathsf{Com}(r', c')$ and $y'' = \mathsf{Com}(r'', c'')$ for random independent coins c', c''.
5. Prove using the WIPOK that either y' or y'' are wellformed using the witness c', r' for y'.
6. If V^* does not send a decommitment to r then output the truncated execution.
7. Prove to V^* using the WIP that either $x \in L$ or $r = r' = r''$ (using of course the witness for the second option).

[2]Note that this is a valid simulation of the honest prover since the honest prover will also abort if V^* does not send a proper decommitment at this stage.

To prove the lemma we need to prove that the output of the simulator S is indistinguishable from the transcript of an interaction between V^* and the honest prover. We'll do so by constructing a sequence of intermediate hybrids.

The first hybrid will be the simulator's output while the last one will be the transcript in a real interaction. We'll show that each hybrid is indistinguishable from the preceding one. As usual, we'll do this by transforming a distinguisher between the hybrid into a distinguisher for one of the cryptographic primitives used (e.g., the commitments or the WI-proofs). When we make such a transformation the distinguisher for the primitive will have to run parts of the simulator or prover/verifier algorithms "in its belly". It is crucial to ensure when doing so that the distinguisher has indeed all the inputs necessary to do these executions. When describing each hybrid, we'll only discuss what are the *changes* in the hybrid's behavior from the previous hybrid.
Note: For starters we will analyze the simulation assuming that V^* always sends a proper decommitment to y. We'll deal with the other case later.

Hybrid 0: The simulator's output Hybrid 0 will simply be the output of the simulator S.

Hybrid 1: Modified simulator Hybrid 1 will be a modified simulator that actually uses the witness w for the statement x in the final WI. Of course the real simulator does not have access to this information, but this is a hybrid for the purposes of the proof only. The reason it is indistinguishable from Hybrid 0 is that the final proof is WI. (Note that when reducing a distinguisher between the hybrids to a distinguisher for the WI proof we are allowed to give the WI distinguisher both the witness w and all coins used in constructing the commitments y, y', y''.)

Hybrid 2: Use $r'' = 0^n$ Hybrid 2 will use $r'' = 0^n$ also in the second time it feeds the commitment y'' to V^*, instead of using $r'' = r$. That is, in Step 4 the simulator will use $y' = \mathsf{Com}(r)$ but $y'' = \mathsf{Com}(0^n)$. Note that this will not cause problems later on, since the simulator no longer uses the fact that $r = r' = r''$ in proving the final WI. The reason Hybrid 2 is indistinguishable form Hybrid 1 is the hiding property of the commitment scheme used to commit to r'' — the coins used in producing the commitment $y'' = \mathsf{Com}(r'')$ are not used anywhere else by the simulator, and so a distinguisher between Hybrid 1 and Hybrid 2 can be converted into a distinguisher breaking the hiding property of the commitment scheme.

Hybrid 3: Prove that y'' is well-formed in WIPOK Hybrid 3 will use the coins c'' and string r'' (which we set in Hybrid 2 to be equal to 0^n) used to produce $y'' = \mathsf{Com}(0^n)$ in the proof of Step 5 that wither y' or y'' are well-formed. It is indistinguishable from Hybrid 2 by the WI property of the WIPOK system - a distinguishing verifier for the WI system can be obtained from a distinguisher for these two hybrids.

Hybrid 4: Use $r' = 0^n$ Hybrid 4 will use $y' = \mathsf{Com}(0^n)$ in the commitment of Step 4. It's indistinguishable from Hybrid 3 by the security of the commitment scheme, in the same way that Hybrid 2 is indistinguishable from Hybrid 1.

Hybrid 5: Prove that y' is well-formed in WIPOK Hybrid 5 will use the coins c' and string r' (equal to 0^n) in the proof of Step 5 that either y' or y'' are well-formed. It is indistinguishable from Hybrid 4 by the Wi property of the WIPOK, for the same reason that Hybrid 3 is indistinguishable from Hybrid 4.

Hybrid 6: Skip steps 2 and 3 Note that we are no longer using the value r learned in steps 3. Thus we can skip Steps 2 and 3, and go straight to Step 4. However, then the output is exactly the distribution of a transcript with the honest prover! Note that skipping Step 3 is only allowed because we're assuming that we never halt in that step, and it is only used to learn r.

This completes the proof for the case that V^* never fails to decommit. Handling the other case is messy but can be done:

Handling a possibly aborting verifier: Let p be the probability that the verifier aborts when interacting with the honest prover. This is the same probability that the verifier aborts in Step 3 (since we feed it the same distribution until that point). Let p' be the probability that the verifier does *not* abort in Step 6. Because of the indistinguishability of the distributions we feed V^* (as we proved above) we know that $|p - p'| < n^{-\omega(1)}$. Let's assume for starters that $p' = p$. Even in this case we have a problem because the simulator will abort with probability $1 - p$ in Step 3, and conditioned on not aborting, it will abort with probability $1 - p$ on Step 6, bringing the total probability of abort to $(1 - p) + p(1 - p) = 1 - p^2$. If p is not tiny then this difference in probability is not negligible.

The idea to solve this is to have the simulator repeat steps 4 and 5 until it manages to get a valid decommitment from the verifier. The expected number of repetitions is $1/p' = 1/p$. We get that with probability $1 - p$, the simulator only gets up to Step 3, and with probability p it performs $1/p$ iterations. Thus the *expected* number of iterations is less than $1 + p \cdot 1/p$ which is still a constant. However, there are two problems with this approach:

Get only expected polynomial-time and not strict polynomial-time Even though the *expected* number of iterations is constant, we can not restrict this simulator to run in any fixed polynomial bound. The reason is that if we make the simulator stop after T iterations, then there still might be a $1/T$ bias between the output of the simulator and the transcript of a real interaction.

p' **is different than** p' The real number of expected iterations is $p \cdot 1/p'$ which may be very large even if p and p' are negligible close. Indeed, suppose that $p = p' + \epsilon$. Then we get that

$$\frac{p}{p'} = \frac{p' + \epsilon}{p'} = 1 + \frac{\epsilon}{p'}$$

For example, think of $p' = 2^{-k}$ and $\epsilon = 2^{-k/2}$. In this case the expected number of iterations of the simulator would be $2^{-k/2}$! (Note that a completely valid simulator in this case would be a simulator that always halts in Step 2, but the problem is that our simulator can not know that it is in this case.)

It turns out that the first problem is inherent: we do not know how to convert the simulator to a simulator with a strict polynomial-time bound, and in fact in a joint paper with Yehuda Lindell we proved that for *every* constant-round zero-knowledge protocol there is no black-box simulator with strict polynomial running time (in the same paper we gave a *non black box* simulator with strict polynomial running time).

The second problem can actually be solved. The idea is that we can estimate p by doing the following: if we don't halt in Step 3 for the first time, then run many iterations of Steps 2 and 3 until we get k times a non-abort response (the expected number of iterations is $p \cdot k/p = k$). By measuring how many iterations we needed, with very high probability (1- exponentially small in k) we can get an estimate $\tilde{p}$ of p up to a factor of two. In the next stage (steps 4–6) we will only allow at most $4k/\tilde{p}$ iterations. The expected number of iterations this simulator will use is

$$1 - p + p \cdot O(k)/p = O(k)$$

Now let's analyze the possible bias: in the case that we passed Step 3, the expected number of times we need to repeat Steps 4–6 is $1/p'$. The probability we need more than k/p' repetitions is negligible (exponentially small in k). Therefore if if $\tilde{p} \leq 4p'$ then the probability that we'll need more than $2k/\tilde{p}$ iterations negligible.

Otherwise, this means that $2p \geq \tilde{p} \geq 4p'$ or that $p' \leq p/2 = p - p/2$. Since $|p - p'| < n^{-\omega(1)}$, this means that $p/2$ must be negligible and hence p, the probability of not aborting is negligible. In this case, since our simulator will output a truncated execution with probability at least $1 - p$, it does not really matter what it outputs with probability p.

$\square$

3 Some related open research questions

- Is the parallel version of the Hamiltonian cycle protocol zero knowledge? (See paper by myself, Lindell and Vadhan for more on this).