# Constant-Round Zero-Knowledge Proofs of Knowledge[*]

Yehuda Lindell[†]

December 26, 2010

**Abstract**

In this note, we show the existence of *constant-round* zero-knowledge *proofs of knowledge* for all $\mathcal{NP}$. The existence of constant-round zero-knowledge proofs was proven by Goldreich and Kahan (Journal of Cryptology, 1996), and the existence of constant-round zero-knowledge *arguments* of knowledge was proven by Feige and Shamir (CRYPTO 1989). Although it is widely believed that there exist constant-round zero-knowledge proofs of knowledge, to the best of our knowledge, no proof of this fact has been published.

## 1 Introduction

In a zero-knowledge proof system, a prover and verifier interact so that the verifier is convinced of the validity of the statement being proved, but learns nothing more [10]. In a proof of knowledge, the verifier is also convinced of the fact that the prover knows a "witness" that testifies to the validity of the statement being proved [10, 1]. Zero-knowledge proofs and zero-knowledge proofs of knowledge are basic primitives in cryptography, and the fact that any statement in $\mathcal{NP}$ can be proved in zero-knowledge has made them widely applicable [9]. An important question that has been considerably studied relates to the round complexity of zero-knowledge protocols. We know the following:

- Assuming the existence of 2-round perfectly-hiding commitments (which can be constructed from families of claw-free functions), there exists a 5-round zero-knowledge proof system with negligible soundness error for all $\mathcal{NP}$. This was proven by Goldreich and Kahan [8]. (We observe that unless $\mathcal{NP} \subseteq \mathcal{MA}$, five rounds is actually the minimal number of rounds [12].)
- Assuming the existence of one-way functions, there exists a 4-round zero-knowledge *argument of knowledge* with negligible soundness error for all $\mathcal{NP}$ (an argument is a "proof" where soundness is only guaranteed computationally in the presence of a polynomial-time cheating prover). This was proven by Feige and Shamir [5, 4].

However, to the best of our knowledge, it has never been proven that there exist constant-round zero-knowledge proofs of knowledge for all $\mathcal{NP}$. In this note, we prove the following theorem:

**Theorem 1** *Assuming the existence of 2-round perfectly-hiding commitments, every $\mathcal{NP}$ relation has a zero-knowledge system for proofs of knowledge with negligible knowledge error.*

[*]This research was generously supported by the European Research Council as part of the ERC project "LAST".
[†]Dept. of Computer Science, Bar-Ilan University, Israel. email: `lindell@cs.biu.ac.il`.

**The Goldreich-Kahan proof system [8].** Our construction is very similar to that of [8], and we thus begin by describing the latter. Informally, the zero-knowledge proof of [8] works as follows:

1. The prover sends the first message of a perfectly-hiding commitment scheme.
2. The verifier commits to a query string $q$ of length $n$ using the perfectly-hiding commitment.
3. The prover begins $n$ parallel executions of the three-round proof system of [9] (or equivalently of [2]) and commits to the first prover message in each execution.
4. The verifier decommits to the query string $q$.
5. The prover concludes the proof based on the query string $q$.

The above is zero-knowledge because by rewinding, the simulator can learn the query string $q$ before it prepares the first message of the proof system. Simulation then follows from known techniques which work when the verifier-queries can be guessed or otherwise obtained ahead of time. Before proceeding, we warn that despite the fact that this strategy is intuitively appealing, and even possibly "obvious", it is highly non-trivial to analyze. Indeed, the proof by [8] that this is zero-knowledge is quite involved, and contains an important and novel proof technique. The problem that arises that makes this non-trivial is discussed at length in [8] and in our proof below. (We remark that this technique is of general importance as it turns out that this problem arises in many cryptographic settings where simulation is used.)

The reason why the protocol of [8] seems to not be a proof of knowledge is that in order to extract, one must obtain multiple different responses from the prover relative to the same first message of the proof system of [9] or [2]. However, the verifier (and thus the extractor) is bound to its query before the prover sends its commitment, and this commitment may in turn be computed as a function of the verifier's first real message (i.e., the commitment). Thus the extractor cannot change the query without the prover changing its first message.

**Our zero-knowledge proof of knowledge.** We solve the aforementioned problem in the protocol of [8] by essentially running a semi-simulatable coin-tossing protocol *in between* the first and second prover messages of the proof system of [9] or [2]. (We do not use a fully simulatable coin-tossing protocol because we need it to be constant-round and secure even if the prover is computationally unbounded. The only such known protocol [13] requires a constant-round zero-knowledge system for proofs of knowledge, which is exactly what we are trying to build.) Informally, our protocol can be described as follows:

1. The prover begins $n$ parallel executions of the proof system of [9] (or equivalently of [2]) and commits to the first prover message in each execution. The prover also sends the first message of a perfectly-hiding commitment scheme.
2. The verifier commits to a query string $q$ of length $n$ using a perfectly-hiding commitment scheme.
3. The prover commits to a string $q'$ of length $n$ using a perfectly-binding commitment scheme.
4. The verifier decommits to the query string $q$.
5. The prover decommits to $q'$ and concludes the proof based on the query string $q \oplus q'$.

The intuitive reasoning as to why this protocol is zero-knowledge is the same as for [8]. The simulator guesses ahead of time a query string $\hat{q}$ and then rewinds the verifier in order to set its $q'$ such that $q \oplus q' = \hat{q}$. However, again proving this requires the techniques of [8].

The reason why this protocol is also a proof of knowledge is that it is now possible for an extractor to rewind the prover multiple times relative to the same first message in order to obtain multiple openings with different query strings $q \oplus q'$. This enables us to apply the extraction strategy of the basic protocols of [9, 2], albeit with some additional complications.

**Semi-simulatable coin tossing.** As we have mentioned, our protocol for constant-round zero-knowledge proofs of knowledge works by having the prover and verifier jointly choose the verifier-query $q$ via a type of coin tossing protocol. In Section 3 we isolate this subprotocol and show that it achieves a level of security, that we call "semi-simulatable coin tossing". Informally speaking, this means that if $P_1$ is corrupted then the protocol is secure according to the standard ideal/real model simulation-based definitions of secure computation. Furthermore, if $P_2$ is corrupted, it is guaranteed that the output of $P_1$ is either "abort" or a uniformly distributed string. We remark that although the case of $P_2$ being corrupted is not simulatable, the fact that $P_1$ is nevertheless guaranteed to output a uniformly distributed string means that a meaningful security level is obtained. We believe that this constant-round coin-tossing protocol, which is highly efficient, is of independent interest.

**Organization.** In order to keep this note brief, we assume familiarity with the definitions of zero-knowledge and zero-knowledge proofs of knowledge; see [6, Chapter 4] for details. We prove that our protocol is a proof of knowledge using Definition 4.7.3 of [6].

## 2 Constant-Round Zero-Knowledge Proof of Knowledge

Our constant-round zero-knowledge proof of knowledge is based on $n$ parallel repetitions of the basic proof system for the *Hamiltonian Cycle* problem which is NP-complete. We therefore obtain a proof system for any language in $\mathcal{NP}$. We consider directed graphs (and the existence of directed Hamiltonian cycles). Our methodology also works for the 3-coloring protocol of [9], but it is simpler to describe it based on Hamiltonicity. See Appendix A for a full description of the basic Hamiltonicity proof system.

We use a two-round *perfectly-hiding* commitment scheme. Such a scheme can be constructed from families of claw-free functions. We denote the first message of such a scheme by $\alpha$, and a commitment to $m$ using $\alpha$ and randomness $r$ by $C^{\alpha}_{\rm ph}(m; r)$. In addition, we use a non-interactive *perfectly-binding* commitment scheme; a commitment to $m$ using randomness $r$ is denoted $C_{\rm pb}(m; r)$. Perfectly-binding commitment schemes can be constructed from 1–1 one-way functions.[1] The zero-knowledge proof of knowledge system can be found in Protocol 2.

[1]We remark that it is also possible to use the 2-round statistically binding commitment scheme of [14] which can be constructed from any one-way function. This enables us to rely on the sole assumption that there exists a two-round perfectly-hiding commitment scheme, since this implies the existence of one-way functions.

**PROTOCOL 2 (Constant-Round ZKPOK)**

- *Common Input:* a directed graph $G = (V, E)$ with $n \stackrel{\text{def}}{=} |V|$.
- *Auxiliary Input to Prover:* a directed Hamiltonian Cycle, $C \subseteq E$, in $G$.
- *The protocol:*
  1. *Prover's first step (P1):* The prover $P$ sends $n$ independent copies of the first message (BP1) for the basic proof of Hamiltonicity, described in Appendix A. In addition, $P$ sends the first message $\alpha$ of a perfectly-hiding commitment scheme.
  2. *Verifier's first step (V1):* The verifier $V$ chooses a random string $\overline{\sigma}_1 \in_R \{0,1\}^n$ and computes $c_1 = C^{\alpha}_{\text{ph}}(\overline{\sigma}_1; r_1)$ for a random $r_1$ of the appropriate length. $V$ sends $c_1$ to $P$.
  3. *Prover's second step (P2):* $P$ chooses a random string $\overline{\sigma}_2 \in_R \{0,1\}^n$ and computes $c_2 = C_{\text{pb}}(\overline{\sigma}_2; r_2)$ for a random $r_2$ of the appropriate length. $P$ sends $c_2$ to $V$.
  4. *Verifier's second step (V2):* $V$ decommits to $c_1$ by sending $\overline{\sigma}_1$ and $r_1$.
  5. *Prover's third step (P3):* If $C_{\text{ph}}(\overline{\sigma}_1; r_1) \neq c_1$, then $P$ aborts and halts. Otherwise, $P$ decommits to $c_2$ by sending $\overline{\sigma}_2$ and $r_2$. $P$ computes $\overline{\sigma} = \overline{\sigma}_1 \oplus \overline{\sigma}_2$. Denoting $\overline{\sigma} = (\sigma_1, \ldots, \sigma_n)$, $P$ sends the second message (BP2) of the basic proof of Hamiltonicity for each of the $n$ copies, based on the verifier query $\sigma_i$ in the $i$th copy.
  6. *Verifier's output:* $V$ computes $\overline{\sigma} = \overline{\sigma}_1 \oplus \overline{\sigma}_2$. If $C_{\text{pb}}(\overline{\sigma}_2; r_2) \neq c_2$ or the response of the prover is not accepting in all $n$ copies, based on the query $\sigma_i$ in the $i$th copy, then $V$ outputs REJECT. Otherwise, $V$ outputs ACCEPT.

**Theorem 3** *Assuming that $C_{\text{ph}}$ is a perfectly-hiding commitment scheme and that $C_{\text{pb}}$ is a perfectly-binding commitment scheme, Protocol 2 is a zero-knowledge proof of knowledge of a Hamiltonian cycle with knowledge error $\kappa(n) = 2^{-n}$.*

**Proof:** We begin by proving that the protocol is a proof of knowledge with knowledge error $\kappa(n) = 2^{-n}$. We use Definition 4.7.3 in [6]. We construct an extractor $K$ that works as follows:

1. $K$ invokes $P^*_{x,y,r}$, where $x$ is the common input graph, and $y$ and $r$ are the auxiliary input and random tape of $P^*$, respectively. $K$ receives the first prover message P1.
2. $K$ continues the execution to the end of the proof, running the honest verifier.
   (a) If the proof is not accepting, then $K$ outputs $\perp$ and halts.
   (b) If the proof is accepting, then $K$ rewinds $P^*_{x,y,r}$ to the beginning, and reruns the execution playing the honest verifier with fresh random coins. $K$ repeats this until another accepting proof is obtained. (Note that since $P^*_{x,y,r}$ is deterministic, the same first prover message is obtained each time.)
3. Let $\overline{\sigma}$ be the resulting query string in the first accepting transcript (where $\overline{\sigma} = \overline{\sigma}_1 \oplus \overline{\sigma}_2$), and let $\overline{\sigma}'$ be the query string in the second accepting transcript (where $\overline{\sigma}' = \overline{\sigma}'_1 \oplus \overline{\sigma}'_2$). If $\overline{\sigma} = \overline{\sigma}'$ then $K$ outputs $\perp$ and halts. Otherwise, let $i$ be such that $\sigma_i \neq \sigma'_i$. Since both transcripts are accepting, $K$ obtained responses to both query $\sigma = 0$ and $\sigma = 1$ relative to the same first prover message. Thus, $K$ can extract a Hamiltonian cycle $C \subseteq E$. $K$ outputs $C$ and halts.

It is immediate that if $K$ does not output $\perp$ then it outputs a valid Hamiltonian cycle $C$. We now claim that $K$ runs in expected polynomial time. Let $p(x,y,r)$ be the probability that $P^*_{x,y,r}$ convinces an honest verifier upon common input $x$ and $(y,r)$ as above. The important point to notice is that the probability that each iteration concludes in Step 2b is exactly $p(x,y,r)$. Thus, the expected number of required iterations is $1/p(x,y,r)$. In addition, the probability that $K$ reaches Step 2b is exactly $p(x,y,r)$. Finally, the cost of each iteration is polynomial in $n$. Thus, the expected running-time of $K$ is

$$p(x,y,r) \cdot \frac{1}{p(x,y,r)} \cdot \mathrm{poly}(n) + (1 - p(x,y,r)) \cdot \mathrm{poly}(n) = \mathrm{poly}(n).$$

It remains to show that the probability that $K$ outputs a valid cycle $C$ is at least $p(x,y,r) - 2^{-n}$. Now, $K$ outputs $\perp$ if the first execution of the proof is not accepting or if $\overline{\sigma} = \overline{\sigma}'$. The probability that the first execution of the proof is not accepting is $1 - p(x,y,r)$, by the definition of $p(x,y,r)$. Next, we claim that the probability that $K$ outputs $\perp$ due to the fact that $\overline{\sigma} = \overline{\sigma}'$ is $2^{-n}$; we denote this event by $\mathsf{collision}$ (because $\overline{\sigma}$ and $\overline{\sigma}'$ collide). Note that this event can only happen if the first execution was accepting. Thus, denoting the event that the first execution was accepting by $\mathsf{accept}_1$ we have that

$$\Pr\left[K^{P^*_{x,y,r}}(x) = \perp\right] \quad = \quad \Pr[\neg\mathsf{accept}_1] + \Pr[\mathsf{accept}_1 \ \wedge \ \mathsf{collision}]$$

Now, let $S \subseteq \{0,1\}^{n+m}$ be the set of pairs of strings $(\overline{\sigma}_1, r_1) \in \{0,1\}^{n+m}$ for which $P^*_{x,y,r}$ concludes with an accepting proof (we denote by $m = m(n)$ the length of the random string needed to commit to an $n$-bit string using $C_{\rm ph}$). We have

$$\Pr[\mathsf{accept}_1] = p(x,y,r) = \Pr_{(\overline{\sigma}_1,r_1)\in_R\{0,1\}^{n+m}}\left[(\overline{\sigma}_1, r_1) \in S\right] = \frac{|S|}{2^{n+m}}.$$

Next, observe that the event $\mathsf{collision}$ depends solely on the values $(\overline{\sigma}_1, r_1)$ used in the first execution and $(\overline{\sigma}'_1, r'_1)$ used in the second execution. In particular, the string $\overline{\sigma}_2$ chosen by $P^*_{x,y,r}$, and thus the string $\overline{\sigma} = \overline{\sigma}_1 \oplus \overline{\sigma}_2$ is a deterministic function of the commitment value $C^{\alpha}_{\rm ph}(\overline{\sigma}_1; r_1)$ computed by $V$. To be concrete, let $g$ be the (inefficient) function such that $\overline{\sigma}_2 = g(C^{\alpha}_{\rm ph}(\overline{\sigma}_1; r_1))$, where $\overline{\sigma}_2$ is the value *committed to* by $P^*_{x,y,r}$ after receiving $C^{\alpha}_{\rm ph}(\overline{\sigma}_1; r_1)$ from $V$. Using this notation, we have that $\mathsf{collision}$ is the event that

$$\overline{\sigma}_1 \oplus g\left(C^{\alpha}_{\rm ph}(\overline{\sigma}_1; r_1)\right) = \overline{\sigma}'_1 \oplus g\left(C^{\alpha}_{\rm ph}(\overline{\sigma}'_1; r'_1)\right).$$

We therefore have

$$\begin{aligned}
&\Pr[\mathsf{accept}_1 \ \wedge \ \mathsf{collision}] \\
&\quad = \Pr_{(\overline{\sigma}_1,r_1)\in\{0,1\}^{n+m},(\overline{\sigma}'_1,r'_1)\in S}\left[\mathsf{accept}_1 \ \wedge \ \left(\overline{\sigma}_1 \oplus g(C^{\alpha}_{\rm ph}(\overline{\sigma}_1; r_1)) = \overline{\sigma}'_1 \oplus g(C^{\alpha}_{\rm ph}(\overline{\sigma}'_1; r'_1))\right)\right] \\
&\quad = \Pr_{(\overline{\sigma}_1,r_1)\in\{0,1\}^{n+m},(\overline{\sigma}'_1,r'_1)\in S}\left[\overline{\sigma}_1 \oplus g(C^{\alpha}_{\rm ph}(\overline{\sigma}_1; r_1)) = \overline{\sigma}'_1 \oplus g(C^{\alpha}_{\rm ph}(\overline{\sigma}'_1; r'_1)) \mid \mathsf{accept}_1\right] \cdot p(x,y,r) \\
&\quad = \Pr_{(\overline{\sigma}_1,r_1),(\overline{\sigma}'_1,r'_1)\in S}\left[\overline{\sigma}_1 \oplus g(C^{\alpha}_{\rm ph}(\overline{\sigma}_1; r_1)) = \overline{\sigma}'_1 \oplus g(C^{\alpha}_{\rm ph}(\overline{\sigma}'_1; r'_1))\right] \cdot p(x,y,r).
\end{aligned}$$

We now prove that

$$\Pr_{(\overline{\sigma}_1,r_1),(\overline{\sigma}'_1,r'_1)\in S}\left[\overline{\sigma}_1 \oplus g(C^{\alpha}_{\rm ph}(\overline{\sigma}_1; r_1)) = \overline{\sigma}'_1 \oplus g(C^{\alpha}_{\rm ph}(\overline{\sigma}'_1; r'_1))\right] \leq \frac{1}{2^n \cdot p(x,y,r)}. \tag{1}$$

For every "commitment value" $t$ (i.e., string in the range of $C^{\alpha}_{\mathrm{ph}}$), and every value $v \in \{0,1\}^n$, define

$$S_{t,v} = \left\{(\overline{\sigma}_1, r_1) \in S \mid C^{\alpha}_{\mathrm{ph}}(\overline{\sigma}_1; r_1) = t \ \wedge \ \overline{\sigma}_1 + g(t) = v\right\}$$

to be the set of pairs $(\overline{\sigma}_1, r_1)$ such that $t = C^{\alpha}_{\mathrm{ph}}(\overline{\sigma}_1; r_1)$ and the resulting query $\overline{\sigma}$ based on $P^*_{x,y,r}$'s reply equals $v$ (i.e., $v = \overline{\sigma}_1 \oplus g(C^{\alpha}_{\mathrm{ph}}(\overline{\sigma}_1; r_1))$). Let $\mathcal{C}$ denote the range of $C^{\alpha}_{\mathrm{ph}}$ for commitments to strings of length $n$; i.e., $\mathcal{C} = \{t \mid \exists v \in \{0,1\}^n, r \in \{0,1\}^m : t = C^{\alpha}_{\mathrm{ph}}(v; r)\}$. In addition, for the sake of clarity of the equations below, we denote the perfectly-hiding commitment $C^{\alpha}_{\mathrm{ph}}$ by $C$. We have:

$$\begin{aligned}
&\Pr_{(\overline{\sigma}_1,r_1),(\overline{\sigma}'_1,r'_1)\in S}\left[\overline{\sigma}_1 + g(C(\overline{\sigma}_1; r_1)) = \overline{\sigma}'_1 + g(C(\overline{\sigma}'_1; r'_1))\right] \\
&\quad = \sum_{t,t'\in\mathcal{C}, v\in\{0,1\}^n} \Pr_{(\overline{\sigma}_1,r_1),(\overline{\sigma}'_1,r'_1)\in S}\left[C(\overline{\sigma}_1; r_1) = t \ \wedge \ C(\overline{\sigma}'_1; r'_1) = t' \ \wedge \ \overline{\sigma}_1 + g(t) = \overline{\sigma}'_1 + g(t') = v\right] \\
&\quad = \sum_{v\in\{0,1\}^n} \sum_{t,t'\in\mathcal{C}} \Pr_{(\overline{\sigma}_1,r_1)\in S}\left[C(\overline{\sigma}_1; r_1) = t \ \wedge \ \overline{\sigma}_1 + g(t) = v\right] \cdot \Pr_{(\overline{\sigma}'_1,r'_1)\in S}\left[C(\overline{\sigma}'_1; r'_1) = t' \ \wedge \ \overline{\sigma}'_1 + g(t') = v\right] \\
&\quad = \sum_{v\in\{0,1\}^n} \left(\sum_{t\in\mathcal{C}} \Pr_{(\overline{\sigma}_1,r_1)\in S}\left[C(\overline{\sigma}_1; r_1) = t \ \wedge \ \overline{\sigma}_1 + g(t) = v\right]\right) \cdot \left(\sum_{t'\in\mathcal{C}} \Pr_{(\overline{\sigma}'_1,r'_1)\in S}\left[C(\overline{\sigma}'_1; r'_1) = t' \ \wedge \ \overline{\sigma}'_1 + g(t') = v\right]\right)
\end{aligned}$$

where the second equality holds because $(\overline{\sigma}_1, r_1)$ and $(\overline{\sigma}'_1, r'_1)$ are chosen independently. Next, observe that for every $v \in \{0,1\}^n$

$$\sum_{t\in\mathcal{C}} \Pr_{(\overline{\sigma}_1,r_1)\in S}\left[C(\overline{\sigma}_1; r_1) = t \ \wedge \ \overline{\sigma}_1 + g(t) = v\right] = \sum_{t'\in\mathcal{C}} \Pr_{(\overline{\sigma}'_1,r'_1)\in S}\left[C(\overline{\sigma}'_1; r'_1) = t' \ \wedge \ \overline{\sigma}'_1 + g(t') = v\right]$$

and thus

$$\begin{aligned}
&\Pr_{(\overline{\sigma}_1,r_1),(\overline{\sigma}'_1,r'_1)\in S}\left[\overline{\sigma}_1 + g(C(\overline{\sigma}_1; r_1)) = \overline{\sigma}'_1 + g(C(\overline{\sigma}'_1; r'_1))\right] \\
&\qquad = \sum_{v\in\{0,1\}^n} \left(\sum_{t\in\mathcal{C}} \Pr_{(\overline{\sigma}_1,r_1)\in S}\left[C(\overline{\sigma}_1; r_1) = t \ \wedge \ \overline{\sigma}_1 + g(t) = v\right]\right)^2 \\
&\qquad = \sum_{v\in\{0,1\}^n} \left(\sum_{t\in\mathcal{C}} \frac{|S_{t,v}|}{|S|}\right)^2 \\
&\qquad = \sum_{v\in\{0,1\}^n} \left(\frac{\sum_{t\in\mathcal{C}} |S_{t,v}|}{|S|}\right)^2 \\
&\qquad = \sum_{v\in\{0,1\}^n} \left(\frac{|S_v|}{|S|}\right)^2 \qquad (2)
\end{aligned}$$

where the second equality follows directly from the definition of $S_{t,v}$, and for every $v \in \{0,1\}^n$, we define

$$S_v = \bigcup_{t\in\mathcal{C}} S_{t,v}.$$

Observe that by the perfect-hiding property of $C$, it holds that $S_v = S_w$ for all $v, w \in \{0,1\}^n$. In particular, this implies that for every $v \in \{0,1\}^n$, the size of $S_v$ is at most $2^m$. Now, for every $v \in \{0,1\}^n$, the value $\sum_{v\in\{0,1\}^n}(|S_v|/|S|)^2$ is maximized when some $S_v$ are of maximal size (i.e., of size $2^m$) and the others are empty. Recalling that $|S| = p(x, y, r) \cdot 2^{n+m}$, and observing that the minimum number of $v \in \{0,1\}^n$ in the sum of Eq. (2) when these are maximal is $p(x,y,r) \cdot 2^n$, we have that:

$$\begin{aligned}
\Pr_{(\overline{\sigma}_1,r_1),(\overline{\sigma}'_1,r'_1)\in S}\left[\overline{\sigma}_1 + g(C(\overline{\sigma}_1;r_1)) = \overline{\sigma}'_1 + g(C(\overline{\sigma}'_1;r'_1))\right] &= \sum_{v\in\{0,1\}^n} \left(\frac{|S_v|}{|S|}\right)^2 \\
&\leq p(x,y,r)\cdot 2^n \cdot \left(\frac{2^m}{p(x,y,r)\cdot 2^{n+m}}\right)^2 \\
&= \frac{p(x,y,r)\cdot 2^n}{(p(x,y,r)\cdot 2^n)^2} \\
&= \frac{1}{p(x,y,r)\cdot 2^n},
\end{aligned}$$

completing the proof of Eq. (1). We therefore conclude that

$$\Pr[\mathsf{accept}_1 \ \wedge \ \mathsf{collision}] \leq \frac{1}{2^n}$$

and so

$$\Pr\left[K^{P^*_{x,y,r}}(x) \neq \bot\right] = p(x,y,r) - \frac{1}{2^n}$$

as required.

***Zero-knowledge.*** Next, we prove that Protocol 2 is zero-knowledge. The proof of this fact is similar to the proof in [8]. We first present a simplified strategy for a black-box simulator $\mathcal{S}$ given oracle access to a verifier $V^*$ (with a fixed input, auxiliary input and random tape), and then explain how to modify it. The simplified simulator $\mathcal{S}$ works as follows:

1. $\mathcal{S}$ chooses a random string $\overline{\sigma} \in_R \{0,1\}^n$. Then, for the prover message in the $i$th execution, $\mathcal{S}$ generates a commitment to a random permutation of $G$ if $\sigma_i = 0$, and to a simple $n$-cycle if $\sigma_i = 1$. $\mathcal{S}$ hands $V^*$ all of the commitments. In addition, $\mathcal{S}$ chooses $\alpha$ like an honest prover would and hands it to $V^*$.

2. $\mathcal{S}$ receives from $V^*$ its commitment $c_1$. $\mathcal{S}$ chooses a random $\overline{\sigma}_2, r_2$, computes $c_2 = C_{\rm pb}(\overline{\sigma}_2; r_2)$ and hands $c_2$ to $V^*$.

3. $\mathcal{S}$ receives the decommitment $\overline{\sigma}_1, r_1$ from $V^*$. If $C^{\alpha}_{\rm ph}(\overline{\sigma}_1; r_1) \neq c_1$, then $\mathcal{S}$ simulates $P$ aborting, outputs whatever $V^*$ outputs, and halts. Otherwise, $\mathcal{S}$ proceeds to the next step.

4. *Rewinding phase: $\mathcal{S}$ rewinds $V^*$ until $\overline{\sigma}_1 \oplus \overline{\sigma}_2 = \overline{\sigma}$:*

   (a) $\mathcal{S}$ fixes $\overline{\sigma}_2 = \overline{\sigma}_1 \oplus \overline{\sigma}$, where $\overline{\sigma}$ is the string it chose initially and $\overline{\sigma}_1$ is the string that it received from $V^*$ in the decommitment.

   (b) $\mathcal{S}$ rewinds $V^*$ back to the point that it needs to send $c_2$. $\mathcal{S}$ chooses a random $r_2$ and hands $V^*$ the commitment $c_2 = C_{\rm pb}(\overline{\sigma}_2; r_2)$.

   (c) $\mathcal{S}$ receives $\overline{\sigma}'_1, r'_1$ from $V^*$.

      i. If $C^{\alpha}_{\rm ph}(\overline{\sigma}'_1; r'_1) \neq c_1$, then $\mathcal{S}$ returns back to Step 4b and repeats using fresh randomness (we stress that $\overline{\sigma}_2$ is the same each time, whereas $r_2$ is fresh).

ii. If $C^{\alpha}_{\text{ph}}(\overline{\sigma}'_1; r'_1) = c_1$ and $\overline{\sigma}'_1 \neq \overline{\sigma}_1$, then $\mathcal{S}$ outputs ambiguous and halts.

iii. Otherwise, $\mathcal{S}$ completes the proof by decommitting either to the entire graph (for $\sigma_i = 0$) or the simple cycle (for $\sigma_i = 1$).

5. $\mathcal{S}$ outputs whatever $V^*$ outputs.

The intuition behind this simulation is clear. $\mathcal{S}$ repeatedly rewinds until the query string $\overline{\sigma}$ is the one that it initially chose. In this case, it can decommit appropriately and conclude the proof. The fact that the result is computationally indistinguishable from a real proof by an honest prover follows from the hiding property of the perfectly-binding commitments.

The problem with the above simplified strategy is that $\mathcal{S}$ actually may not run in expected polynomial time. In order to see this, denote by $\epsilon(n)$ the probability that $V^*$ decommits to $c_1$ in the first iteration (before rewinding) and by $\delta(n)$ the probability that $V^*$ decommits to $c_1$ in all later iterations. We stress that although the initial commitments do not change, $\epsilon(n)$ may not equal $\delta(n)$. This is because $\epsilon(n)$ is based on the case that $c_2$ is a random commitment to a random $\overline{\sigma}_2$, whereas $\delta(n)$ is based on the case that $c_2$ is a random commitment to a *fixed* $\overline{\sigma}_2$. Nevertheless, it follows immediately from the hiding property of $C_{\text{pb}}$ that the difference between $\epsilon(n)$ and $\delta(n)$ is negligible; otherwise, one could use this fact to distinguish commitments. Now, the probability that $\mathcal{S}$ runs the rewinding phase is $\epsilon(n)$, and the expected number of rewinding iterations in the rewinding phase is $1/\delta(n)$. Let $\mu(n)$ be a negligible function, such that $\epsilon(n) - \delta(n) = \mu(n)$. We have that the expected running time of $\mathcal{S}$ is

$$(1-\epsilon(n)) \cdot \text{poly}(n) + \epsilon(n) \cdot \frac{1}{\delta(n)} \cdot \text{poly}(n) = \text{poly}'(n) \cdot \frac{\epsilon(n)}{\epsilon(n) - \mu(n)}.$$

It may be tempting at this point to conclude that the above is polynomial because $\mu(n)$ is negligible, and so $\epsilon(n) - \mu(n)$ is almost the same as $\epsilon(n)$. This is true for "large" values of $\epsilon(n)$. For example, if $\epsilon(n) > 2\mu(n)$ then $\epsilon(n) - \mu(n) > \epsilon(n)/2$. This then implies that $\epsilon(n)/(\epsilon(n) - \mu(n)) < 2$. Unfortunately, however, this is *not* true in general. For example, consider the case that $\mu(n) = 2^{-n}$ and $\epsilon(n) = \mu(n) + 2^{-n/2} = 2^{-n} + 2^{-n/2}$. Then,

$$\frac{\epsilon(n)}{\epsilon(n) - \mu(n)} = \frac{2^{-n} + 2^{-n/2}}{2^{-n/2}} = 2^{n/2} + 1,$$

which is exponential in $n$. This technical problem was observed and solved by [8], and we use their solution here.

The problem described above is solved by ensuring that the simulator $\mathcal{S}$ never runs "too long". Specifically, if $\mathcal{S}$ proceeds to the rewinding phase of the simulation, then it first estimates the value of $\epsilon(n)$. This is done by repeating Steps 2 and 3 of the simulation (choosing random $\overline{\sigma}_2$ and $r_2$ each time) until $m = m(n)$ successful decommits occurs (for a polynomial $m(n)$ to be determined below), where a successful decommit is where $V^*$ decommits to $\overline{\sigma}_1$, the string it first decommit to. We remark that as in the original strategy, if $V^*$ correctly decommits to a different $\overline{\sigma}'_1 \neq \overline{\sigma}_1$ then $\mathcal{S}$ outputs ambiguous. Then, an estimate $\tilde{\epsilon}$ of $\epsilon$ is taken to be $m/T$, where $T$ is the overall number of attempts until $m$ successful decommits occurred. As we will see, this suffices to ensure that the probability that $\tilde{\epsilon}$ is not within a constant factor of $\epsilon(n)$ is at most $2^{-n}$. We show this using the following bound, that is proven in Appendix B:

**Lemma 2.1** (Tail inequality for geometric variables [11]): *Let $X_1, \ldots, X_m$ be $m$ independent random variables with geometric distribution with probability $\epsilon$ (i.e., for every $i$, $Pr[X_i = j] = (1-\epsilon)^{j-1} \cdot \epsilon$). Let $X = \sum_{i=1}^{m} X_i$ and let $\mu = E[X] = m/\epsilon$. Then, for every $\Delta$,*

$$\Pr[X \geq (1+\Delta)\mu] \leq e^{-\frac{m\Delta^2}{2(1+\Delta)}}.$$

Define $X_i$ to be the random variable that equals the number of attempts needed to obtain the $i$th successful decommitment (not including the attempts up until the $i-1$th successful decommitment), and let $\Delta = \pm 1/2$. Clearly, each $X_i$ has a geometric distribution with probability $\epsilon$. It therefore follows that

$$\Pr\left[X \leq \frac{m}{2\epsilon} \vee X \geq \frac{3m}{2\epsilon}\right] \leq 2 \cdot \Pr\left[X \geq \frac{3}{2} \cdot \frac{m}{\epsilon}\right] \leq 2 \cdot e^{-\frac{m}{12}}$$

Stated in words, the probability that the *estimate* $\tilde{\epsilon} = m/X$ is not between $2\epsilon/3$ and $2\epsilon$ is at most $2e^{-m/12}$. Thus, if $m(x) = 12n$ it follows that the probability that $\tilde{\epsilon}$ is not within the above bounds is at most $2^{-n}$, as required.

Next, $\mathcal{S}$ repeats the following rewinding phase up to $n$ times: $\mathcal{S}$ runs the rewinding phase in Step 4 of the simulation. However, $\mathcal{S}$ limits the number of rewinding attempts to $n/\tilde{\epsilon}$ iterations. We have the following cases:

1. If within $n/\tilde{\epsilon}$ rewinding iterations, $\mathcal{S}$ obtains a successful decommitment from $V^*$ to $\overline{\sigma}_1$, then it completes the proof as described. It can do so in this case because the query string is $\overline{\sigma}$ as required.

2. If $\mathcal{S}$ obtains a valid decommitment to some $\overline{\sigma}'_1 \neq \overline{\sigma}$ then it outputs ambiguous.

3. If $\mathcal{S}$ does not obtain any correct decommitment within this time, then $\mathcal{S}$ aborts this attempted rewinding phase.

As mentioned, the above phase is repeated up to $n$ times, each time using independent coins. If the simulator $\mathcal{S}$ doesn't successfully conclude in any of the $n$ attempts, then it halts and outputs fail. We will show that this strategy ensures that the probability that $\mathcal{S}$ outputs fail is negligible.

In addition to the above, $\mathcal{S}$ keeps a count of its overall running time and if it reaches $2^n$ steps, then it halts, outputting fail. (This additional time-out is needed to ensure that $\mathcal{S}$ does not run too long in the case that the estimate $\tilde{\epsilon}$ is not within a constant factor of $\epsilon(n)$. Recall that this "bad event" can only happen with probability $2^{-n}$.)

We first claim that $\mathcal{S}$ runs in expected polynomial-time.

**Claim 2.2** *Simulator $\mathcal{S}$ runs in expected-time that is polynomial in $n$.*

**Proof:** Observe that in the first and all later iterations, all of $\mathcal{S}$'s work takes a strict polynomial-time number of steps. We therefore need to bound only the number of rewinding iterations. Before proceeding, however, we stress that rewinding iterations only take place if $V^*$ provides a valid decommitment in the first place. Thus, all rewinding only occur with probability $\epsilon(n)$.

Now, $\mathcal{S}$ first rewinds in order to obtain an estimate $\tilde{\epsilon}$ of $\epsilon(n)$. This involves repeating until $m(n) = 12n$ successful decommitments are obtained. Therefore, the expected number of repetitions in order to obtain $\tilde{\epsilon}$ equals exactly $12n/\epsilon(n)$ (since the expected number of trials for a single success is $1/\epsilon(n)$). After the estimate $\tilde{\epsilon}$ has been obtained, $\mathcal{S}$ runs the rewinding phase of Step 4 for a maximum of $n$ times, in each phase limiting the number of rewinding attempts to $n/\tilde{\epsilon}$.

Given the above, we are ready to compute the expected running-time of $\mathcal{S}$. In order to do this, we differentiate between two cases. In the first case, we consider what happens if $\tilde{\epsilon}$ is *not* within a constant factor of $\epsilon(n)$. The only thing we can say about $\mathcal{S}$'s running-time in this case is that it is bound by $2^n$ (since this is an overall bound on its running-time). However, since this event happens with probability at most $2^{-n}$, this case adds only a polynomial number of steps to the overall expected running-time. We now consider the second case, where $\tilde{\epsilon}$ *is* within a constant factor of $\epsilon(n)$ and thus $\epsilon(n)/\tilde{\epsilon} = O(1)$. In this case, we can bound the expected running-time of $\mathcal{S}$ by

$$\text{poly}(n) \cdot \epsilon(n) \cdot \left(\frac{12n}{\epsilon(n)} + n \cdot \frac{n}{\tilde{\epsilon}}\right) = \text{poly}(n) \cdot \frac{\epsilon(n)}{\tilde{\epsilon}} = \text{poly}(n)$$

and this concludes the analysis. ■

Next, we prove that the probability that $\mathcal{S}$ outputs fail is negligible.

**Claim 2.3** *The probability that $\mathcal{S}$ outputs* fail *is negligible in $n$.*

**Proof:** Notice that the probability that $\mathcal{S}$ outputs fail is less than or equal to the probability that it does not obtain a successful decommitment in any of the $n$ rewinding phase attempts *plus* the probability that it runs for $2^n$ steps.

We first claim that the probability that $\mathcal{S}$ runs for $2^n$ steps is negligible. We have already shown in Claim 2.2, that $\mathcal{S}$ runs in expected polynomial-time. Therefore, the probability that an execution will deviate so far from its expectation and run for $2^n$ steps is negligible. (It is enough to use Markov's inequality to establish this fact.)

We now continue by considering the probability that in all $n$ rewinding phase attempts, $\mathcal{S}$ does not obtain a successful decommitment within $n/\tilde{\epsilon}$ steps. Consider the following two possible cases (recall that $\epsilon(n)$ equals the probability that $V^*$ decommits when $\overline{\sigma}_2$ is random, and $\mu$ is the negligible difference between $\epsilon(n)$ and $\delta(n)$, the probability that $V^*$ decommits when $\overline{\sigma}_2$ is fixed:

1. *Case 1:* $\epsilon(n) \leq 2\mu(n)$: In this case, $V^*$ decommits to $c_1$ with only negligible probability. This means that the probability that $\mathcal{S}$ even reaches the rewinding phase is negligible. Thus, $\mathcal{S}$ only outputs fail with negligible probability.

2. *Case 2:* $\epsilon(n) > 2\mu(n)$: Recall that $V^*$ successfully decommits in any iteration with probability $\delta(n) = \epsilon(n) - \mu(n)$. Thus, the expected number of iterations needed until $V^*$ successfully decommits is $1/(\epsilon(n) - \mu(n))$. Now, since in this case $\epsilon(n) > 2\mu(n)$ we have that $\mu(n) < \epsilon(n)/2$ and so the expected number of rewinding attempts required to obtain a successful decommitment to $\overline{\sigma}_1$ is less than $2/\epsilon(n)$.

   Assuming that $\tilde{\epsilon}$ is within a constant factor of $\epsilon(n)$, we have that the expected number of rewindings in any given rewinding attempt is bound by $O(1/\tilde{\epsilon})$. Therefore, by Markov's inequality, the probability that $\mathcal{S}$ tries more than $n/\tilde{\epsilon}$ iterations in any given rewinding phase attempt is at most $O(1/n)$. It follows that the probability that $\mathcal{S}$ tries more than this number of iterations in $n$ independent rewinding phases is negligible in $n$ (specifically, it is bound by $O(1/n)^n$). This covers the case that $\tilde{\epsilon}$ is within a constant factor of $\epsilon(n)$. However, the probability that $\tilde{\epsilon}$ is *not* within a constant factor of $\epsilon(n)$ is also negligible.

Putting the above together, we have that $\mathcal{S}$ outputs $\mathsf{fail}$ with negligible probability only. ■

Next, we prove the following:

**Claim 2.4** *The probability that $\mathcal{S}$ outputs* $\mathsf{ambiguous}$ *is negligible in $n$.*

**Proof:** The proof of this claim is *identical* to the proof of this fact in [8]. Intuitively, if there exists an infinite series of inputs $x$ for which $\mathcal{S}$ outputs $\mathsf{ambiguous}$ with non-negligible probability, then this can be used to break the computational binding of the $C_{\mathrm{ph}}$ commitment scheme. The only subtlety is that $\mathcal{S}$ runs in expected polynomial-time, whereas an attacker for the binding of the commitment scheme must run in strict polynomial-time. Nevertheless, this can be overcome by simply truncating $\mathcal{S}$ to twice its expected running time. By Markov's inequality, this reduces the success probability of the binding attack by at most 1/2, and so this is still non-negligible. ■

It remain to prove that the output distribution generated by $\mathcal{S}$ is computationally indistinguishable from the output of $V^*$ in a real proof with an honest prover. We have already shown that $\mathcal{S}$ outputs $\mathsf{fail}$ or $\mathsf{ambiguous}$ with only negligible probability. Thus, the only difference between the output distribution generated by $\mathcal{S}$ and the output distribution generated in a real proof is that in the case that $\overline{\sigma}_i = 0$ the unopened commitments in the simulated transcript are all to 0, and not to the rest of the graph apart from the cycle. The indistinguishability of this is therefore reduced to the hiding property of the perfectly binding (and computationally hiding) commitment scheme. Once again, the proof of this reduction is *identical* to the proof of this fact in [8] and so the details are omitted. This completes the proof. ■

**Reducing the knowledge error to zero.** As shown in [1], if it is possible to find a valid witness to the statement being proved in time $\mathrm{poly}(n)/\kappa(n)$ and it is possible to detect when the extractor "fails" (i.e., in our case, outputs $\perp$ because of the event "$\mathsf{accept}_1 \wedge \mathsf{collision}$"), then the knowledge error of a proof of knowledge can be reduced to 0. This is achieved by running the existing knowledge extractor, and in the case of such a failure, finding a valid witness to the statement being proved and outputting it. For the case of Hamiltonicity, it is possible to naively find a cycle in time $n!$. Thus, in order to reduce the knowledge error to zero using this procedure, we simply need to run the basic Hamiltonicity protocol $n \log n$ times in parallel, instead of just $n$ times in parallel (note that $\log n! < n \log n$). Alternatively, using dynamic programming it is possible to find a Hamiltonian cycle in time and space $\mathrm{poly}(n) \cdot 2^n$. This yields an expected polynomial-time and polynomial-space machine. If this suffices, then $n$ parallel repetitions suffice.

## 3 Semi-Simulatable Coin Tossing

In our protocol for constant-round zero-knowledge proofs of knowledge, the verifier's query $\overline{\sigma}$ is essentially chosen via a type of coin-tossing protocol. See Protocol 4 for a description of this coin-tossing protocol; recall that $C^{\alpha}_{\mathrm{ph}}(x; r)$ is a perfectly-hiding commitment to $x$ using the receiver-message $\alpha$, and $C_{\mathrm{pb}}$ is a perfectly-binding commitment scheme. For the sake of this section, we assume familiarity with the definitions of secure two-party computation; see [7, Chapter 7] and [3].

Protocol 4 has 5 rounds of communication and is highly efficient; in particular, it does not use zero-knowledge proofs or arguments, as does the constant-round coin tossing protocol of [13]. In addition, the proof of the zero-knowledge property in Theorem 3 demonstrates that in the case that $P_1$ (who is the verifier in Protocol 2) is corrupted, it is possible to prove security under

**PROTOCOL 4 (Semi-Simulatable Coin Tossing)**

- *Common Input:* a security parameter $1^n$ and a length parameter $\ell$.
- *The protocol:*
  1. *Party $P_2$'s first step:* $P_2$ sends $P_1$ the receiver-message $\alpha$ of the perfectly-hiding commitment scheme.
  2. *Party $P_1$'s first step:* $P_1$ chooses a random string $x \in_R \{0,1\}^\ell$ and computes $c_1 = C^{\alpha}_{\rm ph}(x; r)$ for a random $r$ of the appropriate length. $P_1$ sends $c_1$ to $P_2$.
  3. *Party $P_2$'s first step:* $P_2$ chooses a random string $y \in_R \{0,1\}^\ell$ and computes $c_2 = C_{\rm pb}(y; s)$ for a random $r_2$ of the appropriate length. $P_2$ sends $c_2$ to $P_1$.
  4. *Party $P_1$'s second step:* $P_1$ decommits to $c_1$ by sending $x$ and $r$.
  5. *Party $P_2$'s second step:* If $C^{\alpha}_{\rm ph}(x; r) \neq c_1$, then $P_2$ outputs $\perp$ and halts. Otherwise, $P_2$ decommits to $c_2$ by sending $y$ and $s$.
  6. *Outputs:*
     (a) $P_1$ checks that $C_{\rm pb}(y; s) = c_2$. If not, it outputs $\perp$. Otherwise, it outputs $x \oplus y$.
     (b) $P_2$ outputs $x \oplus y$.

the standard simulation-based definitions of [3]. This is because the simulation strategy for the proof of zero-knowledge works by the simulator first choosing the query string $\overline{\sigma}$ at random and then obtaining an execution in which the resulting query is $\overline{\sigma}$ (with probability that is negligibly close to the probability that the verifier doesn't abort). Thus, the same strategy works when the simulator receives a random string $R$ from the trusted party and must generate an execution in which $x \oplus y = R$, with probability that is negligibly close to the probability that $P_1$ does not cause $P_2$ to abort. In addition, observe that when $P_2$ is corrupted, the output of $P_1$ from the coin-tossing protocol is *uniformly distributed* or $\perp$. This holds because $P_1$ commits to $x$ using a perfectly-hiding commitment and $P_2$ commits to $y$ using a perfectly-binding commitment. Thus, $P_2$ is fully committed to $y$ before it knows anything (in an information-theoretic sense) about $x$. This implies that $x \oplus y$ is uniformly distributed and so a corrupted $P_2$ can either cause $P_1$ to output this uniform string, or to abort and output $\perp$. We call this level of security "semi-simulatable coin tossing", and define it below. In the following definition, we refer to the coin-tossing functionality $f$ defined by $(1^\ell, 1^\ell) \mapsto (U_\ell, U_\ell)$, meaning that each party inputs the length $1^\ell$ and receives as output the same uniformly-distributed $\ell$-bit string. In addition, we denote by $\mathsf{output}_1(\mathrm{REAL}_{\pi,\mathcal{A}(z)}(1^\ell, 1^\ell, n))$ the output of party $P_1$ after interacting with adversary $\mathcal{A}$ in the protocol $\pi$, with inputs $1^\ell$ and security parameter $n$.

**Definition 5** *A protocol $\pi = (P_1, P_2)$ is a* semi-simulatable coin-tossing protocol *if the following holds:*

1. *For every non-uniform probabilistic polynomial-time adversary $\mathcal{A}$ controlling $P_1$ in the real model, there exists a non-uniform probabilistic polynomial-time adversary/simulator $\mathcal{S}$ for the ideal model such that*

$$\left\{\mathrm{IDEAL}_{f,\mathcal{S}(z)}(1^\ell, 1^\ell, n)\right\}_{z\in\{0,1\}^*;\ell,n\in\mathbb{N}} \stackrel{\rm c}{\equiv} \left\{\mathrm{REAL}_{\pi,\mathcal{A}(z)}(1^\ell, 1^\ell, n)\right\}_{z\in\{0,1\}^*;\ell,n\in\mathbb{N}}.$$

2. *For every non-uniform probabilistic polynomial-time adversary $\mathcal{A}$ controlling $P_2$, and for every $R_1, R_2 \in \{0,1\}^\ell$, it holds that:*

$$\Pr\left[\mathsf{output}_1(\text{REAL}_{\pi,\mathcal{A}(z)}(1^\ell,1^\ell,n)) \in \{R_1,\perp\}\right] = \Pr\left[\mathsf{output}_1(\text{REAL}_{\pi,\mathcal{A}(z)}(1^\ell,1^\ell,n)) \in \{R_2,\perp\}\right].$$

Based on the above discussion, we obtain the following theorem:

**Theorem 6** *If $C_{\rm ph}$ is a two-round perfectly-hiding commitment scheme and $C_{\rm pb}$ is a perfectly-binding commitment scheme, then Protocol 4 is a 5-round semi-simulatable coin-tossing protocol.*

## Acknowledgements

I would like to sincerely thank Oded Goldreich for encouraging me to write up this work, and for providing a proof of Eq. (1). I would also like to thank Eran Omri for helpful discussions.

## References

[1] M. Bellare and O. Goldreich. On Defining Proofs of Knowledge. In *CRYPTO'92*, Springer-Verlag (LNCS 740), pages 390–420, 1992.

[2] M. Blum. How to Prove a Theorem So No One Else Can Claim It. *Proceedings of the International Congress of Mathematicians*, pages 1444–1451, USA.

[3] R. Canetti. Security and Composition of Multiparty Cryptographic Protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[4] U. Feige. *Alternative Models for Zero Knowledge Interactive Proofs.* Ph.D. Thesis, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, 1990. Available from `http://www.wisdom.weizmann.ac.il/~feige`.

[5] U. Feige and A. Shamir. Zero-Knowledge Proofs of Knowledge in Two Rounds. In *CRYPTO'89*, Springer-Verlag (LNCS 435), pages 526–544, 1989.

[6] O. Goldreich. *Foundations of Cryptography: Volume 1 – Basic Tools.* Cambridge University Press, 2001.

[7] O. Goldreich. *Foundations of Cryptography: Volume 2 – Basic Applications.* Cambridge University Press, 2004.

[8] O. Goldreich and A. Kahan. How To Construct Constant-Round Zero-Knowledge Proof Systems for NP. *Journal of Cryptology*, 9(3):167–190, 1996.

[9] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *Journal of the ACM*, 38(1):691–729, 1991.

[10] S. Goldwasser, S. Micali and C. Rackoff The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing,* 18(1):186–208, 1989.

[11] S. Har-Peled. Lecture Notes on Approximation Algorithms in Geometry, Chapter 27, Excercise 27.5.3, 2010. Currently found at `http://valis.cs.uiuc.edu/∼sariel/teach/notes/aprx/`.

[12] J. Katz. Which Languages Have 4-Round Zero-Knowledge Proofs? In the 5*th TCC*, Springer (LNCS 4948), pages 73–88, 2008.

[13] Y. Lindell. Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation. *Journal of Cryptology*, 16(3):143–184, 2003.

[14] M. Naor. Bit Commitment using Pseudorandom Generators. *Journal of Cryptology*, 4(2):151–158, 1991.

## A Blum's Zero-Knowledge Proof for Hamiltonicity [2]

**PROTOCOL 7 (Basic proof system for Hamiltonicity)**

- *Common Input:* a directed graph $G = (V, E)$ with $n \overset{\text{def}}{=} |V|$.
- *Auxiliary Input to Prover:* a directed Hamiltonian Cycle, $C \subseteq E$, in $G$.
- *Prover's first step (BP1):* The prover selects a random permutation $\pi$ over the vertices $V$, and commits (using a perfectly-binding commitment scheme) to the entries of the adjacency matrix of the resulting permuted graph. That is, it sends an $n$-by-$n$ matrix of commitments so that the $(\pi(i), \pi(j))^{\text{th}}$ entry is a commitment to 1 if $(i, j) \in E$, and is a commitment to 0 otherwise.
- *Verifier's first step (BV1):* The verifier uniformly selects $\sigma \in \{0, 1\}$ and sends it to the prover.
- *Prover's second step (BP2):*
  - If $\sigma = 0$, the prover sends $\pi$ to the verifier and decommits to all of the commitments in the adjacency matrix.
  - If $\sigma = 1$, the prover decommits to the commitments of entries $(\pi(i), \pi(j))$ for which $(i, j) \in C$ (and only to these commitments).
- *Verifier's second step (BV2):*
  - If $\sigma = 0$, the verifier checks that the revealed graph is indeed isomorphic, via $\pi$, to $G$.
  - If $\sigma = 1$, the verifier checks that all revealed values are 1 and that the corresponding entries form a simple $n$-cycle. In both cases the verifier checks that the decommitments are proper (i.e., that they fits the corresponding commitments). The verifier accepts if and only if the corresponding condition holds.

## B A Proof of Lemma 2.1

**Lemma B.1** (Tail inequality for geometric variables [11] – Lemma 2.1 restated): *Let $X_1, \ldots, X_m$ be $m$ independent random variables with geometric distribution with probability $\epsilon$ (i.e., for every $i$, $Pr[X_i = j] = (1-\epsilon)^{j-1} \cdot \epsilon$). Let $X = \sum_{i=1}^m X_i$ and let $\mu = E[X] = m/\epsilon$. Then, for every $\Delta$,*

$$\Pr[X \geq (1+\Delta)\mu] \leq e^{-\frac{m\Delta^2}{2(1+\Delta)}}$$

**Proof:** In order to prove this lemma, we define a new random variable $Y_\alpha$ for any $\alpha \in \mathbb{N}$ as follows. Consider an infinite series of independent Bernoulli trials with probability $\epsilon$ (i.e., the probability of any given trial being 1 is $\epsilon$). Then, write the results of these trials as a binary string and let $Y_\alpha$ be the number of *ones* appearing in the prefix of length $\alpha$. It is clear that

$$\mu_\alpha = E[Y_\alpha] = \alpha \cdot \epsilon \ .$$

Furthermore,

$$\Pr[X \geq (1+\Delta)\mu] = \Pr[Y_\alpha < m]$$

for $\alpha = (1+\Delta)\mu$. In order to see why this holds, observe that one can describe the random variable $X = \sum_{i=1}^m X_i$ by writing an infinite series of Bernoulli trials with probability $\epsilon$ (as above), and then taking $X$ to be the index of the $m$th one to appear in the string. Looking at it in this way, we have that $X$ is greater than or equal to $(1+\Delta)\mu$ if and only if $Y_{(1+\Delta)\mu} < m$ (because if $Y_{(1+\Delta)\mu} < m$ then this means that $m$ successful trials have not yet been obtained). Observe now that $\mu_\alpha = \alpha \cdot \epsilon$, $\alpha = (1+\Delta)\mu$, and $\mu = m/\epsilon$. Thus, $\mu_\alpha = (1+\Delta) \cdot m$. This implies that

$$\left(1 - \frac{\Delta}{1+\Delta}\right) \cdot \mu_\alpha = \left(1 - \frac{\Delta}{1+\Delta}\right) \cdot (1+\Delta) \cdot m = (1+\Delta) \cdot m - \Delta \cdot m = m \ ,$$

and so

$$\Pr[Y_\alpha < m] = \Pr\left[Y_\alpha < \left(1 - \frac{\Delta}{1+\Delta}\right) \cdot \mu_\alpha\right].$$

Applying the Chernoff bound[2], we have that

$$\Pr\left[Y_\alpha < m\right] = \Pr\left[Y_\alpha < \left(1 - \frac{\Delta}{1+\Delta}\right)\mu_\alpha\right] < e^{-\frac{\mu_\alpha}{2}\cdot\left(\frac{\Delta}{1+\Delta}\right)^2}$$

Once again using the fact that $\mu_\alpha = (1+\Delta) \cdot m$ we conclude that

$$\Pr[X \geq (1+\Delta)\mu] = \Pr\left[Y_\alpha < m\right] < e^{-\frac{(1+\Delta)m}{2}\cdot\left(\frac{\Delta}{1+\Delta}\right)^2} = e^{-\frac{m\Delta^2}{2(1+\Delta)}}$$

as required. ■

[2]We use the following version of the Chernoff bound. Let $X_1, \ldots, X_m$ be independent Bernoulli trials where $\Pr[X_i = 1] = \epsilon$ for every $i$, and let $X = \sum_{i=1}^m X_i$ and $\mu = E[X] = m\epsilon$. Then, for every $\Delta$ it holds that $\Pr[X < (1-\beta)\mu] < e^{-\frac{\mu}{2}\cdot\beta^2}$.