Achieving Constant Round Leakage-Resilient Zero-Knowledge*

Omkant Pandey

omkant@uiuc.edu
University of Illinois at Urbana-Champaign

Abstract. Recently there has been a huge emphasis on constructing cryptographic protocols that maintain their security guarantees even in the presence of side channel attacks. Such attacks exploit the physical characteristics of a cryptographic device to learn useful information about the internal state of the device. Designing protocols that deliver meaningful security even in the presence of such leakage attacks is a challenging task.

The recent work of Garg, Jain, and Sahai formulates a meaningful notion of zero-knowledge in presence of leakage; and provides a construction which satisfies a weaker variant of this notion called $(1+\epsilon)$-leakage-resilient-zero-knowledge, for every constant $\epsilon > 0$. In this weaker variant, roughly speaking, if the verifier learns ℓ bits of leakage during the interaction, then the simulator is allowed to access $(1+\epsilon)\cdot\ell$ bits of leakage. The round complexity of their protocol is $\lceil \frac{n}{\epsilon} \rceil$.

In this work, we present the first construction of leakage-resilient zero-knowledge satisfying the ideal requirement of $\epsilon = 0$. While our focus is on a feasibility result for $\epsilon = 0$, our construction also enjoys a constant number of rounds. At the heart of our construction is a new "public-coin preamble" which allows the simulator to recover arbitrary information from a (cheating) verifier in a "straight line." We use non-black-box simulation techniques to accomplish this goal.

1 Introduction

The concept of zero-knowledge interactive proofs, originating in the seminal work of Goldwasser, Micali, and Rackoff [39], is a fundamental concept in theoretical cryptography. Informally speaking, a zero-knowledge proof allows a prover P to prove an assertion x to a verifier V such that V learns "nothing more" beyond the validity of x. The proof is an interactive and randomized process. To formulate "nothing more," the definition of zero-knowledge requires that for every malicious V^* attempting to lean more from the proof, there exists a polynomial time simulator S which on input *only* x, simulates a "real looking" interaction for V^*.

In formulating the zero-knowledge requirement, it is assumed that the prover P is able to keep its internal state — the witness and the random coins — perfectly hidden from the verifier V^*. It has been observed, however, that this

* IACR Eprint Archive Report 2012/362

assumption may not hold in many settings where an adversary has the ability to perform *side channel attacks*. These attacks enable the adversary to learn useful information about the internal state of a cryptographic device (see e.g., [48, 6, 68, 59] and the references therein). In presence of such attacks, standard cryptographic primitives often fail to deliver any meaningful notion of security. As a matter of fact, even *formulating* a meaningful security notion under such attacks—as is the case with leakage-resilient zero-knowledge—can be a challenging task.

To deliver meaningful security in the presence side channel attacks, many recent works consider stronger adversarial models in which the device implementing the honest algorithm *leaks* information about its internal state to the adversary. The goal of these works is then to construct cryptographic primitives that are "resilient" to such leakage. Leakage resilient constructions for many basic cryptographic tasks are now known [29, 3, 64, 26, 4, 5, 57, 47, 15, 25, 24, 50, 30, 49, 14, 2].

Leakage-resilient zero-knowledge. Very recently Garg, Jain, and Sahai [33] initiated an investigation of leakage-resilient zero-knowledge (LRZK). Their notion considers a cheating verifier V^* who can learn an arbitrary amount of leakage on the internal state of the honest prover, including the witness. This is formulated by allowing V^* to make leakage queries $F_1, F_2, \ldots$ throughout the execution of the protocol. Then the definition of LRZK, roughly speaking, captures the intuition that no such V^* learns anything *beyond the validity of the assertion* and *the leakage*.

The actual formulation of this intuition is slightly more involved. Observe that during the simulation, S will need to answer leakage queries of V^*, which may contain information about the witness to V^*. Simulator S cannot answer such queries without having access to the witness. The definition of [33] therefore provides S with access to a *leakage oracle* which holds a witness to x. The oracle, $\mathcal{L}^n_w(\cdot)$, is parameterized by the witness w and $n = |x|$; on input a function F expressed as a boolean circuit, it returns $F(w)$. To ensure that S can answer leakage requests of V^*, the simulator is also allowed to query $\mathcal{L}^n_w$ on leakage functions of its choice. Of course, providing S with uncontrolled access to the witness will render the notion meaningless.[1] Therefore, to ensure that the notion delivers meaningful security, the LRZK definition requires the following restriction on the length of bits that S can read from $\mathcal{L}^n_w$. Suppose that $S^{\mathcal{L}^n_w}$ outputs a simulated view v for V^*. Denote by $\ell_S(v)$ the number of bits S reads from $\mathcal{L}^n_w$ in generating this particular view v. Denote by $\ell_{V^*}(v)$ the total length of the leakage answers that S provides to V^* (which are already included in v, and can be different from answers received by S). Then, it is required that:

$$\ell_S(v) \leq \ell_{V^*}(v). \tag{1}$$

[1] S can simply access the entire witness, and then simulate.

More precisely, in [33], a slightly more general notion of $(1+\epsilon)$-LRZK is defined in which the above condition is relaxed to:

$$\ell_S(\upsilon) \leq (1+\epsilon) \cdot \ell_{V^*}(\upsilon),$$

where $\epsilon > 0$ is a constant. In addition, [33] also present a protocol of $\lceil \frac{n}{\epsilon} \rceil$ rounds, which achieves $(1+\epsilon)$-LRZK for every a-priori fixed constant $\epsilon > 0$. Since $\epsilon > 0$, the resulting notion is weaker than the one required by equation 1. Nevertheless, [33] show that despite this relaxation, $(1+\epsilon)$-LRZK still delivers meaningful security. By applying this notion in the context of cryptography based on hardware-tokens, [33] were able to weaken the requirements of tamper-proofness on the hardware tokens.

Our main result. Although a protocol with $\epsilon > 0$ is still useful certain contexts, it is significantly weaker than the ideal requirement of $\epsilon = 0$—both qualitatively and philosophically. Qualitatively, a constant $\epsilon > 0$ allows the simulator to learn *strictly more* information about the internal secret than the actual leakage allows! Qualitatively, it means that a protocol proven to be $(1+\epsilon)$-LRZK "secure" may actually expose additional parts of the internal secret than the actual leakage. Furthermore, even in situations where $(1+\epsilon)$-LRZK is sufficient, protocol of [33] requires a large round complexity, which continues to increase as we lower the value of ϵ.

Philosophically, an $\epsilon > 0$ essentially defies the very nature of simulation-based security. In particular, as argued above, since it allows S to learn strictly more than what the verifier does, it is not "zero" knowledge, but only an "ϵ-approximation" of it, and closer in spirit to super-polynomial time simulation [61, 67, 65]. Furthermore, this is not merely a philosophical issue—$(1+\epsilon)$-LRZK can be particularly problematic in protocol composition [16, 17]. For example, using such a simulator in place of a cheating party may result in learning more "outputs" than allowed.

In this work, we present the first construction of an LRZK protocol satisfying $\epsilon = 0$. Although our main goal is to obtain a feasibility result, our protocol also enjoys a *constant* number of rounds. Our protocol uses standard cryptographic tools. However, it requires some of them – particularly, oblivious transfer – to have an *invertible sampling* property [20, 42]. To the best of our knowledge, instantiations satisfying this property are known only based on the decisional Diffie-Hellman assumption (DDH) [23]. We leave constructing an LRZK proof system based on general assumption as an interesting open question.

Theorem 1 (Main Result). *Suppose that the decision Diffie-Hellman assumption holds. Then, there exists a constant-round leakage-resilient zero-knowledge proof system for all languages in $\mathcal{NP}$.*

We remark that the low round-complexity is usually a desirable protocol feature [37, 7, 69, 66, 70]. In the context of side channel attacks, however, it can be a particularly attractive one to have. This is because a protocol with high round complexity may require the device to maintain state for more rounds,

and therefore may give an adversary more opportunities to launch side-channel attacks.

1.1 An Overview of Our Approach

Let us start by recalling the main difficulty in constructing an LRZK protocol. Recall that a zero-knowledge simulator S "cheats" in the execution of the protocol so that it can produce a convincing view. When dealing with leakage, not only the simulator needs to continue executing the protocol, but it also needs to "explain its actions" so far by maintaining a state consistent with an honest prover.

To be able to simultaneously perform these two actions, the GJS simulator does the following. It combines the following two different but well-known methods of "cheating." The first method, due to Goldreich-Kahan [37], requires the verifier to commit its challenge ch; the second method, due to Feige-Shamir [31], requires the prover to use equivocal[2] commitments. The GJS simulator uses these methods *together*. It uses ch to perform its main simulation (by using [37] strategy), and uses the trapdoor of equivocal commitments, denoted t_1, to "explain its actions" so far. We call (t_1, ch) the double trapdoor.

The GJS simulator "rewinds" the verifier to obtain the two trapdoors before it actually enters the main proof stage. By using a precise rewinding strategy [53], GJS achieves $(1+\epsilon)$-LRZK. However, since rewinding strategy is crucial to their simulation, this approach by itself seems insufficient for achieving LRZK.

A fundamentally different simulation strategy, in which the simulator uses the program of the malicious verifier V^*, was presented in Barak's work [7]. This method does not need to "rewind" the verifier to produce its output. Our first idea there is to somehow use Barak's simulation strategy along with the use of equivocal commitments as in [31]. Unfortunately, this does not work since the trapdoor t_1 for equivocation has to be somehow recovered and only then any other simulation strategy (such as knowing the challenge ch) can be used.

We therefore modify this approach so that we can use Barak's method to recover arbitrary information from the verifier during the simulation. For the purpose of this discussion, let us assume that Barak's technique provides a way for P and V to interactively generate a statement σ for some $\mathcal{NP}$-relation $\mathbf{R}_{sim}$ so that no cheating prover P^* can prove $\sigma \in \mathbf{L}_{sim}$, but a simulator S holding the program of the cheating verifier V^* will always have a witness ω such that $\mathbf{R}_{sim}(\sigma, \omega) = 1$. At this point, let us just assume that the verifier does not ask leakage queries.

Then, we need to design a two party protocol for the following task. The first party P holds a private input ω, the second party V holds an arbitrary private message m, the common input to both parties is σ. The protocol allows P to learn m if and only if $\mathbf{R}_{sim}(\sigma, \omega) = 1$, nothing otherwise; V learns nothing. This is similar in spirit to the *conditional disclosure* primitive of [34], except that

[2] These are commitments which, given appropriate trapdoor information, can be opened to both 0 or 1.

here the condition is an arbitrary $\mathcal{NP}$-relation $\mathbf{R}_{sim}(\sigma, \omega) = 1$. Constructing such protocols for $\mathcal{NP}$-conditions has not been studied, since they follow from work on secure two-party computation [71, 35, 54]. Clearly, we cannot directly use secure two-party computation since their security-guarantee is often *simulation-based*—which is essentially what our protocol is trying to achieve in the first place.

Our next observation is that we do not really require the strong simulation-based guarantee. We only need to construct a conditional disclosure protocol for a very specific $\mathcal{NP}$-relation. We construct such a protocol based on Yao's garbled circuit technique. We show that if we use properly chosen OT protocols (constructed in [1, 56]) — then we get a conditional disclosure protocol. In addition, the protocol ensures that the messages of P are pseudorandom (more precisely, invertible samplable [20, 42]). As a result, the protocol maintains its security claims even in the presence of leakage. This is a two-round protocol, and a crucial ingredient in achieving leakage resilience.

Armed with this new tool, simulation now seems straightforward: use the conditional disclosure protocol to recover both (t_1, ch) and then use the GJS-simulator. While this idea works, there is a difficulty in proving the *soundness* of this protocol. Recall that in Barak's protocol, one must find collisions in the hash function h to prove that no cheating P^* can succeed in learning a witness to statement σ. Typically, this is achieved by applying "rewinding techniques" to extract knowledge P^*. However, ensuring this typical requires the simulator to demonstrate "knowledge"—which is difficult to "explain" later when leakage queries are asked by the cheating prover.

To resolve this difficulty, we need to ensure that *extraction* can be performed from a party *without requiring it to maintain knowledge*.[3] We ensure this by using a variant of the commitment protocol of Barak and Lindell [11]. We use this protocol to extract useful information directly from Barak's preamble [7], without requiring the honest party to maintain knowledge explicitly.

Recall that we work in the model of [33]. In this model the verifier is allowed to ask arbitrary leakage queries $F_1, F_2, \ldots$ on prover's state. The state of the prover at any given round only consists of its witness and the randomness up to that round. In particular, the randomness of future rounds is determined only at the beginning of those round. Observe that all ingredients described by us so far actually require the prover to send only random strings. Therefore, it is easy to asnswer the leakage queries up to this point in the simulation. By the time simulator enters the main body, it recovers (t_1, ch) and use them to answer leakage queries as in [33].

1.2 Related Work

Relevant to our work are the works on zero-knowledge proofs in other more complex attack models such as man-in-middle attacks [27], concurrent attacks [28], resettable attacks [19, 8], and so on. Also relevant to our work are different

[3] Indeed, there is a difference between the two, see discussion in [11].

variants of non-black-box simulation used in the literature [7, 9, 62, 63, 22] as well as efficient and universal arguments [46, 52, 10, 43].

The explicit study of leakage-resilient cryptography was started by Dziembowski and Pietrzak [29]. Related study on protecting devices appears in the works of Ishai, Prabhakaran, Sahai, and Wagner [45, 44]. After these works a long line of research has focussed on constructing primitives resilient to leakage including public-key encryption and signatures [3, 64, 26, 4, 5, 57, 47, 15, 25, 24, 49, 14, 50], devices [30, 2], and very recently interactive protocols [33, 12].

Also relevant to our work are the works on adaptive security [18] and invertible sampling [20, 42]. Adaptively secure protocols and leakage-resilience in interactive protocols were shown to be tightly connected in the work of Bitansky, Canetti, and Halevi [12].

2 Notation and Definitions

Notation. For a randomized algorithm A we write $A(x; r)$ the process of evaluating A on input x with random coins r. We write $A(x)$ the process of sampling a uniform r and then evaluating $A(x; r)$. We define $A(x, y; r)$ and $A(x, y)$ analogously. The set of natural numbers is represented by $\mathbb{N}$. Unless specified otherwise, $n \in \mathbb{N}$ represents a security parameter available as an implicit input when necessary. All inputs are assumed to be of length at most polynomial in n. We assume familiarity with standard concepts such as interactive Turing machines (ITM), computational indistinguishability, commitment schemes, $\mathcal{NP}$-languages, witness relations and so on (see [36]).

For two randomized ITMs A and B, we denote by $[A(x, y) \leftrightarrow B(x, z)]$ the interactive computation between A and B, with A's inputs (x, y) and B's inputs (x, z), and uniform randomness; and $[A(x, y; r_A) \leftrightarrow B(x, z; r_B)]$ when we wish to specify randomness. We denote by $\text{VIEW}_B[A(x, y) \leftrightarrow B(x, z)]$ and $\text{OUT}_B[A(x, y) \leftrightarrow B(x, z)]$ the view and output of B in this computation; VIEW_A, OUT_A are defined analogously. Finally, $\text{TRANS}[A(x, y) \leftrightarrow B(x, z)]$ denotes the public transcript of the interaction $[A(x, y) \leftrightarrow B(x, z)]$.

For two probability distributions D_1 and D_2, we write $D_1 \stackrel{\text{c}}{\equiv} D_2$ to mean that D_1 and D_2 are computationally indistinguishable.

Definition 1 (Interactive Proofs) *A pair of probabilistic polynomial time interactive Turing machines $\langle P, V \rangle$ is called an interactive proof system for a language $L \in \mathcal{NP}$ with witness relation R if the following two conditions with respect to some negligible function* $\text{negl}(\cdot)$*:*

- *Completeness: for every $x \in L$, and every witness w such that $R(x, w) = 1$,*

$$\Pr\left[\text{OUT}_V[P(x, w) \leftrightarrow V(x)] = 1\right] \geq 1 - \text{negl}(|x|).$$

- *Soundness: for every $x \notin L$, every interactive Turing machine P^*, and every $y \in \{0, 1\}^*$,*

$$\Pr\left[\text{OUT}_V[P^*(x, y) \leftrightarrow V(x)] = 1\right] \leq \text{negl}(|x|).$$

If the soundness condition holds only against polynomial time machines P^*, $\langle P, V\rangle$ is called an argument system. We will only need/construct argument systems in this work.

Leakage attack. Machine P and V are modeled as randomized ITM which interact in rounds. It is assumed that the the random coins used by a party in any particular round are determined only at the beginning of that round. Denote by `state` a variable initialized to prover's private input w. At the end beginning of each round i, P flips coins r_i to be used for that round, and updates $\texttt{state} := \texttt{state} \parallel r_i$. A leakage query on prover's state in round i corresponds to verifier sending a function F_i (represented as a polynomial-sized circuit), to which the prover responds with $F_i(\texttt{state})$. The verifier is allowed to any number of arbitrary leakage queries throughout the interaction. A malicious verifier who obtains leakage under this setting is said to be launching a *leakage attack*.

To formulate zero-knowledge under a leakage attack, we consider a PPT machine S called the simulator, which receives access to an oracle $\mathcal{L}_w^n(\cdot)$. $\mathcal{L}_w^n(\cdot)$ is called the leakage oracle, and parametrized by the witness w and the security parameter n. A query to the leakage oracle consists of an efficiently computable function F, to which the oracle responds with $F(w)$. The leakage-resilient zero-knowledge is defined by requiring that the output of S be computationally indistinguishable from the real view; in addition the length of all bits read by S from $\mathcal{L}_w^n$ in producing a particular view v is at most the length of leakage answers contained in the v.

For $x \in L$, w such that $R(x, w) = 1$, $z \in \{0,1\}^*$, and randomness $r \in \{0,1\}^*$ defining the output $v = S^{\mathcal{L}_w^n(\cdot)}(x, z; r)$, we let the function $\ell_S(v, r)$ denote the number of bits that S receives from $\mathcal{L}_w^n(\cdot)$ in generating view v with randomness r. Further, we let the function $\ell_{V^*}(v)$ denote the total length of leakage answers that V^* receives in the output v. By convention, randomness r will be included in the notation only when we need to be explicit about it.

Definition 2 (Leakage-resilient Zero-knowledge) *We say that an interactive proof system $\langle P, V\rangle$ for a language $L \in \mathcal{NP}$ with a witness relation R, is leakage-resilient zero-knowledge if for every probabilistic polynomial time machine V^* launching a* leakage attack *on P, there exists a probabilistic polynomial time machine S such that the following two conditions hold:*

1. *For every $x \in L$, every w such that $R(x, w) = 1$, and every $z \in \{0,1\}^*$, distributions $\text{VIEW}_{V^*}[P(x, w) \leftrightarrow V^*(x, z)]$ and $S^{\mathcal{L}_w^n(\cdot)}(x, z)$ are computationally indistinguishable.*
2. *For every $x \in L$, every w such that $R(x, w) = 1$, every $z \in \{0,1\}^*$, and every sufficiently long $r \in \{0,1\}^*$ defining the output $v = S^{\mathcal{L}_w^n(\cdot)}(x, z; r)$, it holds that $\ell_S(v, r) \leq \ell_{V^*}(v)$.*

The definition of standard zero-knowledge is obtained by removing condition 2, and enforcing that no leakage queries are allowed to any machine.

3 Cryptographic Tools

We start by recalling some standard cryptographic tools and two-party protocols. Looking ahead, we will require that our protocols satisfy the following important property. For a specific party (chosen depending upon the protocol), all messages sent by this party be pseudorandom strings. In some cases where this is not possible, it will be sufficient if the messages are pseudorandom elements of group (e.g., a prime-order subgroup of $\mathbb{Z}_p^*$ for a (safe) prime p of length n).[4] We will provide necessary details when appropriate.

Statistically-binding commitments. We use Naor's scheme [55], based on a pseudorandom generator (prg). Recall that in this scheme, first the receiver sends a random string τ of length $3n$; to commit to bit b, the sender selects a uniform seed s of length n and sends y such that if $b = 0$ then $y = \mathsf{prg}(s)$, otherwise $y = \tau \oplus \mathsf{prg}(s)$. This scheme is statistically binding; in addition, *sender's message is pseudorandom*. A string can be committed by committing bitwise, and it suffices to use same τ for all the bits. We write $\mathsf{sbcom}_\tau(m; s)$ to represent sender's string, when receiver's first message is τ.

Statistically-hiding commitments. We use a statistically hiding commitment scheme as well. We require the receiver of this scheme to be *public coin*. Such schemes are known, including a two-round string commitment scheme, based on collision-resistant hash functions (crhf) [58, 41, 21]. We write $\mathsf{shcom}_\rho(m; s)$ to denote sender's commitment string, when receiver's first message is ρ. Without loss of generality, $|\rho| = n$.

Zero-knowledge proofs. We use a statistical zero-knowledge argument-of-knowledge (szkaok) protocol for proving $\mathcal{NP}$-statements. We require the verifier of this protocol to be *public coin*. Such protocols are known to exist; including a constant-round protocol based on crhf [7, 10, 63], and a $\omega(1)$-round protocol based on statistically-hiding commitments [38, 13].

We choose the constant-round protocol of Pass and Rosen, denoted Π_{PR}, as our candidate szkaok. Let S_{PR} denote the corresponding simulator for Π_{PR}. We remark that S_{PR} is a "straight-line" simulator, with strict polynomial running time.

3.1 Oblivious Transfer

We will use a *two-round* oblivious transfer protocol $\mathsf{OT} := \langle S_{\text{OT}}, R_{\text{OT}} \rangle$. For the choice bit b of the receiver, we denote by $\{R_{\text{OT}}(1^n, b)\}_{n \in \mathbb{N}}$ the message sent by R_{OT} on input $(1^n, b)$.

[4] This will be sufficient since public sampling from such a group admits *invertible sampling* [20, 42]. However, it is more convenient to directly work with the assumption that algorithms can receive random elements in such a group as part of their random tape.

Let p, q be primes such that $p = 2q + 1$ and $|p| = n$. Then, we require the OT protocol to satisfy the following requirement. There exists a randomized PPT algorithm R_{OT}^{pub} such that for every $n \in \mathbb{N}$, every $b \in \{0, 1\}$, and every safe prime $p = 2q + 1$, the following two conditions hold:

1. $R_{\mathrm{OT}}(1^n, 0) \stackrel{\mathrm{c}}{\equiv} R_{\mathrm{OT}}^{pub}(1^n, p)$
2. The output of $R_{\mathrm{OT}}^{pub}(1^n, p)$ consists of components $\{\alpha_i\}_{i=1}^{\mathrm{poly}(n)}$ such that every α_i is a uniform and independent element in an order q subgroup of $\mathbb{Z}_p^*$.

We can formulate the second requirement by simply requiring the output to contain independent and random bits. The difficulty is that we do not know any OT protocol that would satisfy such a requirement. We therefore choose the above formulation. Note that without loss of generality, we can assume that uniform and independent elements can be provided as part of the random tape.[5] We will call algorithm R_{OT}^{pub} the "fake" receiver algorithm.

Concrete instantiation: The existence of R_{OT}^{pub} is extremely crucial for our construction. Unfortunately, no OT protocol satisfying this requirement are known to exist based on *general* assumptions. However, two round OT protocols of [56, 1] based on the DDH assumption, do satisfy both of our requirements. For concreteness, we fix the Naor-Pinkas oblivious transfer (protocol 4.1 in [56]) as our choice, and denote it by $\mathsf{OT}_{\mathrm{NP}}$. Algorithm R_{OT}^{pub} in this protocol consists of sending random and independent elements in order q subgroup of $\mathbb{Z}_p^*$. When multiple secrets must be exchanged we simply repeat this protocol in parallel.

Security of OT. In terms of security, the protocols in [56, 1] are secure against malicious adversaries. However, they do not satisfy the usual simulation based (i.e., "ideal/real") security. Instead, they satisfy the following (informally stated) security notions:

1. *Indistinguishability for receiver:* it ensures that $\{R_{\mathrm{OT}}(1^n, 0)\}_{n\in\mathbb{N}} \stackrel{\mathrm{c}}{\equiv} \{R_{\mathrm{OT}}(1^n, 1)\}_{n\in\mathbb{N}}$, where $\{R_{\mathrm{OT}}(1^n, b)\}_{n\in\mathbb{N}}$ denotes the message sent by honest receiver on input $(1^n, b)$.
2. *Statistical secrecy for sender:* it ensures either $\{S_{\mathrm{OT}}(1^n, m_0, m_1, q)\}_{n\in\mathbb{N}} \stackrel{\mathrm{s}}{\equiv} \{S_{\mathrm{OT}}(1^n, m_0, m')\}_{n\in\mathbb{N}}$ or $\{S_{\mathrm{OT}}(1^n, m_0, m_1, q)\}_{n\in\mathbb{N}} \stackrel{\mathrm{s}}{\equiv} \{S_{\mathrm{OT}}(1^n, m', m_1)\}_{n\in\mathbb{N}}$, where m' is an arbitrary message and $S_{\mathrm{OT}}(1^n, m_0, m_1, q)$ denotes the message sent by the honest sender on input $(1^n, m_0, m_1)$ when receiver's message is q.

This type of security notion is sufficient for our purpose. A formal description, following [40], is given in the full version of this work [60].

[5] This assumption is easily removed by requiring an invertible sampling algorithm for R_{OT}^{pub}, which are known to exist. Also, the two-round requirement is not essential and can be relaxed.

3.2 Extractable Commitments

We will need a perfectly-binding commitment scheme which satisfies the following two properties. First, if a cheating committer C^* successfully completes the protocol, then there exists an extractor algorithm E which outputs the value committed by C^* during the commit stage. Second, there exists a public-coin algorithm C_{pub} such that no cheating receiver can tell if it is interacting with C_{pub} or the honest committing algorithm C. Algorithm C_{pub} is essentially the "fake" committing algorithm for C (much like the fake receiver R^p_{OT} define above). Let us first define these properties.

Commit-with-extract. We will actually need a slightly property than mere extraction, called *commit-with-extract* [11, 51]. Informally, commit-with-extract requires that for every cheating C^*, there exists an extractor E which first simulates the view of the cheating *committer* in an execution with honest receiver; further, if the view is accepting then it also outputs the value committed to in this view. Our specific use requires that the quality of simulation be *statistical*.

Definition 3 (Commit-with-extract [11]) *Let $n \in \mathbb{N}$ be the security parameter. A perfectly-binding commitment scheme $\Pi_{com} := \langle \mathsf{C}, \mathsf{R} \rangle$ is a* commit-with-extract *scheme if the following holds: there exists a strict* PPT *commitment-extractor E such that for every* PPT *committer C^*, for every $m \in \{0,1\}^n$, every (advice) $z \in \{0,1\}^*$ and every $r \in \{0,1\}^*$, upon input (C^*, m, z, r), machine E outputs a pair, denoted $(E_1(\mathsf{C}^*, m, z, r), E_2(\mathsf{C}^*, m, z, r))$, such that the following conditions hold:*

1. $E_1(\mathsf{C}^*, m, z, r) \stackrel{\text{s}}{\equiv} \text{VIEW}_{\mathsf{C}^*}[\mathsf{C}^*(m, z; r) \leftrightarrow \mathsf{R}()]$
2. $\Pr[E_2(\mathsf{C}^*, m, z, r)] = \mathsf{value}(E_1(\mathsf{C}^*, m, z, r)) \geq 1 - \text{negl}(n)$

where $\mathsf{value}(\cdot)$ *is a deterministic function which outputs either the* unique *value committed to in the view $E_1(\mathsf{C}^*, m, z, r)$, or $\perp$ if no such value exists.*

We say that a perfectly binding commitment scheme Π_{com} admits *public decommitment* if there exists a deterministic polynomial time algorithm D_{com} which on input the *public transcript* of interaction $\widehat{m}$, and the decommitment information d, outputs the *unique* value m committed in $\widehat{m}$. If there is no such value, the algorithm outputs $\perp$. For perfectly binding commitment schemes, the function value is well defined on the public transcripts as well. Therefore, we can write $D_{com}(d, \widehat{m}) = \mathsf{value}(\widehat{m})$.

We now specify our "fake" public-coin sender requirement. Since we are working with DDH based construction, we will use a safe prime $p = 2q + 1$ of length n, (as used in R^{pub}_{OT}).

Let $n \in \mathbb{N}$ be the security parameter. We say that a perfectly binding commitment scheme $\Pi_{com} := \langle \mathsf{C}, \mathsf{R} \rangle$ has a *fake public-coin sender* if there exists an algorithm C_{pub} such that for every malicious PPT R^*, every $m \in \{0,1\}^n$, every safe prime p of length n, every advice $z \in \{0,1\}^*$, the following two conditions hold:

1. $\text{VIEW}_{\mathsf{R}^*}[\mathsf{C}(m) \leftrightarrow \mathsf{R}^*(z)] \stackrel{c}{\equiv} \text{VIEW}_{\mathsf{R}^*}[\mathsf{C}_{pub}(p) \leftrightarrow \mathsf{R}^*(z)]$
2. The output of $\mathsf{C}_{pub}(p)$ consists of components $\{\alpha_i\}_{i=1}^{\text{poly}(n)}$ such that for every i: α_i is a uniform and independent element either in $\{0,1\}$ or in an order q subgroup of $\mathbb{Z}_p^*$.

Concrete instantiation. Unfortunately, no commitment protocol satisfying these requirements is known. The central reason behind this is that the fake public-coin sender C_{pub} requirement interferes with the commit-with-extract requirement. In [11], Barak and Lindell constructed a commitment protocol with the goal of *strict polynomial time* extraction. We observe that somewhat surprisingly, with some very minor changes, this protocol actually satisfies all our requirements. In particular, this commitment scheme is a *commit-with-extract* scheme, has a *fake public-coin sender*, and admits *public decommitment*. However, as with the OT protocol, this change requires us to use ElGamal [32] and hence DDH (instead of a general trapdoor permutation). For completeness, we present the protocol of [11] and explain the required modifications in the full version of this work [60].

Important notation. For concreteness, fix $\Pi_{com} := \langle \mathsf{C}, \mathsf{R} \rangle$ to be a specific commitment protocol satisfying all three conditions above, and let D_{com} denote it's public decommitment algorithm. Let $\mathbf{L}_{com} := \{(m, \widehat{m}) : \exists d \text{ s.t. } D_{com}(\widehat{m}, d) = m\}$. That is, $\mathbf{L}_{com}$ is an $\mathcal{NP}$-language containing statements $(m, \widehat{m})$ such that $\widehat{m}$ is a commitment-transcript for value m. Let $\mathbf{R}_{com}$ be the corresponding $\mathcal{NP}$-relation so that $\mathbf{R}_{com}((m, \widehat{m}), d) = 1$ if $D_{com}(\widehat{m}, d) = m$ and 0 otherwise.

3.3 Barak's Preamble

In this section, we will recall Barak's non-black-box simulation method. In addition, we will make a slight change to this protocol which requires us to reprove some of the claims. We start by recalling Barak's relation for the complexity class $\mathbf{NTIME}(n^{\log\log(n)})$.

Barak's relation. Let $n \in \mathbb{N}$ be the security parameter, and $\{\mathcal{H}_n\}_n$ be a family of crhf, $h : \{0,1\}^* \to \{0,1\}^n$. Since we are using Naor's commitment scheme, we will have an extra string τ for the commitment scheme sbcom. Barak's relation, $\mathbf{R}_{\text{B}}$ takes as input an instance of the form $\langle h, \tau, c, r \rangle \in \{0,1\}^n \times \{0,1\}^{3n} \times \{0,1\}^{3n^2} \times \{0,1\}^{n+n^2}$ and a witness of the form $\langle M, y, s \rangle \in \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^{\text{poly}(n)}$.

Relation: $\mathbf{R}_{\text{B}}(\langle h, \tau, c, r \rangle, \langle M, y, s \rangle) = 1$ if and only if:
1. $|y| \leq |r| - n$.
2. $c = \mathsf{sbcom}_\tau(h(M); s)$.
3. $M(c, y) = r$ within $n^{\log\log n}$ steps.

Let $\mathbf{L}_{\text{B}}$ be the language corresponding to $\mathbf{R}_{\text{B}}$. We use this more complex version involving y, since it will allow us to successfully simulate even in the

presence of leakage queries, which a cheating verifier obtains during the protocol execution.[6]

Universal arguments and statement generation. Universal arguments (UARG) are four-round public-coin interactive argument systems [46, 52, 10], which can be used to prove statements in $\mathbf{L}_{\mathrm{B}}$. Let $\langle P_{\mathrm{UA}}, V_{\mathrm{UA}} \rangle$ be such a system. We will denote the four rounds of this UARG by $\langle \alpha, \beta, \gamma, \delta \rangle$. Consider the following protocol between a party P_{B} and a party V_{B}.

Protocol GENSTAT: Let $\{\mathcal{H}_n\}_n$ be a family of crhf functions.
1. V_{B} sends $h \leftarrow \mathcal{H}_n$ and $\tau \leftarrow \{0,1\}^{3n}$
2. P_{B} sends $c \leftarrow \{0,1\}^{3n^2}$
3. V_{B} sends $r \leftarrow \{0,1\}^{n+n^2}$.

Note that that length of r is $n^2 + n$ which allows y to be of length at most n^2. Length of c is $3n^2$ since it is supposed to be a commitment to n bits. We have the following lemma:[7]

Composed protocol $\langle P^{\otimes}, V^{\otimes} \rangle$. We define this for convenience. The composed protocol is simply the GENSTAT protocol followed by an universal argument that the transcript $\sigma := \langle h, \tau, c, r \rangle$ is in $\mathbf{R}_{\mathrm{B}}$. More precisely, strategy $P^{\otimes} := P_{\mathrm{B}} \odot P_{\mathrm{UA}}$ is the composed prover, and $V^{\otimes} := V_{\mathrm{B}} \odot V_{\mathrm{UA}}$ is the composed verifier, where $A \odot B$ denotes the process of running ITM A first, and then continuing ITM B from then onwards.[8] The following lemma states that the composed verifier $V^{\otimes}$ almost always rejects in an interaction with any PPT prover (i.e., it always rejects that $\sigma \in \mathbf{L}_{\mathrm{B}}$).

Lemma 1 ([7]) *Suppose that* $\{\mathcal{H}_n\}_n$ *is a family of* crhf *functions. There exists a negligible function* negl *such that for every* PPT *strategy* P^**, every* $z \in \{0,1\}^*$*, every* $r \in \{0,1\}^*$*, and every sufficiently large* n*,*

$$\Pr\left[\mathrm{OUT}_{V^{\otimes}}[P^*(z;r) \leftrightarrow V^{\otimes}()]\right] \leq \mathrm{negl}(n)$$

where the probability is taken over the randomness of $V^{\otimes}$*.*

The "encrypted" version. In Barak's protocol, an "encrypted" version of the above protocol is used in which the honest prover sends commitments to its UARG-messages (instead of the messages themselves). This is possible to do since the verifier is public coin.

[6] This relation is identical to the one used for constructing bounded concurrent zero-knowledge in constant rounds in [7].

[7] The version of Barak's relation that we use is actually a somewhat simplified form of the relation given in [10], which results only in a reduction to hash functions that are crhf against circuits of size $n^{\log n}$. By using the more complex version of [10], we get a reduction to standard crhf, without affecting any of our claims.

[8] A and B do not share states and run with their own independent inputs.

We will use our commit-with-extract scheme $\Pi_{com} := \langle \mathsf{C}, \mathsf{R} \rangle$ for this purpose.[9] Recall that for Π_{com}, there exists a fake public-coin sender algorithm C_{pub} whose execution is indistinguishable from that of C. During the commitment phase, our prover algorithm will follow instructions of C_{pub}; the verifier will continue to use the normal receiver strategy R.

"Encrypted" preamble. $\langle \widehat{P}_{\text{B}}, \widehat{V}_{\text{B}} \rangle$: Let $\{\mathcal{H}_n\}_n$ be a family of crhf functions.

1. $\widehat{P}_{\text{B}}$ and $\widehat{V}_{\text{B}}$ run the GenStat protocol.
   Let $\langle h, \tau, c, r \rangle$ denote the resulting statement.
2. $\widehat{P}_{\text{B}}$ and $\widehat{V}_{\text{B}}$ execute uarg for the statement $\langle h, \tau, c, r \rangle$.
   (a) $\widehat{V}_{\text{B}}$ sends α, obtained from V_{UA}.
   (b) $\widehat{P}_{\text{B}}$ runs C_{pub}, and $\widehat{V}_{\text{B}}$ runs R;
   Let $\widehat{\beta}$ be the commitment transcript.
   (c) $\widehat{V}_{\text{B}}$ sends γ, obtained from V_{UA}.
   (d) $\widehat{P}_{\text{B}}$ runs C_{pub}, and $\widehat{V}_{\text{B}}$ runs R;
   Let $\widehat{\delta}$ be the commitment transcript.

The full transcript of the preamble is $\langle h, \tau, c, r, \alpha, \widehat{\beta}, \gamma, \widehat{\delta} \rangle$.

Since the prover messages are committed, we cannot make a claim along the lines of lemma 1. Therefore, we define the following $\mathcal{NP}$-relation $\mathbf{R}_{sim}$ and claim that it is a "hard" relation. This relation simply tests that there exist valid decommitments (d_1, d_2) for strings $\widehat{\beta}, \widehat{\delta}$ so that the transcript is accepted by the uarg verifier.

Relation: $\mathbf{R}_{sim}(\langle h, \tau, c, r, \alpha, \widehat{\beta}, \gamma, \widehat{\delta} \rangle, \langle \beta, d_1, \delta, d_2 \rangle) = 1$ if and only if:

1. $\mathbf{R}_{com}(\langle \beta, \widehat{\beta} \rangle, d_1) = 1$.
2. $\mathbf{R}_{com}(\langle \delta, \widehat{\delta} \rangle, d_2) = 1$.
3. $V_{\text{UA}}(h, \tau, c, r, \alpha, \beta, \gamma, \delta) = 1$.

The language corresponding to relation $\mathbf{R}_{sim}$ is denoted by $\mathbf{L}_{sim}$. Also note that $\widehat{P}_{\text{B}}$ sends either random strings of uniform elements in a prime order group of $\mathbb{Z}_p^*$. The proof of the following lemma appears in the full version of this work [60].

Lemma 2 *Suppose that $\{\mathcal{H}_n\}_n$ is a family of* crhf *functions. There exists a negligible function* negl *such that for every* ppt *strategy P^*, every $z \in \{0,1\}^*$, every $r \in \{0,1\}^*$, and every sufficiently large n,*

$$\Pr\left[\sigma \leftarrow \text{TRANS}[P^*(z;r) \leftrightarrow \widehat{V}_{\text{B}}()]; \sigma \in \mathbf{L}_{sim}\right] \leq \text{negl}(n)$$

[9] Recall that Π_{com} is perfectly-binding commitment scheme which satisfies the commit-with-extract notion. In addition, the protocol has a public decommitment algorithm D_{com}, an associated $\mathcal{NP}$-relation $\mathbf{R}_{com}$, and $\mathcal{NP}$-language $\mathbf{L}_{com}$, and a fake public-coin sender algorithm. See section 3.2.

where the probability is taken over the randomness of $\widehat{V}_{\text{B}}$.

4 Conditional Disclosure via Garbled Circuits

Yao's garbled circuit method [72] allows two parties to compute any arbitrary function f of their inputs in a "secure" manner. Without loss of generality, let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$.

The method. The garbled circuit method specifies two polynomial time algorithms $(\mathsf{Garble}, \mathsf{Eval})$. Algorithm Garble is randomized; on input 1^n and the description of a circuit (that computes) f, it outputs a triplet $(\mathcal{C}, \mathsf{key}_0, \mathsf{key}_1)$. $\mathcal{C}$, which consists of a set of tables containing encrypted values, is called the *garbled circuit*; and $\mathsf{key}_0 = \{(k^0_{0,i}, k^1_{0,i})\}_{i=1}^n$ and $\mathsf{key}_1 = \{(k^0_{1,i}), k^1_{1,i}\}_{i=1}^n$ are called the keys. Let $a = a_1, \ldots, a_n$ and $b = b_1, \ldots, b_n$ be binary strings. Algorithm Eval, on input $(\mathcal{C}, K_a, K_b)$ outputs a value $v \in \{0,1\}^n$ such that if $K_a = \{k^{a_i}_{0,i}\}$ and $K_b = \{k^{b_i}_{1,i}\}$ then $v = f(a,b)$.

For an $\mathcal{NP}$-relation R, and $\sigma \in \{0,1\}^*$, let $f_{\sigma,R}$ be the following function.

> **Function** $f_{\sigma,R}(\omega, m)$:
> If $R(\sigma, \omega) = 1$, `output` m; otherwise `output` $0^{|m|}$.

That is, $f_{\sigma,R}$ discloses m if and only if ω is a valid witness for the statement σ. We will use the garbled circuit method for such functions $f_{\sigma,R}$. Jumping ahead, we will use $f_{\sigma,\mathbf{R}_{sim}}$ for the $\mathcal{NP}$-relation $\mathbf{R}_{sim}$ described in section 3.3.

Conditional disclosure via garbled-circuits. In the two party setting, one party prepares the garbled circuit $\mathcal{C}$ and sends the keys K_b corresponding to her input b to the other party. An OT protocol is used by the first party to receive keys K_a for her input a, so that it can execute the evaluation algorithm. This allows the receiver of the garbled circuit (and OT) to learn $f(a,b)$ but "nothing more". In addition, receiver's input remains secure due to OT-security for receiver.

Looking forward, we will require our protocol so that it will admit a "fake" receiver algorithm. Therefore, we will use the Naor-Pinkas OT protocol, denoted OT_{NP} (see section 3.1). For a technical reason, our protocol starts by first executing steps of OT_{NP}, and then executes the garbled circuit step. Note that the first step involves n parallel executions of OT, one for each input bit. The resulting two-round conditional disclosure protocol, Π_{cd}, is as follows.

Protocol Π_{cd} for computing $f_{\sigma,R}(\omega, m)$: The protocol consists of two parties, a receiver R_{cd} and a sender S_{cd}. R_{cd}'s private input is bit string $\omega = \omega_1, \ldots, \omega_n$, and S_{cd}'s private input is bit string $m = m_1, \ldots, m_n$. The common input to the parties is the description of the function $f_{\sigma,R}$ as a circuit (equivalently, just σ).

1. R_{cd} computes $v = (v_1, \ldots, v_n)$, where v_i is the first message of OT_{NP} using the input ω_i and fresh randomness for $i \in [n]$. It then sends v.

2. S_{cd} prepares a garbled circuit for the function $f_{\sigma,R}$: $(\mathcal{C}_{\sigma,R}, \mathsf{key}_0, \mathsf{key}_1) \leftarrow \mathsf{Garble}(f_{\sigma,R})$. Next, S_{cd} prepares $v' = (v'_1, \ldots, v'_n)$ where v'_i is the second message of OT_{NP} computed using $(k^0_{0,i}, k^1_{0,i})$ as sender's input and v_i as receiver's first message. Here the keys $(k^0_{0,i}, k^1_{0,i})$ are the i^{th} component of key_0. Finally, let K_m denote the keys taken from key_1 corresponding to m. S_{cd} sends $(\mathcal{C}_{\sigma,R}, v', K_m)$.

Recall that OT_{NP} is a two-round protocol, it provides statistical secrecy for the sender, and has a fake public-coin receiver. Also recall that OT_{NP} does not satisfy the the standard simulation-based security. As a result, we cannot directly use known results about the security of Yao's protocol. Nevertheless, we can make weaker indistinguishability-style claims which suffice for our purpose. First notice that the OT-security for receiver, intuitively guarantees indistinguishability for the input of R_{cd}. For the sender, we can prove the following claim, whose proof appears in the full version of this work [60].

Lemma 3 (Security for sender) *Let $L \in \mathcal{NP}$ with witness relation R and $\sigma \in \{0,1\}^*$. For the security parameter n, let $S_{cd}(1^n, f_{\sigma,R}, m, q)$ represent the response of the honest sender (of protocol Π_{cd}), with input $(f_{\sigma,R}, m)$ when receiver's first message is q. Then, for every pair of distinct messages (m, m'), every $q \in \{0,1\}^*$ (from a possibly malicious* PPT *receiver), and every $\sigma \notin L$, it holds that*

$$S_{cd}(1^n, f_{\sigma,R}, m, q) \stackrel{c}{\equiv} S_{cd}(^n, f_{\sigma,R}, m', q).$$

5 A Constant Round Protocol

In this section we will present our constant round protocol. The protocol will use the dual simulation idea, introduced in [33], as an important tool. To simplify the exposition and the proofs, we isolate a part of the protocol from [33], and present it as a separate building block.[10]

Shortened GJS protocol $\langle P_{\text{GJS}}, V_{\text{GJS}} \rangle$. The common input is an n vertex graph G in the form of an adjacency matrix, and prover's auxiliary input is a Hamiltonian cycle H in G. The protocol proceeds in following three steps.

1. Commitment stage:
   (a) P_{GJS} sends a random string ρ.
   (b) V_{GJS} sends $\widehat{t_1} = \mathsf{shcom}_\rho(t_1; s_1)$ and $\widehat{ch} = \mathsf{shcom}_\rho(ch; s_2)$, where $t_1 \leftarrow \{0,1\}^{3n^4}$, $ch \in \{0,1\}^n$, and $s_1, s_2 \leftarrow \{0,1\}^{\text{poly}(n)}$.

[10] The only difference is that the challenge-response slots in the [33] protocol have been removed. As a result, many other parameters of their protocol become irrelevant, and also do not appear in this protocol. This does not affect the soundness of the protocol.

2. Coin flipping stage:
   (a) P_{GJS} sends a random string t_2.
   (b) V_{GJS} opens $\widehat{t_1}$ by sending (t_1, s_1).
   Let $\mathbf{t} = t_1 \oplus t_2$.
3. Blum Hamiltonicity protocol:
   (a) Let $\mathbf{t} = \mathbf{t}_1, \ldots, \mathbf{t}_{n^3}$ so that $|\mathbf{t}_i| = 3n$ for $i \in [n^3]$.
   Prover chooses n random permutations $\pi_1, \ldots, \pi_n$ and sets $G_i = \pi_i(G)$
   for each $i \in [n]$. It then commits to each bit b_j in G_i using $\mathsf{sbcom}_{\mathbf{t}_{i\times j}}$.
   (b) Verifier opens to $\widehat{ch}$ by sending (ch, s_2).
   (c) Let $ch = ch_1, \ldots, ch_n$. For every $i \in [n]$, if $ch_i = 0$ then prover opens
   each edge in G_i and reveals π_i; else, it opens edges of the cycle in G_i.

The following lemma has been shown in [33].

Lemma 4 ([33]) *Protocol $\langle P_{\text{GJS}}, V_{\text{GJS}}\rangle$ is a sound interactive argument system for all of $\mathcal{NP}$.*

5.1 Our Protocol

We are now ready to present our protocol $\langle P, V\rangle$. The protocol starts with an execution of the "encrypted" preamble protocol (see section 3.3); this is followed by the first i.e., commitment, stage of the GJS protocol. Before completing the GJS protocol, verifier executes the garbled-circuit protocol Π_{cd} for $f_{\sigma,\mathbf{R}_{sim}}$ and a specific m (described shortly), and proves using an szkaok that this step was performed honestly. This will enable the simulator to extract useful information in m. Finally, the rest of the GJS protocol is executed to complete the proof. The full description of the protocol is given in figure 1. It is easy to see that our protocol has constant rounds. The completeness of the protocol follows directly from the completeness of $\langle P_{\text{GJS}}, V_{\text{GJS}}\rangle$. In next two sections, we prove the soundness and zero-knowledge of this protocol. Note that the the prover is actually "public coin" *up until the final step.*

Proving leakage-resilient zero-knowledge. Due to space constraints, the proof of security of this protocol—theorem 1—appears in the full version of this work [60]. At a high level, we use Barak's non-black-box simulation idea along with GJS simulation. Let V^* be an arbitrary PPT verifier whose program is given as an input to the simulator S. There are four main ideas:

1. First, the simulation uses V^*'s code to execute the preamble in such a way, that at the end of the preamble, $\sigma \in \mathbf{L}_{sim}$. In addition, the simulator will also have a witness ω so that $\mathbf{R}_{sim}(\sigma, \omega) = 1$. The properties of the components used in the preamble (in particular the use of fake sampling algorithms

Protocol $\langle P, V\rangle$. The common input consists of 1^n, and an n vertex graph G in the form of its adjacency matrix. Prover's private input is a Hamiltonian cycle H in G.

1. **"Encrypted" preamble:** $P \Rightarrow V$
   P and V run Barak's encrypted preamble. P runs the public-coin strategy $\widehat{P}_{\text{B}}$, and V runs strategy $\widehat{V}_{\text{B}}$. Let the transcript be $\sigma := \langle h, \tau, c, r, \alpha, \widehat{\beta}, \gamma, \widehat{\delta}\rangle$.
2. **Commitment step:** $V \Rightarrow P$
   P and V run the first, i.e. commitment, step of $\langle P_{\text{GJS}}, V_{\text{GJS}}\rangle$.
   (a) P sends a random string ρ
   (b) V sends $\widehat{t_1} = \mathsf{shcom}_\rho(t_1; s_1)$ and $\widehat{ch} = \mathsf{shcom}_\rho(ch; s_2)$, where $t_1 \leftarrow \{0,1\}^{3n^4}$,
   $ch \leftarrow \{0,1\}^n$, and $s_1, s_2 \leftarrow \{0,1\}^{\text{poly}(n)}$; let $m := (t_1, s_1, ch, s_2)$.
3. **Garbled-circuit step:** $V \Rightarrow P$
   P and V run the *two-round* garbled circuit protocol, Π_{cd}, for the function $f_{\sigma,\mathbf{R}_{sim}}$. V acts as the sender with private input m.
   (a) P runs the fake receiver, $v_1 \leftarrow R_{\text{OT}}^{pub}(1^n, p)$ for a random safe prime p; sends v_1.
   (b) V sends $(\mathcal{C}, v_2, K_m) \leftarrow S_{cd}(f_{\sigma,R}, m, v_1; s_3)$, using fresh coins s_3.
4. **Proof of correctness:** $V \Rightarrow P$
   V proves to P using public-coin szkaok Π_{PR} the knowledge of s_3 and $m = (t_1, s_1, ch, s_2)$ so that:
   (a) $\widehat{t_1} = \mathsf{shcom}_\rho(t_1; s_1)$,
   (b) $\widehat{ch} = \mathsf{shcom}_\rho(ch; s_2)$,
   (c) $S_{cd}(f_{\sigma,R}, m, v_1; s_3) = (\mathcal{C}, v_2, K_m)$.
5. **Final step:** $P \Rightarrow V$
   P and V complete all remaining five rounds of $\langle P_{\text{GJS}}, V_{\text{GJS}}\rangle$. P uses H as the witness.

Fig. 1. Our Constant Round LRZK Protocol.

that are public coin) guarantee that simulator's actions in the preamble are indistinguishable from a real execution with an honest prover. In addition, it is easy to answer leakage queries since the messages exchanged so far represent the entire random-tape of the prover at this point. This allows the simulator to answer leakage queries by simply appending these messages to the state, and sending an appropriate query to the leakage oracle.

2. Next, the simulator will use ω in the garbled circuit step to obtain keys K_ω. Once again, since the first message of OT_{NP} provides indistinguishability for receiver's input, this step does not affect the simulation. Further, since P is public coin in this step as well, the simulator can continue to answer leakage queries as before.
3. Having obtained K_ω along with $\mathcal{C}, K_m$ in the garbled circuit step, the simulator can evaluate the $\mathcal{C}$ and learn $f_{\sigma,\mathbf{R}_{sim}}(\omega, m)$ to learn m. By the soundness

of szkaok of the next step, it is guaranteed that m contains valid openings (t_1, s_1, ch, s_2) for $\widehat{t_1}$ and $\widehat{ch}$.

4. Finally, observe that (t_1, ch) is precisely the information needed by the GJS simulation method to successfully simulate the last step, while answering leakage queries properly. Briefly, ch is the challenge for Blum's protocol, and a first message can be created by the simulator to successfully answer V^*'s challenge in the last message. At the same time, since t_1 is known prior to the coin-flipping stage of the GJS protocol (see section 5), the simulator will have the ability to equivocate in Naor's commitment scheme, allowing it to successfully answer leakage queries.

An important point to note is that if V^* asks more than n^2 bits of leakage after receiving c and before sending r (see GenStat), the simulator will not be able to ensure that $\sigma \in \mathbf{L}_{sim}$. However, if this happens, the simulator can simply ask for the entire witness H from the leakage oracle since the length of leakage is more than the witness size. The simulator can then continue to run like the honest prover and output a view. See full proof in the full version of this work [60].

References

1. William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In *EUROCRYPT*, pages 119–135, 2001.
2. Miklós Ajtai. Secure computation with information leaking to an adversary. In *STOC*, pages 715–724, 2011.
3. Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC*, pages 474–495, 2009.
4. Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO*, pages 36–54, 2009.
5. Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO*, pages 36–54, 2009.
6. Ross J. Anderson and Markus G. Kuhn. Low cost attacks on tamper resistant devices. In *Security Protocols Workshop*, pages 125–136, 1997.
7. B. Barak. How to go beyond the black-box simulation barrier. In *FOCS*, pages 106–115, 2001.
8. B. Barak, O. Goldreich, S. Goldwasser, and Y. Lindell. Resettably-sound zero-knowledge and its applications. In *FOCS 2001*, pages 116–125, 2001.
9. Boaz Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In *FOCS*, 2002.
10. Boaz Barak and Oded Goldreich. Universal arguments and their applications. In *Annual IEEE Conference on Computational Complexity (CCC)*, volume 17, 2002. Preliminary full version available as Cryptology ePrint Archive, Report 2001/105.
11. Boaz Barak and Yehuda Lindell. Strict polynomial-time in simulation and extraction. *SIAM Journal on Computing*, 33(4):783–818, August 2004. Extended abstract appeared in STOC 2002.
12. Nir Bitansky, Ran Canetti, and Shai Halevi. Leakage-tolerant interactive protocols. In *TCC*, pages 266–284, 2012.

13. Manuel Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, pages 1444–1451, 1987.
14. Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. In *EUROCRYPT*, pages 89–108, 2011.
15. Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *FOCS*, pages 501–510, 2010.
16. Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 13(1):143–202, 2000.
17. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In Bob Werner, editor, *Proc. 42nd FOCS*, pages 136–147, 2001. Preliminary full version available as Cryptology ePrint Archive Report 2000/067.
18. Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *STOC*, pages 639–648, 1996.
19. Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge. In *Proc. 32th STOC*, pages 235–244, 2000.
20. Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In *CRYPTO*, pages 432–450, 2000.
21. Ivan Damgård, Torben P. Pedersen, and Birgit Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. *J. Cryptology*, 10(3):163–194, 1997.
22. Yi Deng, Vipul Goyal, and Amit Sahai. Resolving the simultaneous resettability conjecture and a new non-black-box simulation strategy. In *FOCS*, 2009.
23. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
24. Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *FOCS*, pages 511–520, 2010.
25. Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In *ASIACRYPT*, pages 613–631, 2010.
26. Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In *STOC*, pages 621–630, 2009.
27. Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *STOC*, pages 542–552, 1991.
28. Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero knowledge. In *Proc. 30th STOC*, pages 409–418, 1998.
29. Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302, 2008.
30. Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In *EUROCRYPT*, pages 135–156, 2010.
31. U. Feige and A. Shamir. Zero knowledge proofs of knowledge in two rounds. In *CRYPTO*, pages 526–545, 1989.
32. Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO*, pages 10–18, 1984.
33. Sanjam Garg, Abhishek Jain, and Amit Sahai. Leakage resilient zero knowledge. In *Advances in Cryptology – CRYPTO*, 2011. Full version at: `http://www.cs.ucla.edu/~abhishek/papers/lrzk.pdf`.

34. Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. *J. Comput. Syst. Sci.*, 60(3):592–629, 2000.
35. O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In ACM, editor, *Proc. 19th STOC*, pages 218–229, 1987. See [36, Chap. 7] for more details.
36. Oded Goldreich. *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004.
37. Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–189, Summer 1996.
38. Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, July 1991. Preliminary version in FOCS' 86.
39. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *Proc. 17th STOC*, pages 291–304, Providence, 1985. ACM.
40. Shai Halevi and Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. *J. Cryptology*, 25(1):158–193, 2012.
41. Shai Halevi and Silvio Micali. Practical and provably-secure commitment schemes from collision-free hashing. In *CRYPTO*, pages 201–215, 1996.
42. Yuval Ishai, Abishek Kumarasubramanian, Claudio Orlandi, and Amit Sahai. On invertible sampling and adaptive security. In *ASIACRYPT*, pages 466–482, 2010.
43. Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient arguments without short pcps. In *IEEE Conference on Computational Complexity*, pages 278–291, 2007.
44. Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits ii: Keeping secrets in tamperable circuits. In *EUROCRYPT*, pages 308–327, 2006.
45. Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO*, pages 463–481, 2003.
46. Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proc. 24th STOC*, pages 723–732, 1992.
47. Eike Kiltz and Krzysztof Pietrzak. Leakage resilient elgamal encryption. In *ASIACRYPT*, pages 595–612, 2010.
48. Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *CRYPTO*, pages 104–113, 1996.
49. Allison B. Lewko, Yannis Rouselakis, and Brent Waters. Achieving leakage resilience through dual system encryption. In *TCC*, pages 70–88, 2011.
50. Allison B. Lewko and Brent Waters. On the insecurity of parallel repetition for leakage resilience. In *FOCS*, pages 521–530, 2010.
51. Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. In *CRYPTO*, pages 171–189, 2001.
52. S. Micali. CS proofs. In *Proc. 35th FOCS*, pages 436–453, 1994.
53. Silvio Micali and Rafael Pass. Local zero knowledge. In Jon M. Kleinberg, editor, *STOC*, pages 306–315. ACM, 2006.
54. Silvio Micali and Phillip Rogaway. Secure computation. In *Crypto '91*, pages 392–404, 1991.
55. Moni Naor. Bit commitment using pseudo-randomness (extended abstract). In *CRYPTO*, pages 128–136, 1989.
56. Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *SODA*, pages 448–457, 2001.
57. Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO*, pages 18–35, 2009.

58. Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proc. 21st STOC*, pages 33–43, 1989.
59. Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures: The case of aes. In *CT-RSA*, pages 1–20, 2006.
60. Omkant Pandey. Achieving constant round leakage-resilient zero-knowledge. *IACR Cryptology ePrint Archive*, 2012. Full version: `http://eprint.iacr.org/2012/362.pdf`.
61. Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In *Eurocrypt '03*, 2003.
62. Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *Proc. 36th STOC*, pages 232–241, 2004.
63. Rafael Pass and Alon Rosen. New and improved constructions of non-malleable cryptographic protocols. In *STOC*, 2005.
64. Krzysztof Pietrzak. A leakage-resilient mode of operation. In *EUROCRYPT*, pages 462–482, 2009.
65. Manoj Prabhakaran. *New Notions of Security*. PhD thesis, Department of Computer Science, Princeton University, Princeton, NJ, USA, 2005.
66. Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *FOCS*, 2002.
67. Manoj Prabhakaran and Amit Sahai. New notions of security: achieving universal composability without trusted setup. In *STOC*, pages 242–251, 2004.
68. Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *E-smart*, pages 200–210, 2001.
69. Alon Rosen. A note on the round-complexity of concurrent zero-knowledge. In *Crypto '00*, pages 451–468, 2000.
70. Alon Rosen. A note on constant-round zero-knowledge proofs for NP. In *TCC*, pages 191–202, 2004.
71. Andrew C. Yao. Theory and applications of trapdoor functions. In *Proc. 23rd FOCS*, pages 80–91, 1982.
72. Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Proc. 27th FOCS*, pages 162–167, 1986.