Constant-round Leakage-resilient Zero-knowledge from Collision Resistance*

Susumu Kiyoshima

NTT Secure Platform Laboratories, Tokyo, Japan
kiyoshima.susumu@lab.ntt.co.jp

August 20, 2018

Abstract

In this paper, we present a constant-round leakage-resilient zero-knowledge argument system for $\mathcal{NP}$ under the assumption of the existence of collision-resistant hash function family. That is, using collision-resistant hash functions, we construct a constant-round zero-knowledge argument system that has the following zero-knowledge property: Even against any cheating verifier that obtains arbitrary amount of leakage on the prover's internal secret state, a simulator can simulate the verifier's view by obtaining the same amount of leakage on the witness. Previously, leakage-resilient zero-knowledge proofs/arguments for $\mathcal{NP}$ were constructed only under a relaxed security definition (Garg, Jain, and Sahai, CRYPTO'11) or under the DDH assumption (Pandey, TCC'14).

Our leakage-resilient zero-knowledge argument system satisfies an additional property that it is simultaneously leakage-resilient zero-knowledge, meaning that both zero-knowledgeness and soundness hold in the presence of leakage.

*This article is based on an earlier article: Constant-round Leakage-resilient Zero-knowledge from Collision Resistance, in Proceedings of EUROCRYPT 2016, ©IACR 2016, https://doi.org/10.1007/978-3-662-49896-5_4.

1 Introduction

Zero-knowledge (ZK) *proofs* and *arguments* [GMR89] are interactive proof/argument systems with which a prover can convince a verifier of the correctness of a mathematical statement while providing "zero additional knowledge." This zero-knowledge property is formalized thorough the *simulation paradigm*: An interactive proof or argument is said to be zero-knowledge if for any cheating verifier there exists a *simulator* that can output simulated view of the verifier.

Recently, Garg et al. [GJS11] introduced a new notion of zero-knowledgeness called *leakage-resilient zero-knowledge* (LRZK). Informally speaking, LRZK is a notion of zero-knowledgeness in the setting where cheating verifiers can obtain some leakage on the internal state of the provers (including the witnesses and the randomness that they have) during the entire protocol executions, and is motivated by the studies of *side-channel attacks* (e.g., [Koc96, AK96, QS01]), which demonstrated that in some cases cheating parties can indeed obtain leakage on honest parties' internal states by attacking physical implementations of cryptographic algorithms.

LRZK requires that cheating verifiers cannot learn anything from honest provers beyond the validity of the statement *and the leakage that they have obtained*. More formally, LRZK is defined by modifying the standard definition of zero-knowledgeness as follows. First, the cheating verifier V^* is allowed to make arbitrary number of *leakage queries* during the interaction with the honest prover, where when querying a function f as a leakage query, V^* obtains $f(w, \mathsf{tape})$ as the answer, where w is the witness that the honest prover has and tape is the randomness that the honest prover used thus far. Second, the simulator $\mathcal{S}$ is allowed to make queries to the *leakage oracle* $\mathcal{L}_w$, where $\mathcal{L}_w$ is parametrized by the witness w and outputs $f(w)$ on input any function f. Finally, it is required that for any $\ell \in \mathbb{N}$, when V^* obtains ℓ bits of leakage on the prover's state via leakage queries, $\mathcal{S}$ can simulate V^*'s view by obtaining ℓ bits of leakage on the witness via queries to the leakage oracle $\mathcal{L}_w$.[1]

In [GJS11], Garg et al. showed a proof system that satisfies a weaker notion of LRZK called $(1+\epsilon)$-LRZK. Specifically, they showed that for any $\epsilon > 0$, there exists a proof system such that when V^* obtains ℓ bits of leakage from the prover, a simulator can simulate V^*'s view by obtaining at most $(1+\epsilon)\cdot\ell$ bits of leakage from $\mathcal{L}_w$. The round complexity of this protocol is at least $\omega(\log n)/\epsilon$, and its security is proven under a standard general assumption (the existence of statistically hiding commitment scheme that is public-coin w.r.t. the receiver).

A natural question that was left open by Garg et al. [GJS11] is whether a LRZK protocol can be constructed without weakening the security requirement. That is, the question is whether one can reduce ϵ to 0 in the protocol of Garg et al. [GJS11]. This question is important because $(1+\epsilon)$-LRZK does not necessarily provide sufficient level of security in several applications—indeed, since $(1+\epsilon)$-LRZK allows the simulator to obtain strictly more leakage than cheating verifiers, $(1+\epsilon)$-LRZK protocols can potentially reveal secret information to the verifier in addition to the leakage. This question is also of theoretical interest because reducing ϵ to 0 is optimal in the sense that λ-LRZK for $\lambda < 0$ is impossible to achieve in the plain model [GJS11].

Recently, this open question was solved affirmatively by Pandey [Pan14], who constructed the first LRZK argument system by using the DDH assumption and collision-resistant hash functions. Pandey's protocol has only constant number of rounds; hence, his result implies that asymptotically optimal round complexity can be achievable even in the presence of leakage.

A question that was explicitly left open by Pandey [Pan14, Section 1] is whether one can construct LRZK protocols under a standard *general* assumption. Indeed, although the protocol of Pandey [Pan14] is superior to the protocol of Garg et al. [GJS11] in terms of both leakage resilience (LRZK

[1] In [OPV15], it is pointed out that nowadays *leakage tolerance* is the commonly accepted term for this security notion. Nevertheless, in this paper we use the term "leakage resilience" for this security notion for consistency with previous works [GJS11, Pan14].

v.s. $(1+\epsilon)$-LRZK) and round complexity (constant v.s. $\omega(\log n)/\epsilon$), the assumption of the former is seemingly much stronger than that of the latter (the DDH assumption v.s. the existence of statistically hiding commitment scheme that is public-coin w.r.t. the receiver, which is implied by, say, the existence of collision-resistant hash function family or even the existence of one-way functions[2]).

> **Question.** *Can we construct a (constant-round) leakage-resilient zero-knowledge protocol under standard general assumptions?*

1.1 Our Results

In this paper, we answer the above question affirmatively by constructing a LRZK argument from collision-resistant hash functions (CRHFs). Like the protocol of Pandey [Pan14], our protocol has only constant number of rounds. Also, our protocol has an additional property that it is public coin, i.e., that the verifier does not have any secret state and just sends the outcome of coin-tossing in each round.

Main Theorem. *Assume the existence of collision-resistant hash function family. Then, there exists a constant-round public-coin leakage-resilient zero-knowledge argument for $\mathcal{NP}$.*

(The formal statement of our result is given as Theorem 1 in Section 5.) We notice that the existence of LRZK protocols under CRHFs is somewhat surprising because the only known LRZK protocol [Pan14] crucially relies on the secure two-party computation protocol of Yao [Yao86], which requires an assumption that is seemingly stronger than the existence of CRHFs (namely, the existence of oblivious transfer protocols). One of our technical novelties is constructing a LRZK protocol without relying on Yao's secure computation protocol.

Simultaneously leakage-resilient zero-knowledge. Our protocol has an additional property that it is *simultaneously leakage-resilient zero-knowledge* [GJS11], meaning that not only zero-knowledgeness but also soundness holds in the presence of leakage. The *leakage-resilient (*LR*) soundness* (i.e., soundness in the presence of leakage) of our protocol follows immediately from its public-coin property—as shown by Garg et al. [GJS11], any public-coin interactive proof/argument system is LR sound for arbitrary amount of leakage on the verifier's state since the verifier has no secret state in public-coin protocols.

To the best of our knowledge, our protocol is the first simultaneously LRZK protocol. The $(1+\epsilon)$-LRZK protocol of Garg et al. [GJS11] is LR sound in a weak sense—it is LR sound when there is an a-priori upper bound on the amount of leakage—but is not LR sound when the amount of leakage is unbounded,[3] and similarly, the LRZK protocol of Pandey [Pan14] is also not LR sound with unbounded amount of leakage. In contrast, our protocol is sound even when cheating provers obtain arbitrary amount of leakage on the secret states of the verifiers.

The summary of the previous results and ours is given in Table 1. In the table, "bounded-LR sound" means that the soundness holds when there is an a-priori upper bound on the amount of leakage from the verifier.

[2] A constant-round one can be constructed from collision-resistant hash functions [NY89, DPP98] and a polynomial-round one can be constructed from one-way functions [HNO+09].

[3] This is because in the protocol of [GJS11], the verifier commits to the challenge bits of Blum's Hamiltonicity protocol in advance and hence an cheating prover can easily break the soundness by obtaining the challenge bits via leakage.

	LR ZKness	LR soundness	#(round)	Assumptions
Garg et al. [GJS11]	$(1+\epsilon)$-LRZK	bounded-LR sound	$\mathsf{poly}(n) + \omega(\log n)/\epsilon$	OWFs
			$\omega(\log n)/\epsilon$	CRHFs
Pandey [Pan14]	LRZK	-	$O(1)$	DDH + CRHFs
This work	LRZK	LR sound	$O(1)$	CRHFs

Table 1: Summary of the results on LRZK protocols. The round complexity of the protocol of Garg et al. [GJS11] depends on the assumption that is used to instantiate the underlying statistically-hiding commitment scheme. In particular, when only one-way functions (OWFs) are used, there is a polynomial additive overhead because statistically hiding commitment schemes currently require polynomial number of rounds in this case [HNO+09].

1.2 Open Questions

Reducing assumption to the existence of one-way functions. An important open question is whether one can construct constant-round LRZK argument systems under the assumption of the existence of one-way functions.

We notice that solving this question affirmatively seems to require an advancement on "straight-line" simulation techniques (i.e., techniques that do not use rewinding). This is because, as will become clear in Section 2, constant-round LRZK seems to require straight-line simulation, and currently the only known straight-line simulation technique, the one by Barak [Bar01], requires collision-resistant hash functions.[4]

Constructing LRZK **proof system.** Another open question is whether one can construct LRZK proof systems (instead of argument ones).

We notice that solving this question affirmatively also seems to require an advancement on straight-line simulation techniques. This is because the straight-line simulation technique of Barak [Bar01] is currently inherently only computationally sound.

1.3 Related Works

The works relevant to ours are the works that study interactive protocols in the presence of arbitrary leakage in the models other than the plain model. These works include, for example, the works about leakage-tolerant UC-secure protocols in the CRS model [BCH12], non-transferable interactive proof systems in the CRS model with leak-free input encoding/updating phase [AGP14], and secure computation protocols in the CRS model with leak-free preprocessing/input-encoding phase and constant fraction of honest parties [BGJK12, BGJ+13, BDL14]. We remind the readers that, like [GJS11, Pan14], this work considers LRZK protocols in the plain model without any leak-free phase.

In [OPV15], Ostrovsky et al. showed an impossibility result about black-box LRZK in the model with only leak-free input-encoding phase (i.e., without CRS and preprocessing). We notice that there is no contradiction between this impossibility result and our result since the definition of LRZK in [OPV15] is different from the one we use. Specifically, in the definition of Ostrovsky et al. [OPV15], the simulator is not allowed to obtain any leakage whereas in the definition that we use, the simulator can obtain leakage on the witness (in other words, Ostrovsky et al. [OPV15] considers leakage resilience whereas we consider leakage tolerance; see Footnote 1).

[4] Chung et al. [CPS16] showed that the simulation technique of Barak can be modified so that it requires only one-way functions. However, the simulation technique of Chung et al. involves rewinding of the adversary and therefore is no longer straight-line simulation.

1.4 Outline

In Section 2, we give an overview of our techniques. In Section 3, we give the notations and definitions that are used throughout the paper. In Section 4, we show the two new building blocks that we use in our LRZK protocol. In Section 5, we describe our LRZK protocol and prove its security.

2 Overview of Our Techniques

In this section, we give an informal overview of our LRZK protocol. We remind the readers that, as informally stated in Section 1, a proof/argument system is leakage-resilient zero-knowledge if for any cheating verifier V^* there exists a simulator $\mathcal{S}$ such that for any $\ell \in \mathbb{N}$, if V^* obtains ℓ bits of leakage on the prover's state, $\mathcal{S}$ can simulate V^*'s view by obtaining ℓ bit of leakage on the witness w from leakage oracle $\mathcal{L}_w$.

2.1 Previous Techniques

Since our techniques rely on the techniques that are used in the LRZK protocol of Garg et al. [GJS11] and that of Pandey [Pan14], we start by recalling their protocols.

Protocol of Garg et al. [GJS11]

In [GJS11], Garg et al. constructed a $(1+\epsilon)$-leakage-resilient zero-knowledge proof system from a statistically hiding commitment scheme that is public-coin w.r.t. the receiver. That is, by using such a commitment scheme, they constructed a proof system such that, when V^* obtains ℓ bits of leakage from the prover, its view can be simulated by obtaining at most $(1+\epsilon)\cdot\ell$ bits of leakage from $\mathcal{L}_w$.

A key idea behind the protocol of Garg et al. [GJS11] is to give the simulator two independent ways of cheating—one is for simulating the prover's messages and the other is for simulating the leakages. Concretely, Garg et al. constructed their protocol by combining two well-known techniques on constant-round zero-knowledge protocols—the technique of Goldreich and Kahan [GK96] that requires the verifier to commit to its challenges in advance and the technique of Feige and Shamir [FS89] that uses equivocal commitment schemes. At a high level, Garg et al. proved the security by considering a simulator that simulates the prover's messages by extracting the challenges and simulates the leakages by using the equivocality of the underlying commitment scheme.

In more details, the protocol of Garg et al. [GJS11] consists of the following two phases. In the first phase, the verifier uses an extractable commitment scheme to commit to a challenge string ch of Blum's Hamiltonicity protocol as well as trapdoor information td of an equivocal commitment scheme.[5] In the second phase, the prover and the verifier execute Blum's Hamiltonicity protocol that is instantiated with the equivocal commitment scheme. In simulation, the simulator extracts ch and td in the first phase and then simulates the prover's messages and the leakages in the second phase by using the knowledge of ch and td in the following way. (For simplicity, we assume that Blum's protocol is executed only once instead of many times in parallel.)

- When the extracted challenge ch is 0, the simulator commits to a randomly permuted adjacent matrix of the statement G, and after V^* reveals the challenge ch (which must be 0), the simulator decommits all the commitment to reveal the permuted adjacent matrix of G.

[5] Actually, there is also a coin-tossing protocol that determines the parameter of the equivocal commitment scheme, and td is the trapdoor for biasing the outcome of the coin-tossing. However, for simplicity, we think that td is trapdoor for the equivocal commitment scheme in this overview.

Notice that the simulator does exactly the same things as an honest prover. Hence, the simulator can simulate prover's randomness tape easily and therefore can answer any leakage query f from V^* by querying $f(\cdot, \mathsf{tape})$ to $\mathcal{L}_w$.

- When the extracted challenge ch is 1, the simulator commits to the adjacent matrix of a randomly chosen cycle graph H, and after V^* reveals the challenge ch (which must be 1), the simulator decommits some of the commitments so that only the edges on the cycle are revealed.

  When V^* makes a leakage query, the simulator answers it by using the fact that, given w and td, it is possible to compute randomness that "explains" the commitment to H as a commitment to a permuted G (that is, randomness that explains the commitment that the simulator has sent as a commitment that an honest prover would have sent). Specifically, the simulator answers a leakage query f from V^* by querying the following function $\widetilde{f}(\cdot)$ to $\mathcal{L}_w$.

  1. On input w, function $\widetilde{f}$ first computes a permutation π that maps the Hamiltonian cycle w in G to the cycle in H (i.e., computes π such that $\pi(G)$ has the same cycle as H).
  2. Next, by using equivocality[6] with trapdoor td, it computes randomness tape that explains the commitment to H as a commitment to $\pi(G)$ (i.e., it computes tape such that committing to $\pi(G)$ with randomness tape will generate the same commitment as the one that the simulator has sent to V^* by committing to H).
  3. Finally, it outputs $f(w, \mathsf{tape})$.

  It is easy to see that this simulation strategy correctly simulates leakage on the prover's state. Furthermore, since the simulator chooses π so that $\pi(G)$ has the same cycle as H, the simulated leakages (from which V^* may be able to compute $\pi(G)$) are consistent with the cycle of H that is revealed in the last round of the protocol.

We remark that the reason why the protocol of Garg et al. [GJS11] satisfies only $(1 + \epsilon)$-LRZK for $\epsilon > 0$ is that the extraction of ch and td involves rewinding of V^*. Indeed, if V^* always makes new leakage queries after being rewound, the simulator need to obtain new leakages from $\mathcal{L}_w$ in each rewinding and hence need to obtain more bits of leakage than V^*. From this observation, it seems that to achieve LRZK, we need to avoid the use of rewinding simulation techniques.

Protocol of Pandey [Pan14]

In [Pan14], Pandey constructed a constant-round LRZK argument system under the DDH assumption. Roughly speaking, Pandey's idea is to replace the rewinding simulation technique in the protocol of Garg et al. [GJS11] with the "straight-line" simulation technique of Barak [Bar01]. In particular, Pandey replaced the first phase of the protocol of Garg et al. [GJS11] with the following one.

1. First, the prover and the verifier execute an encrypted version of so called *Barak's preamble* [Bar01, PR08b, PR08a], which determines a "fake statement" that is false except with negligible probability.

2. Next, the prover and the verifier execute Yao's garbled circuit protocol [Yao86] in which the prover can obtain ch and td only when it has a valid witness for the fake statement.

From the security of the encrypted Barak's preamble, no cheating prover can make the fake statement true; hence, ch and td are hidden from the cheating prover. In contrast, a non-black-box simulator can

[6] What is actually used here is *adaptive security*, which guarantees that for each underlying commitment, it is possible to compute randomness tape_0 and tape_1 such that tape_b explains the commitment as a commitment to b for each $b \in \{0, 1\}$.

make the fake statement true by using the knowledge of the code of the verifier; hence, the simulator can obtain ch and td without rewinding V^*. An issue is that, to guarantee leakage resilience, it is required that Yao's protocol is executed in a way that all prover's messages are pseudorandom (since otherwise it is hard to simulate randomness that explains the simulated prover's messages as honest prover's messages during the simulation of the leakages). Since Yao's protocol involves executions of an oblivious transfer protocol (in which the prover behaves as a receiver), this property is hard to satisfy in general. Pandey solved this problem by using the DDH assumption, under which there exists an oblivious transfer protocol such that all receiver's messages are indistinguishable from random group elements.

2.2 Our Techniques

The reason why the protocols of Garg et al. and Pandey [GJS11, Pan14] either guarantee only weaker security or rely on a stronger assumption is that the simulation involves extraction from V^*. Indeed, in [GJS11], the simulator need to obtain more amount of leakage than V^* because it rewinds V^* during extraction, and in [Pan14], the DDH assumption is required because Yao's protocol is used for extraction.

Based on this observation, our strategy is to modify the protocols of Garg et al. and Pandey [GJS11, Pan14] so that no extraction is required in simulation. We first remove the extraction of trapdoor td and next remove the extraction of challenge ch. We remark that the latter is much harder than the former.

Removing Extraction of Trapdoor td

We first modify the protocols of [GJS11, Pan14] so that leakages can be simulated without extracting the trapdoor td of the equivocal commitment scheme.

Our main tool is Hamiltonicity commitment scheme $\mathsf{H\text{-}Com}$ [FS89, CLOS02], which is a well-known instance-dependent equivocal commitment scheme based on Blum's Hamiltonicity protocol. $\mathsf{H\text{-}Com}$ is parametrized by a graph G with $q = \mathsf{poly}(n)$ vertices. To commit to 0, the committer chooses a random permutation π and commits to the adjacent matrix of $\pi(G)$ by using any statistically binding commitment scheme Com; in the decommit phase, the committer reveals π and decommits the commitments to reveal all the entries of the matrix. To commit to 1, the committer commits to the adjacent matrix of a random q-cycle graph; in the decommit phase, the committer decommits some of the commitments so that only the entries that corresponds to the edges on the cycle are revealed. $\mathsf{H\text{-}Com}$ satisfies equivocality when G has a Hamiltonian cycle; this is because after committing to 0, the committer can decommit it to both 0 and 1 given a Hamiltonian cycle w in G.

Given $\mathsf{H\text{-}Com}$, we remove the extraction of td by combining $\mathsf{H\text{-}Com}$ with an encrypted variant of Barak's preamble. Specifically, we replace the equivocal commitment scheme in the protocols of [GJS11, Pan14] with $\mathsf{H\text{-}Com}$ that depends on the fake statement G' that is obtained from the encrypted Barak's preamble. From the security of Barak's preamble, any cheating prover cannot make G' true and hence cannot use the equivocality of $\mathsf{H\text{-}Com}$, whereas the simulator can make G' true and hence can use the equivocality of $\mathsf{H\text{-}Com}$ as desired.

Remark 1. As observed in [Pan14], it is not straightforward to use the encrypted Barak's preamble in the presence of leakage. Roughly speaking, in the encrypted Barak's preamble, the prover commits to its messages instead of sending them in clear, and in the proof of soundness, it is required that the prover's messages are extractable from the commitments. The problem is that it is not easy to guarantee this extractability in the presence of leakage (this is because the prover's messages are typically not pseudorandom in the techniques of extractability). Pandey [Pan14] solved this problem by having the prover use a specific extractable commitment scheme based on the DDH assumption. In this paper, we instead have the prover use a commitment scheme that only satisfies very weak extractability but

the prover's messages of which are pseudorandom and the security of which is based on the existence of CRHFs.[7] For details, see Section 4.1.

Removing Extraction of Challenge *ch*

Next, we modify the protocols of [GJS11, Pan14] so that prover's messages can be simulated without extracting the challenge *ch* of Hamiltonicity protocol.

We first notice that, although the simulator can use equivocality without extraction as shown above, it is not easy for the simulator to use equivocality for simulating prover's messages. This is because when the leakages to V^* includes the randomness that is used for commitments, V^* may be able to determine the committed values from the leakages and therefore equivocation may be detected by V^*.

As our main technical tool, we introduce a specific instance-dependent equivocal commitment scheme GJS-Com that we obtain by considering the technique of Garg et al. [GJS11] on Hamiltonicity protocol in the context of H-Com. Recall that in [GJS11], Garg et al. use Blum's Hamiltonicity protocol that is instantiated with an equivocal commitment scheme. Here, we use H-Com that is instantiated with an equivocal commitment scheme (i.e., we use H-Com in which the adjacent matrix is committed to by an equivocal commitment scheme). The equivocal commitment scheme that we use is, as above, H-Com that depends on the fake statement that is generated by the encrypted Barak's preamble.[8] Hence, the commitment scheme GJS-Com is a version of H-Com that is instantiated by using H-Com itself as the underling commitment scheme.[9] GJS-Com depends on two statements of the Hamiltonicity problem: The outer H-Com (the H-Com that is implemented with H-Com) depends on the real statement G, and the inner H-Com (the H-Com that is used to implement H-Com) depends on the fake statement G'. GJS-Com inherits equivocality from the outer H-Com, i.e., given a witness for the real statement G, a GJS-Com commitment to 0 can be decommitted to both 0 and 1.

Since GJS-Com is obtained by considering the technique of Garg et al. [GJS11] in the context of H-Com, we can see that GJS-Com satisfies a property that is useful for proving LRZK property. First, observe that given GJS-Com, the second phase of the LRZK protocol of [GJS11] (i.e., Blum's Hamiltonicity protocol phase) can be viewed as follows.

1. The prover commits to 0 by using GJS-Com.
2. The verifier reveals the challenge $ch \in \{0, 1\}$ that is committed to in the first phase.
3. When $ch = 0$, the prover decommits the GJS-Com commitment to 0 honestly, and when $ch = 1$, the prover decommits it to 1 by using the equivocality with the knowledge of Hamiltonian cycle w in G.

When the second phase of the protocol of [GJS11] is viewed in this way, the key property that is used in the simulation of the leakages in [GJS11] is the following.

- Given a Hamiltonian cycle in G and that in G', a GJS-Com commitment to 1 (in which a random cycle graph is committed) can be "explained" as a commitment to 0 (in which a permuted G is committed) by using the equivocality of the inner H-Com (cf. the function $\widetilde{f}$ in Section 2.1). Furthermore, even after being explained as a commitment to 0, the commitment can later be decommitted to 1 in a consistent way with the explained randomness.

[7] This extractability is used only in the proof of soundness. Hence, the proof of zero-knowledgeness works even in the presence of this extractable commitment scheme.

[8] Actually, we use an adaptively secure H-Com [CLOS02, LZ11]. See footnote 6.

[9] In the "inner" H-Com, the underlying commitment scheme is Com as before.

Because of this property, even when the simulator commits to 1 instead of 0 using GJS-Com to simulate the messages in the protocol of [GJS11], the simulator can answer any leakage query f from V^* by querying $\mathcal{L}_w$ a function $\widetilde{f}$ that, on input w, computes randomness tape that explains the commitment to 1 as a commitment to 0 and then outputs $f(w, \mathsf{tape})$.

A problem of this property is that it can be used only in a very limited situation. Specifically, this property can be used only when the simulator knows which GJS-Com commitment will be decommitted to 1, and this is the reason why the extraction of ch is required in the simulation strategy of [GJS11, Pan14]. Hence, to remove the extraction of ch, we need to use GJS-Com in a way that, given a witness for the fake statement, the simulator can predict which value each GJS-Com commitment will be decommitted to.

Then, our key observation is that we can use this property if we use GJS-Com to implement the Hamiltonicity protocol *in which the fake statement is proven*.[10] Concretely, we consider the following protocol.

1. The prover and the verifier execute an encrypted variant of Barak's preamble. Let G' be the fake statement and let q' be the number of the vertices of G'.

2. (a) The prover commits to a $q' \times q'$ zero matrix by using GJS-Com.
   (b) The verifier sends a challenge $ch \in \{0, 1\}$.
   (c) When $ch = 0$, the prover sends a random permutation π over G' to the verifier and then decommit the GJS-Com commitments to the adjacent matrix of $\pi(G')$ by using the equivocality of GJS-Com with the knowledge of a witness for the real statement.
   When $ch = 1$, the prover chooses a random q'-cycle graph H and decommits some of the GJS-Com commitments to 1 by using the equivocality of GJS-Com so that the decommitted entries of the matrix correspond to the cycle in H.
   (d) When $ch = 0$, the verifier verifies whether the decommitted graph is $\pi(G')$. When $ch = 1$, the verifier verifies whether the decommitted entries corresponds to a q'-cycle in a graph.

Since any cheating prover cannot make the fake statement G' true, GJS-Com is statistically binding when the real statement G is false, and hence soundness follows. In contrast, the simulator can cheat in Barak's preamble so that it knows a Hamiltonian cycle w' in the fake statement G', and therefore can simulate the prover's messages by "honestly" proving the fake statement, i.e., by committing to $\pi(G')$ in step 2(a) for a randomly chosen π and then revealing the entire graph $\pi(G')$ when $ch = 0$ and the cycle $\pi(w')$ when $ch = 1$. Furthermore, since in step 2(a) the simulator do know which value each GJS-Com commitment will be decommitted to (the commitments to the edges on $\pi(w')$ will be always decommitted to 1 and others will be decommitted honestly or will not be decommitted), the simulator can simulate the leakage in the same way as in the protocol of Garg et al. [GJS11] by using the property of GJS-Com described above.

Since Barak's preamble is based on the existence of CRHFs and has constant rounds, our protocols is based on the existence of CRHFs and has constant rounds. This completes the overview of our techniques.

[10] Hence, we use Hamiltonicity protocol recursively *three times*: We instantiate Hamiltonicity commitment with Hamiltonicity commitment to obtain GJS-Com, and then instantiate Blum's Hamiltonicity protocol with GJS-Com.

3 Preliminaries

3.1 Notations

Throughout the paper, we use n to denote the security parameter. We use $\mathbb{N}$ to denote the set of all natural numbers, $\bot$ to denote a special error symbol, poly to denote an arbitrary polynomial, and negl to denote an arbitrary negligible function, where a function f is negligible if it grows slower than the inverse of any polynomial (i.e., $f(n) = 1/n^{\omega(1)}$). We use PPT as an abbreviation of "probabilistic polynomial time." For any $k \in \mathbb{N}$, we use $[k]$ to denote the set $\{1, \dots, k\}$. For any randomized algorithm Algo, we use $\mathsf{Algo}(x; r)$ to denote an execution of Algo with input x and randomness r, and we use $\mathsf{Algo}(x)$ to denote an execution of Algo with input x and uniformly chosen randomness. For any two-party protocol $\langle A, B \rangle$, we use $\mathsf{trans}\left[A(x) \leftrightarrow B(y)\right]$ to denote the random variable representing the transcript of the interaction between A and B with input x and y respectively, and use $\mathsf{output}_A\left[A(x) \leftrightarrow B(y)\right]$ (resp., $\mathsf{output}_B\left[A(x) \leftrightarrow B(y)\right]$) to denote the random variable representing the output of A (resp., B) in the interaction between A and B with input x and y respectively.

We use $\mathbf{L}_{\mathrm{HC}}$ to denote the languages of the Hamiltonian graphs. For any $G \in \mathbf{L}_{\mathrm{HC}}$, we use $\mathbf{R}_{\mathrm{HC}}(G)$ to denote the set of the Hamiltonian cycles in G. More generally, for any language $\mathbf{L}$ and any instance $x \in \mathbf{L}$, we use $\mathbf{R}_{\mathbf{L}}(x)$ to denote the set of the witnesses for $x \in \mathbf{L}$.

We assume familiarity with the notion of computational indistinguishability (see, e.g., [Gol01]). For any two probabilistic ensembles $\mathcal{X} = \{X_k\}_{k\in\mathbb{N}}$ and $\mathcal{Y} = \{Y_k\}_{k\in\mathbb{N}}$, we use $\mathcal{X} \approx \mathcal{Y}$ to denote that they are computationally indistinguishable, and use $\mathcal{X} \equiv \mathcal{Y}$ to denote that they are identically distributed.

3.2 Collision-resistant Hash Functions

In this subsection, we recall the definition of collision resistant hash functions.

Definition 1 (Collision-resistant hash functions)**.** *A family of functions $\mathcal{H} = \{h_s : \{0,1\}^* \to \{0,1\}^{|s|}\}_{s\in\{0,1\}^*}$ is **collision resistant** if the following two conditions hold.*

- ***Easy to compute:*** *There exists a deterministic polynomial-time algorithm M such that $M(s, x) = h_s(x)$ holds for every $s \in \{0,1\}^*$ and $x \in \{0,1\}^*$.*
- ***Hard to find collision:*** *For any PPT adversary $\mathcal{A}$, consider the following probabilistic experiment $\mathrm{Exp}^{\mathrm{coll}}(\mathcal{H}, \mathcal{A}, n)$ between $\mathcal{A}$ and a challenger.*

  1. *The challenger chooses $h_s \in \mathcal{H}_n$ uniformly at random, where $\mathcal{H}_n \stackrel{\mathrm{def}}{=} \{h_s \in \mathcal{H} \text{ s.t. } |s| = n\}$.*
  2. *On input 1^n and h_s, the adversary $\mathcal{A}$ outputs x, x'.*

  Then, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $n \in \mathbb{N}$, it holds $\Pr[x \neq x' \wedge h_s(x) = h_s(x')] \leq \mathsf{negl}(n)$ in the experiment $\mathrm{Exp}^{\mathrm{coll}}(\mathcal{H}, \mathcal{A}, n)$.

◇

Remark 2. Compared with the definitions in, e.g., [Gol04], the above definition is simplified since it is assumed that (1) all strings correspond to valid keys (i.e., $h_s \in \mathcal{H}$ exists for every $s \in \{0,1\}^n$) and that (2) the image is $\{0,1\}^n$ when the key length is n (i.e., the image of h_s is $\{0,1\}^n$ when $|s| = n$.) Nonetheless, the results in this paper hold even when the definition in [Gol04] are used.

3.3 Leakage-resilient Zero-knowledge Arguments

In this subsection, we recall the definition of leakage-resilient zero-knowledgeness argument systems [GJS11].

3.3.1 Interactive Arguments

We first recall the definition of interactive argument systems [GMR89, BCC88].

Definition 2 (Interactive argument)**.** *For an $\mathcal{NP}$ language $\mathbf{L}$ with witness relation $\mathbf{R}_{\mathbf{L}}$, a pair of interactive Turing machines $\langle P, V\rangle$ is an **interactive argument system** for $\mathbf{L}$ if it satisfies the following properties.*

- ***Completeness:*** *For every $x \in \mathbf{L}$ and $w \in \mathbf{R}_{\mathbf{L}}(x)$,*

$$\Pr\left[\mathsf{output}_V\left[P(x,w) \leftrightarrow V(x)\right] = 1\right] = 1 \ .$$

- ***Soundness:*** *For every PPT Turing machine P^*, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $x \notin \mathbf{L}$ and $z \in \{0,1\}^*$,*

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

*An interactive argument system $\langle P, V\rangle$ is **public coin** if V sends only the outcome of coin tossing in each round of interaction with P.* ◇

In an interactive argument system $\langle P, V\rangle$, P is called a *prover* and V is called a *verifier*.

3.3.2 Leakage-resilient Zero-knowledge

We next recall the definition of leakage-resilient zero-knowledgeness. (For convenience, we use a formulation that is slightly different from that of [GJS11].)

For any interactive argument system $\langle P, V\rangle$, any PPT cheating verifier V^*, any statement $x \in \mathbf{L}$, any witness $w \in \mathbf{R}_{\mathbf{L}}(x)$, any $z \in \{0,1\}^*$, and any oracle machine $\mathcal{S}$ called *simulator*, consider the following two experiments.

$\underline{\mathrm{REAL}_{V^*}(x,w,z)}$

1. Execute $V^*(x,z)$ with an honest prover $P(x,w)$ of $\langle P, V\rangle$. During the interaction, V^* can make an arbitrary number of adaptive *leakage queries* on the state of P. A leakage query consists of an efficiently compatible function f (described as a circuit) and is answered with $f(w, \mathsf{tape})$, where tape is the randomness that has been used by P thus far.
2. Output the view of V^*.

$\underline{\mathrm{IDEAL}_{\mathcal{S}}(x,w,z)}$

1. Execute $\mathcal{S}(x,z)$ with a *leakage oracle* $\mathcal{L}_w$. A query to $\mathcal{L}_w$ from $\mathcal{S}$ consists of an efficiently computable function f and is answered with $f(w)$. Let τ be the output of $\mathcal{S}$.
2. If τ is not valid view of V^*, the output of the experiment is $\perp$. Otherwise, let ℓ be the total length of the leakage that V^* obtains in τ. If the total length of the answers that $\mathcal{S}$ obtained from $\mathcal{L}_w$ is larger than ℓ, the output of the experiment is $\perp$. Otherwise, the output is τ.

Let $\mathsf{REAL}_{V^*}(x,w,z)$ be the random variable representing the output of $\mathrm{REAL}_{V^*}(x,w,z)$ and $\mathsf{IDEAL}_{\mathcal{S}}(x,w,z)$ be the random variable representing the output of $\mathrm{IDEAL}_{\mathcal{S}}(x,w,z)$. Then, leakage resilient zero-knowledgeness is defined as follows.

Definition 3 (Leakage-resilient zero-knowledge)**.** *An interactive argument system $\langle P, V\rangle$ for an $\mathcal{NP}$ language $\mathbf{L}$ with witness relation $\mathbf{R}_{\mathbf{L}}$ is **leakage-resilient zero knowledge** if for every PPT Turing machine V^* and every sequence $\{w_x\}_{x\in\mathbf{L}}$ such that $w_x \in \mathbf{R}_{\mathbf{L}}(x)$, there exists a PPT oracle Turing machine $\mathcal{S}$ such that the following hold.*

- ***Indistinguishability condition:***

$$\{\mathsf{REAL}_{V^*}(x, w_x, z)\}_{x\in\mathbf{L}, z\in\{0,1\}^*} \approx \{\mathsf{IDEAL}_{\mathcal{S}}(x, w_x, z)\}_{x\in\mathbf{L}, z\in\{0,1\}^*} \quad .$$

- ***Leakage-length condition:*** *For every $x \in \mathbf{L}$ and $z \in \{0,1\}^*$,*

$$\Pr\left[\mathsf{IDEAL}_{\mathcal{S}}(x, w_x, z) = \bot\right] = 0 \quad .$$

◇

3.4 Commitment Schemes

In this subsection, we recall the definition of commitment schemes and describe the existing instantiations that we use in this paper.

3.4.1 Basic Definitions

We first recall the basic security requirements for commitment schemes. Commitment schemes are two-party protocols between a *committer* and a *receiver*, and their executions consist of two phases, the *commit phase* and the *decommit phase*. In the commit phase, the committer *commits* to a secret input $v \in \{0,1\}^n$ by interacting with the receiver; the transcript of the commit phase is called the *commitment*. In the decommit phase, the committer *decommits* the commitment by sending the receiver the secret value v along with a message called the *decommitment*; the receiver then outputs either 1 (accept) or 0 (reject). It is required that the receiver accepts the decommitment with probability 1 when both the committer and the receiver behaved honestly. Additionally, it is required that the committer cannot decommit a commitment to two different values and that the committed value is hidden from the receiver in the commit phase; the former is called the *binding* property and the latter is called the *hiding* property. Formal definitions of these two properties are given below.

Definition 4 (Binding property)**.** *For a commitment scheme $\langle C, R\rangle$ and any (not necessarily PPT) cheating committer C^*, consider the following probabilistic experiment $\mathrm{Exp}^{\mathrm{bind}}(\langle C, R\rangle, C^*, n, z)$ for any $n \in \mathbb{N}$ and $z \in \{0,1\}^*$.*

> *On input 1^n and auxiliary input z, the cheating committer C^* interacts with an honest receiver in the commit phase of $\langle C, R\rangle$ and then outputs two pairs, (v_0, d_0) and (v_1, d_1). Then, C^* is said to win the experiment if $v_0 \neq v_1$ but the receiver accepts both (v_0, d_0) and (v_1, d_1) in the decommit phase.*

*Then, $\langle C, R\rangle$ is **statistically binding** if for any sequence of auxiliary inputs $\{z_n\}_{n\in\mathbb{N}}$, the probability that C^* wins the experiment $\mathrm{Exp}^{\mathrm{bind}}(\langle C, R\rangle, C^*, n, z_n)$ is negligible.* ◇

Definition 5 (Hiding property)**.** *For a commitment scheme $\langle C, R\rangle$ and any PPT cheating receiver R^*, consider the following probabilistic experiment $\mathrm{Exp}^{\mathrm{hide}}_b(\langle C, R\rangle, R^*, n, z)$ for any $b \in \{0,1\}$, $n \in \mathbb{N}$, and $z \in \{0,1\}^*$.*

> *On input 1^n and auxiliary input z, the cheating receiver R^* chooses a pair of challenge values $v_0, v_1 \in \{0,1\}^n$ and then interacts with an honest committer in the commit phase of $\langle C, R\rangle$, where the committer commits to v_b. The output of the experiment is the view of R^**

Let $\mathsf{Exp}_b^{\mathsf{hide}}(\langle C, R\rangle, R^*, n, z)$ *be the random variable representing the output of experiment* $Exp_b^{\mathsf{hide}}(\langle C, R\rangle, R^*, n, z)$*. Then,* $\langle C, R\rangle$ *is* ***computationally hiding*** *if the following indistinguishability holds.*

$$\left\{\mathsf{Exp}_0^{\mathsf{hide}}(\langle C, R\rangle, R^*, n, z)\right\}_{n\in\mathbb{N}, z\in\{0,1\}^*} \approx \left\{\mathsf{Exp}_1^{\mathsf{hide}}(\langle C, R\rangle, R^*, n, z)\right\}_{n\in\mathbb{N}, z\in\{0,1\}^*} .$$

◇

In this paper, we say that a commitment is *valid* if there is a value to which the commitment can be correctly decommitted. We denote by $\mathsf{value}(\cdot)$ a function that, on input a commitment (i.e., a transcript in the commit phase), outputs its committed value if it is uniquely determined and outputs $\perp$ otherwise.

3.4.2 Naor's Commitment Scheme

We next recall Naor's statistically binding commitment scheme Com, which can be constructed from one-way functions [Nao91, HILL99].

Commit phase. The commit phase consists of two rounds. In the first round, the receiver sends a random $3n$-bit string $r \in \{0,1\}^{3n}$. In the second round, the committer chooses a random seed $s \in \{0,1\}^n$ for a pseudorandom generator $\mathsf{PRG} : \{0,1\}^n \to \{0,1\}^{3n}$ and then sends $\mathsf{PRG}(s)$ if it wants to commit to 0 and sends $\mathsf{PRG}(s) \oplus r$ if it wants to commit to 1.[11]

We use $\mathsf{Com}_r(\cdot)$ to denote an algorithm that, on input $b \in \{0,1\}$, computes a commitment to b as above by using r as the first-round message.

Decommit phase. In the decommit phase, the committer reveals the seed s as the decommitment.

Security. Com is statistically binding and computational hiding. Furthermore, the binding and hiding property hold even when the same first-round message r is used in multiple commitments.

Committing to strings. For any $\ell \in \mathbb{N}$, one can commit to an ℓ-bit string by committing to each bit using Com, where the same first-round message r is used in all the commitments.

We abuse the notation and use $\mathsf{Com}_r(\cdot)$ to denote an algorithm that, on input $m \in \{0,1\}^*$, computes a commitment to m as above by using r as the first-round message. Notice that $\mathsf{Com}_r(\cdot)$ has pseudorandom range. Hence, by using a public-coin algorithm $\mathsf{Com}_{\mathrm{pub}}$ that outputs a random $3n\ell$-bit string on input 1^ℓ, we obtain a "fake commitment" that is indistinguishable from a real commitment to an ℓ-bit string.

3.4.3 Hamiltonicity Commitment Scheme

We next recall a well-known instance-dependent commitment scheme $\mathsf{H\text{-}Com}$ [FS89, CLOS02] that is based on Blum's zero-knowledge proof for Hamiltonicity.

Commit phase. $\mathsf{H\text{-}Com}$ is parametrized by a graph G. Let q be the number of its vertices. To commit to 0, the committer chooses a random permutation π over the vertices of G and then commits to the adjacent matrix of $\pi(G)$ by using Com. To commit to 1, the committer chooses a random q-cycle graph and then commits to its adjacent matrix by using Com.

We use $\mathsf{H\text{-}Com}_{G,r}(\cdot)$ to denote an algorithm that, on input $b \in \{0,1\}$, computes a commitment to b as above by using r as the first-round message of all the Com commitments.

[11] For the definition of pseudorandom generators, see [Gol01].

Decommit phase. When the committer committed to 0, it reveals π, and also reveals all the entries of the adjacent matrix by decommitting all the Com commitments. When the committer committed to 1, it reveals the entries that correspond to the edges on the q-cycle by decommitting the corresponding Com commitments.

Security. $\mathsf{H\text{-}Com}$ is computationally hiding both when $G \in \mathbf{L}_{\mathrm{HC}}$ and when $G \notin \mathbf{L}_{\mathrm{HC}}$, and it is statistically binding when $G \notin \mathbf{L}_{\mathrm{HC}}$.

Equivocality. When $G \in \mathbf{L}_{\mathrm{HC}}$, a commitment to 0 can be decommitted to 1 given a Hamiltonian cycle $w \in \mathbf{R}_{\mathrm{HC}}(G)$ in G. Specifically, a commitment to 0 can be decommitted to 1 by revealing the entries that corresponds to the edges on $\pi(w)$ (i.e., the edges on the cycle that is obtained by applying π on w).

3.4.4 Adaptive Hamiltonicity Commitment Scheme

We next recall the adaptively secure Hamiltonicity commitment scheme $\mathsf{AH\text{-}Com}$, which was used in, e.g., [CLOS02, LZ11].

Commit phase. $\mathsf{AH\text{-}Com}$ is parametrized by a graph G. Let q be the number of its vertices. To commit to 0, the committer does the same things as in $\mathsf{H\text{-}Com}$, i.e., it chooses a random permutation π over the vertices of G and then commits to the adjacent matrix of $\pi(G)$ by using Com. To commit to 1, the committer chooses a random q-cycle graph and then commits to its adjacent matrix in the following way: For all the entries that correspond to the edges on the q-cycle, it commits to 1 by using Com, and for all the other entries, it simply sends random $3n$-bit strings instead of committing to 0. (Since Com has pseudorandom range, random $3n$-bit strings are indistinguishable from Com commitments.)

We use $\mathsf{AH\text{-}Com}_{G,r}(\cdot)$ to denote an algorithm that, on input $b \in \{0,1\}$, computes a commitment to b as above by using r as the first-round message of all the Com commitments.

Decommit phase. To decommit, the committer reveals all the randomness that was used in the commit phase. We use $\mathsf{AH\text{-}Dec}_r(\cdot,\cdot,\cdot)$ to denote an algorithm that, on input c, b, ρ such that $\mathsf{AH\text{-}Com}_r(b;\rho) = c$, outputs a decommitment d as above.

Security. Like $\mathsf{H\text{-}Com}$, $\mathsf{AH\text{-}Com}$ is computationally hiding both when $G \in \mathbf{L}_{\mathrm{HC}}$ and when $G \notin \mathbf{L}_{\mathrm{HC}}$, and it is statistically binding when $G \notin \mathbf{L}_{\mathrm{HC}}$.

Adaptive security. When $G \in \mathbf{L}_{\mathrm{HC}}$, a commitment to 0 can be "explained" as a valid commitment to 1 given a witness $w \in \mathbf{R}_{\mathrm{HC}}(G)$; that is, for a commitment c to 0, one can compute ρ such that $\mathsf{AH\text{-}Com}(1;\rho) = c$. This is because commitments to the entries that do not correspond to the edges on $\pi(w)$ are indistinguishable from random strings.

Formally, there exists an algorithm $\mathsf{AH\text{-}ExplainAsOne}$ such that for security parameter $n \in \mathbb{N}$, graphs $G \in \mathbf{L}_{\mathrm{HC}}$, witness $w \in \mathbf{R}_{\mathrm{HC}}(G)$, and string $r \in \{0,1\}^{3n}$, the following hold.

Correctness. Given witness $w \in \mathbf{R}_{\mathrm{HC}}(G)$ and c, ρ such that $\mathsf{AH\text{-}Com}_{G,r}(0;\rho) = c$, $\mathsf{AH\text{-}ExplainAsOne}_{G,r}$ outputs ρ' such that $\mathsf{AH\text{-}Com}_{G,r}(1;\rho') = c$.

Indistinguishability. Consider the following two probabilistic experiments.

$\underline{\mathrm{EXP}_0^{\mathrm{AH}}(n, G, w, r)}$

```
/* commit to 1 and reveal randomness */
```

1. Computes $c \leftarrow \mathsf{AH\text{-}Com}_{G,r}(1)$.
   Let ρ_1 be the randomness that was used in $\mathsf{AH\text{-}Com}$.
2. Output (c, ρ_1).

$\underline{\mathrm{EXP}_1^{\mathrm{AH}}(n, G, w, r)}$
`/* commit to 0 and explain it as commitment to 1 */`

1. Computes $c \leftarrow \mathsf{AH\text{-}Com}_{G,r}(0)$.
   Let ρ_0 be the randomness that was used in $\mathsf{AH\text{-}Com}$.
   Compute $\rho_1 := \mathsf{AH\text{-}ExplainAsOne}_{G,r}(w, c, \rho_0)$.
2. Output (c, ρ_1).

Let $\mathsf{EXP}_b^{\mathrm{AH}}(n, G, w, r)$ be the random variable representing the output of $\mathrm{EXP}_b^{\mathrm{AH}}(n, G, w, r)$ for each $b \in \{0, 1\}$. Then, the following two ensembles are computationally indistinguishable.

- $\left\{\mathsf{EXP}_0^{\mathrm{AH}}(n, G, w, r)\right\}_{n \in \mathbb{N}, G \in \mathbf{L}_{\mathrm{HC}}, w \in \mathbf{R}_{\mathrm{HC}}(G), r \in \{0,1\}^{3n}}$
- $\left\{\mathsf{EXP}_1^{\mathrm{AH}}(n, G, w, r)\right\}_{n \in \mathbb{N}, G \in \mathbf{L}_{\mathrm{HC}}, w \in \mathbf{R}_{\mathrm{HC}}(G), r \in \{0,1\}^{3n}}$

3.5 Barak's Non-black-box Zero-knowledge Argument

In this subsection, we recall Barak's non-black-box zero-knowledge argument [Bar01]. As mentioned in Section 2, a variant of its preamble stage (more precisely, a variant of so called "encrypted" Barak's preamble [PR08b, PR08a]) will be used in our LRZK protocol.

3.5.1 Universal Arguments

We first recall the definition of universal argument systems [BG08], which is a key building block of Barak's non-black-box zero-knowledge argument.

Universal language. For the purpose of this paper, it suffices to give the definition of universal arguments only w.r.t. the membership of a single "universal" language $\mathbf{L}_{\mathcal{U}}$. For triplet $y = (M, x, t)$, we have $y \in \mathbf{L}_{\mathcal{U}}$ if non-deterministic machine M accepts x within t steps. (Here, all components of y, including t, are encoded in binary.) Let $\mathbf{R}_{\mathcal{U}}$ be the witness relation of $\mathbf{L}_{\mathcal{U}}$, i.e., $\mathbf{R}_{\mathcal{U}}$ is a polynomial-time decidable relation such that for any $y = (M, x, t)$, we have $y \in \mathbf{L}_{\mathcal{U}}$ if and only if there exists $w \in \{0, 1\}^{\leq t}$ such that $(y, w) \in \mathbf{R}_{\mathcal{U}}$.

Universal argument. Roughly speaking, universal arguments are "efficient" arguments of knowledge for proving the membership in $\mathbf{L}_{\mathcal{U}}$, where they are efficient in the sense that the prover's running time is bounded by the time that is needed for verifying the validity of the witness that the prover has. A formal definition is given below. In the following, for any $y = (M, x, t) \in \mathbf{L}_{\mathcal{U}}$, we use $T_M(x, w)$ to denote the running time of M on input x with witness w, and let $\mathbf{R}_{\mathcal{U}}(y) = \{w \mid (y, w) \in \mathbf{R}_{\mathcal{U}}\}$.

Definition 6 (Universal argument)**.** *A pair of interactive Turing machines $\langle P, V\rangle$ is a* ***universal argument system*** *if it satisfies the following properties.*

- ***Efficient verification:*** *There exists a polynomial p such that for any $y = (M, x, t)$, the total time spent by (probabilistic) verifier strategy V on inputs y is at most $p(|y|)$.*

- ***Completeness by a relatively efficient prover:*** *For every $y = (M, x, t) \in \mathbf{L}_{\mathcal{U}}$ and $w \in \mathbf{R}_{\mathcal{U}}(y)$,*

$$\Pr\left[\mathsf{output}_V\left[P(y, w) \leftrightarrow V(y)\right] = 1\right] = 1 \ .$$

Furthermore, there exists a polynomial q such that the total time spent by P, on input (y, w), is at most $q(|y| + T_M(x, w)) \leq q(|y| + t)$.

- ***Computational soundness:*** *For every PPT Turing machine P^*, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $y = (M, x, t) \notin \mathbf{L}_{\mathcal{U}}$ and $z \in \{0,1\}^*$,*

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

- ***Weak proof of knowledge:*** *For every polynomial $p(\cdot)$ there exists a polynomial $p'(\cdot)$ and a PPT oracle machine E such that the following holds: For every PPT Turing machine P^*, every sufficiently long $y = (M, x, t) \in \{0,1\}^*$, and every $z \in \{0,1\}^*$, if*

$$\Pr\left[\mathsf{output}_V\left[P^*(y, z) \leftrightarrow V(y)\right] = 1\right] \geq \frac{1}{p(|y|)} \ ,$$

then

$$\Pr_r\left[\exists w = w_1 \cdots w_t \in \mathbf{R}_{\mathcal{U}}(y) \text{ s.t. } \forall i \in [t], E_r^{P^*(y,z)}(y, i) = w_i\right] \geq \frac{1}{p'(|y|)} \ ,$$

where $E_r^{P^*(y,z)}(\cdot, \cdot)$ denotes the function defined by fixing the randomness of E to r, and providing the resulting E_r with oracle access to $P^*(y, z)$. ◇

The weak proof-of-knowledge property of universal arguments only guarantees that each individual bit w_i of a witness w can be extracted in probabilistic polynomial time. However, for any $y = (M, x, t) \in \mathbf{L}_{\mathcal{U}}$, since any witness $w \in \mathbf{R}_{\mathcal{U}}(y)$ is of length at most t, there exists an extractor (called the *global extractor*) that extracts the whole witness in time polynomial in $\mathsf{poly}(|y|) \cdot t$. In this paper, this property is called the *global proof-of-knowledge property* of a universal argument.

3.5.2 Barak's Non-black-box Zero-knowledge Argument

We next describe Barak's non-black-box zero-knowledge argument, which can be constructed from any collision-resilient hash function family $\mathcal{H}$. Informally speaking, Barak's protocol BarakZK proceeds as follows.

Protocol BarakZK

1. The verifier V sends a random hash function $h \in \mathcal{H}$ and the first-round message $r_1 \in \{0,1\}^{3n}$ of Com to the prover P.
2. P sends $c \leftarrow \mathsf{Com}_{r_1}(0^n)$ to V. Then, V sends random string r_2 to P.
3. P proves the following statement by a witness-indistinguishable argument.
   - $x \in L$, or
   - $(h, c, r_2) \in \Lambda$, where $(h, c, r_2) \in \Lambda$ holds if and only if there exists a machine Π such that c is a commitment to $h(\Pi)$ and Π outputs r_2 in $n^{\log\log n}$ steps.

Note that the statement proven in the last step is not in $\mathcal{NP}$. Thus, P proves this statement by a witness-indistinguishable universal argument (WIUA), with which P can prove any statement in $\mathcal{NEXP}$ in the relatively efficient manner. Intuitively, BarakZK is sound since $\Pi(c) \neq r$ holds with overwhelming probability even when a cheating prover P^* commits to $h(\Pi)$ for any machine Π. On the other hand, the zero-knowledge property can be proven by using a simulator that commits to $h(\Pi)$ such that Π is a machine that emulates the cheating verifier V^*; since $\Pi(c) = V^*(c) = r$ holds from the definition, the simulator can give a valid proof in the last step.

For our purpose, it is convenient to consider a variant of BarakZK that we denote by $\langle P_B, V_B \rangle$. $\langle P_B, V_B \rangle$ is the same as BarakZK except that in the last step, instead of proving $x \in L \vee (h, c, r_2) \in \Lambda$ by using a WIUA, P proves $(h, c, r_2) \in \Lambda$ by using the four-round public-coin universal argument system UA of Barak and Goldreich [BG08]. (Hence, $\langle P_B, V_B \rangle$ is no longer zero-knowledge protocol.) The formal description of $\langle P_B, V_B \rangle$ is shown in Figure 1. We remark that in $\langle P_B, V_B \rangle$, the language proven in the last step is replaced with a slightly more complex language as in, e.g., [Bar01, PR08b, PR08a, Pan14]. (Roughly speaking, this replacement is necessary for using $\langle P_B, V_B \rangle$ in the setting of leakage-resilient zero-knowledge since the cheating verifier can obtain arbitrary information as leakage before sending r_2.)

Stage 1:

The verifier V_B sends a random hash function $h \in \mathcal{H}$ to the prover P_B, where the domain of h is $\{0,1\}^*$ and the range of h is $\{0,1\}^n$. V_B also sends $r_1 \in \{0,1\}^{3n}$ (the first-round message of Com) to P_B.

Stage 2:

1. P_B computes $c \leftarrow \mathsf{Com}_{r_1}(0^n)$ and send c to V_B.
2. V_B sends random $r_2 \in \{0,1\}^{n+n^2}$ to P_B.

Stage 3: P_B proves statement $(h, r_1, c, r_2) \in \Lambda$ by using UA.

1. V_B sends the first-round message α.
2. P_B sends the second-round message β.
3. V_B sends the third-round message γ.
4. P_B sends the fourth-round message δ.

Language Λ:

$(h, r_1, c, r_2) \in \Lambda$ if and only if there exist

- a machine Π
- randomness rand for Com
- a string y such that $|y| \leq n^2$

such that

- $c = \mathsf{Com}_{r_1}(h(\Pi); \mathsf{rand})$, and
- $\Pi(c, y)$ outputs r_2 within $n^{\log\log n}$ steps.

Figure 1: Encrypted Barak's preamble $\langle P_B, V_B \rangle$.

In essentially the same way as the soundness of BarakZK [Bar01], we can prove the following lemma on $\langle P_{\mathrm{B}}, V_{\mathrm{B}} \rangle$, which roughly states that there exists a "hard" language $\mathbf{L}_{\mathrm{B}}$ on the transcript of $\langle P_{\mathrm{B}}, V_{\mathrm{B}} \rangle$ such that no cheating prover can generate a transcript that is included in $\mathbf{L}_{\mathrm{B}}$.

Lemma 1 (Soundness)**.** *Let $\mathbf{L}_{\mathrm{B}}$ be the language defined in Figure 2. Then, for any cheating prover P^* against $\langle P_{\mathrm{B}}, V_{\mathrm{B}} \rangle$, any $n \in \mathbb{N}$, and any $z \in \{0,1\}^*$,*

$$\Pr\left[\tau \in \mathbf{L}_{\mathrm{B}} \mid \tau \leftarrow \mathsf{trans}\left[P^*(1^n, z) \leftrightarrow V_{\mathrm{B}}(1^n)\right]\right] \leq \mathsf{negl}(n) \enspace .$$

Language $\mathbf{L}_{\mathrm{B}}$:
$\tau = (h, r_1, c, r_2, \alpha, \beta, \gamma, \delta) \in \mathbf{L}_{\mathrm{B}}$ if and only if $(\alpha, \beta, \gamma, \delta)$ is an accepting transcript of UA for statement $(h, r_1, c, r_2) \in \Lambda$.

Figure 2: A "hard" language $\mathbf{L}_{\mathrm{B}}$.

Proof sketch of Lemma 1. We first remark that the language Λ depicted in Figure 1 is overly simplified and therefore we can prove this lemma only when the underlying hash function family $\mathcal{H}$ is secure against $\mathsf{poly}(n^{\log\log n})$-time adversaries. By using the language given in [BG08], we can prove this lemma even when $\mathcal{H}$ is secure only against polynomial-time adversaries.

Assume for contradiction that there exists P^* such that for infinitely many n's, there exists $z \in \{0,1\}^*$ such that the following holds for a polynomial $p(\cdot)$.

$$\Pr\left[\tau \in \mathbf{L}_{\mathrm{B}} \mid \tau \leftarrow \mathsf{trans}\left[P^*(1^n, z) \leftrightarrow V_{\mathrm{B}}(1^n)\right]\right] \geq \frac{1}{p(n)} \enspace .$$

Fix any such P^*, n, and z. Then, consider interacting with P^* in the following way.

1. Interacts with P^* as an honest V_{B} until the end of $\langle P_{\mathrm{B}}, V_{\mathrm{B}} \rangle$. Let (h, r_1, c, r_2) be the transcript of the first two stages. If the UA proof in the last stage in not accepting, abort. Otherwise, extracts witness $w = (\Pi, R, y)$ for $(h, r_1, c, r_2) \in \Lambda$ using the global extractability of UA. (From the definition, this extraction takes at most $\mathsf{poly}(n^{\log\log n})$ steps.)

2. Rewind P^* to the point just before sending r_2 to P^*, and interacts with P^* again as an honest V_{B} with fresh randomness until the end of $\langle P_{\mathrm{B}}, V_{\mathrm{B}} \rangle$. Let (h, r_1, c, r_2') be the transcript of the first two stages. If the UA proof in not accepting, abort. Otherwise, extracts witness $w' = (\Pi', R', y')$ for $(h, r_1, c, r_2') \in \Lambda$ using the global extractability of UA.

From an average argument and the extractability of UA, we can obtain w and w' with probability $1/p'(n)$ for a polynomial $p'(\cdot)$. We then show that when we obtain w and w', we can obtain a collision of h. First, observe that since Π is deterministic, we have

$$\left|\left\{r \;\middle|\; \exists y \in \{0,1\}^* \text{ s.t. } |y| \leq n^2 \wedge \Pi(c, y) = r\right\}\right| \leq 2^{n^2+1} \enspace .$$

Since r_2' is chosen uniformly at random from $\{0,1\}^{n+n^2}$, the probability that there exists $y \in \{0,1\}^{\leq n^2}$ such that $\Pi(c, y) = r_2'$ is at most $2^{n^2+1}/2^{n+n^2} = 1/2^{n-1}$. Then, since we have $\Pi'(c, y_2') = r_2'$ because w' is a valid witness, we have $\Pi \neq \Pi'$ except with probability $1/2^{n-1}$. Furthermore, since both $h(\Pi)$ and $h(\Pi')$ are the committed value of c, from the statistical binding property of Com, $h(\Pi) = h(\Pi')$ holds except with negligible probability. Hence, the pair (Π, Π') is a collision of h except with negligible probability. □

3.6 Somewhat Extractable Commitment Scheme

In this subsection, we introduce a commitment scheme that satisfies only very weak extractability that we call *somewhat extractability*. This scheme will be used in our variant of encrypted Barak's preamble in Section 4.1. As mentioned in Remark 1 in Section 2.2, an important point on this scheme is that the committer sends only pseudorandom messages while it can be constructed from one-way functions.

Concretely, we consider the commitment scheme SWExtCom in Figure 3. SWExtCom is the same as the extractable commitment scheme of Pass and Wee [PW09] except that in the last step, the committer simply reveals the values that it committed to in the first step (in other words, the committer does not send the decommitments). Because of this simplification, SWExtCom does not satisfy extractability in the standard sense. Still, it is not hard to see that SWExtCom satisfies extractability in the sense that, given two valid commitments c and c' such that the transcripts of the commit stage are identical but those of the challenge stage are different, then the committed value of c can be extracted. Formally, SWExtCom satisfies the following extractability.

Commit phase. The committer C and the receiver R receive common inputs 1^n. To commit to $v \in \{0,1\}^n$, the committer C does the following with the receiver R.

Commit stage.
For each $i \in [n]$, the committer C chooses a pair of random n-bit strings $(a_{i,0}, a_{i,1})$ such that $a_{i,0} \oplus a_{i,1} = v$. Then, for each $i \in [n]$ in parallel, C commits to $a_{i,0}$ and $a_{i,1}$ by using Com. For each $i \in [n]$ and $b \in \{0,1\}$, let $c_{i,b}$ be the commitment to $a_{i,b}$.

Challenge stage.
R sends random n-bit string $e = (e_1, \ldots, e_n)$ to C.

Reply stage.
For each $i \in [n]$, C sends a_{i,e_i} to R.
COMMENT: *C just sends a_{i,e_i} and does not decommit c_{i,e_i}.*

Decommit phase. C sends v to R and decommits $c_{i,b}$ to $a_{i,b}$ for all $i \in [n]$ and $b \in \{0,1\}$. R checks whether $a_{1,0} \oplus a_{1,1} = \cdots = a_{n,0} \oplus a_{n,1} = v$ holds and whether $a_{1,e_1}, \ldots, a_{n,e_n}$ are equal to the values that were revealed in the commit phase.

. . . . . .

Extracting algorithm Extract**.**
On input two commitments $c = (\{c_{i,b}\}_{i\in[n],b\in\{0,1\}}, \{e_i\}_{i\in[n]}, \{a_{i,e_i}\}_{i\in[n]})$ and $c' = (\{c'_{i,b}\}_{i\in[n],b\in\{0,1\}}, \{e'_i\}_{i\in[n]}, \{a'_{i,e_i}\}_{i\in[n]})$ such that $c_{i,b} = c'_{i,b}$ for every $i \in [n]$ and $b \in \{0,1\}$, do the following.

1. Find any $i \in [n]$ such that $e_i \neq e'_i$. If no such i exist, output fail.
2. Output $\widetilde{v} \stackrel{\text{def}}{=} a_{i,e_i} \oplus a'_{i,e'_i}$.

Figure 3: A somewhat extractable commitment scheme SWExtCom

Lemma 2 (Somewhat extractability)**.** *Let us say that two commitments $c = (\{c_{i,b}\}_{i\in[n],b\in\{0,1\}}, \{e_i\}_{i\in[n]}, \{a_{i,e_i}\}_{i\in[n]})$ and $c' = (\{c'_{i,b}\}_{i\in[n],b\in\{0,1\}}, \{e'_i\}_{i\in[n]}, \{a'_{i,e_i}\}_{i\in[n]})$ are **admissible** if*

- *$c_{i,b} = c'_{i,b}$ for every $i \in [n]$ and $b \in \{0,1\}$,*

- *there exists* $i^* \in [n]$ *such that* $e_{i^*} \neq e'_{i^*}$, *and*
- *the committed value of* $c_{i,b}$ *is uniquely determined for every* $i \in [n]$ *and* $b \in \{0,1\}$.

Let $\mathsf{Extract}(\cdot,\cdot)$ *be the algorithm shown in Figure 3. Then, for any two admissible commitments* c *and* c', *if both* c *and* c' *are valid,* $\widetilde{v} \stackrel{\text{def}}{=} \mathsf{Extract}(c,c')$ *is equal to* $\mathsf{value}(c)$ *(i.e.,* $\widetilde{v}$ *is the committed value of* c*).*

Proof. First, when c and c' are valid, $a_{i^*,e_{i^*}}$ and $a'_{i^*,e'_{i^*}}$ are the committed values of $c_{i^*,e_{i^*}}$ and $c_{i^*,e'_{i^*}}$ (since otherwise, any decommitments of c and c' would be rejected because the decommitted values of $c_{i^*,e_{i^*}}$ and $c_{i^*,e'_{i^*}}$ are not consistent with $a_{i^*,e_{i^*}}$ and $a'_{i^*,e'_{i^*}}$). Second, when c and c' are valid, the committed value of c can be computed by XORing the committed values of $c_{i^*,e_{i^*}}$ and $c_{i^*,e'_{i^*}}$ (since otherwise, any decommitments of c and c' would be rejected). From these, the lemma follows. □

A nice property of $\mathsf{SWExtCom}$ is that all the messages that the committer sends in the commit phase are pseudorandom. Formally, we have the following lemma.

Lemma 3 (Existence of public-coin fake committing algorithm)**.** *Let* C *be an honest committer algorithm of* $\mathsf{SWExtCom}$. *There exists a* PPT *public-coin algorithm* C_{pub} *such that for any* PPT *cheating receiver* R^* *that interacts with* C *in the commit phase of* $\mathsf{SWExtCom}$, *the following ensembles are computationally indistinguishable.*

- $\{\mathsf{output}_{R^*}[C(v) \leftrightarrow R^*(1^n,z)]\}_{n\in\mathbb{N},v\in\{0,1\}^n,z\in\{0,1\}^*}$
- $\left\{\mathsf{output}_{R^*}\left[C_{\mathrm{pub}}(1^n) \leftrightarrow R^*(1^n,z)\right]\right\}_{n\in\mathbb{N},v\in\{0,1\}^n,z\in\{0,1\}^*}$

Proof. C_{pub} is an algorithm that is the same as C except that, instead of sending commitments of Com, it sends fake commitments of Com using $\mathsf{Com}_{\mathrm{pub}}$ (i.e., sends random strings with the same length as the Com commitments). Since Com has pseudorandom range, the indistinguishability can be proven by using a standard hybrid argument (in which the commitments of Com are replaced with random strings one by one). □

4 Building Blocks

In this section, we introduce two new building blocks that we use in our LRZK protocol.

4.1 Special-purpose Encrypted Barak's Preamble

In our LRZK protocol, we use a variant of so called "encrypted" Barak's preamble [PR08b, PR08a]. The encrypted Barak's preamble is the same as (a variant of) Barak's non-black-box zero-knowledge protocol $\langle P_{\mathrm{B}}, V_{\mathrm{B}}\rangle$ in Section 3.5 except that P_{B} commits to its UA messages β and δ instead of sending them in clear. In this paper, we use a variant in which, instead of giving valid commitments, P_{B} gives fake commitments of Com and $\mathsf{SWExtCom}$ by using $\mathsf{Com}_{\mathrm{pub}}$ and C_{pub} (cf. Sections 3.4.2 and 3.6). A nice property of this variant is that the prover sends only random strings; as will become clear later, this property is useful for constructing leakage-resilient protocols. The formal description of this variant, which we denote by $\langle \mathbb{P}_{\mathrm{B}}, \mathbb{V}_{\mathrm{B}}\rangle$, is shown in Figure 4.

We first show that, as in the case of $\langle P_{\mathrm{B}}, V_{\mathrm{B}}\rangle$, there exists a "hard" language on the transcript of $\langle \mathbb{P}_{\mathrm{B}}, \mathbb{V}_{\mathrm{B}}\rangle$.

Stage 1:

The verifier $\mathbb{V}_B$ sends a random hash function $h \in \mathcal{H}$ to the prover $\mathbb{P}_B$. $\mathbb{V}_B$ also sends $r_1 \in \{0,1\}^{3n}$ (the first-round message of Com) to $\mathbb{P}_B$.

Stage 2:

1. $\mathbb{P}_B$ gives a fake commitment c of Com to $\mathbb{V}_B$ by running $c \leftarrow \mathsf{Com}_{\mathrm{pub}}(1^n)$.
2. $\mathbb{V}_B$ sends random $r_2 \in \{0,1\}^{n+n^2}$ to $\mathbb{P}_B$.

Stage 3 (Encrypted UA):

1. $\mathbb{V}_B$ sends the first-round message α of UA for statement $(h, r_1, c, r_2) \in \Lambda$.
2. $\mathbb{P}_B$ gives a fake commitment of $\mathsf{SWExtCom}$ to $\mathbb{V}_B$ by running $C_{\mathrm{pub}}(1^n)$. Let $\widehat{\beta}$ be the fake commitment (i.e., the transcript of this step).
3. $\mathbb{V}_B$ sends the third-round message γ of UA for statement $(h, r_1, c, r_2) \in \Lambda$.
4. $\mathbb{P}_B$ gives a fake commitment of $\mathsf{SWExtCom}$ to $\mathbb{V}_B$ by running $C_{\mathrm{pub}}(1^n)$. Let $\widehat{\delta}$ be the fake commitment.

. . . . . .

Language Λ (same as the one in Figure 1):

$(h, r_1, c, r_2) \in \Lambda$ if and only if there exist

- a machine Π
- randomness rand for Com
- a string y such that $|y| \leq n^2$

such that

- $c = \mathsf{Com}_{r_1}(h(\Pi); \mathsf{rand})$, and
- $\Pi(c, y)$ outputs r_2 within $n^{\log\log n}$ steps.

Figure 4: Special-purpose encrypted Barak's preamble $\langle \mathbb{P}_B, \mathbb{V}_B \rangle$.

Language $\mathbb{L}_B$:

$(h, r_1, c, r_2, \alpha, \widehat{\beta}, \gamma, \widehat{\delta}) \in \mathbb{L}_B$ if and only if there exist

- decommitments $d_1, d_2 \in \{0,1\}^{\mathsf{poly}(n)}$ for $\mathsf{SWExtCom}$
- the second-round and the fourth-round messages $\beta, \delta \in \{0,1\}^n$ of UA

such that

- d_1 is a valid decommitment of $\widehat{\beta}$ to β, and
- d_2 is a valid decommitment of $\widehat{\delta}$ to δ, and
- $(\alpha, \beta, \gamma, \delta)$ is an accepting transcript of UA for statement $(h, r_1, c, r_2) \in \Lambda$.

Figure 5: Language $\mathbb{L}_B$.

Lemma 4 (Soundness). *Let $\mathbb{L}_{\mathrm{B}}$ be the language defined in Figure 5. Then, for any cheating prover $\mathbb{P}^*$ against $\langle \mathbb{P}_{\mathrm{B}}, \mathbb{V}_{\mathrm{B}} \rangle$, any $n \in \mathbb{N}$, and any $z \in \{0,1\}^*$,*

$$\Pr\left[\tau \in \mathbb{L}_{\mathrm{B}} \mid \tau \leftarrow \mathsf{trans}\left[\mathbb{P}^*(1^n, z) \leftrightarrow \mathbb{V}_{\mathrm{B}}(1^n)\right]\right] \leq \mathsf{negl}(n) \ .$$

Proof. Assume for contradiction that there exists $\mathbb{P}^*$ such that for infinitely many n's, there exists $z \in \{0,1\}^*$ such that

$$\Pr\left[\tau \in \mathbb{L}_{\mathrm{B}} \mid \tau \leftarrow \mathsf{trans}\left[\mathbb{P}^*(1^n, z) \leftrightarrow \mathbb{V}_{\mathrm{B}}(1^n)\right]\right] \geq \frac{1}{p(n)}$$

for a polynomial $p(\cdot)$. We use $\mathbb{P}^*$ to construct a cheating prover P^* against $\langle P_{\mathrm{B}}, V_{\mathrm{B}} \rangle$ and show that it breaks the soundness of $\langle P_{\mathrm{B}}, V_{\mathrm{B}} \rangle$ (i.e., Lemma 1).

Consider the following cheating prover P^* against $\langle P_{\mathrm{B}}, V_{\mathrm{B}} \rangle$. First, P^* internally invokes $\mathbb{P}^*$. Then, while externally interacting with an honest V_{B} of $\langle P_{\mathrm{B}}, V_{\mathrm{B}} \rangle$, P^* internally interacts with $\mathbb{P}^*$ as a verifier of $\langle \mathbb{P}_{\mathrm{B}}, \mathbb{V}_{\mathrm{B}} \rangle$ in the following way.

- In Stage 1 and 2 (of $\langle \mathbb{P}_{\mathrm{B}}, \mathbb{V}_{\mathrm{B}} \rangle$), P^* forwards all messages from external V_{B} to internal $\mathbb{P}^*$ and forwards all messages from internal $\mathbb{P}^*$ to external V_{B}. (Notice that the verifier of $\langle P_{\mathrm{B}}, V_{\mathrm{B}} \rangle$ and that of $\langle \mathbb{P}_{\mathrm{B}}, \mathbb{V}_{\mathrm{B}} \rangle$ are identical.) Let (h, r_1, c, r_2) be the transcript of these stages.
- In Stage 3-1, P^* forwards α from external V_{B} to internal $\mathbb{P}^*$.
- In Stage 3-2, P^* interacts with internal $\mathbb{P}^*$ as an honest receiver of $\mathsf{SWExtCom}$ and obtains $\widehat{\beta}_1$. Let st be the current state of $\mathbb{P}^*$. Then, P^* rewinds $\mathbb{P}^*$ to the point just before the challenge stage of $\mathsf{SWExtCom}$, interacts with $\mathbb{P}^*$ again, and obtains $\widehat{\beta}_2$. Then, P^* computes $\widetilde{\beta} \stackrel{\mathrm{def}}{=} \mathsf{Extract}(\widehat{\beta}_1, \widehat{\beta}_2)$ as a potential committed value of $\widehat{\beta}_1$ (recall that $\mathsf{Extract}$ is the extracting algorithm of $\mathsf{SWExtCom}$ shown in Figure 3) and sends $\widetilde{\beta}$ to external V_{B}.
- In Stage 3-3, P^* receives γ from V_{B} and sends it to internal $\mathbb{P}^*$ (which is restarted from state st).
- In Stage 3-4, P^* interacts with internal $\mathbb{P}^*$ as an honest receiver of $\mathsf{SWExtCom}$ and obtains $\widehat{\delta}_1$. Then, P^* rewinds $\mathbb{P}^*$ to the point just before the challenge stage of $\mathsf{SWExtCom}$, interacts with $\mathbb{P}^*$ again, and obtains $\widehat{\delta}_2$. Then, P^* computes $\widetilde{\delta} := \mathsf{Extract}(\widehat{\delta}_1, \widehat{\delta}_2)$ and sends $\widetilde{\delta}$ to external V_{B}.

If internal $\mathbb{P}^*$ aborts while interacting with it as above, P^* also aborts.

Before analyzing the success probability of P^*, we first introduce some terminologies regarding the internally emulated interaction between $\mathbb{P}^*$ and $\mathbb{V}_{\mathrm{B}}$. Let $\tau = (h, r_1, c, r_2, \alpha, \widehat{\beta}_1, \gamma, \widehat{\delta}_1)$ be its transcript. Notice that since P^* emulates $\mathbb{V}_{\mathrm{B}}$ for internal $\mathbb{P}^*$ perfectly, we have $\tau \in \mathbb{L}_{\mathrm{B}}$ with probability at least $1/p(n)$.

- We say that a transcript τ_1 up until the commit stage of $\mathsf{SWExtCom}$ in Stage 3-2 is *good* if under the condition that τ_1 is a prefix of τ, the probability that $\tau \in \mathbb{L}_{\mathrm{B}}$ holds is at least $1/2p(n)$.
- We say that a transcript τ_2 up until the commit stage of $\mathsf{SWExtCom}$ in Stage 3-4 is *good* if (1) a prefix of τ_2 up until the commit stage of $\mathsf{SWExtCom}$ in Stage 3-2 is good and (2) under the condition that τ_2 is a prefix of τ, the probability that $\tau \in \mathbb{L}_{\mathrm{B}}$ holds is at least $1/4p(n)$.

Now, we analyze the success probability of P^* as follows. Let $\mathtt{GOOD}_1$ be the event that a prefix of τ up until the commit stage of $\mathsf{SWExtCom}$ in Stage 3-2 is good, and let $\mathtt{GOOD}_2$ be the event that a

prefix of τ up until the commit stage of $\mathsf{SWExtCom}$ in Stage 3-4 is good. From an average argument, we have

$$\Pr[\mathsf{GOOD}_1] \geq \frac{1}{2p(n)} \qquad \text{and} \qquad \Pr[\mathsf{GOOD}_2 \mid \mathsf{GOOD}_1] \geq \frac{1}{4p(n)} \ .$$

Hence, we have

$$\Pr[\mathsf{GOOD}_2] \geq \Pr[\mathsf{GOOD}_1 \wedge \mathsf{GOOD}_2] \geq \frac{1}{8\,(p(n))^2} \ . \tag{1}$$

Also, from the definition of GOOD_2, we have

$$\Pr[\tau \in \mathbb{L}_{\mathrm{B}} \mid \mathsf{GOOD}_2] \geq \frac{1}{4p(n)} \ . \tag{2}$$

Hence, from Equation (1) and (2), we have

$$\Pr[\mathsf{GOOD}_1 \wedge \mathsf{GOOD}_2 \wedge \tau \in \mathbb{L}_{\mathrm{B}}] = \Pr[\mathsf{GOOD}_2 \wedge \tau \in \mathbb{L}_{\mathrm{B}}] \geq \frac{1}{32\,(p(n))^3} \ . \tag{3}$$

Next, we observe that when the transcript up until the commit stage of $\mathsf{SWExtCom}$ in Stage 3-2 is good, $\mathbb{P}^*$ gives a valid commitment of $\mathsf{SWExtCom}$ in Stage 3-2 with probability at least $1/2p(n)$, and similarly, when the transcript up until the commit stage of $\mathsf{SWExtCom}$ in Stage 3-4 is good, $\mathbb{P}^*$ gives a valid commitment of $\mathsf{SWExtCom}$ in Stage 3-4 with probability at least $1/4p(n)$. (This is because when the transcript is in $\mathbb{L}_{\mathrm{B}}$, the $\mathsf{SWExtCom}$ commitments in Stage 3-2 and 3-4 are valid.) Hence, under the condition that $\mathsf{GOOD}_1 \wedge \mathsf{GOOD}_2 \wedge \tau \in \mathbb{L}_{\mathrm{B}}$, the probability that both of $\widehat{\beta}_2$ and $\widehat{\delta}_2$ are valid is at least $1/8(p(n))^2$. Also, from the definition of $\mathbb{L}_{\mathrm{B}}$, both of $\widehat{\beta}_1$ and $\widehat{\delta}_1$ are valid when $\tau \in \mathbb{L}_{\mathrm{B}}$, and furthermore, $\widehat{\beta}_1$ and $\widehat{\beta}_2$ (resp, $\widehat{\delta}_1$ and $\widehat{\delta}_2$) are admissible except with negligible probability. Hence, from Lemma 2, for $\widetilde{\beta} = \mathsf{Extract}(\widehat{\beta}_1, \widehat{\beta}_2)$ and $\widetilde{\delta} = \mathsf{Extract}(\widehat{\delta}_1, \widehat{\delta}_2)$ we have

$$\begin{aligned} &\Pr\left[\widetilde{\beta} = \mathsf{value}(\widehat{\beta}_1) \wedge \widetilde{\delta} = \mathsf{value}(\widehat{\delta}_1) \mid \mathsf{GOOD}_1 \wedge \mathsf{GOOD}_2 \wedge \tau \in \mathbb{L}_{\mathrm{B}}\right] \\ &\geq \frac{1}{8(p(n))^2} - \mathsf{negl}(n) \ . \end{aligned} \tag{4}$$

Hence, from Equation (3) and (4), we have

$$\begin{aligned} &\Pr\left[\mathsf{GOOD}_1 \wedge \mathsf{GOOD}_2 \wedge \tau \in \mathbb{L}_{\mathrm{B}} \wedge \widetilde{\beta} = \mathsf{value}(\widehat{\beta}_1) \wedge \widetilde{\delta} = \mathsf{value}(\widehat{\delta}_1)\right] \\ &\geq \frac{1}{256(p(n))^5} - \mathsf{negl}(n) \ . \end{aligned}$$

Notice that from the definition of $\mathbb{L}_{\mathrm{B}}$, when $\tau \in \mathbb{L}_{\mathrm{B}} \wedge \widetilde{\beta} = \mathsf{value}(\widehat{\beta}_1) \wedge \widetilde{\delta} = \mathsf{value}(\widehat{\delta}_1)$, it holds that $(\alpha, \widetilde{\beta}, \gamma, \widetilde{\delta})$ is an accepting UA proof for $(h, r_1, c, r_2) \in \Lambda$. Hence, we have

$$\Pr\left[(h, r_1, c, r_2, \alpha, \widetilde{\beta}, \gamma, \widetilde{\delta}) \in \mathbf{L}_{\mathrm{B}}\right] \geq \frac{1}{256(p(n))^5} - \mathsf{negl}(n) \ ,$$

which contradicts Lemma 1. □

We next note that a non-black-box simulator can simulate the transcript τ in such a way that $\tau \in \mathbb{L}_{\mathrm{B}}$ holds, and the simulator can additionally output a witness for $\tau \in \mathbb{L}_{\mathrm{B}}$.

Lemma 5 (Simulatability)**.** *Let $\mathbb{L}_{\mathrm{B}}$ be the language defined in Figure 5. Then, for any PPT cheating verifier $\mathbb{V}^*$ against $\langle \mathbb{P}_{\mathrm{B}}, \mathbb{V}_{\mathrm{B}} \rangle$, there exists a PPT simulator $\mathcal{S}$ such that the following hold.*

- *Let $\mathcal{S}_1(x,z)$ be the random variable representing the first output of $\mathcal{S}(x,z)$. Then, the following indistinguishability holds.*

$$\{\mathsf{view}_{\mathbb{V}^*}\left[\mathbb{P}_{\mathrm{B}}(1^n) \leftrightarrow \mathbb{V}^*(1^n,z)\right]\}_{n\in\mathbb{N},z\in\{0,1\}^*} \approx \{\mathcal{S}_1(1^n,z)\}_{n\in\mathbb{N},z\in\{0,1\}^*}$$

- *For any $n \in \mathbb{N}$ and $z \in \{0,1\}^*$, the following holds.*

$$\Pr\left[w \in \mathbf{R}_{\mathbb{L}_{\mathrm{B}}}(\tau) \;\middle|\; \begin{array}{l}(v,w) \leftarrow \mathcal{S}(1^n,z);\\ \textit{reconstruct transcript } \tau \textit{ from view } v \textit{ of } \mathbb{V}^*\end{array}\right] \geq 1 - \mathsf{negl}(n)$$

This lemma can be proven in essentially the same way as the zero-knowledge property of Barak's non-black-box zero-knowledge argument. For completeness, a proof sketch is given below.

Proof sketch of Lemma 5. To simulate the view of $\mathbb{V}^*$, the simulator $\mathcal{S}$ internally invokes $\mathbb{V}^*$ and interacts with it as follows.

- After receiving h and r_1 in Stage 1, $\mathcal{S}$ sends $c \leftarrow \mathsf{Com}_{r_1}(h(\mathbb{V}^*))$ to $\mathbb{V}^*$ in Stage 2-1. Let rand be the randomness that was used in this step.
- After receiving r_2 in Stage 2-2 and α in Stage 3-1, $\mathcal{S}$ computes the second-round UA message β by using witness $(\mathbb{V}^*, \mathsf{rand}, \varepsilon)$ for $(h, r_1, c, r_2) \in \Lambda$ (where ε is an empty string) and then honestly commits to β by using $\mathsf{SWExtCom}$. Let $\widehat{\beta}$ be the commitment and d_1 be the decommitment.
- After receiving γ in Stage 3-3, $\mathcal{S}$ computes the fourth-round UA message δ and then honestly commits to δ by using $\mathsf{SWExtCom}$. Let $\widehat{\delta}$ be the commitment and d_2 be the decommitment.

$\mathcal{S}$ then outputs (v,w), where v is the view of internal $\mathbb{V}^*$ and $w \stackrel{\text{def}}{=} (d_1, d_2, \beta, \delta)$. Let $\tau := (h, r_1, c, r_2, \alpha, \widehat{\beta}, \gamma, \widehat{\delta})$.

We analyze $\mathcal{S}$ as follows. First, from the relatively efficient prover property of universal arguments, $\mathcal{S}$ runs in polynomial time. (Notice that $(\mathbb{V}^*, \mathsf{rand}, \varepsilon)$ is a valid witness for $(h, r_1, c, r_2) \in \Lambda$ and can be verified in time $\mathsf{poly}(T_{\mathbb{V}^*})$, where $T_{\mathbb{V}^*}$ is the running time of $\mathbb{V}^*$.) Next, from the hiding property of Com and the indistinguishability of C_{pub} (Lemma 3), v is indistinguishable from the real view of $\mathbb{V}^*$. Finally, from the definitions of Λ and $\mathbb{L}_{\mathrm{B}}$, we have that $\tau \in \mathbb{L}_{\mathrm{B}}$ and w is its witness. Hence, the lemma follows. □

4.2 Special-purpose Instance-dependent Commitment

In our LRZK protocol, we use a special-purpose instance-dependent commitment scheme $\mathsf{GJS\text{-}Com}$, which is shown in Figure 6. $\mathsf{GJS\text{-}Com}$ is parametrized by two graphs, G and G', and obtained by modifying Hamiltonicity commitment scheme $\mathsf{H\text{-}Com}_{G,r}$ in such a way that the adjacent matrix is committed to by using $\mathsf{AH\text{-}Com}_{G',r}$ instead of Com_r. $\mathsf{GJS\text{-}Com}$ inherits many properties from $\mathsf{H\text{-}Com}$—hiding, binding, and equivocality—and additionally, thanks to the adaptive security of $\mathsf{AH\text{-}Com}$, it provides adaptive security in the following sense: When $G \in \mathbf{L}_{\mathrm{HC}}$ and $G' \in \mathbf{L}_{\mathrm{HC}}$, a commitment to 1 can be explained as a valid commitment to 0, and furthermore, even after being explained as a commitment to 0, it can be decommitted to 1 in a consistent way. Details follow.

Lemma 6 (Hiding and binding)**.** $\mathsf{GJS\text{-}Com}$ *is computationally hiding. Furthermore, it is statistically binding when $G \notin \mathbf{L}_{\mathrm{HC}}$ and $G' \notin \mathbf{L}_{\mathrm{HC}}$.*

Proof . The hiding property follows directly from the hiding property of $\mathsf{AH\text{-}Com}_{G'}$. To see the binding property, observe the following: When $G' \notin \mathbf{L}_{\mathrm{HC}}$, $\mathsf{AH\text{-}Com}_{G'}$ is statistically binding and therefore the matrix that is committed to in the commit phase of $\mathsf{GJS\text{-}Com}$ is uniquely determined except with negligible probability; furthermore, when the committed matrix is uniquely determined and $G \notin \mathbf{L}_{\mathrm{HC}}$, decommitting to both 0 and 1 is clearly impossible; hence, when $G \notin \mathbf{L}_{\mathrm{HC}}$ and $G' \notin \mathbf{L}_{\mathrm{HC}}$, a commitment of $\mathsf{GJS\text{-}Com}$ can be decommitted to both 0 and 1 only with negligible probability. □

Parameters:

- Security parameter n.
- Two graphs G and G', where the number of vertices in G is $q = \mathsf{poly}(n)$ and that in G' is $q' = \mathsf{poly}'(n)$.

Inputs:

- C has secret input $b \in \{0,1\}$, which is the value to be committed to.

Commit phase:

1. R sends the first-round message $r \in \{0,1\}^{3n}$ of Com.
2. **To commit to 0,** C chooses a random permutation π over the vertices of G, computes $H_0 := \pi(G)$, and commits to its adjacent matrix $A_0 = \{a_{0,i,j}\}_{i,j\in[q]}$ by using $\mathsf{AH\text{-}Com}_{G',r}$, i.e., sends $c_{i,j} \leftarrow \mathsf{AH\text{-}Com}_{G',r}(a_{0,i,j})$ for every $i, j \in [q]$.

   To commit to 1, C chooses a random q-cycle graph H_1 and commits to its adjacent matrix $A_1 = \{a_{1,i,j}\}_{i,j\in[q]}$ by using $\mathsf{AH\text{-}Com}_{G',r}$, i.e., sends $c_{i,j} \leftarrow \mathsf{AH\text{-}Com}_{G',r}(a_{1,i,j})$ for every $i, j \in [q]$.

Let $\mathsf{GJS\text{-}Com}_{G,G',r}(\cdot)$ be a function that, on input $b \in \{0,1\}$, computes a commitment to b as above by considering r as the first-round message from the receiver.

Decommit phase:

- **When C committed to 0,** it reveals π and decommits $c_{i,j}$ to $a_{0,i,j}$ for every $i, j \in [q]$. R verifies whether the decommitted matrix is the adjacent matrix of $\pi(G)$.
- **When C committed to 1**, it decommits $c_{i,j}$ to 1 for every i, j such that edge $(i.j)$ is on the q-cycle in H_1 (i.e., every i, j such that $a_{1,i,j} = 1$). R verifies whether the decommitted entries correspond to the edges on a Hamilton cycle.

Let $\mathsf{GJS\text{-}Dec}_r(\cdot)$ be a function that, on input (c, b, ρ) such that $\mathsf{GJS\text{-}Com}_{G,G',r}(b;\rho) = c$, outputs a decommitment to b as above.

Figure 6: Special-purpose instance-dependent commitment GJS-Com.

Lemma 7 (Equivocality)**.** *There exists an algorithm* GJS-EquivToOne *that is parametrized by graphs G, G' and a string $r \in \{0,1\}^{3n}$ and satisfies the following: When $G \in \mathbf{L}_{\mathrm{HC}}$, on input any $w \in \mathbf{R}_{\mathrm{HC}}(G)$ and any c and ρ such that* $\mathsf{GJS\text{-}Com}_{G,G',r}(0;\rho) = c$, $\mathsf{GJS\text{-}EquivToOne}_{G,G',r}$ *outputs a valid decommitment of c to* 1.

Proof . We need to show that, on inputs a commitment c to 0, a witness $w \in \mathbf{R}_{\mathrm{HC}}(G)$, and randomness ρ that was used to compute c, an algorithm GJS-EquivToOne can decommit c to 1.

GJS-EquivToOne decommits c to 1 as follows. From the construction of GJS-Com, commitment c consists of $\{c_{i,j}\}_{i,j\in[q]}$, which are AH-Com commitments to the adjacent matrix of $H_0 = \pi(G)$. To decommit c to 1, GJS-EquivToOne need to decommit some of $\{c_{i,j}\}_{i,j\in[q]}$ to 1 so that decommitted entries of the matrix correspond to the edges on a Hamiltonian cycle in a q-vertex graph. To do such decommitments, GJS-EquivToOne first computes a Hamiltonian cycle $\pi(w)$ in H_0 by using Hamiltonian cycle w in G and permutation π (which is included in ρ). Then, GJS-EquivToOne decommits $c_{i,j}$ to $a_{i,j}$ honestly for every i, j such that (i, j) is an edge on $\pi(w)$. Clearly, this is a valid decommitment to 1. □

Lemma 8 (Adaptive security)**.** *There exists an algorithm* GJS-ExplainAsZero *that is parametrized by graphs G, G' and a string $r \in \{0,1\}^{3n}$ and satisfies the following.*

Correctness. *When $G, G' \in \mathbf{L}_{\mathrm{HC}}$, on input any $w \in \mathbf{R}_{\mathrm{HC}}(G)$ and $w' \in \mathbf{R}_{\mathrm{HC}}(G')$ and any c and ρ_1 such that* $\mathsf{GJS\text{-}Com}_{G,G',r}(1;\rho_1) = c$, $\mathsf{GJS\text{-}ExplainAsZero}_{G,G',r}$ *outputs ρ_0 such that* $\mathsf{GJS\text{-}Com}_{G,G',r}(0;\rho_0) = c$.

Indistinguishability. *For security parameter $n \in \mathbb{N}$, graphs $G, G' \in \mathbf{L}_{\mathrm{HC}}$, witnesses $w \in \mathbf{R}_{\mathrm{HC}}(G)$ and $w' \in \mathbf{R}_{\mathrm{HC}}(G')$, and string $r \in \{0,1\}^{3n}$, consider the following two probabilistic experiments.*

$\underline{\mathrm{EXP}_0^{\mathrm{GJS}}(n, G, G', w, w', r)}$
`/* commit to 0 and decommit it to 1 using equivocality */`

1. *Compute* $c \leftarrow \mathsf{GJS\text{-}Com}_{G,G',r}(0)$.
   Let ρ_0 be the randomness used in GJS-Com.
2. *Compute* $d_1 := \mathsf{GJS\text{-}EquivToOne}_{G,G',r}(c, w, \rho_0)$.
3. *Output* (c, ρ_0, d_1).

$\underline{\mathrm{EXP}_1^{\mathrm{GJS}}(n, G, G', w, w', r)}$
`/* commit & decommit to 1 and explain it as commitment to 0 */`

1. *Compute* $c \leftarrow \mathsf{GJS\text{-}Com}_{G,G',r}(1)$.
   Let ρ_1 be the randomness used in GJS-Com.
   Compute $d_1 := \mathsf{GJS\text{-}Dec}_{G,G',r}(c, 1, \rho)$.
2. *Compute* $\rho_0 := \mathsf{GJS\text{-}ExplainAsZero}_{G,G',r}(c, w, w', \rho_1)$.
3. *Output* (c, ρ_0, d_1).

Let $\mathsf{EXP}_b^{\mathrm{GJS}}(n, G, G', w, w', r)$ *be the random variable representing the output of* $\mathrm{EXP}_b^{\mathrm{GJS}}(n, G, G', w, w', r)$ *for each $b \in \{0,1\}$. Then, the following two ensembles are computationally indistinguishable.*

- $\left\{\mathsf{EXP}_0^{\mathrm{GJS}}(n, G, G', w, w', r)\right\}_{n\in\mathbb{N}, G, G'\in\mathbf{L}_{\mathrm{HC}}, w\in\mathbf{R}_{\mathrm{HC}}(G), w'\in\mathbf{R}_{\mathrm{HC}}(G'), r\in\{0,1\}^{3n}}$
- $\left\{\mathsf{EXP}_1^{\mathrm{GJS}}(n, G, G', w, w', r)\right\}_{n\in\mathbb{N}, G, G'\in\mathbf{L}_{\mathrm{HC}}, w\in\mathbf{R}_{\mathrm{HC}}(G), w'\in\mathbf{R}_{\mathrm{HC}}(G'), r\in\{0,1\}^{3n}}$

Proof. GJS-ExplainAsZero is shown in Figure 7. A key idea behind GJS-ExplainAsZero is that given the ability to explain AH-Com commitments to 0 as AH-Com commitments to 1, we can explain a GJS-Com commitment to 1 (which is AH-Com commitments to the adjacent matrix of a cycle graph) as a GJS-Com commitment to 0 (which is AH-Com commitments to the adjacent matrix of a Hamiltonian graph G). Intuitively, this is because a cycle graph can be transformed to any Hamiltonian graph by appropriately adding edges (which corresponds to changing some entries of the adjacent matrix from 0 to 1).

We first prove the correctness. A key is that since H_0 is defined in such a way that H_0 has the same q-cycle as H_1, for every $i, j \in [q]$ we have only the following three cases regarding the values of $a_{0,i,j}$ and $a_{1,i,j}$.

Case 1. $a_{0,i,j} = 0, a_{1,i,j} = 0$

Case 2. $a_{0,i,j} = 1, a_{1,i,j} = 1$

Case 3. $a_{0,i,j} = 1, a_{1,i,j} = 0$

Parameter:

- Graphs $G, G' \in \mathbf{L}_{\mathrm{HC}}$
- String $r \in \{0,1\}^{3n}$

Input:

- Witnesses $w \in \mathbf{R}_{\mathrm{HC}}(G)$ and $w' \in \mathbf{R}_{\mathrm{HC}}(G')$
- Commitment c and randomness ρ_1 s.t. $\mathsf{GJS\text{-}Com}_{G,G',r}(1;\rho_1) = c$

Output:

1. Parse c as $\{c_{i,j}\}_{i,j\in[q]}$, where each $c_{i,j}$ is a $\mathsf{AH\text{-}Com}$ commitment. Also, from ρ_1, reconstruct $A_1 = \{a_{1,i,j}\}_{i,j\in[q]}$ and $\{\sigma_{1,i,j}\}_{i,j\in[q]}$ such that A_1 is the adjacent matrix of a q-cycle graph H_1 and $\mathsf{AH\text{-}Com}_{G',r}(a_{1,i,j};\sigma_{1,i,j}) = c_{i,j}$ for every $i, j \in [q]$.
2. Choose a random permutation π under the condition that a q-cycle in $H_0 \stackrel{\mathrm{def}}{=} \pi(G)$ coincides with the q-cycle in H_1 (i.e., H_0 has the same cycle as H_1).[a] Let $A_0 = \{a_{0,i,j}\}_{i,j\in[q]}$ be the adjacent matrix of H_0.
3. For every $i, j \in [q]$, define $\sigma_{0,i,j}$ by $\sigma_{0,i,j} \stackrel{\mathrm{def}}{=} \sigma_{1,i,j}$ when $a_{0,i,j} = a_{1,i,j}$ and by $\sigma_{0,i,j} \stackrel{\mathrm{def}}{=} \mathsf{AH\text{-}ExplainAsOne}_{G',r}(w', c_{i,j}, \sigma_{1,i,j})$ when $a_{0,i,j} \neq a_{1,i,j}$.[b]
4. Outputs $\rho_0 \stackrel{\mathrm{def}}{=} (\pi, \{\sigma_{0,i,j}\}_{i,j\in[q]})$.

[a] Given w, this can be done efficiently.
[b] When $a_{0,i,j} \neq a_{i,j}$, it holds that $a_{0,i,j} = 1$ and $a_{1,i,j} = 0$; see the proof.

Figure 7: $\mathsf{GJS\text{-}ExplainAsZero}$.

In particular, we do not have the case that $a_{0,i,j} = 0$ and $a_{1,i,j} = 1$ because when $a_{1,i,j} = 1$, edge (i, j) is on the q-cycle in H_1, and therefore edge (i, j) is also on a q-cycle in H_0 and thus $a_{0,i,j} = 1$. Then, since we have only these three cases, from the property of $\mathsf{AH\text{-}ExplainAsOne}$ we have $\mathsf{AH\text{-}Com}_{G',r}(a_{0,i,j};\sigma_{0,i,j}) = c_{i,j}$ for every i, j such that $a_{0,i,j} \neq a_{1,i,j}$. Therefore, the output ρ_0 satisfies $\mathsf{GJS\text{-}Com}_{G,G',r}(1;\rho) = c$.

We next prove the indistinguishability. Toward this end, we consider the following hybrid experiments.

Hybrid $\mathbf{HYB}_0$ is the same as $\mathrm{EXP}_0^{\mathrm{GJS}}(n, G, G', w, w', r)$. Recall that in $\mathrm{EXP}_0^{\mathrm{GJS}}$, output (c, ρ_0, d_1) is computed as follows:

- Choose $\rho_0 = (\pi, \{\sigma_{0,i,j}\}_{i,j\in[q]})$, where π is a randomly chosen permutation and each $\sigma_{0,i,j}$ is randomly chosen randomness for $\mathsf{AH\text{-}Com}$.
- Compute $c = \{c_{i,j}\}_{i,j\in[q]}$ by $c_{i,j} := \mathsf{AH\text{-}Com}_{G',r}(a_{0,i,j};\sigma_{0,i,j})$ for each $i, j \in [q]$, where $A_0 = \{a_{0,i,j}\}_{i,j\in[q]}$ is the adjacent matrix of $H_0 = \pi(G)$.
- Define $d_1 \stackrel{\mathrm{def}}{=} \{\sigma_{0,i,j}\}_{(i,j)\in\pi(w)}$, where $\pi(w)$ is the set of the edges on the Hamiltonian cycle in H_0 that is obtained by applying π on Hamiltonian cycle w in G.

Hybrid $\mathbf{HYB}_1$ is the same as HYB_0 except that π is chosen as follows:

1. Choose a random q-cycle graph H_1. Let $A_1 = \{a_{1,i,j}\}_{i,j\in[q]}$ be the adjacent matrix of H_1.

2. Choose a random permutation π under the condition that a q-cycle in $H_0 = \pi(G)$ coincides with the q-cycle in H_1.

Hybrid $\mathbf{HYB}_2$ is the same as HYB_1 except for the following.

- $c_{i,j}$ is computed by $c_{i,j} := \mathsf{AH\text{-}Com}_{G',r}(a_{1,i,j}; \sigma_{1,i,j})$ for every $i, j \in [q]$, where $\sigma_{1,i,j}$ is randomly chosen randomness.
- $\sigma_{0,i,j}$ is defined by $\sigma_{0,i,j} \stackrel{\mathrm{def}}{=} \sigma_{1,i,j}$ when $a_{0,i,j} = a_{1,i,j}$ and by $\sigma_{0,i,j} \stackrel{\mathrm{def}}{=} \mathsf{AH\text{-}ExplainAsOne}_{G',r}(w', c_{i,j}, \sigma_{1,i,j})$ when $a_{0,i,j} \neq a_{1,i,j}$.

Hybrid $\mathbf{HYB}_3$ is the same as $\mathrm{EXP}_1^{\mathrm{GJS}}(n, G, G', w, w', r)$.

From a hybrid argument, we can show the indistinguishability of $\mathrm{EXP}_0^{\mathrm{GJS}}$ and $\mathrm{EXP}_1^{\mathrm{GJS}}$ by showing the indistinguishability of each neighboring hybrids.

Claim 1. *The outputs of HYB_0 and HYB_1 are identically distributed.*

Proof. HYB_0 and HYB_1 differ only in the way π is chosen. However, the distribution of π is uniformly random in both hybrids. (In particular, the distribution of π is uniformly random in HYB_1 since H_1 is chosen randomly.) Hence, the claim follows. □

Claim 2. *The outputs of HYB_1 and HYB_2 are computationally indistinguishable.*

Proof. We first remark that, as noted above, we have $a_{0,i,j} = 1$ and $a_{1,i,j} = 0$ when $a_{0,i,j} \neq a_{1,i,j}$. Because of this, HYB_1 and HYB_2 differ only in that for every i, j such that $a_{0,i,j} \neq a_{1,i,j}$,

- in the case of HYB_1, $c_{i,j}$ is a commitment to 1 and $\sigma_{0,i,j}$ is randomly chosen randomness that was used to generate $c_{i,j}$, whereas
- in the case of HYB_2, $c_{i,j}$ is a commitment to 0 and $\sigma_{0,i,j}$ is the randomness that was computed by $\mathsf{AH\text{-}ExplainAsOne}$.

Hence, the indistinguishability follows from the adaptive security of $\mathsf{AH\text{-}Com}$. In particular, we can prove the indistinguishability by considering a sequence of intermediate hybrids $\mathrm{HYB}_{1,0}, \ldots, \mathrm{HYB}_{1,q^2}$ such that

- $\mathrm{HYB}_{1,0}$ is the same as HYB_1, and
- for every $u, v \in [q]$, $\mathrm{HYB}_{1,(u-1)q+v}$ is the same as $\mathrm{HYB}_{1,(u-1)q+v-1}$ except that $c_{u,v}$ and $\sigma_{0,u.v}$ are computed in the same way as in HYB_2,

and then proving the indistinguishability of each neighboring intermediate hybrids by designing an adversary against the adaptive security of $\mathsf{AH\text{-}Com}$ in a straight-forward manner so that, depending on the value of (c, ρ_1) that it receives externally, it internally emulates either $\mathrm{HYB}_{1,(u-1)q+v}$ or $\mathrm{HYB}_{1,(u-1)q+v-1}$ (i.e., when c is a commitment to 1 and ρ is its randomness, the adversary internally emulates $\mathrm{HYB}_{1,(u-1)q+v-1}$, and when c is a commitment to 0 and ρ is the randomness that is generated by $\mathsf{AH\text{-}ExplainAsOne}$, the adversary internally emulates $\mathrm{HYB}_{1,(u-1)q+v}$). □

Claim 3. *The outputs of HYB_2 and HYB_3 are identically distributed.*

Proof. It can be seen by inspection that in HYB_2, the output (c, ρ_0, d_1) is computed in exactly the same way as in $\mathrm{EXP}_1^{\mathrm{GJS}}$. Hence, the claim follows. □

From these claims, we obtain the indistinguishability of $\mathrm{EXP}_0^{\mathrm{GJS}}$ and $\mathrm{EXP}_1^{\mathrm{GJS}}$. This concludes the proof of Lemma 8. □

5 Our Leakage-resilient Zero-knowledge Argument

In this section, by using the two building blocks $\langle \mathbb{P}_B, \mathbb{V}_B \rangle$ and GJS-Com in Section 4, we construct a constant-round LRZK argument system for $\mathcal{NP}$.

Theorem 1. *Assume the existence of collision-resistant hash function family. Then, there exists a constant-round public-coin leakage-resilient zero-knowledge argument system for $\mathcal{NP}$.*

Proof. Our leakage-resilient zero-knowledge argument system LR-ZK is shown in Figure 8. (As usual, we obtain an argument system for $\mathcal{NP}$ by constructing that for a specific $\mathcal{NP}$-complete language, namely $\mathbf{L}_{\mathrm{HC}}$.) Since $\langle \mathbb{P}_B, \mathbb{V}_B \rangle$ can be constructed from any collision-resistant hash function family and SWExtCom can be constructed from any one-way function (which can be obtained from any collision-resistant hash function family), LR-ZK can be constructed from any collision-resistant hash function family. Also, by inspection, it can be seen that LR-ZK is public-coin and has constant number of rounds.

In the following, we prove completeness and soundness in Section 5.1 and leakage-resilient zero-knowledgeness in Section 5.2.

5.1 Completeness and Soundness

Lemma 9. LR-ZK *is an argument system for* $\mathbf{L}_{\mathrm{HC}}$.

Proof. Completeness follows directly from the equivocality of GJS-Com, so we focus on proving soundness.

For any cheating PPT prover P^*, we show that P^* cannot give an accepting proof for a false statement $G \notin \mathbf{L}_{\mathrm{HC}}$ except with negligible probability. Notice that from the soundness of $\langle \mathbb{P}_B, \mathbb{V}_B \rangle$, the statement τ generated in Stage 1 satisfies $\tau \notin \mathbb{L}_B$ except with negligible probability. Hence, it suffices to show that under the condition that $\tau \notin \mathbb{L}_B$ (and hence $G' \notin \mathbf{L}_{\mathrm{HC}}$), P^* cannot give an accepting proof except with negligible probability.

A key is that when $G \notin \mathbf{L}_{\mathrm{HC}}$ and $G' \notin \mathbf{L}_{\mathrm{HC}}$, $\mathsf{GJS\text{-}Com}_{G,G'}$ is statistically binding, and therefore the matrix that is committed to in Stage 2-1 is uniquely determined in each of the n iterations except with negligibly probability. Using this fact, we can prove the soundness in essentially the same way as the soundness of Blum's Hamiltonicity protocol. Specifically, when the committed matrix is uniquely determined in each of the n iterations, P^* can give a valid response in Stage 2-3 with probability at most $1/2$ in each of n iterations. (This is because, when $G' \notin \mathbf{L}_{\mathrm{HC}}$, no Hamiltonian graph is isomorphic to G'.) Hence, under the condition that $\tau \notin \mathbb{L}_B$, P^* can give an accepting proof with only negligible probability. This completes the proof of soundness. □

5.2 Leakage-resilient Zero-knowledgeness

Lemma 10. LR-ZK *is leakage-resilient zero-knowledge.*

In the following, we prove this lemma only w.r.t. a simplified version of LR-ZK in which Stage 2-1, 2-2, and 2-3 are executed only once (instead of executed n times in parallel). The proof w.r.t. the original version of LR-ZK can be obtained by modifying the following proof in a straight-forward way by using the fact that our simulator runs in a "straight-line" manner.

Proof. Without loss of generality, we assume that after receiving each message from the prover, the cheating verifier makes exactly a single leakage query. To see that we indeed do not lose generality, observe that instead of making two queries f_1 and f_2, the cheating verifier can always query a single query f such that, on input witness w and prover's randomness tape, it computes the first leakage $L_1 :=$

Input.

- Common input is graph $G \in \mathbf{L}_{\mathrm{HC}}$.
  Let $n \stackrel{\mathrm{def}}{=} |G|$, and q be the number of vertices in G.
- Private input to the prover P is witness $w \in \mathbf{R}_{\mathrm{HC}}(G)$.

Stage 1.

- P and V execute special-purpose encrypted Barak's preamble $\langle \mathbb{P}_{\mathrm{B}}, \mathbb{V}_{\mathrm{B}} \rangle$. Let τ be the transcript.
- P and V reduce statement "$\tau \in \mathbb{L}_{\mathrm{B}}$" to Hamiltonicity problem via general $\mathcal{NP}$ reduction. Let G' be the graph that P and V obtained. Let q' be the number of vertices in G'.

Stage 2.

- V sends the first-round message $r \in \{0,1\}^{3n}$ of Com to P.
- P and V do the following for n times in parallel.
  1. P commits to a $q' \times q'$ zero matrix in a bit-by-bit manner by using $\mathsf{GJS\text{-}Com}_{G,G',r}$. That is, P sends $c_{i,j} \leftarrow \mathsf{GJS\text{-}Com}_{G,G',r}(0)$ to V for every $i, j \in [q']$. Let $\rho_{i,j}$ be the randomness that was used to compute $c_{i,j}$.
  2. V sends a random bit $ch \in \{0,1\}$ to P.
  3. **When $ch = 0$:**
     - P chooses a random permutation π and computes $H_0 := \pi(G')$. Let $A_0 = \{a_{0,i,j}\}_{i,j \in [q']}$ be the adjacent matrix of H_0.
     - P sends π to V and decommits the $\mathsf{GJS\text{-}Com}$ commitments in Stage 2-1 to A_0 by using the equivocality of $\mathsf{GJS\text{-}Com}$. That is, for every $i, j \in [q]$, P sends an honest decommitment $d_{i,j} := \mathsf{GJS\text{-}Dec}_{G,G',r}(c_{i,j}, 0, \rho_{i,j})$ to V when $a_{0,i,j} = 0$ and sends a fake decommitment $d_{i,j} := \mathsf{GJS\text{-}EquivToOne}_{G,G',r}(c_{i,j}, w_0, \rho_{i,j})$ to V when $a_{0,i,j} = 1$.
     - V computes $H_0 = \pi(G')$ and verifies whether the decommitted matrix is equal to the adjacent matrix of H_0.

     When $ch = 1$:
     - P chooses a random q'-cycle graph H_1. Let $A_1 = \{a_{1,i,j}\}_{i,j \in [q']}$ be the adjacent matrix of H_1.
     - P decommits $c_{i,j}$ to $a_{1,i,j}$ for every i, j such that $a_{1,i,j} = 1$ (i.e., for every i, j such that edge (i, j) is on the q'-cycle of H_1). That is, for every such i and j, P sends a fake decommitment $d_{i,j} := \mathsf{GJS\text{-}EquivToOne}_{G,G',r}(c_{i,j}, w_0, \rho_{i,j})$ to V.
     - V checks whether the decommitted entries of the matrix correspond to the edges on a q'-cycle.

Figure 8: Constant-round leakage-resilient zero-knowledge argument $\mathsf{LR\text{-}ZK}$.

$f_1(w, \mathsf{tape})$, chooses the second query f_2 adaptively, computes the second leakage $L_2 := f_2(w, \mathsf{tape})$, and outputs (L_1, L_2).

In the following, we describe our simulator, observe that our simulator obtains the same amount

of leakage as the adversary, and prove the indistinguishability of views.

5.2.1 Description of the Simulator

Given input (G, z) and access to leakage oracle $\mathcal{L}_w$, our simulator $\mathcal{S}$ simulates the view of cheating verifier V^* by internally invoking $V^*(G, z)$ and interacting with it as follows.

Simulating messages and leakages in Stage 1. Roughly speaking, $\mathcal{S}$ simulates the messages in Stage 1 by interacting with V^* in the same way as the simulator of $\langle \mathbb{P}_B, \mathbb{V}_B \rangle$ (cf. Lemma 5). To simulate the leakages in Stage 1, on the other hand, $\mathcal{S}$ uses the fact that Stage 1 of LR-ZK is public coin w.r.t. the prover and therefore all the randomness that an honest prover generates during Stage 1 is the messages themselves (specifically, $\mathcal{S}$ simulates the leakages by considering the messages msgs that it sent to V^* thus far as the randomness of the prover). An issue is that due to the existence of leakage queries, $\mathcal{S}$ cannot use the simulator of $\langle \mathbb{P}_B, \mathbb{V}_B \rangle$ in a modular way for simulating the prover messages. Nonetheless, it can be shown that $\mathcal{S}$ can still use the technique used in the simulator of $\langle \mathbb{P}_B, \mathbb{V}_B \rangle$ as long as the length of the leakages is bounded by n^2. (Notice that when the length of leakage exceeds n^2, $\mathcal{S}$ can simply obtain the Hamiltonian cycle w of G from $\mathcal{L}_w$.[12])

Formally, $\mathcal{S}$ interacts with V^* as follows.

1. After receiving h and r_1 from V^*, $\mathcal{S}$ sends $c \leftarrow \mathsf{Com}_{r_1}(h(V^*))$ to V^*. Let rand be the randomness that was used in this step.
   Leakage query: When V^* makes a leakage query f, $\mathcal{S}$ does the following.
   - Let $\mathsf{tape} := c$.
   - If the output length of f is more than n^2, $\mathcal{S}$ obtains w from $\mathcal{L}_w$ and returns $f(w, \mathsf{tape})$ to V^*.
   - Otherwise, $\mathcal{S}$ queries $f(\cdot, \mathsf{tape})$ to $\mathcal{L}_w$, obtains reply L from $\mathcal{L}_w$, and forwards L to V^*.

   If $\mathcal{S}$ obtained w, from now on $\mathcal{S}$ interacts with V^* in the same way as an honest prover. Otherwise, $\mathcal{S}$ does the following.

2. After receiving r_2 and α from V^*, $\mathcal{S}$ computes the second-round UA message β by using witness (V^*, rand, L) and then honestly commits to β by using $\mathsf{SWExtCom}$. Let $\widehat{\beta}$ be the commitment and d_1 be the decommitment.
   Leakage query: When V^* makes a leakage query f, $\mathcal{S}$ sets $\mathsf{tape} := \mathsf{msgs}$, queries $f(\cdot, \mathsf{tape})$ to $\mathcal{L}_w$, and forwards the reply from $\mathcal{L}_w$ to V^*, where msgs is the messages that $\mathcal{S}$ sent to V^* thus far.

3. After receiving γ from V^*, $\mathcal{S}$ computes the fourth-round UA message δ and then honestly commits to δ by using $\mathsf{SWExtCom}$. Let $\widehat{\delta}$ be the commitment and d_2 be the decommitment.
   Leakage query: When V^* makes a leakage query f, $\mathcal{S}$ answers it as above.

Let $\tau \stackrel{\text{def}}{=} (h, r_1, c, r_2, \alpha, \widehat{\beta}, \gamma, \widehat{\delta})$ and $\bar{w} \stackrel{\text{def}}{=} (d_1, d_2, \beta, \delta)$. Since (V^*, rand, L) is a valid witness for $(h, r_1, c, r_2) \in \Lambda$, we have $\tau \in \mathbb{L}_B$ and $\bar{w} \in \mathbf{R}_{\mathbb{L}_B}(\tau)$. Let G' and w' be the graph and its Hamiltonian cycle that are obtained by reducing statement "$\tau \in \mathbb{L}_B$" to Hamiltonicity problem through the $\mathcal{NP}$ reduction.

[12] We assume that w is encoded as an adjacent matrix of n-vertex graph.

Simulating messages Stage 2. If $\mathcal{S}$ obtained w during Stage 1, it interacts with V^* in the same way as an honest prover. Otherwise, $\mathcal{S}$ interacts with V^* as follows. The idea is that, since $\mathcal{S}$ know a witness w' for $G' \in \mathbf{L}_{\mathrm{HC}}$, $\mathcal{S}$ can correctly respond to the challenge for both $ch = 0$ and $ch = 1$ by committing to a random permutation of G' in the first step.

1. $\mathcal{S}$ chooses a random permutation π and computes $H := \pi(G')$. Then, $\mathcal{S}$ commits to the adjacent matrix $A = \{a_{i,j}\}_{i,j\in[q']}$ of H by using $\mathsf{GJS\text{-}Com}_{G,G',r}$. That is, $\mathcal{S}$ sends $c_{i,j} \leftarrow \mathsf{GJS\text{-}Com}_{G,G',r}(a_{i,j})$ to V^* for every $i, j \in [q']$.

   Let $\{\rho_{i,j}\}_{i,j\in[q']}$ be the randomness that was used in the $\mathsf{GJS\text{-}Com}$ commitments and $\pi(w')$ be the Hamiltonian cycle in H that is obtained by applying π on Hamiltonian cycle w' in G'.

2. $\mathcal{S}$ receives a random bit $ch \in \{0, 1\}$ from V^*.

3. **When $ch = 0$,** $\mathcal{S}$ sends π to V and decommits $c_{i,j}$ to $a_{i,j}$ honestly for every $i, j \in [q']$. That is, $\mathcal{S}$ sends $d_{i,j} := \mathsf{GJS\text{-}Dec}_{G,G',r}(c_{i,j}, a_{i,j}, \rho_{i,j})$ to V for every $i, j \in [q']$.

   When $ch = 1$, $\mathcal{S}$ decommits $c_{i,j}$ to 1 honestly for every i, j such that edge (i, j) is on the Hamiltonian cycle $\pi(w')$ in H. That is, for every such i and j, $\mathcal{S}$ sends $d_{i,j} := \mathsf{GJS\text{-}Dec}_{G,G',r}(c_{i,j}, a_{i,j}, \rho_{i,j})$ to V^*.

Simulating leakage queries in Stage 2. When V^* makes a leakage query f, $\mathcal{S}$ simulates the leakage as follows. Recall that in Stage 2-1, an honest prover commits to a $q' \times q'$ zero matrix whereas $\mathcal{S}$ commits to the adjacent matrix of H. Hence, $\mathcal{S}$ simulates the leakage by "explaining" commitments $\{c_{i,j}\}_{i,j\in[q']}$ to $\{a_{i,j}\}_{i,j\in[q']}$ as commitments to $\{0\}$ by using the adaptive security of $\mathsf{GJS\text{-}Com}$ and the knowledge of w'. Concretely, $\mathcal{S}$ does the following.

- First, for each $i, j \in [q']$, $\mathcal{S}$ constructs a function $F_{i,j}(\cdot)$ such that on input w, it outputs $\widetilde{\rho}_{i,j}$ such that $\mathsf{GJS\text{-}Com}_{G,G',r}(0;\widetilde{\rho}_{i,j}) = c_{i,j}$. Concretely, when $a_{i,j} = 0$, $F_{i,j}(\cdot)$ is a function that always outputs $\rho_{i,j}$, and when $a_{i,j} = 1$, $F_{i,j}(\cdot) \stackrel{\mathrm{def}}{=} \mathsf{GJS\text{-}ExplainAsZero}_{G,G',r}(c_{i,j}, \cdot, w', \rho_{i,j})$.
- Next, $\mathcal{S}$ constructs a function $\widetilde{f}$ such that on input w, it computes $\mathsf{tape} := \mathsf{msgs} \| \{F_{i,j}(w)\}_{i,j\in[q']}$ and outputs $f(w, \mathsf{tape})$.
- Finally, $\mathcal{S}$ queries $\widetilde{f}$ to $\mathcal{L}_w$ and forwards the reply from $\mathcal{L}_w$ to V^*.

5.2.2 Amount of Total Leakage

From the construction of $\mathcal{S}$, it always obtains at most the same amount of leakages as V^*. Hence, we have

$$\Pr\left[\mathsf{IDEAL}_{\mathcal{S}}(x, w_x, z) = \bot\right] = 0 \ .$$

5.2.3 Indistinguishability of Views

We show that for any cheating verifier V^* and any sequence $\{w_G\}_{G\in\mathbf{L}_{\mathrm{HC}}}$ such that $w_G \in \mathbf{R}_{\mathrm{HC}}(G)$, the following indistinguishability holds.

$$\{\mathsf{REAL}_{V^*}(G, w_G, z)\}_{G\in\mathbf{L}_{\mathrm{HC}},z\in\{0,1\}^*} \approx \{\mathsf{IDEAL}_{\mathcal{S}}(G, w_G, z)\}_{G\in\mathbf{L}_{\mathrm{HC}},z\in\{0,1\}^*} \ . \tag{5}$$

Toward this end, we consider the following hybrid experiments.

Hybrid $\mathbf{HYB}_0(G, z)$ is identical with experiment $\mathrm{REAL}_{V^*}(G, w, z)$. That is, V^* interacts with honest $P(G, w)$ and obtains leakage that is computed honestly based on witness w and the prover's randomness. The outputs of this hybrid is the view of V^*.

Hybrid $\mathbf{HYB}_1(G, z)$ is the same as HYB_0 except for the following.

- In Stage 1, an honest prover is replaced with the simulator. That is, c is computed by committing to $h(V^*)$, $\widehat{\beta}$ is computed by committing to β, and $\widehat{\delta}$ is computed by committing to δ.

  Let τ and $\bar{w}$ be the statement and the witness generated in it. Let G' and w' be the graph and its Hamiltonian cycle that are obtained by reducing statement "$\tau \in \mathbb{L}_{\mathrm{B}}$" to Hamiltonicity problem through the $\mathcal{NP}$ reduction.
- The leakage queries during Stage 1 are answered as in the simulator, i.e., by considering that the randomness generated by the prover during Stage 1 is equal to the messages sent to V^* during Stage 1.

Hybrid $\mathbf{HYB}_2(G, z)$ is the same as HYB_1 except for the following.

- As in $\mathcal{S}$, a random permutation π is chosen randomly at the beginning of Stage 2-1. Let $H \stackrel{\mathrm{def}}{=} \pi(G')$, and $A = \{a_{i,j}\}_{i,j\in[q']}$ be the adjacent matrix of H. Let $\pi(w')$ be the Hamiltonian cycle in H that is obtained by applying π on Hamiltonian cycle w' in G'.

  We remark that in this hybrid, the prover still commits to a $q' \times q'$ zero matrix as in HYB_1. Also, the leakage query immediately after Stage 2-1 is answered as in HYB_1 (in particular, when the leakage query is answered, π is not included in the randomness generated by the prover in Stage 2-1).
- In Stage 2-3, graph H_0 or H_1 is chosen as follows.

  When $ch = 0$, $H_0 := H$.

  When $ch = 1$, H_1 is the graph that is obtained by removing every edge in H except for the ones on Hamiltonian cycle $\pi(w')$.

  The leakage query immediately after Stage 2-3 is answered as in HYB_1 by considering that H_0 or H_1 was chosen during Stage 2-3 as in HYB_1.

Hybrid $\mathbf{HYB}_3(G, z)$ is the same as HYB_2 except for the following.

- In Stage 2-1, for every $i, j \in [q']$, commitment $c_{i,j}$ is computed by committing to $a_{i,j}$ (instead of 0), i.e., $c_{i,j} \leftarrow \mathsf{GJS\text{-}Com}_{G,G',r}(a_{i,j})$.
- In Stage 2-3, for every $i, j \in [q']$, if commitment $c_{i,j}$ need to be decommitted, it is decommitted to $a_{i,j}$ honestly.
- When the leakage queries are answered during Stage 2, the randomness $\rho_{i,j}$ used for computing $c_{i,j}$ is simulated by $\widetilde{\rho}_{i,j}$ that is computed by function $F_{i,j}$ as in $\mathcal{S}$ for every $i, j \in [q']$.

Hybrid $\mathbf{HYB}_4(G, z)$ is identical with $\mathrm{IDEAL}_{\mathcal{S}}(x, w, z)$. That is, $\mathcal{S}(G, z)$ is executed given access to $\mathcal{L}_w$. The outputs of this hybrid is that of $\mathcal{S}$.

From a hybrid argument, we can obtain Equation (5) by showing that the outputs of each neighboring hybrids are indistinguishable. Let $\mathsf{HYB}_i(x, z)$ be the random variable representing the output of $\mathrm{HYB}_i(x, z)$ for each $i \in \{0, \ldots, 4\}$.

Claim 4. *We have the following indistinguishability.*

$$\{\mathsf{HYB}_0(G, z)\}_{G\in\mathbf{L}_{\mathrm{HC}},z\in\{0,1\}^*} \approx \{\mathsf{HYB}_1(G, z)\}_{G\in\mathbf{L}_{\mathrm{HC}},z\in\{0,1\}^*} \ .$$

Proof. This claim is proven in a similar way to Lemma 5. Specifically, since HYB_1 differs from HYB_0 only in that the fake commitments of Com and $\mathsf{SWExtCom}$ are replaced with real commitments, the indistinguishability follows from the security of $\mathsf{Com}_{\mathrm{pub}}$ and C_{pub} (cf. Section 3.4.2 and 3.6). □

Claim 5. *We have the following indistinguishability.*

$$\{\mathsf{HYB}_1(G,z)\}_{G\in\mathbf{L}_{\mathrm{HC}},z\in\{0,1\}^*} \equiv \{\mathsf{HYB}_2(G,z)\}_{G\in\mathbf{L}_{\mathrm{HC}},z\in\{0,1\}^*} \ .$$

Proof. This claim can be proven by inspection. Observe that HYB_2 differs from HYB_1 only in the way graph H_0 or H_1 is chosen in Stage 2. When $ch = 0$, the distribution of H_0 in HYB_2 is the same as that in HYB_1 since H_0 is obtained both in HYB_2 and HYB_1 by applying a random permutation on G'. When $ch = 1$, the distribution of H_1 in HYB_2 is the same as that in HYB_1 since the Hamiltonian cycle w' in G' is mapped to a random q-cycle by π. Hence, the output of HYB_2 is identically distributed with that of HYB_1. □

Claim 6. *We have the following indistinguishability.*

$$\{\mathsf{HYB}_2(G,z)\}_{G\in\mathbf{L}_{\mathrm{HC}},z\in\{0,1\}^*} \approx \{\mathsf{HYB}_3(G,z)\}_{G\in\mathbf{L}_{\mathrm{HC}},z\in\{0,1\}^*} \ .$$

Proof. Assume for contradiction that for infinitely many $G \in \mathbf{L}_{\mathrm{HC}}$, there exists $z \in \{0,1\}^*$ such that a distinguisher $\mathcal{D}$ distinguishes $\mathsf{HYB}_2(G,z)$ and $\mathsf{HYB}_3(G,z)$ with advantage $1/p(n)$ for a polynomial $p(\cdot)$. Fix any such G and z. To derive a contradiction, we consider the following intermediate hybrids.

Hybrid $\mathbf{HYB}_{2:0}(G,z)$ is identical with $\mathrm{HYB}_2(G,z)$.

Hybrid $\mathbf{HYB}_{2:k}(G,z)$, where $k \in [q'^2]$, is the same as $\mathrm{HYB}_{2:k-1}$ except for the following. Let $u, v \in [q']$ be such that $(u-1)q' + v = k$.

- In Stage 2-1, commitment $c_{u,v}$ is computed by committing to $a_{u,v}$ (instead of 0), i.e., $c_{u,v} \leftarrow \mathsf{GJS\text{-}Com}_{G,G',r}(a_{u,v})$.
- In Stage 2-3, if commitment $c_{u,v}$ need to be decommitted, it is decommitted to $a_{u,v}$ honestly.
- When the leakage queries are answered during Stage 2, the randomness $\rho_{u,v}$ used for computing $c_{u,v}$ is simulated by $\widetilde{\rho}_{u,v}$ that is computed by function $F_{u,v}$ as in $\mathcal{S}$.

Clearly, $\mathrm{HYB}_{2:q'^2}$ is identical with HYB_3. Hence, there exists $k^* \in [q'^2]$ such that the output of $\mathrm{HYB}_{2:k^*-1}$ and that of $\mathrm{HYB}_{2:k^*}$ can be distinguished with advantage $1/q'^2p(n)$. Furthermore, from an average argument, there exists a prefix σ of the execution of HYB_{k^*-1} up until permutation π is chosen in Stage 2-1 (i.e., just before $\{c_{i,j}\}_{i,j\in[q']}$ is sent to V^*) such that under the condition that a prefix of the execution is σ, the output of $\mathrm{HYB}_{2:k^*-1}$ and that of $\mathrm{HYB}_{2:k^*}$ can be distinguished with advantage $1/q^2p(n)$. Notice that σ determines G', w', r, $\{a_{i,j}\}_{i,j\in[q']}$.

We derive a contradiction by showing that we can break the adaptive security of $\mathsf{GJS\text{-}Com}$ (Lemma 8). Specifically, we show that $\mathsf{EXP}_0^{\mathrm{GJS}}(n,G,G',w,w',r)$ and $\mathsf{EXP}_1^{\mathrm{GJS}}(n,G,G',w,w',r)$ can be distinguished with advantage $1/q^2p(n)$. Toward this end, consider the following distinguisher $\mathcal{D}'$.

- Externally, $\mathcal{D}'$ takes (c,ρ_0,d_1) as well as (n,G,G',w,w',r) as input. $\mathcal{D}'$ also takes (σ,z) as non-uniform input.
- Internally, $\mathcal{D}'$ invokes V^* and simulates $\mathrm{HYB}_{2:k^*-1}(G,z)$ for V^* from σ honestly except for the following. Let $u^*, v^* \in [q']$ be such that $(u^*-1)q' + v^* = k^*$. Notice that it must hold that $a_{u^*,v^*} = 1$ since $\mathrm{HYB}_{2:k^*}$ is identical with $\mathrm{HYB}_{2:k^*-1}$ when $a_{u^*,v^*} = 0$.
  - In Stage 2-1, commitment c_{u^*,v^*} is defined by setting $c_{u^*,v^*} := c$.

– In Stage 2-3, when commitment c_{u^*,v^*} is decommitted, it is decommitted to $a_{u^*,v^*} = 1$ by sending d_1.
– When the leakage queries are answered during Stage 2, the randomness ρ_{u^*,v^*} used for computing c_{u^*,v^*} is simulated by setting $\widetilde{\rho}_{u^*,v^*} := \rho_0$.

Let view be the view of V^*. Then, $\mathcal{D}'$ outputs $\mathcal{D}(\mathsf{view})$.

When $(c, \rho_0, d_1) \leftarrow \mathsf{EXP}_0^{\mathrm{GJS}}(n, G, G', w, w', r)$ (i.e., when c is a commitment to 0, ρ_0 is the randomness that is used to generate c, and d_1 is a decommitment to 1 that is computed by GJS-EquivToOne), $\mathcal{D}'$ emulates $\mathrm{HYB}_{2:k^*-1}$ for V^* perfectly. On the other hand, when $(c, \rho_0, d_1) \leftarrow \mathsf{EXP}_1^{\mathrm{GJS}}(n, G, G', w, w', r)$ (i.e., when c is a commitment to 1, ρ_0 is randomness that is computed by GJS-ExplainAsZero, and d_1 is a decommitment to 1 that is computed honestly), $\mathcal{D}'$ emulates $\mathrm{HYB}_{2:k^*}$ for V^* perfectly. Hence, from our assumption, $\mathcal{D}'$ distinguishes $\mathsf{EXP}_0^{\mathrm{GJS}}(n, G, G', w, w', r)$ and $\mathsf{EXP}_1^{\mathrm{GJS}}(n, G, G', w, w', r)$ with advantage $1/q^2 p(n)$, and therefore we reach a contradiction. □

Claim 7. *We have the following indistinguishability.*

$$\{\mathsf{HYB}_3(G, z)\}_{G \in \mathbf{L}_{\mathrm{HC}}, z \in \{0,1\}^*} \equiv \{\mathsf{HYB}_4(G, z)\}_{G \in \mathbf{L}_{\mathrm{HC}}, z \in \{0,1\}^*} \ .$$

Proof. In HYB_3, the prover interacts with V^* in exactly the same way as $\mathcal{S}$. Hence, the claim follows. □

From Claim 4, 5, 6, and 7, we obtain Equation (5). This concludes the proof of Lemma 10. □

This concludes the proof of Theorem 1. □

References

[AGP14] Prabhanjan Ananth, Vipul Goyal, and Omkant Pandey. Interactive proofs under continual memory leakage. In *CRYPTO*, pages 164–182, 2014.

[AK96] Ross Anderson and Markus Kuhn. Tamper resistance: A cautionary note. In *WOEC*, pages 1–11, 1996.

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *FOCS*, pages 106–115, 2001.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[BCH12] Nir Bitansky, Ran Canetti, and Shai Halevi. Leakage-tolerant interactive protocols. In *TCC*, pages 266–284, 2012.

[BDL14] Nir Bitansky, Dana Dachman-Soled, and Huijia Lin. Leakage-tolerant computation with input-independent preprocessing. In *CRYPTO*, pages 146–163, 2014.

[BG08] Boaz Barak and Oded Goldreich. Universal arguments and their applications. *SIAM Journal on Computing*, 38(5):1661–1694, 2008.

[BGJ+13] Elette Boyle, Sanjam Garg, Abhishek Jain, Yael Tauman Kalai, and Amit Sahai. Secure computation against adaptive auxiliary information. In *CRYPTO*, pages 316–334, 2013.

[BGJK12] Elette Boyle, Shafi Goldwasser, Abhishek Jain, and Yael Tauman Kalai. Multiparty computation secure against continual memory leakage. In *STOC*, pages 1235–1254, 2012.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *STOC*, pages 494–503, 2002.

[CPS16] Kai-min Chung, Rafael Pass, and Karn Seth. Non-Black-Box Simulation from One-Way Functions and Applications to Resettable Security. *SIAM Journal on Computing*, 45(2):415–458, 2016.

[DPP98] Ivan Damgård, Torben P. Pedersen, and Birgit Pfitzmann. Statistical secrecy and multibit commitments. *IEEE Transactions on Information Theory*, 44(3):1143–1151, 1998.

[FS89] Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In *CRYPTO*, pages 526–544, 1989.

[GJS11] Sanjam Garg, Abhishek Jain, and Amit Sahai. Leakage-resilient zero knowledge. In *CRYPTO*, pages 297–315, 2011.

[GK96] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–190, 1996.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, August 2001.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, May 2004.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[HNO+09] Iftach Haitner, Minh-Huyen Nguyen, Shien Jin Ong, Omer Reingold, and Salil P. Vadhan. Statistically hiding commitments and statistical zero-knowledge arguments from any one-way function. *SIAM Journal on Computing*, 39(3):1153–1218, 2009.

[Koc96] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *CRYPTO*, pages 104–113, 1996.

[LZ11] Yehuda Lindell and Hila Zarosim. Adaptive zero-knowledge proofs and adaptively secure oblivious transfer. *Journal of Cryptology*, 24(4):761–799, 2011.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *STOC*, pages 33–43, 1989.

[OPV15] Rafail Ostrovsky, Giuseppe Persiano, and Ivan Visconti. Impossibility of black-box simulation against leakage attacks. In *CRYPTO*, pages 130–149, 2015.

[Pan14] Omkant Pandey. Achieving constant round leakage-resilient zero-knowledge. In *TCC*, pages 146–166, 2014.

[PR08a] Rafael Pass and Alon Rosen. Concurrent Nonmalleable Commitments. *SIAM Journal on Computing*, 37(6):1891–1925, 2008.

[PR08b] Rafael Pass and Alon Rosen. New and Improved Constructions of Nonmalleable Cryptographic Protocols. *SIAM Journal on Computing*, 38(2):702–752, 2008.

[PW09] Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In *TCC*, pages 403–418, 2009.

[QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In *E-smart*, pages 200–210, 2001.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.