Constant-Round Leakage-Resilient Zero-Knowledge Arguments of Knowledge for NP

Hongda Li, Qihua Niu, Guifang Huang

[1] The Data Assurance and Communication Security Research Center
[2] State Key Lab of Information Security, Institute of Information Engineering
Chinese Academy of Sciences, Beijing 100093, China

Abstract. Garg, Jain, and Sahai first consider zero knowledge proofs in the presence of leakage on the local state of the prover, and present a leakage-resilient-zero-knowledge proof system for HC (Hamiltonian Cycle) problem. Their construction is called $(1+\varepsilon)$-leakage-resilient zero-knowledge, for any constant $\varepsilon > 0$, because the total length of the leakage the simulator needs is $(1+\varepsilon)$ times as large as that of the leakage received by the verifier. In recent, Pandey provides a constant-round leakage-resilient zero-knowledge argument satisfying the ideal requirement of $\varepsilon = 0$. Whether there exist constant round leakage-resilient zero-knowledge arguments of knowledge for all NP languages is an interesting problem. This paper focuses on this problem and presents a constant-round construction of leakage-resilient zero-knowledge arguments of knowledge for the HC problem.

Key word: zero-knowledge proofs, proofs of knowledge, leakage-resilient, non-black-box simulation, constant-round.

1 Introduction

The zero-knowledge proofs, first introduced by Goldwasser et al. in [4], allow the prover to convince the verifier of the validity of a statement $x \in L$, while providing no additional knowledge to the verifier apart from the fact that the statement is indeed true. A fundamental variant, known as zero knowledge arguments, relaxed the soundness condition only with respect to efficient provers. The zero-knowledge property is formalized by requiring that for any probabilistic polynomial-time (PPT) verifier V^* there exists a PPT simulator such that the view of V^* interaction with the prover can be "indistinguishably simulated" by the simulator on input x.

The conventional zero knowledge property assume that the prover's internal state, including the witness and the random coins, is perfectly hidden from the verifier. The verifier is given only black-box access to the honest prover's algorithm during the interaction. After a class of attack (known as side-channel attacks) is noticed, this idealized assumption becomes unreasonable and is often hard to satisfy in many real setting where an malicious verifier has the ability to obtain leakage of prover's internal

[0] Corresponding author e-mail: lihongda@iie.ac.cn

state by launching a side-channel attacks. In order to deal with this problem, Garg et. al. [13] recently investigated zero knowledge proofs in such a leaking setting (known as leakage-resilient zero-knowledge proof, LR-ZKP), where the malicious verifier is able to learn an arbitrary amount of leakage on the internal state of the honest prover. Such a leakage attack is modeled as leakage-queries in the execution of the protocol, denoted as a series of polynomial-time computable functions $f_1, f_2, ...$ There exist several model to restrict leakage functions. The most general one requires that leakage functions are hard-to-invert. Ideally, LR-ZKP require that no such malicious verifier can learn anything beyond the validity of the assertion and the leakage. Concretely, it is required that, for any malicious verifier V^*, there exists a simulator which is given access to a leakage oracle, such that the output of the simulator and the view of V^* are computationally indistinguishable. [13] defined λ-leakage-resilient zero knowledge, which allows the simulator to obtain at most $\lambda \cdot l$-bits leakage from the leakage oracle if V^* obtains l-bit leakage. [13] provided super-constant-round $(1+\varepsilon)$-leakage-resilient-zero-knowledge proof for HC problem for any $\varepsilon > 0$. Very recently, Pandey in [20] presented a constant-round leakage-resilient zero-knowledge argument satisfying the ideal requirement of $\varepsilon = 0$.

If zero knowledge proofs or arguments are also "proof of knowledge", it is known as zero knowledge proofs or arguments of knowledge (ZKPoK or ZKAoK). In fact, [13] discussed how to modify their protocol such that the modified protocol is a proof of knowledge. The presented method needs a public-coin zero knowledge proof of knowledge, in which the verifier proves to the prover that the previous commitment is "valid". Concretely, the verifier first commit to a random challenge ch. At a later decommitment stage, the verifier prove, by a public-coin ZKPoK , that the committed value is indeed ch. [7] first presented a simple construction of super-constant-round LR-ZKPoK for HC problem. An interesting problem left by [13, 20, 7] is how to construct constant-round leakage-resilient ZKPoK (or ZKAoK) for NP. In this paper, We focus on how to construct constant-round LR-ZKAoK for NP and give a construction for HC problem by means of non-black-box simulation techniques.

1.1 Related Works

In the past few years there have been a few works on leakage-resilient interactive protocols. Bitansky et. al. [9] considered leakage-resilient protocols for general functionality which are secure against semi-honest adversaries in the UC framework. Boyle et. al. [11] studied leakage-resilient multi-party coin tossing protocol. Damgard et. al. [12] considered leakage-resilient two-party secure protocols against semi-honest adversary. Two types of leakage attack are considered in [12]: global leakage model (the adversary use a leakage function on the input and the entire randomness of an honest party) and local leakage model (the adversary chooses different leakage functions at different points of time during the protocol execution). Very recently, Boyle et. al. [10] constructed a general leakage-resilient multiparty computation (MPC) protocol with more strong security notion: an adversary obtaining leakage on the honest party's secret state is guaranteed to learn nothing beyond the input and output of corrupted parties. Garg et. al. [13] first formulated leakage-resilient zero-knowledge proofs and provided a super-constant-round construction for HC problem. Following

[13], [7] showed a construction of leakage-resilient zero-knowledge proofs of knowledge. [20] presented a constant-round leakage-resilient zero-knowledge argument.

1.2 Our results

In this work, we adopt the basic model of [13] and present the first constant-round construction of LR-ZKAoK protocol.

Main result: *Assume that collision-resistant hash functions and pseudorandom permutations exist, and DDH assumption holds. Then, there exists constant-round LR-ZKAoK system for HC.*

The key problem in constructing LR-ZK protocol is how to respond to leakage queries in simulation such that the responses are consistent to the simulated view. To solve this problem, [13] adds a preamble to Blum's 3-round zero knowledge proof. In the preamble, the verifier first commits to its random challenge ch and a random string r' using a extractable commitment scheme originally proposed by Prabhakaran et al. in [21], and then the prover sends a random string r''. In subsequent Blum's protocol, the prover uses $r = r' \oplus r''$ as common random reference of Naor's non-interactive commitment scheme to compute its commitments. There are many challenge-response slots in the extractable commitment scheme. This provides chances for the simulator to learn ch and r' by rewinding the verifier's program. By Learning ch in advance the simulator can prepare its commitment in Blum's protocol according to ch and then can answer this challenge correctly. By obtaining r' the simulator can control r and then have the ability to respond leakage queries under the help of the leakage oracle. The construction in [20] use the same idea but the preamble use non-black-box methods. In this approach, however, black-box knowledge extractors $\mathcal{K}$ cannot work efficiently. Therefore, the protocols in [13, 20] are no longer a proof of knowledge.

To obtain LR-ZKAoK, we modified the construction in [20] such that the following two condition hold: (1) the challenge ch is generated after the prover sends its commitments, as in [15, 16]; (2) the simulator has chance to control over the generated challenge and the prover cannot. Thus, the simulator can simulate the prover's commitments according to a-priori randomly selected challenge and the prover cannot break the soundness. On the other hand, black-box knowledge extractors $\mathcal{K}$ can work efficiently. To this end, we use a jointly coin-flipping protocol to generate ch after the prover sends its commitments. In order to let the simulator have ability to control over ch, a preamble in which the simulator obtains the trapdoors is needed.

As in [20], we use Barak's non-black-box techniques to construct such a preamble. The difference is that we use a new variant of Barak's relation to define a NP-relation $\mathcal{R}_{sim}$. In general Barak's non-black-box simulation, the simulator needs the description of the verifier's strategy, denoted as $desc(V)$, to accomplish a task that the prover cannot achieve no matter what it does. That is, the simulator's advantage over the prover is that it is given the verifier's program.

To improve the simulator's ability, we permit the simulator to receive leakage queries apart from this advantage. We define a variant of Barak's relation by simultaneously using $desc(V)$ and the received leakage queries. The simulator can take advantage of knowing $desc(V)$ and the leakage queries to obtain an instance $\sigma \in L_{\mathcal{R}_{sim}}$ and a corresponding witness w such that $\mathcal{R}_{sim}(\sigma, w) = 1$, and no prover can obtain

$\sigma \in L_{\mathcal{R}_{sim}}$. This advantage over the prover enables the simulator to control over ch and r. In the jointly coin-flipping protocols, we use the modified Yao's garbled-circuit method, known as augmented garbled-circuit method, to achieve the conditional disclosure of a random secret selected by the verifier. Thus, the verifier needs not use a zero-knowledge proof to convince the prover that the coin-flipping protocol is executed honestly.

2 Preliminaries

In this paper, we use some standard notations. If $A(\cdot)$ is a probabilistic algorithm, $A(x)$ is the result of running A on input x and $y = A(x)$ (or $y \leftarrow A(x)$) denote that y is set to $A(x)$. For a finite set $\mathcal{S}$, we denote by $y \in_R S$ that y is uniformly selected from $\mathcal{S}$. U_l denotes uniform distribution on $\{0,1\}^l$. We write $[n]$ for any $n \in \mathbb{N}$ to denote the set $\{1, \cdots, n\}$ and $poly(\cdot)$ to denote an unspecified polynomial.

We use the following standard definitions and tools.

2.1 Leakage-resilient zero-knowledge proof

We first recall the definitions of leakage-resilient zero-knowledge.

Let P and V be a pair of randomized interactive Turing machines, $\langle P, V\rangle(x)$ be a random variable representing the local output of Turing machine V when interacting with machine P on common input x, when the random input to each machine is uniformly and independently chosen. Customarily, machine P is called the prover and machine V is called the verifier. We denote by $\langle P, V\rangle(x) = 1$ ($\langle P, V\rangle(x) = 0$) that machine V accepts (rejects) the proofs given by machine P.

Definition 1. *A pair of interactive Turing machines $\langle P, V\rangle$ is called an interactive proof system for a language L if machine V is polynomial-time and the following two conditions hold:*

- *Completeness: there exists a negligible function c (known as completeness error) such that for every $x \in L$,*
$$\mathbf{Pr}[\langle P, V\rangle(x) = 1] > 1 - c(|x|)$$
- *Soundness: there exists a negligible function s (known as soundness error) such that for every $x \notin L$ and every interactive machine B,*
$$\mathbf{Pr}[\langle B, V\rangle(x) = 1] < s(|x|)$$

In the execution of interactive proofs, P has the ability to select a random coins r_i at the beginning of round i and uses it in round i. In the interaction P synchronously update his current secret state. That is, let *state* denote P's secret state (initialized to the private auxiliary input z), P updates *state* by setting $state = state||r_i$ in round i. The cheating verifier V^* launches a leakage attack by means of any number of arbitrary leakage queries throughout the interaction. Without loss of generality, we assume that the cheating verifier launches one and only one leakage query after receiving a message from the prover. The leakage query in round i is denoted as a

leakage function f_i, to which the prover responds with $f_i(state)$. We denote by ℓ the total number of leaking bits, i.e. $\ell = \sum_i |f_i(state)|$.

General zero knowledge is formalized by requiring that for any polynomial-time verifier V^* there exists a polynomial-time algorithm $\mathcal{S}$ (a.k.a. the simulator) such that the view of V^*, which consists of the random tosses of V and the messages received by V, can be simulated by $\mathcal{S}$. Under the leakage attack setting, leakage information $f_i(state)$ obtained by V^* cannot be simulated by any polynomial-time algorithm $\mathcal{S}$ only upon common input x. To formulate leakage-resilient zero-knowledge, $\mathcal{S}$ is given black-box access to an leakage oracle to respond the verifier's leakage queries. The leakage oracle, denoted by $L_z^n(\cdot)$, is determined by the security parameter n and the auxiliary input z (the witness). A query to the oracle is an efficiently computable function $f'_j(\cdot)$, to which $L_z^n(\cdot)$ responds with $f'_j(z)$. We denote by ℓ' the number of bits that $\mathcal{S}$ receives from the leakage oracle, i.e. $\ell' = \sum_j |f'_j(state_j)|$. The leakage resilient zero-knowledge property essentially requires the condition $\ell' \leq \ell$ holds. [13] defined $(1+\varepsilon)$-LR-ZKP that require a relaxed condition, $\ell' \leq (1+\varepsilon)\ell$, holds for a-priori fixed $\varepsilon > 0$. This is reasonable and necessary for black-box zero knowledge. Recently, [20] presented a non-black-box zero knowledge protocol that satisfies the condition $\ell' \leq \ell$.

The leakage-resilient zero knowledge proofs require that no malicious verifier, launching leakage attack, can learn anything beyond the validity of the assertion and the leakage. This is formulated by requiring that for any polynomial-time verifier V^* with any auxiliary input aux there exists a polynomial-time algorithm $\mathcal{S}$ such that the output of $S^{L_z^n(\cdot)}(x, aux)$ is indistinguishable from the view of V^*. Since this paper focuses on non-black-box zero knowledge, we refer to the definition of LR-ZKP in [20].

Definition 2 (Leakage-Resilient Zero Knowledge). *An interactive proof system (P, V) for a language L with a witness relation $\mathcal{R}$ is said to be* ***leakage-resilient zero knowledge*** *if for every* PPT *machine V^* that makes any arbitrary polynomial number of leakage queries on P's state (in the manner as described above), there exists a* PPT *algorithm $\mathcal{S}$ such that the following two conditions hold:*

- *For every $(x, z) \in \mathcal{R}$, and every $aux \in \{0,1\}^*$, $\{View_{V^*}(x, aux)\}_{x \in L, aux \in \{0,1\}^*}$ and $\left\{\mathcal{S}^{L_z^n(\cdot)}(x, aux)\right\}_{x \in L, aux \in \{0,1\}^*}$ are computationally indistinguishable.*
- *For every $(x, z) \in \mathcal{R}$, every $aux \in \{0,1\}^*$, and sufficiently long $r \in \{0,1\}^*$, it holds that $\ell_{\mathcal{S},r}(v) \leq \ell_{V^*}(v)$, where $\ell_{\mathcal{S},r}(v)$ is the number of bits that $\mathcal{S}$ receives from L_z^n when generating the view v, $\ell_{V^*}(v)$ is total length of leakage answers that V^* receives in the view v.*

2.2 Proof of knowledge

In a proof of knowledge for a relationship R, the prover, holding a secret input z such that $(x, z) \in R$, and the verifier interact on a common input x. The goal of the protocol is to convince the verifier that the prover indeed knows such z. This is in contrast to a regular interactive proof, where the verifier is just convinced of the validity of the statement.

The concept of "knowledge" for machines is formalized by saying that if the prover can convince the verifier, then there exists an efficient procedure that can "extract" a witness from this prover (thus the prover knows the witness because it could run the extraction procedure on itself).

Definition 3. *An interactive protocol $\langle P, V\rangle$ is a system of proofs of knowledge for a (poly-balanced) relation R with knowledge error κ if the following conditions hold:*

- *(efficiency): $\langle P, V\rangle$ is polynomially bounded, and V is computable in probabilistic polynomial time.*
- *(non-triviality): There exists an interactive machine P such that for every $(x, z) \in R$ all possible interactions of V with P on common input x and auxiliary y are accepting.*
- *(validity with knowledge error κ): Denote by $p(x, y, r)$ the probability that the interactive machine V accepts, on input x, when interacting with the prover specified by $P^*_{x,y,r}$ (the prover's strategy when fixing common x, auxiliary input y and random tape r). There exists an expected polynomial-time oracle machine $\mathcal{K}$ and a polynomial q such that if $p(x, y, r) > \kappa$, on input x and access to oracle $P_{x,y,r}$, $\mathcal{K}^{P_{x,y,r}}(x)$ outputs z, satisfying $(x, z) \in R$, with probability of at least $(p(x, y, r) - \kappa(|x|))/q(|x|)$*

2.3 Universal arguments

The notion of universal arguments, introduced by Barak and Goldreich in [3], is used to present interactive proofs for language

$$L_{\mathcal{U}} = \{(M, x, t) : \text{non-deterministic machine } M \text{ accepts } x \text{ within } t \text{ steps}\}$$

The corresponding relation is denoted by $\mathcal{R}_{\mathcal{U}}$, that is, $\mathcal{R}_{\mathcal{U}}((M, x, t), w) = 1$ if and only if M (viewed here as a two-input deterministic machine) accepts (x, w) within t steps. We can handle all NP language by a universal argument for $L_{\mathcal{U}}$, because every language in NP is linear time reducible to $L_{\mathcal{U}}$.

For a super-polynomial function $T : N \to N$, define the relation $\mathcal{R}^T_{\mathcal{U}}$ as follows: $\mathcal{R}^T_{\mathcal{U}}(y, w) = 1$ if $\mathcal{R}_{\mathcal{U}}(y, w) = 1$ and $t \le T(|y|)$, where $y = (M, x, t)$. The corresponding language is denoted by $L^T_{\mathcal{U}}$.

Definition 4. *(Universal arguments)[3]. A universal-argument system for $Ntime(T(n))$ is a pair of strategies, denoted $\langle P, V\rangle$, that satisfies the following properties:*

- ***Efficient verification**: There exists a polynomial p such that for any $y = (M, x, t)$, the total time spent by the (probabilistic) verifier strategy V, on common input y, is at most $p(|y|)$. In particular, all messages exchanged in the protocol have length smaller than $p(|y|)$. (Here, $|y|$ is assumed polynomial bounded).*
- ***Completeness by a relatively-efficient prover**: For every $((M, x, t), w)$ in $\mathcal{R}^T_{\mathcal{U}}$,*

  $$Pr[\langle P(w), V\rangle(M, x, t) = 1] = 1$$

  Furthermore, there exists a polynomial p such that the total time spent by $P(w)$, on common input (M, x, t), is at most $p(T_M(x, w)) \le p(t)$, where $T_M(x, w)$ denote the number of steps made by M on input (x, w).

– ***Computational Soundness****: For every polynomial-size circuit family* $\{\widetilde{P}_n\}_{n\in N}$*, and every* $(M,x,t)\in\{0,1\}^n - L_{\mathcal{U}}$*,*

$$Pr[\langle \widetilde{P}_n, V\rangle(M,x,t)=1] < neg(n)$$

– ***Weak Proof of Knowledge Property****: There exists a probabilistic* $T(n)^{O(1)}$*-time oracle machine* E *such that the following holds: For every polynomial-size circuit family* $\{\widetilde{P}_n\}_{n\in N}$*, and every sufficiently long* $y=(M,x,t)\in\{0,1\}^n$

$$Pr[E^{\widetilde{P}_n}(y)\in L_{\mathcal{U}}^T] \geq \Pr\left[\langle \widetilde{P}_n(y), V(y)\rangle = 1\right] - neg(n)$$

The oracle machine E *is called a (knowledge) extractor.*

2.4 Commitment Schemes

Commitment schemes are used to enable a party, known as the sender, to commit itself to a value while keeping it secret from another party, called the receiver. This property is called hiding. Furthermore, the commitment is binding, and thus at a later stage when the commitment is opened, it is guaranteed that the "opening" can yield only a single value determined in the committing phase. We sketch the two properties. General definitions can be found in [5].

– **Statistically binding commitments:** the binding property holds against unbounded senders, while the hiding property only holds against computationally bounded receivers.
– **Statistically hiding commitments:** the hiding property holds against unbounded receivers, while the binding property only holds against computationally bounded senders.

In this paper, we use two types of commitment schemes. One is Naor's statistically binding commitment scheme[18], denoted by $Comm_{sb}$. This scheme is constructed from pseudorandom generator $g:\{0,1\}^n\to\{0,1\}^{3n}$. Recall that in the commitment phase of this scheme, the receiver first selects $\tau\in_R\{0,1\}^{3n}$ and sends it to the sender. To commit to bit b, the sender selects $s\in_R\{0,1\}^n$, and then sends to the receiver $c=g(s)$ if $b=0$, or $c=\tau\oplus g(s)$ otherwise. In the decommitment phase, the sender reveals b and s.

Obviously, when $\tau\in\{g(s_0)\oplus g(s_1): s_0,s_1\in\{0,1\}^n\}$, the commitment to b by sending $g(s_0)$ to the receiver can be decommited to two different values. Concretely, the commitment c can decommit to 0 by revealing $(0,s_0)$, or to 1 by revealing $(1,s_1)$. For convenience, we denote by $Comm_{sb}^{\tau}(v;r)$ the commitment to v with the random coin r when the receiver's message is τ. To commit to a string $r=r_1\cdots r_k\in\{0,1\}^k$, the sender and the receiver run k instances of this scheme in parallel. Concretely speaking, the receiver sends $\tau=\tau_1\cdots\tau_k\in\{0,1\}^{k\cdot 3n}$, and the sender computes and sends $c_i=Comm_{sb}^{\tau_i}(r_i;s_i)$ to the receiver, where $s_i\in\{0,1\}^n$, $i=1,\cdots,k$. Similarly, we denote by $Comm_{sb}^{\tau}(r;s=s_1\cdots s_k)$ the commitment to $r=r_1\cdots r_k\in\{0,1\}^k$.

The other is a two-round statistically hiding commitment scheme $Comm_{sh}(\cdot;\cdot)$. In such a scheme, the first message denoted by m is from the receiver, and a corresponding commitment algorithm is denoted by $Comm_{sh}^{m}(\cdot;\cdot)$. In particular, thisscheme can be constructed based on claw-free collections[14].

3 Tools

3.1 Barak's non-black-box preamble

Barak's non-black-box simulation method enable the simulator to have access to "trapdoor" without using rewinding technique. By this "trapdoor", the simulator can simulate the real interaction. Concretely, the simulator take the advantage of holding the description of the verifier's strategy $desc(V)$ to generate such a "trapdoor" by means of "generation protocol" and then it can work efficiently. The actual prover without $desc(V)$, however, cannot generate any "trapdoor", and then the soundness of the protocol holds.

In the leaking setting, except fore interacting with P according to the protocol, V^* has the ability to obtain additional information $f(state)$ (where $state$ consists of the witness and all used random value so far) by launching a leakage attack. The prover, however, is not aware of the leakage attack. The leakage query $f(\cdot)$ is a part of the strategy of V^*, and so the messages sent by the verifier may depend on the leaking information $f(state)$. Therefore, to show leakage-resilient zero knowledge property, the simulator should "correctly" respond V^*'s leakage queries. However, owing to without any witness, the simulator does not have the ability to reply directly to any leakage queries by itself. In order to solve this problem, [13] introduced the leakage-oblivious simulation, in which the simulator directly receives V^*'s leakage queries and responds each leakage query $f(\cdot)$ under the help of the leakage oracle holding the auxiliary input of P (a witness for $x \in L$). More precisely, after receiving a leakage query $f(\cdot)$, the simulator constructs a new function $f'(\cdot)$ only on the witness, and query the leakage oracle with $f'(\cdot)$. Finally, V^* receives $f'(z)$ directly from the leakage oracle. Notice that, the simulator in leakage-oblivious simulation is authorized to receive the leakage query, and so it obtains an advantage over the prover who is not aware of the existence of the leakage attack. To use such the advantage, we consider a new variant of Barak's relation.

For any $x \in L$, let Ψ be the verifier's interactive strategy. Let $\{\mathcal{H}_n\}$ be a family of hash functions, where $h \in \mathcal{H}_n : \{0,1\}^* \to \{0,1\}^n$. Let $\mathcal{T}_n$ be a family of pseudorandom permutations on $\{0,1\}^n$. Define relation $\mathcal{R}$ as follows (depicted in Figure 1).

Instance: $(h, t, \tau, c, d) \in \mathcal{H}_n \times \mathcal{T}_n \times \{0,1\}^{2n \cdot 3n} \times \{0,1\}^{2n \cdot 3n} \times \{0,1\}^{6n^2}$.
Witness: (Ψ, u, ζ, s), where Ψ is an interactive program , $u \in \{0,1\}^n$, $\zeta \in \{0,1\}^*$, $s \in \{0,1\}^{2n \cdot n}$.
Relation: $\mathcal{R}((h, t, \tau, c, d), (\Psi, u, \zeta, s)) = 1$ if and only if

1. $c = Comm^{\tau}_{sb}(\eta; s)$, where $\eta = (t(h(\Psi) \oplus u), u)$.
2. $\Psi(c)$ outputs a computable leakage function, denoted by f_Ψ, such that $\zeta = f_\Psi(z, \eta, s)$, where $z \in \{0,1\}^*$ is a witness for $x \in L$.
3. $\Psi(c, \zeta) = d$ within $T(n) = n^{\log \log n}$ steps.

Figure 1: Relation $\mathcal{R}$

Denote by $L_{\mathcal{R}}$ the language corresponding to the variant of Barak's relation $\mathcal{R}$. Notice that $\mathcal{R}$ is dependent on a given instance $x \in L$. Next, we present an instance

generation protocol, denoted by Φ, for $L_{\mathcal{R}}$ (depicted in Figure 2). It is similar to Barak's generation protocol.

Instance generation protocol Φ:
- V selects $h \in_R \mathcal{H}_n$, $t \in_R \mathcal{T}_n$, $\tau \in_R \{0,1\}^{2n \cdot 3n}$, and then sends (h, t, τ) to P.
- P selects $\eta \in_R \{0,1\}^{2n}; s \in_R \{0,1\}^{2n^2}$ and sends $c = Comm^{\tau}_{sb}(\eta; s)$ to V.
- V selects $d \in_R \{0,1\}^{6n^2}$ and sends it to P.

Figure 2: Generation protocol Φ

The generation protocol Φ is sound if V honestly executes the protocol.

Lemma 1. *For any $T(n)^{O(1)}$-size P, let $\sigma = (h, t, \tau, c, d)$ be the transcript of execution of Φ. If V is honest, then the probability that P outputs w such that $\mathcal{R}(\sigma, w) = 1$ is negligible.*

Proof. Define the event E to be "*P outputs a witness for $\sigma \in L_{\mathcal{R}}$*", and let $\varepsilon = Pr[E]$. Suppose $H_{set} = \{(h, t, \tau) : Pr[E|(h, t, \tau)] \geq \varepsilon/2\}$. That is, for any $(h, t, \tau) \in H_{set}$, it holds that $Pr[E|(h, t, \tau)] \geq \varepsilon/2$. Then, the probability that V randomly selects (h, t, τ) such that $(h, t, \tau) \in H_{set}$ is at least $\varepsilon/2$. Without loss of generality, we can assume that P is deterministic. Thus, for any fixed (h, t, τ), c and z (a witness for x) are also fixed. It follows that V's leakage query f is determined if it exists. By the assumption that P obtains a witness for σ with probability ε, we have that P^*, with probability $\varepsilon^2/4$, obtains two witness respectively for $\sigma = (h, t, \tau, c, d)$ and $\sigma' = (h, t, \tau, c, d')$, where $d \neq d'$. This means that P^* can find two different programs Ψ, Ψ' and random values $\eta, \eta' \in \{0,1\}^n$, such that

$$\eta = (t(h(\Psi) \oplus u), u),\ \eta' = (t(h(\Psi') \oplus u'), u')$$
$$Comm^{\tau}_{sb}(\eta; s) = Comm^{\tau}_{sb}(\eta'; s')$$

By the statistically binding property of $Comm_{sb}(\cdot)$, we have that $Pr[\eta \neq \eta']$ is negligible. This means that $Pr[t(h(\Psi) \oplus u) \neq t(h(\Psi') \oplus u')]$ and $\Pr[u \neq u']$ all are negligible. It follows that $Pr[h(\Psi) \neq h(\Psi')]$ is negligible since t is a permutation. Therefore, P^* can find a collision for h with a non-negligible probability if ε is non-negligible. This contradicts with the assumption for $\mathcal{H}_n$. ■

In a leaking setting where V carries out leakage attacks, P is not ware of any leakage attacks. If the simulator is allowed to see V's leakage queries, it will get an extra advantage than P. In other words,

Lemma 2. *For any V^* that possibly launches leakage attack, there exists an oracle PPT algorithm $\mathcal{S}_{V^*}$, on inputting the description of V^* and being allowed to obtain V^*'s leakage queries, outputs (v, w) such that the following conditions hold:*

- *v is indistinguishable from the view of V^* interacting with an honest P.*
- *Let σ be the transcript that is contained in v. w is a witness for $\sigma \in L_{\mathcal{R}}$.*

Proof. For any V^*, let Ψ denote the description of V^*. Define $\mathcal{S}_{V^*}$, which has black-box access to the leakage oracle, as follows:

- Uniformly select random coins R_{V^*} for V^*.

- Obtain (h, t, τ) by invoking V^*.
- Randomly select $u \in \{0,1\}^n$ and $s \in \{0,1\}^{2n \cdot n}$, compute $c = Comm^{\tau}_{sb}(\eta; s)$, where $\eta = (t(h(\Psi) \oplus u), u)$.
- Execute $\Psi(c)$ and then obtain leakage query $f_\Psi = \Psi(c)$. (If V^* does not launch leakage query, f_Ψ is the empty.)
- Query the leakage oracle with $f_\Psi(\cdot, \eta, s)$ (if f_Ψ is not empty) and obtain $\zeta = f_\Psi(z, \eta, s)$.
- Invoke V^* with (c, ζ) and it returns d.
- Output $v = ((h, \tau, c, d, R_{V^*}), w = (\Psi, \zeta, s)$.

It follows from the hiding property of $Comm_{sb}$ that v is indistinguishable from the view of V^*. Obviously, $(h, t, \tau, c, d) \in L_{\mathcal{R}}$ and the corresponding witness is $w = (\Psi, u, \zeta, s)$. ■

Remark: Since the auxiliary input z (witness for $x \in L$) is not given to $\mathcal{S}_{v^*}$, $\mathcal{S}_{V^*}$ cannot compute $f_\Psi(z, \eta, s)$ by itself even if $\mathcal{S}_{V^*}$ has obtained $f_\Psi(\cdot)$. Fortunately, $\mathcal{S}_{V^*}$ here is given access to the leakage oracle. After obtaining $f_\Psi(\cdot)$, $\mathcal{S}_{V^*}$ queries the leakage oracle with $f_\Psi(\cdot, \eta, s)$ and then receives $f_\Psi(z, \eta, s)$ from the leakage oracle.

The hardness condition of $\mathcal{R}$ holds against $T(n)^{O(1)}$-size adversaries. Barak in [3] used an error correcting code $(ECC)$to relax the assumption such that the hardness condition holds under the existence of the standard collusion-resisted hash functions. Let ECC be a polynomial-time computable function with polynomial-time encoding and with constant distance. If one replaces the condition $\eta = (t(h(\Psi) \oplus u), u)$ with $\eta = (t(h(ECC(\Psi)) \oplus u), u)$, the conclusion of lemma 1 holds for polynomial sized P^*.

3.2 A new preamble

The fact that $\mathcal{R}$ can be verified in $T(n) = n^{\log \log n}$ under the help of the leakage oracle implies that $L_{\mathcal{R}}$ does not lie in NP. Therefore, all the traditional proof systems for NP cannot deal with $L_{\mathcal{R}}$. Barak and Goldreich in [3] introduced universal arguments for the languages $Ntime(T(n))$ and presented a 4-round public-coin protocol, denoted by $\langle P_{ua}, V_{ua} \rangle$, for such languages. Their construction is as follows:

Common input: (h, t, τ, c, d). Security parameter 1^n.

- V_{ua} sends its first random massage α to P_{ua}.
- P_{ua} computes the response, denoted by β, and sends it to V_{au}. Let $l_1 = |\beta|$.
- V_{ua} sends its second random massage γ to V_{ua}.
- P_{ua} computes the response, denoted by δ, and sends it to V_{au}. Let $l_2 = |\delta|$.

The above system $\langle P_{ua}, V_{ua} \rangle$ can be used to deal with $L_{\mathcal{R}}$ although the verification of the relation $\mathcal{R}$ needs access to the leakage oracle. Suppose that $Comm(\cdot;\cdot)$ is a non-interactive statistically binding commitment scheme. By combining $Comm(\cdot;\cdot)$ with universal argument $\langle P_{ua}, V_{ua} \rangle$, one can construct an "encrypted" universal argument, denoted by $\langle \widetilde{P}_{ua}, \widetilde{V}_{ua} \rangle$ for $L_{\mathcal{R}}$ (Figure 3).

‘Encrypted” universal argument $\langle \widetilde{P}_{ua}, \widetilde{V}_{ua} \rangle$ for the statement that $(h, \tau, c, d) \in L_{\mathcal{R}}$.

- $\widetilde{V}_{ua}$ executes V_{ua} and sends α to $\widetilde{P}_{ua}$.
- $\widetilde{P}_{ua}$ executes P_{ua} to computes β, and then sends $\widetilde{\beta} = (t_\beta, Comm(t_\beta(\beta); r_\beta))$ to $\widetilde{V}_{ua}$, where $t_\beta \in_R \mathcal{T}_{l_1}, r_\beta \in_R \{0,1\}^*$.
- $\widetilde{V}_{ua}$ executes V_{ua} and then sends γ to $\widetilde{P}_{ua}$.
- $\widetilde{P}_{ua}$ executes P_{ua} to computes δ and then sends $\widetilde{\delta} = (t_\delta, Comm(t_\delta(\delta); r_\delta))$ to $\widetilde{V}_{ua}$, where $t_\delta \in_R \mathcal{T}_{l_2}, r_\delta \in_R \{0,1\}^*$.

Figure 3: “Encrypted” universal argument $\langle \widetilde{P}_{ua}, \widetilde{V}_{ua} \rangle$

Next, we define the following relation $\mathcal{R}_{sim}$ corresponding to $\langle \widetilde{P}_{ua}, \widetilde{V}_{ua} \rangle$ (depicted in Figure 4).

Instance: $\sigma = ((h, t, \tau, c, d), (\alpha, \widetilde{\beta} = (t_\beta, \widetilde{\beta}_1), \gamma, \widetilde{\delta} = (t_\delta, \widetilde{\delta}_1)))$.
Witness: $(\beta, r_\beta, \delta, r_\delta)$.
Relation: $\mathcal{R}_{sim}(\sigma, (\beta, r_\beta, \delta, r_\delta)) = 1$ if and only if the following conditions hold.

1. $\widetilde{\beta}_1 = Comm(t_\beta(\beta); r_\beta)$.
2. $\widetilde{\delta}_1 = Comm(t_\delta(\delta); r_\delta)$.
3. $V_{ua}((h, \tau, c, d), (\alpha, \beta, \gamma, \delta)) = 1$.

Figure 4: Relation $\mathcal{R}_{sim}$

Note that V_{ua} is efficient. It follows that $\mathcal{R}_{sim}$ is an NP-relation. The corresponding language is denoted $L_{\mathcal{R}_{sim}} \in NP$. We construct the following preamble Σ for $L_{\mathcal{R}_{sim}}$ (depicted in Figure 5).

Preamble Σ:

- P and V run Φ. Let $\sigma' = (h, t, \tau, c, d)$ be the transcript.
- P and V run $\langle \widetilde{P}_{au}, \widetilde{V}_{au} \rangle$ for σ'.
  - V sends V_{ua}’s first massage α to P .
  - P sends $\widetilde{\beta} = (t_\beta, Comm(\beta; r_\beta))$ to V, where $\beta \in_R \{0,1\}^{l_1}, t_\beta \in_R \mathcal{T}_{l_1}$, $r_\beta \in_R \{0,1\}^*$.
  - V sends V_{ua}’s second massage γ to P.
  - P sends $\widetilde{\delta} = (t_\delta, Comm(\delta; r_\delta))$ to V, where $\delta \in_R \{0,1\}^{l_2}, t_\delta \in_R \mathcal{T}_{l_2}$, $r_\delta \in_R \{0,1\}^*$.

The transcript is denoted by $\sigma'' = (\alpha, \widetilde{\beta}, \gamma, \widetilde{\delta})$.

Figure 5: Preamble Σ

The preamble Σ is sound for $L_{\mathcal{R}_{sim}}$. For any P^* interacting with V, let σ be the transcript of the protocol, then the probability that P^* obtains a witness for $\sigma \in L_{\mathcal{R}_{sim}}$ is negligible.

Lemma 3. *No matter how P^* does, the probability that P^* obtains a witness w such that $\mathcal{R}_{sim}(\sigma, w) = 1$ is negligible when V executes the protocol honestly, where σ is the transcript of protocols.*

Proof. Let $\sigma = (\sigma', \sigma'')$, where σ' and σ'' are the transcript Φ and $\langle \widetilde{P}_{ua}, \widetilde{V}_{ua} \rangle$, respectively. Assume the probability that, at the end of Σ, P^* obtains w such that $\mathcal{R}_{sim}(\sigma, w) = 1$ is non-negligible. Notice that $w = (\beta, r_\beta, \delta, r_\delta)$, meeting with

– $\widetilde{\beta} = (t_\beta, Comm(\beta; r_\beta))$, $\widetilde{\delta} = (t_\delta, Comm(\delta; r_\delta))$, $\sigma'' = (\alpha, \widetilde{\beta}, \gamma, \widetilde{\delta})$.
– $V_{ua}(\sigma', (\alpha, \beta, \gamma, \delta)) = 1$

It is easy to construct a prover of $\langle P_{ua}, V_{ua}\rangle$ from P^*, denoted by P^*_{au}, such that $\Pr[\langle P^*_{ua}, V_{ua}\rangle(\sigma') = 1]$ is non-negligible. Therefore, from the proof of knowledge property of $\langle P_{ua}, V_{ua}\rangle$, there exist a probabilistic $T(n)$-time oracle machine E such that $Pr[E^{P^*_{au}}(\sigma') \in L^T_{\mathcal{U}}]$ is non-negligible. It follows that P^* can obtain w' at the end of Φ such that $\mathcal{R}(\sigma', w') = 1$ with a non-negligible probability. By lemma 1, this is impossible. ■

3.3 Augmented garbled-circuit

Garbled circuit method allows two parties, P_0 and P_1, to evaluate an arbitrary function $f(x, y)$ on their respective inputs $x = x_1 \cdots x_n$ and $y = y_1 \cdots y_n$, without leaking any information about their inputs beyond what is implied by $f(x, y)$.

Let C be a acyclic boolean circuit such that $C(x, y) = f(x, y)$ for any x, y. The basic idea of garbled-circuits is that one party generates an "garbled-circuit" $\widehat{C}$ of C, the second party then obliviously computes the output without learning any intermediate values.

Suppose that $e = poly(n)$ is the number of wires (including input wires of C and output wires of every gate in the circuit C). All the wires are labeled by $W_1, \cdots, W_e$, where $(W_1, \cdots, W_n)$ and $(W_{n+1}, \cdots, W_{2n})$ correspond to x and y respectively. Let $OUT = \{W_{e-k+1}, \cdots, W_e\}$ be all the output wires and C_g be all gates in C. Let (G, E, D) be a symmetric encryption scheme that has indistinguishable encryptions under chosen-plaintext attacks. In addition, assume that (G, E, D) has elusive and efficiently verifiable range. Such a encryption scheme can be constructed from a family of pseudorandom functions. One party garbles the circuit C as follows:

1. For each wire W_i, choose two independent keys (k^0_i, k^1_i) by a algorithm $G(1^n)$.
2. Given these keys, the four garbled values of each gate g, with input wires W_i, W_j and output wire W_m, are computed as follows:
$$c_{a,b} = E_{k^a_i}(E_{k^b_j}(k^{g(a,b)}_m)),\ a = 0, 1; b = 0, 1$$
where the inputs of W_i and W_j are a and b respectively, the output of g is $g(a, b)$. Let $T_g = (c_{0,0}, c_{0,1}, c_{1,0}, c_{1,1})$.
3. For each T_g, randomly choose a permutation π_g and set $T^*_g = \pi_g(T_g) \stackrel{\triangle}{=} (c_0, c_1, c_2, c_3)$. Let $T^* = \{T^*_g\}_{g \in C_g}$.
4. Construct decryption tables $T_{dec} = \{(0, k^0_i; 1, k^1_i)\}^{i=e}_{i=e-k+1}$, and define garbled-circuit $\widehat{C} = (T^*, T_{dec})$.

Garbled circuit method is formally denoted by two PPT algorithms $(Garble, Eval)$. On inputting 1^n and C, algorithm $Garble$ first executes the above process and then outputs $(\widehat{C}, K_0, K_1)$, where $K_0 = \left\{k^0_{0,i}, k^1_{0,i}\right\}^n_{i=1}$ is the keys of the input wires of x, $K_1 = \left\{k^0_{1,i}, k^1_{1,i}\right\}^n_{i=1}$ is the keys of the input wires of y. Let $K_x = \left\{k^{x_i}_{0,i}\right\}^n_{i=1}$, $K_y = \left\{k^{y_i}_{1,i}\right\}^{2n}_{i=n+1}$. Algorithm $Eval$, on inputting K_x, K_y and $\widehat{C}$, outputs $C(x, y) = f(x, y)$. Roughly speaking, for every gate g in C with input wires W_i, W_j and output wire W_m, $Eval$ uses the keys K^a_i and K^b_j respectively corresponding W_i, W_j to decrypt

$T_g^* = \pi_g(T_g) = (c_0, c_1, c_2, c_3)$. If more than one decryption or no decryption succeed, then fails and stops. Otherwise, obtain a key $K_m^{g(a,b)}$ corresponding W_m. Using K_x, K_y and starting from the input gates, $Eval$ can obtain the keys of all the output wires and return $C(x, y)$ by requiring the decryption tables.

Notice that if π_g is omitted in garbling C, that is, $T_g^* = (c_{0,0}, c_{0,1}, c_{1,0}, c_{1,1})$, the keys K_i^a and K_j^b (corresponding the inputs wires of the gate g) in fact will reveal the real inputs a and b, since the subscript of the cipher that can be decrypted is (a, b).

Here, assumed that $|x| = |y|$ for simplicity. In fact, this condition is unnecessary.

Garbled circuit method presents an approach for secure two-party computation. Let OT be 2-round 1-out-of-2 oblivious transfer protocol. Yao's two-party protocol is described as follows:

Yao's protocol Π_{cd}

Common input: $C(x, y) = f(x, y)$, security parameter n and m of $Comm_{sh}^m()$.
Private inputs: P_0's input: x; P_1's input: y.

- P_0 prepares a garbled-circuit $(\widehat{C}, K_0, K_1)$ by $(\widehat{C}, K_0, K_1) \leftarrow Garble(C, 1^n)$ for the $C(x)$ as above. Then P_0 sends $\widehat{C}$ and K_x to P_1.
- P_0 and P_1 execute OT such that P_1 obtains K_y.
  - P_1 computes $v = (v_1, \cdots, v_n)$, where v_i is the first message of OT protocol using the input y_i and fresh randomness r_i for $i \in [n]$. It then sends v.
  - P_0 prepares $v' = (v'_1, \cdots, v'_n)$, where v'_i is the second message of OT when taking $(k_{1,i}^0, k_{1,i}^1)$ as sender's input and v_i as receiver's first message. Finally, P_0 sends v'.
  - P_1 computes K_y from v, v'.
- P_1 obtains $f(x, y)$ by computing $Eval(\widehat{C}, K_x, K_y)$.

The construction of LR-ZKA in [20] used Yao's protocol to compute a conditional disclosure function such that the verifier reveals a random string only if the prover holds some special "witness". In order to force the verifier to compute conditional disclosure function honestly, a public-coin statistical ZKAoK, in which the verifier plays the role of a prover, follows this conditional disclosure.

Let $\mathcal{R}_{sim}$ be as above and $\sigma \in \{0,1\}^*$ be the transcript of Σ. As in [20], we use Yao's protocol Π_{cd} to compute the following conditional disclosure function:

$$f_{\sigma,\mathcal{R}_{sim}}(r, w) = \begin{cases} r, & \mathcal{R}_{sim}(\sigma, w) = 1 \\ 0^{|r|}, & others \end{cases}$$

The common input to P_0 and P_1 is the description of the function $f_{\sigma,\mathcal{R}_{sim}}$ or a circuit C computing $f_{\sigma,\mathcal{R}_{sim}}$.

P_0 (known as a sender, denoted by S_{cd}) randomly selects input $r \in \{0,1\}^{poly(n)}$ and want to disclose r to P_1 (known as a receiver, denoted by R_{cd}). The receiver holding w obtains r^* after the computation has been performed. If $\mathcal{R}(\sigma, w) = 1$ then $r^* = r$, otherwise $r^* = 0^{|r|}$. That is, the sender discloses r to the receiver if and only if $\mathcal{R}_{sim}(\sigma, w) = 1$. Here, we will use Naor's 2-round OT protocol (presented in [19], denoted as OT_{NP}) in Π_{cd}. OT_{NP} is secure if the DDH assumption holds. In addition, OT_{NP} holds the following properties: 1) the sender's security is statistical; and 2)

the receiver's first message is indistinguishable from one selected uniformly from the specific message space. It follows from [17] that Π_{cd} securely computes the conditional disclosure function $f_{\sigma,\mathcal{R}_{sim}}(r,w)$. Specially, the receiver taking w as the input does not obtain any information about the sender's input r except for $f_{\sigma,\mathcal{R}_{sim}}(r,w)$. Therefore, the receiver can only obtain $0^{|r|}$ when w is not a witness for $\sigma \in L_{\mathcal{R}_{sim}}$.

The computation of the conditional disclosure function $f_{\sigma,\mathcal{R}_{sim}}(r,w)$ in our construction of LR-ZKAoK, in fact, is a commitment stage, where the verifier acts as a sender to commit to a random value r'. If the prover does not have any witness for $\sigma \in L_{\mathcal{R}_{sim}}$, it only obtains $0^{|r'|}$ at the end of this commitment stage. So, the hiding property holds when the prover does not have any witness. In later stage when receiving a random value r'' from the prover, the verifier will reveal r'. Finally, the prover and the verifier obtain a common random value $r = r' \oplus r''$. Obviously, After receiving the revealment to r' from the verifier the prover cannot verify whether it is correct. That is, the binding property is in doubt.

The binding property is unnecessary to real interaction, since the prover completely ignores the computation of $f_{\sigma,\mathcal{R}_{sim}}(r',w)$ and accepts any r' revealed by the verifier. In simulation, however, simulator needs to learn r' by computing $f_{\sigma,\mathcal{R}_{sim}}(r',w)$ such that $r' \oplus r''$ equals a specified value by selecting a special r''. Therefore, if the verifier does not correctly reveal r', the simulator will fail even though the prover does not abort in that case. In order to ensure that the simulator fails if and only if the prover aborts in real interaction, the prover needs to check weather the verifier correctly reveal r' and aborts when the verifier does not. Therefore, we need to modify the algorithm *Garble* by replacing the step 3 of garbling circuit with the following process.

3'. For each T_g, choose randomly permutation π_g, and computes $C_{T_g} = Comm^m_{sh}(\pi_g)$ and $T^*_g = \pi_g(T_g) \stackrel{\triangle}{=} (c_0, c_1, c_2, c_3)$. Let $T^* = \{T^*_g\}_{g \in C_g}$ and $C_T = \{C_{T_g}\}_{g \in C_g}$.

The augmented garbled-circuit is $\widehat{C} = (T^*, C_T, T_{dec})$. It is easy to see that the modified garbled-circuit can be evaluated by the algorithm $Eval$ when given K_x, K_y. Therefore, Yao's protocol Π_{cd} works well in the context of $\widehat{C} = (T^*, C_T, T_{dec})$ and the proved security has not been impacted.

On the other hand, C_T in $\widehat{C} = (T^*, C_T, T_{dec})$ contains the commitments of all π_g's and can be used to verify the computation of $C(x,y)$. Recall that the receiver computes the garbled-circuit $\widehat{C}$ of C by executing $Eval(\widehat{C}, K_x, K_y)$. Starting from K_x, K_y, every gate in C can be "decrypted" correctly. Concretely, for every gate g $Eval$ uses the keys corresponding the input wires of g to decrypt $T^*_g = \pi_g(T_g) = (c_0, c_1, c_2, c_3)$ and then obtain the key corresponding the output wires of g. Notice π_g is a permutation over $T_g = (c_{0,0}, c_{0,1}, c_{1,0}, c_{1,1})$, where $c_{a,b} = E_{k^a_i}(E_{k^b_j}(k_m^{g(a,b)})$. Therefore, if the receiver obtains π_g, it can recover $T_g = (c_{0,0}, c_{0,1}, c_{1,0}, c_{1,1})$ by computing $\pi_g^{-1}(T^*_g)$. By requiring which one of T_g can be decrypted correctly, the receiver can obtain the inputs of g. Therefore, if the sender opens correctly all π_g's, the receiver can verifies the sender's input x and $C(x,y)$.

In our construction, the prover and the verifier (playing the role of sender) jointly compute $f_{\sigma,\mathcal{R}_{sim}}(r',w)$ by the augmented garbling circuit. This sub-protocol, in substance, is a commitment scheme in which the verifier commits to a random value r'.

In subsequent opening phase, the verifier is required to reveal r' and all the commitments to π_g's. Since by π_g's the prover can verify weather the sender correctly reveal r', the binding property holds since $Comm^m_{sh}$ is computationally binding.

4 Constant-round LR-ZKAoK for HC

Let HC be all directed graphs containing Hamiltonian cycles. Our goal in this section is to construct LR-ZKAoK for NP-relation R_{HC}. Recall Blum's protocol for HC.

Common input: $G = (\overline{V}, E) \in HC$ (containing a Hamiltonian cycle H), $|\overline{V}| = n$.

- Prover's first step(P1): The prover selects a random permutation π over $\overline{V}$, and sends the commitmentsto the adjacency of $G_i = \pi(G)$ to the verifier.
- Verifier's first step(V1): The verifier uniformly selects a challenge $ch \in_R \{0, 1\}$ and sends it to the prover.
- Prover's second step(P2): If $ch = 1$, the prover reveals some commitments corresponding to the edges of the Hamiltonian cycle $\pi(H)$. If $ch = 0$ then the prover reveals all the commitments and π.
- Verifier's second step(V2): The verifier V checks P's revealment.

Garg et al. in [13] first presented a LR-ZKP system for NP by adding a preamble to the paralleled Blum's protocol. In the preamble, the verifier first commits to its random challenge ch and a random string r' using a extractable commitment scheme, and then the prover sends a random string r''. In subsequent Blum's protocol, the prover uses $r = r' \oplus r''$ as the first random message of Naor's commitment scheme to compute its commitments. From the extractability of the commitment scheme, the simulator can learn ch and r' in advance by rewinding the verifier's strategy. Therefore, the simulator can answer this challenge correctly and has the ability to respond leakage queries under the help of the leakage oracle. More precisely, their construction consist of three stages. In Stage 1, V commits to its challenge ch and a random string r' using a public-coin statistically hiding commitment scheme, followed by many challenge-response slots. In Stage 2, P selects a random string r'' and V reveals r'. And then, P and V compute $r = r' \oplus r''$. Finally, in Stage 3, P and V run n parallel repetitions of Blum's protocol. In the first round, P uses Naor's two-round commitment scheme, using r as the first message, to commit to the adjacency of the permuted graphs. In the second round, V reveals the commitment to its challenge ch. Finally, P responds to this challenge and V verifies the response. The construction of [13] is a black-box LR-ZKP, since the simulator needs to rewind the verifier's program V^* such that it can extract the challenge ch and force r to a special distribution.

In the model of [13], cheating verifiers are allowed to launch a leakage query every time the prover sends its message. To obtain ch and r', the (black-box) simulator needs to rewind the challenge-response many times. In each rewinding, the simulator must inquire the leakage oracle again. Therefore, rewinding will increase the number of bits that the simulator receives from the leakage oracle. In order to allow the simulator to obtain ch and r' with minimal increase of the number of bits from the leakage oracle, the number of challenge-response slots in Stage 1 is not a constant. This results in the presented construction is super-constant-round.

The LR-ZK property requires that for any V^* there exists a simulator (with a leakage oracle) that can simulate the view of V^*, while proofs of knowledge require that there exists a (black-box) knowledge extractor $\mathcal{K}$ to extract a witness from P^* if P^*'s proof is accepted. In the protocol of [13], V commit to its challenges ch before P sends its commitment. This implies that the permutation that P use to compute its commitments may depend on the challenge selected by V. This results in that the black-box knowledge extractors cannot work efficiently. Therefore, the protocol in [13] is no longer a proof of knowledge. The approach of requiring the verifier to commit to its challenges ch in advance can ensure LR-ZK property, but then seemingly destroy the proof of knowledge property.

Recently, [20] presented a constant round leakage-resilient zero-knowledge proof for HC by combining the idea of [13] with Barak's non-black-box simulation technique. Roughly speaking, the protocol of [20] replaced the stage 1 of the protocol of [13] with a constant round preamble consisting of four stage such that the simulator can extract the challenge ch and force r to a special distribution of its choice without rewinding the verifier's algorithm V^*. Unfortunately, the protocol of [20] is not a proof of knowledge because of the same reason as mentioned above.

The key problem in constructing LR-ZKP is how to respond leakage queries in simulation such that the responses are consistent to the simulated view. To obtain LR-ZKAoK, we modified the construction in [20] such that the following two conditions hold: (1) the challenge ch is generated after the prover sends its commitments, as in [15, 16]; (2) the simulator have chance to control over the generated challenge and the prover cannot. Thus, the simulator can simulate the prover's commitments according to a-priori randomly selected challenge. On the other hand, black-box knowledge extractors $\mathcal{K}$ can work efficiently. To this end, we use a jointly coin-flipping protocol to generate ch after the prover sends its commitments. In addition, we also need a preamble in which the simulator obtains "trapdoors". As in [20], we use Barak's non-black-box techniques to construct such a preamble. The difference between this preamble and that in [20] is that it use a new variant of Barak's relation to define a NP-relation $\mathcal{R}_{sim}$. In general Barak's non-black-box simulation, the simulator needs the description of the verifier's strategy to accomplish a task that the prover cannot. That is, the simulator's advantage over the prover is that it is given the verifier's program. Notice that apart from the description of the verifier strategy is allowed to see the verifier's leakage queries. We define a variant of Barak's relation by simultaneously using the description of the verifier's strategy and leakage queries. The simulator can take advantage of knowing the description of the verifier's strategy and leakage queries to obtain an instance $\sigma \in L_{\mathcal{R}_{sim}}$ and a corresponding witness w such that $\mathcal{R}_{sim}(\sigma, w) = 1$, and no prover can obtain $\sigma \in L_{\mathcal{R}_{sim}}$. Therefore, the protocol is sound and meanwhile the simulator have the ability to control over ch and r.

Our protocol, denoted by Π, consists of 5 stages (depicted in Figure 6). In Stage 0, P and V run preamble Σ. Stage 1 is a coin-flipping protocol to produce a random string $r = r' \oplus r''$, where r' and r'' are randomly picked by P and V respectively. In this stage, V first commits to a random string r' by means of computing the conditional disclosure function $f_{\sigma,\mathcal{R}_{sim}}(r', w)$. Here, the augmented garbled-circuit $\widehat{C} = (T^*, C_T, T_{dec})$ is used to achieve the conditional disclosure of a random secret r'.

Subsequently, P sends a random string r'', and then V sets $r = r' \oplus r''$ and reveals r' along with and the all permutations committed in C_T. If r' pass the verification, P compute $r = r' \oplus r''$. In Stage 2, the prover commits n random permuted graphes using Naor's commitment scheme with r as the first message. Stage 3 is another coin-flipping protocol to produce a random challenge ch. Similarly as in Stage 1, the augmented garbled-circuit method is used to achieve the conditional disclosure of a random secret ch_V selected by V. And then, P randomly selects ch_P and V reveals ch_V. If ch_V is passed the verification, the two parties obtain a common random string $ch = ch_P \oplus ch_V$. In Stage 4, P uses ch as challenge to execute prover's second step of Blum's protocol P2. Then, V executes verifier's second step of Blum's V2 with ch.

Π: LR-ZKAoK for HC

Common input: $G = (V, E) \in HC$, $|V| = n$. Let $t = 3n^4$.
Auxiliary input to the prover: A Hamilton cycle H in G.

– Stage 0: Preamble.
  - P and V run Σ. The transcript is denoted by $\sigma = (\sigma', \sigma'')$.
– Stage 1: Jointly coin-tossing.
  - P sends the first random message m to V. (The first message of the statistically hiding commitment scheme $Comm^m_{sh}(\cdot)$).
  - V (the sender) selects $r' \in \{0,1\}^t$, and run Π_{cd} with P to compute $f_{\sigma,\mathcal{R}_{sim}}(r', \rho)$, where ρ is selected by P. Finally, P obtains r^*
  - P selects and sends $r'' \in_R \{0,1\}^t$ to V.
  - V opens r' and the commitments in C_T computed in Π_{cd}. V sets $r = r' \oplus r''$.
  - P verifies r' (verifying the decommitment to C_T and that r' is V's input in Π_{cd}) and aborts if the verification fails. Otherwise, P sets $r = r' \oplus r''$.
– Stage 2: The prover's commitment.
  - Suppose $r = r_1 \cdots r_n$, where $r_i = r_{i,1} \cdots r_{i,n^2}$ satisfying $|r_{i,j}| = 3n$ for $i \in [n]$ and $j \in [n^2]$. For every $i \in [n]$, P picks a random permutation π_i and computes $C_i = Comm^{r_i}_{sb}(\pi_i(G); \theta_i)$ (Naor's commitment scheme with r_i as the first message), and sends $C = (C_1, \cdots, C_n)$ to V.
– Stage 3: Jointly coin-tossing:
  - V (the sender) selects $ch_V \in \{0,1\}^n$ and and run Π_{cd} with P (the receiver) to compute $f_{\sigma,\mathcal{R}_{sim}}(ch_V, \rho)$, where ρ is selected by P. Finally, P obtains ch'_V
  - P selects and sends $ch_P \in_R \{0,1\}^n$ to V.
  - V opens ch_V and the commitments in C_T. V sets $ch = ch_P \oplus ch_V$.
  - P verifies ch_V (as the verification of r' in Stage 1) and aborts if the verification fails. Otherwise, P sets $ch = ch_P \oplus ch_V \stackrel{\Delta}{=} ch_1 \cdots ch_n$.
– Stage 4: P responds to the challenge ch, and V verifies this response.
  - P reveals some commitments in C_i corresponding to the edges of the Hamiltonian cycle $\pi_i(H)$ if $ch_i = 1$, or reveals all the commitments in C_i and π_i if $ch_i = 0$.
  - V verifies the decommitments according to ch.

Figure 6: LR-ZKAoK

The presented protocol uses non-black-box techniques such that the total of the leakage information received by the simulator is as much as the verifier. Concretely, since the simulator holds the description of the verifier's strategy V^* and is allowed to receive leakage queries from V^*, it can obtain a witness for $\sigma \in L_{\mathcal{R}_{sim}}$, where σ is the transcript of running Σ. By this witness, the simulator has the ability to force r to a special distribution in Stage 1 and ensure ch equal to a-priori randomly selected value. This results in that the simulator can interact with V^* in later stages as an honest prover and correctly respond to V^*'s leakage queries.

Theorem 1. *Let $\mathcal{H}$ be a family of collision-resistant pseudorandom hash function and $\mathcal{T}$ be a family of pseudorandom permutation. Assume that DDH assumption holds. Protocol Π is a LR-ZKAoK for HC with knowledge error $\kappa = 2^{-n}$, if $Comm_{sb}(\cdot;\cdot)$ is a Naor's 2-round statistically binding commitment scheme and $Comm_{sh}(\cdot;\cdot)$ is 2-round statistically hiding commitment scheme.*

Proof. **Completeness:** Completeness is obvious.

Soundness: Suppose $G \notin HC$. For any P^*, let ε^* be the probability that P^* successfully cheats V. By Lemma 3, we have that for any P^* the probability that P^* holds a witness for $\sigma \in L_{\mathcal{R}_{sim}}$ at the end of Stage 0 is negligible. Next, we need to show that the jointly coin-tossing sub-protocol in Stage 1 is secure.

To show this, define

$$\mathcal{B} = \{r : \exists s_0, s_1 \in \{0,1\}^n, \text{ such that } r = g(s_0) \oplus g(s_1)\}$$

for $g : \{0,1\}^n \to \{0,1\}^{3n}$ (pseudorandom generator used in Naor's commitment). Let r'' be a string that P^* sends to V in Stage 1. Then, after V open r', P^* obtains $r = r' \oplus r'' = r_{1,1} \cdots r_{n,n^2}$ at the end of Stage 1. We show the following lemma holds.

Lemma 4. *No matter what P^* does, the probability that there exist $i \in [n]$ and $j \in [n^2]$ such that $r_{i,j} \in \mathcal{B}$ is negligible, if V honestly executes the protocol.*

Proof. Denote by E_1 the event that there exist $i \in [n]$ and $j \in [n^2]$ such that $r_{i,j} \in \mathcal{B}$. Denote by E_2 the event that P^* holds a witness for $\sigma \in L_{\mathcal{R}_{sim}}$ at the end of Stage 0.

$$Pr[E_1] = Pr[E_1 \wedge E_2] + Pr[E_1 \wedge \overline{E_2}] \leq Pr[E_2] + Pr[E_1|\overline{E_2}]$$

First, it follows from Lemma 3 that $Pr[E_2]$ is negligible. Next, under the condition that P^* does not obtain a witness for $\sigma \in L_{\mathcal{R}_{sim}}$ in Stage 0, P^* only obtains 0^t by means of executing Π_{cd} and cannot learn r' before sending r'' to V. Therefore, $r = r' \oplus r''$ is uniformly distributed on $\{0,1\}^t$ except for a negligible probability if V is honest. It follows that $Pr[E_1|\overline{E_2}]$ is negligible. Therefore, $Pr[E_1]$ is negligible. □

By the same reason as above, the following lemma holds.

Lemma 5. *For any priori fixed $ch' \in \{0,1\}^n$, the probability that P^* successfully selects ch_{P^*} such that $ch = ch'$ at the end of Stage 3 is negligible, if V honestly executes the protocol.*

By Lemma 4, r generated in Stage 1 is uniformly distributed except for a negligible probability. So, the commitment scheme Naor's $Comm^r_{sb}(\cdot;\cdot)$ is statistically binding. If $G \notin HC$, by the binding property of $Comm^r_{sb}(\cdot;\cdot)$, we have that, except for a

negligible probability, there is only one $ch' \in \{0,1\}^n$ such that $C = (C_1, \cdots, C_n)$ can be successfully opened according ch' no matter what P^* does. P^* computes the commitments $C = (C_1, \cdots, C_n)$ before the challenge ch is generated. Therefore, if V is convinced of $G \in HC$ with a noticeable probability, then P^* is able to select ch_{P^*} such that $ch_{P^*} \oplus ch_V = ch'$ in Stage 3. By Lemma 5, this is impossible. And then we have that ε^* is negligible.

Leakage-resilient zero knowledge: To prove leakage-resilient zero knowledge property, we need to construct a simulator which not only output a simulated transcript but also responds V^*'s leakage query. The key problem is that the leakage must conform to the simulated transcript.

We first show how the simulator answers the verifier's leakage queries. The simulator is permitted to directly receive leakage query $f(state)$ from V^*, where $state$ consists of the witness and all used random value so far. Without holding any witness, however, the simulator cannot respond to leakage query f directly. It is just well that the simulator has access to a leakage oracle L_z^n that holds the witness z. To respond to a leakage query f, the simulator construct a new function f' such that $f'(z)$ is indistinguishable from $f(state)$, and then sends f' to the leakage oracle L_z^n. The leakage oracle L_z^n leaks $f'(z)$ directly to verifier V^*, and does not leak any information to the simulator.

Notice that leakage query f takes the form of $f(z, R)$, where z is the prover's witness (a Hamiltonian cycle H in G), $R = R(z)$ is a function that, on inputting the prover's witness z, outputs the prover's random coins. To create a new query f' after receiving a query $f(z, R(z))$ from the verifier, the simulator first selects an appropriate random coins $\widehat{R}$, and then presents a strategy based on z to select R' from $\widehat{R}$ such that $f(z, R')$ is indistinguishable from $f(z, R)$. Finally, define $f'(z) = f(z, R')$.

Non-black-box simulator: given $desc(V^*)$ and permitted to receive V^*'s leakage query, the non-black-box simulation is very straightforward. Roughly speaking, the simulator $\mathcal{S}$ plays the role of prover to interact with V^* and then outputs a simulated transcript. Concretely, in Stage 0, $\mathcal{S}$ can obtain a instance $\sigma = (\sigma', \sigma'')$ and a corresponding witness $w = (\beta, r_\beta, \delta, r_\delta)$, where $\sigma' = (h, \tau, t, c, d)$, $\sigma'' = (\alpha, \widetilde{\beta}, \gamma, \widetilde{\delta})$, $\sigma = (\sigma', \sigma'') \in L_{\mathcal{R}_{sim}}$. In Stage 1, $\mathcal{S}$ and V^* jointly generate r, which will be used as the first message of Naor's commitment scheme used in Stage 2. Using the witness w, $\mathcal{S}$ is able to select a special r in Stage 1, such that all the commitments generated in Stage 2 can be opened in two different manner. In Stage 2 and Stage 4, $\mathcal{S}$ acts as an honest prover. $\mathcal{S}$ completes Stage 3 using the same strategy as that in Stage 1, such that the generated challenge ch is equal to a specified one which depend on $\mathcal{S}$'s response to the leakage query in Stage 2. The reason for this is that, in Stage 2, $\mathcal{S}$ must determine all random value of Naor's commitment scheme in order to response V^*'s leakage query. This results in that $\mathcal{S}$ must specify the challenge ch in advance.

Let $t = 3n^4$. We denote by $\mathcal{S} \leftarrow V^*$ that $\mathcal{S}$ receives a message from V^*, by $\mathcal{S} \rightarrow V^*$ that $\mathcal{S}$ sends a message to V^*. For simplicity, we use $\mathcal{S} \leftrightarrows V^*$ to denote a interactive process between $\mathcal{S}$ and V^*. The process that V^* carries out a leakage query and obtains the response to it is denoted as $S \leftrightarrow V^*$.

On inputting common n-vertex graph G and the description of the code of the strategy of V^* (denoted by $desc(V^*)$), the simulator $\mathcal{S}^{L_z^n}(desc(V^*), G, aux)$ (where aux is V^*'s axuiliary. We omit aux sometimes) first selects uniformly random string R_{V^*} for V^* and then proceeds as follows.

If V^* aborts at any point, $\mathcal{S}$ outputs V^*'s current View and stops.

Simulating Stage 0: $\mathcal{S}^{L_z^n}(desc(V^*), G)$ deals with V^*'s leakage queries as follows. Initially, let $R_0 = \emptyset$ be the empty bit string. At any point, $\mathcal{S}$ resets $R_0 = R_0||R'$, where R' is the random coins used in the current process. When receiving a leakage query $f(\cdot,\cdot)$ from V^*, $\mathcal{S}^{L_z^n}(desc(V^*), G)$ sets $f'(z) \stackrel{\triangle}{=} f(z, R_0)$, and then queries $L_z^n(\cdot)$ with $f'(\cdot)$. Finally, V^* obtains $f'(z)$. It is easy to see that the leakage information, $f'(z) = f(z; R_0)$, is indistinguishable from one obtained by V^* in the real execution.

1. $\mathcal{S} \leftrightarrows V^*$: $\mathcal{S}$ runs Φ with V^*: $\mathcal{S}$ first invokes V^* and receives (h, t, τ). Thus, $\mathcal{S}$ randomly selects $u \in \{0,1\}^n, s \in \{0,1\}^{2n^2}$ and computes
   $c = Comm_{sb}^{\tau}(\eta; s)$, where $\eta = (t(h(desc(V^*)) \oplus u), u)$
   Finally, $\mathcal{S}$ invoke V^* with c.
2. $\mathcal{S} \leftrightarrow V^*$: After receiving a new query f, $\mathcal{S}$ sets $R_0 = t(h(desc(V^*)) \oplus u)||u$. Thus, $\mathcal{S}$ queries $L_z^n(\cdot)$ with $f'(\cdot) \stackrel{\triangle}{=} f(\cdot, R)$. Finally, V^* obtains $\zeta = f'(z)$ returned from $L_z^n(\cdot)$.
   Remark 1: Notice that $R = t(h(desc(V^*)) \oplus u)||u$ is indistinguishable from $\theta||u$ (corresponding random value generated in real interaction), where $\theta \in_R \{0,1\}^n$. It follows that $\zeta = f'(z)$ is indistinguishable from the leakage query in real interaction.
3. $\mathcal{S} \leftarrow V^*$: $\mathcal{S}$ receives d from V^*.
4. $\mathcal{S} \leftrightarrows V^*$: $\mathcal{S}$ executes $\widetilde{P}_{au}$ of $\langle \widetilde{P}_{au}, \widetilde{V}_{au} \rangle$ with the witness $(desc(V^*), u, \zeta, s)$. First, after receiving α from V^*, $\mathcal{S}$ computes β as an honest P_{ua}. Then, sends $\widetilde{\beta} = (t_\beta, \widetilde{\beta}_1)$ to V^*, where $\widetilde{\beta}_1 = Comm(t_\beta(\beta); r_\beta)$, $t_\beta \in_R \mathcal{T}_{l_1}, r_\beta \in_R \{0,1\}^*$.
5. $\mathcal{S} \leftrightarrow V^*$: After receiving a new query f from V^*, $\mathcal{S}$ prepares random string $R_0 = R_0||t_\beta||t_\beta(\beta)||r_\beta$. Then, $\mathcal{S}$ queries the leakage oracle $L_z^n(\cdot)$ with $f'(\cdot) \stackrel{\triangle}{=} f(\cdot, R)$, and $L_z^n(\cdot)$ returns $\zeta = f'(z)$ to V^*.
   Remark 2: In real interaction, the prover uses $t_\beta||\beta'||r_\beta$ as random value to response leakage query, where $\beta' \in_R \{0,1\}^{l1}$. Obviously, it is indistinguishable from $t_\beta||t_\beta(\beta)||r_\beta$ used by $\mathcal{S}$.
6. $\mathcal{S} \leftrightarrows V^*$: $\mathcal{S}$ receives γ from V^*, and then computes δ as an honest P_{au}. Finally, $\mathcal{S}$ sends $\widetilde{\delta} = (t_\delta, \widetilde{\delta}_1)$ to V^*, where $\widetilde{\delta}_1 = Comm(t_\delta(\delta); r_\delta)$, $t_\delta \in_R \mathcal{T}_{l_2}, r_\delta \in_R \{0,1\}^*$.
7. $\mathcal{S} \leftrightarrow V^*$: After receiving a new leakage query f from V^*, $\mathcal{S}$ sets $R_0 = R_0||t_\delta||t_\delta(\delta)||r_\delta$. Then, $\mathcal{S}$ queries $L_z^n(\cdot)$ with $f'(\cdot) \stackrel{\triangle}{=} f(\cdot, R)$. Finally, $L_z^n(\cdot)$ returns $\zeta = f'(z)$ to V^*.
   Remark 3: Similar to Remark 2.

Let σ be the simulated transcript of Stage 0, $fake = (\beta, r_\beta, \delta, r_\delta)$.

Simulating Stage 1: $\mathcal{S}^{L_z^n}(desc(V^*), G)$ deals with V^*'s leakage queries as in simulating Stage 0. Initially, let $R_1 = R_0$. At any point, $\mathcal{S}$ resets $R_1 = R_1||R'$, where

R' is the random coins used in the current process. To respond to V^*'s query f, $\mathcal{S}^{L_z^n}(desc(V^*), G)$ defines $f'(\cdot) \stackrel{\triangle}{=} f(\cdot, R_1)$ and then queries the leakage oracle $L_z^n(\cdot)$ with $f'(\cdot)$. Finally, V^* obtains $f'(z)$ from $L_z^n(\cdot)$. The details are as follows.

1. $\mathcal{S} \rightarrow V^*$: $\mathcal{S}$ sends a random value m to V^*.
2. $\mathcal{S} \leftrightarrow V^*$: After receiving a leakage query f from V^*, $\mathcal{S}$ sets $R_1 = R_1||m$. Thus, $\mathcal{S}$ queries $L_z^n(\cdot)$ with $f'(\cdot) \stackrel{\triangle}{=} f(\cdot, R)$. Thus, V^* obtains $f'(z)$ returned from $L_z^n(\cdot)$.
3. $\mathcal{S} \rightleftharpoons V^*$: $\mathcal{S}$ honestly executes Π_{cd} taking $fake = (\beta, r_\beta, \delta, r_\delta)$ as the input. $\mathcal{S}$ first receives $\widehat{C} = (T^*, C_T, T_{dec})$ and $K_{r'}$. Then, $\mathcal{S}$ prepares the first message $v_1 = (v_{1,1}, \cdots, v_{1,t})$ of OT_{NP} and sends it to V^*.
4. $\mathcal{S} \leftrightarrow V^*$: After receiving a leakage query f from V^*, $\mathcal{S}$ sets $R_1 = R_1||R'$, where R' is random coins used in preparing v. Thus, $\mathcal{S}$ queries $L_z^n(\cdot)$ with $f'(\cdot) \stackrel{\triangle}{=} f(\cdot, R_1)$. Thus, V^* obtains $f'(z)$ returned from $L_z^n(\cdot)$.
5. $\mathcal{S} \leftarrow V^*$: After receiving V^*'s response $v_1' = (v_{1,1}', \cdots, v_{1,t}')$, $\mathcal{S}$ first computes K_{fake} from v, v'. Finally, $\mathcal{S}$ computes $Eval(\mathcal{C}, K_{fake}, K_{r'})$ and then obtains r^*.
6. $\mathcal{S} \rightarrow V^*$: Assume $r^* = r_1' \cdots r_n'$, where $r_i' = r_{i,1}' \cdots r_{i,n^2}'$ and $|r_{i,j}'| = 3n$. For $i \in [n], j \in [n^2]$, $\mathcal{S}$ selects $z_{i,j}^0, z_{i,j}^1 \in \{0,1\}^n$ and sets
$$r_{i,j}'' = r_{i,j}' \oplus g(z_{i,j}^0) \oplus g(z_{i,j}^1),\ r_i'' = r_{i,1}'' \cdots r_{i,n^2}''$$
Then, $\mathcal{S}$ sends $r'' = r_1'' \cdots r_n''$ to V^* and receives r' along with the decommitments to all C_{T_g}'s committed in C_T from V^*.
7. Output: $\mathcal{S}$ verifies r' as an honest prover. If the verification fails, $\mathcal{S}$ outputs $(R_{V^*}; c, \widetilde{\beta}, \widetilde{\delta}, v_1, r'')$ and then aborts. Otherwise, $\mathcal{S}$ sets $r = r'' \oplus r^*$ and proceeds the next step.

Simulating Stage 2: Assume $r = r_1 \cdots r_n$, where $r_i = r_{i,1} \cdots r_{i,n^2}$ and $|r_{i,j}| = 3n$. Obviously, $r_{i,j} = g(z_{i,j}^0) \oplus g(z_{i,j}^1)$ for $(i,j) \in [n] \times [n^2]$ if $\mathcal{S}$ does not abort. In this stage, $\mathcal{S}^{L_z^n}(desc(V^*), G)$ completes the commitments as follows.

1. $\mathcal{S} \rightarrow V^*$: For every $i \in [n]$, $\mathcal{S}$ picks a random permutation π_i and sets $G_i = \pi_i(G)$. Then, $\mathcal{S}$ sends to V^* the commitments to the adjacency of G_i using Naor's commitment scheme with r_i as the first message, $i \in [n]$. Accurately, $\mathcal{S}$ sets $C_i = \{g(z_{i,j}^0)\}_{j=1}^{j=n^2}$ for $i \in [n]$, and sends $C = (C_1, \cdots, C_n)$ to V^*.
2. $\mathcal{S} \leftrightarrow V^*$: $\mathcal{S}$ uses $L_z^n(\cdot)$ to answer the leakage queries. The detail is as follows.

Response to query leakage in simulating Stage 2: It is not easy to deal with leakage queries in this stage although $\mathcal{S}$ acts seemingly as the same as an honest prover. The witness for $x \in L$ is a Hamiltonian cycle H in G. For simplicity, we replace z with H in what follows. After receiving V^*'s leakage query $f(\cdot)$, $\mathcal{S}$ needs to construct a new leakage query f' such that $f'(H)$ is indistinguishable from $f(state)$. To this end, $\mathcal{S}$ selects random permutation π_i' and $\widetilde{\theta}_i \in \{0,1\}^n$, $i = 1, \cdots, n$, such that the following two conditions hold.

1) $C_i = Comm_{sb}^{r_i}(\pi_i'(G); \widetilde{\theta}_i)$.

2) $\mathcal{S}$ is able to answer the challenge in Stage 4.

It is easy to see that if $\mathcal{S}$ has selected such $(\pi'_1, \widetilde{\theta}_1), \cdots, (\pi'_n, \widetilde{\theta}_n)$ and $\widetilde{\theta}_i \in \{0,1\}^n$, it can construct f' by letting $f'(\cdot) = f(\cdot, R_1||\pi'_1||\widetilde{\theta}_1|| \cdots ||\pi'_n||\widetilde{\theta}_n)$.

The condition 1) is easy to meet. But the condition 2), which is related to the witness H, is not easy to meet owing to lack of the witness. To solve this problem, $\mathcal{S}$ specifies in advance the challenge ch to respond to in Stage 4, and then define a random function, denoted as $Select(H, \{z^0_{i,j}, z^1_{i,j}\})$, to determine π'_i and $\widetilde{\theta}_i$ $(i = 1, \cdots, n)$ from ch and the witness H. $Select(H, \{z^0_{i,j}, z^1_{i,j}\})$ proceeds as follows:

- Randomly select a challenge $ch = ch_1 \cdots ch_n \in \{0,1\}^n$.
- For every $i \in [n]$
  - If $ch_i = 0$, define $\rho(0, H)$ and $\varphi(0, H, \{z^0_{i,j}, z^1_{i,j}\})$ as follows:
    (1) $\rho(0, H)$ returns the permutation $\pi'_i = \pi_i$ selected by $\mathcal{S}$ in Stage 2. Let $\{g_{i,j}\}_{j\in[n^2]}$ be the adjacency matrix of $G_i = \pi_i(G)$.
    (2) $\varphi(0, H, \{z^0_{i,j}, z^1_{i,j}\})$ returns $\widetilde{\theta}_i = y_{i,1}|| \cdots ||y_{i,n^2}$, where $y_{i,j} = z^b_{i,j}$ if $g_{i,j} = b$ for each $j \in [n^2]$.

    Remark: When $ch_i = 0$, $\mathcal{S}$ will ask to reveal π_i and decommit C_i to $\pi_i(G)$ in Stage 4. $\varphi(0, H, \{z^0_{i,j}, z^1_{i,j}\})$ returns $\theta_i = y_{i,1}|| \cdots ||y_{i,n^2}$ such that $C_i = Comm^{r_i}_{sb}(\pi_i(G); \theta_i)$.
  - If $ch_i = 1$, select a random cycle H_i, and define two functions $\rho(1, H, H_i)$ and $\varphi(1, H, H_i, \{z^0_{i,j}, z^1_{i,j}\})$ as follows:
    (1) $\rho(1, H, H_i)$ first selects a permutation $\overline{\pi}$ such that $H_i = \overline{\pi}_i(\pi_i(H))$. Thus, $\rho(1, H, H_i)$ returns $\pi'_i = \overline{\pi}_i \circ \pi_i$.
    (2) $\varphi(1, H, H_i, \{z^0_{i,j}, z^1_{i,j}\})$ computes π'_i such that $H_i = \pi'_i(H)$. Let $\{g'_{i,j}\}_{j\in[n^2]}$ be the adjacency matrix of $G_i = \pi'_i(G)$. $\varphi(1, H, H_i, \{z^0_{i,j}, z^1_{i,j}\})$ returns $\theta_i = y_{i,1}|| \cdots ||y_{i,n^2}$, where $y_{i,j} = z^b_{i,j}$ if $g'_{i,j} = b$ for each $j \in [n^2]$.

    Remark: When $ch_i = 1$, $\mathcal{S}$ will ask to reveal a random hamiltonian cycle committed in C_i in Stage 4. $\mathcal{S}$ first selects randomly H_i and then the leakage oracle can determine a permutation π'_i such that $H_i = \pi'_i(H)$ by executing $\rho(1, \cdot, H_i)$ with H, and compute $\widetilde{\theta}_i$ such that $C_i = Comm^{r_i}_{sb}(\pi'(G); \widetilde{\theta}_i)$ by executing $\varphi(1, \cdot, H_i, \{z^0_{i,j}, z^1_{i,j}\})$ with H.
- Output $R' = \pi'||\widetilde{\theta}_1|| \cdots ||\pi'_n||\widetilde{\theta}_n$.

After receiving V^*'s query $f(\cdot)$, $\mathcal{S}$ defines a new leakage query

$$f'(H) = f(H, R_1||Select(H, \{z^0_{i,j}, z^1_{i,j}\}))$$

Then, $\mathcal{S}$ queries $L^n_z(\cdot)$ with $f'(\cdot)$ and V^* obtains $f'(H)$ from $L^n_z(\cdot)$. It is easy to see that $f'(H)$ is the same as one leaked from P. Finally, let $R_2(H) = R_1||Select(H, \{z^0_{i,j}, z^1_{i,j}\})$.

Simulating Stage 3: $\mathcal{S}$ honestly executes Π_{cd} taking $fake$ as the inputs of the garbled-circuit. $\mathcal{S}$ handles leakage queries as in simulating Stage 1. Initially, let $R_3 = R_2(H)$.

1. $\mathcal{S} \rightleftharpoons V^*$: $\mathcal{S}$ honestly executes Π_{cd} taking $fake$ as the inputs of the garbled-circuit. $\mathcal{S}$ first receives $\widehat{C} = (T^*, C_T, T_{dec})$ and K_{ch_V} (keys corresponding ch_{V^*}). Then, $\mathcal{S}$ prepares the first message $v_2 = (v_{2,1}, \cdots, v_{2,t})$ of OT_{NP} and sends it to V^*.
2. $\mathcal{S} \leftrightarrow V^*$: After receiving a leakage query f from V^*, $\mathcal{S}$ sets $R_3 = R_3(H)||R'$, where R' is random coins used in preparing v. Thus, $\mathcal{S}$ queries $L_z^n(\cdot)$ with $f'(\cdot) \triangleq f(\cdot, R)$. Thus, V^* obtains $f'(z)$ returned from $L_z^n(\cdot)$.
3. $\mathcal{S} \leftarrow V^*$: After receiving V^*'s response $v_2' = (v_{2,1}', \cdots, v_{2,t}')$, $\mathcal{S}$ first computes K_{fake} from v, v'. Finally, $\mathcal{S}$ computes $Eval(\mathcal{C}, K_{fake}, K_{ch_V})$ and then obtains ch_{V^*}.
4. $\mathcal{S} \rightarrow V^*$: $\mathcal{S}$ sets $ch_P = ch \oplus ch_{V^*}$ and sendsch_P to V^*. V^* returns the decommitments to C_{T_g} to $\mathcal{S}$.
5. Output: $\mathcal{S}$ verifies ch_{V^*} as an honest prover. If the verification fails, $\mathcal{S}$ outputs the view $(R_{V^*}; c, \widetilde{\beta}, \widetilde{\delta}, v_1, r'', \overrightarrow{C}, v_2, ch_P)$ and then aborts. Otherwise, $\mathcal{S}$ sets $ch = ch_P \oplus ch_{V^*}$ and proceeds the next step.

Simulating Stage 4:

1. $\mathcal{S} \rightarrow V^*$: $\mathcal{S}$ reveals π_i and all the commitments of C_i when $ch_i = 0$, or reveals some commitments of C_i corresponding to the edges to H_i when $ch_i = 1$, for every $i \in [n]$. Formally, $\mathcal{S}$ sets
$$Ans_i = \begin{cases} \left(\pi_i, \{z_{i,j}^{g'_{i,j}}\}_{j=1}^{n^2}\right), & ch_i = 0 \\ \left\{z_{i,j}^1 : h'_{i,j} = 1, 1 \le j \le n^2\right\}, & ch_i = 1 \end{cases}$$
where $\{g'_{i,j}\}_{j=1}^{n^2}$ and $\{h'_{i,j}\}_{j=1}^{n^2}$ are the adjacencies of $G_i = \pi(G)$ and H_i, respectively. $\mathcal{S}$ sends $Ans = (Ans_1, \cdots, Ans_n)$ to V^*.

Outputting : Finally, $\mathcal{S}$ outputs $(R_{V^*}; c, \widetilde{\beta}, \widetilde{\delta}, v_1, r'', \overrightarrow{C}, v_2, ch_P, Ans)$ and stops.

Simulator's output distribution: Next, we use the hybrid argument method to prove that $\left\{\mathcal{S}^{L_z^n(\cdot)}(G, desc(V^*), aux)\right\}_{x \in L, aux \in \{0,1\}^*}$ is indistinguishable from the real view $\{View_{V^*}(G, aux)\}_{x \in L, aux \in \{0,1\}^*}$.

For simplicity, we will omit V^*'s auxiliary aux in what follows.

First, the prover may abort in real interaction when V^* fails in revealing r' correctly in Stage 1(or ch_{V^*} in Stage 3). Notice that $\mathcal{S}$ aborts in simulating Stage 1 (or Stage 3) under the same condition, and meanwhile outputs the current simulated view. For simplicity, we assume $\mathcal{S}$ does not abort in simulating Stage 1 or Stage 3 in what follows.

Suppose there is a hybrid simulator $\widehat{\mathcal{S}}$ which receives a Hamiltonian cycle $H \subseteq G$ as the input. We define hybrid experiments as follows:

Exp_0: $\widehat{\mathcal{S}}$ holding a hamiltonian cycle H, executes the protocol with V^* just like the honest prover.

Exp_1: This experiment is the same as Exp_0 except that in Stage 0 $\widehat{\mathcal{S}}$ interacts with V^* as $\mathcal{S}$. Specially, $\widehat{\mathcal{S}}$ computes $c = Comm^{\tau}_{sb}(\eta; s)$ after receiving (h, t, τ) from V^*, where $u \in_R \{0,1\}^n$, $\eta = t(h(desc(V^*)) \oplus u, u)$, and then sends c to V^*. After receiving a leakage query f from V^*, $\widehat{\mathcal{S}}$ computes the leakage information $\zeta = f(H, R)$. Finally, $\widehat{\mathcal{S}}$ can compute $\widetilde{\beta}, \widetilde{\delta}$ as an honest $\widetilde{P}_{ua}$.

Exp_2: This experiment is the same as Exp_1 except that in Stage 1 $\widehat{\mathcal{S}}$ interacts with V^* and responds to leakage queries as $\mathcal{S}$.

Exp_3: This experiment is the same as Exp_2 except in Stage 2 and Stage 4. In Stage 2, $\widehat{\mathcal{S}}$ interact with V^* as $\mathcal{S}$ to commit to $\pi_i(G)$ by $C_i = \{g(z^0_{i,j})\}_{j=1}^{j=n^2}$ for a randomly selected π_i. Let $ch = ch_1 \cdots ch_n$ be the challenge selected by $\widetilde{\mathcal{S}}$ in Stage 2, $ch' = ch'_1 \cdots ch'_n$ be the challenge generated in Stage 3. The probability that $ch = ch'$ is negligible since $\widehat{\mathcal{S}}$ proceeds as the honest prover in Stage 3. Therefore, $\widehat{\mathcal{S}}$ should use a special strategy to answer the challenge ch'. More precisely, $\widehat{\mathcal{S}}$ proceeds in Stage 4 as follows.

- When $ch_i = 0 \wedge ch'_i = 0$, $\widehat{\mathcal{S}}$ simply reveals π_i and $\{z^{g'_{i,j}}_{i,j}\}_{j=1}^{n^2}$ to show C_i is the commitments to $\pi_i(G) = (g'_{i,j})$.
- When $ch_i = 0 \wedge ch'_i = 1$, using H and π_i (selected in Stage 2), $\widehat{\mathcal{S}}$ can open some commitments of C_i according to the Hamiltonian cycle $\pi_i(H)$ in Stage 4. (Because $\widehat{\mathcal{S}}$ replies to the leakage query in Stage 2 as an honest prover ($ch_i = 0$), $\widehat{\mathcal{S}}$'s respond to ch'_i is coincident with its leakage answer in Stage 2.)
- When $ch_i = 1 \wedge ch'_i = 0$, $\widehat{\mathcal{S}}$ first invokes $\rho(1, H, H_i)$ and $\varphi(1, H, H_i, \{z^0_{i,j}, z^1_{i,j}\})$ with H and H_i (selected by $\mathcal{S}$ in Stage 2) to determine π'_i and the random coins $z^{g'_{i,j}}_{i,j}$ for each $j \in [n^2]$, where $\{g'_{i,j}\}_{j\in[n^2]}$ be the adjacency matrix of $G_i = \pi'_i(G)$, and then reveals π_i and $\{z^{g'_{i,j}}_{i,j}\}_{j=1}^{n^2}$ to show that C_i is the commitment to G_i. (Here, $\widehat{\mathcal{S}}$ uses the same strategy as in Stage 2 to determine π'_i and the random coins $z^{g'_{i,j}}_{i,j}$.)
- if $ch_i = 1, ch'_i = 1$, $\widehat{\mathcal{S}}$ opens some commitments of C_i according to the Hamiltonian cycle H_i (the Hamiltonian cycle picked in Stage 2).

Exp_4: This experiment is the same as Exp_3 except that in Stage 3 $\widehat{\mathcal{S}}$ interact with V^* and responds to leakage queries as $\mathcal{S}$.

Exp_5: This experiment is the same as Exp_4 except that $\widehat{\mathcal{S}}$ interact with V^* like $\mathcal{S}$ in Stage 4. That is, $\widehat{\mathcal{S}}$ in Exp_5 is the same as $\mathcal{S}$.

Let D_i be the output of $\widehat{\mathcal{S}}$ in Exp_i ($i = 0, 1, 2, 3, 4, 5$). In Exp_i, V^*'s leakage queries are dealt with as $\mathcal{S}$ from Stage 0 to Stage $i-1$. It is easy to see that $D_0 = View_{V^*}(G)$ and $D_5 = \mathcal{S}^{L^n_z(\cdot)}(G, desc(V^*))$.

Notice the differences between Exp_0 and Exp_1 are that the values committed by $c, \widehat{\beta}_1, \widehat{\delta}_1$ respectively are different. Thus, it follows from the hiding property of the commitment schemes that D_0 and D_1 are indistinguishable.

Exp_2 is different from Exp_1 in two aspects: (1) the input to Π_{cd} in Stage 1, and (2) the distribution of r'' (and then r). More precisely, $\widehat{\mathcal{S}}$ computes Π_{cd} with a random input in Exp_1, whereas with a witness of $\sigma \in L_{\mathcal{R}_{sim}}$ in Exp_2. Additionally,

instead of selecting uniformly r'' in Exp_1, $\widehat{\mathcal{S}}$ randomly selects $z^0_{i,j}, z^1_{i,j} \in_R \{0,1\}^n$ and then constructs r'' such that $r = r' \oplus r'' = r_1 \cdots r_n = r_{1,1} \cdots r_{n,n^2}$ meeting with $r_{i,j} = g(z^0_{i,j}) \oplus g(z^1_{i,j})$ for every $i \in [n], j \in [n^2]$, where r' be the output of Π_{cd} received by $\widehat{\mathcal{S}}$. Therefore, the fact that D_2 and D_1 are indistinguishable is implied by the receiver's security of OT_{NP} and the pseudo-randomness of g.

The differences between Exp_3 and Exp_2 are in two aspects. One is $\widehat{\mathcal{S}}$ computes $C_i = Comm^{r_i}_{sb}(\pi_i(G); \theta_i)$ (the commitment to $\{g'_{i,j}\}_{j=1}^{n^2}$, the adjacency of $G_i = \pi_i(G)$ for a randomly selected π_i) using different $\theta_i = \theta_{i,1} \cdots \theta_{i,n^2}$. In Exp_2, π_i is uniformly selected is Stage 2 and $\theta_{i,j} \in_R \{0,1\}^n$ is independent with $z^b_{i,j}$ $(b = 0, 1)$, whereas in Exp_3, π_i is determined from H and H_i (a random Hamiltonian cycle selected in Stage 2), $\theta_{i,j} = z^0_{i,j}$ if $g'_{i,j} = 0$ or $z^1_{i,j}$ if $g'_{i,j} = 1$. So, $C_i = \{g(z^0_{i,j})\}_{j=1}^{n^2}$ in Exp_3.

The other is the approach of replying to the challenge $ch' = ch'_1 \cdots ch'_n$ generated in Stage 3. In Exp_2, $\widehat{\mathcal{S}}$ honestly replies ch' in Stage 4. In Exp_3, $\widehat{\mathcal{S}}$ (in Stage 4) first determines π_i from H and H_i by requiring $H_i = \pi_i(H)$. Next, $\widehat{\mathcal{S}}$ sets $\theta_{i,j} = z^{g'_{i,j}}_{i,j}$, $j = 1, \cdots, n^2$. After then, $\widehat{\mathcal{S}}$ reply ch'_i as in Exp_2. $\widehat{\mathcal{S}}$ needs the witness H in Stage 4.

Notice π_i is also uniformly distributed in Exp_3 although it is determined from H and H_i. It is easy to see that the difference between D_3 and D_2 results only from C_i's. D_3 and D_2 are distinguishable means $g(z^1_{i,j})$ and $g(\theta_{i,j}) \oplus g(z^0_{i,j}) \oplus g(z^1_{i,j})$ (the commitments to 1 in Exp_3 and Exp_2, respectively) are distinguishable for $z^b_{i,j}, \theta_{i,j} \in_R \{0,1\}^n$. This is contradict with the pseudo-randomness of g. It follows that D_3 and D_2 are computationally indistinguishable.

The difference between Exp_4 and Exp_3 lies in Stage 3. Notice that Stage 3 is aimed at generating a uniform challenge ch'. In Exp_4, $\widehat{\mathcal{S}}$ first randomly selects ch in Stage 2, and then take $fake = (\beta, r_\beta, \delta, r_\delta)$ (the witness of $\sigma \in \mathcal{L}_{sim}$) as the input to run Π_{cd} in Stage 3, such that the generated challenge $ch' = ch_P \oplus ch_{V^*} = ch$. In Exp_3, however, $\widehat{\mathcal{S}}$ honestly executes Stage 3 and generates an uniformly independent $ch' = ch_P \oplus ch_{V^*}$. Obviously, the difference between Exp_4 and Exp_3 in essential is that $\widehat{\mathcal{S}}$ computes Π_{cd} with the different inputs. Therefore, the fact that D_4 and D_3 are indistinguishable is implied by the receiver's security of OT_{NP}.

Because of $ch = ch'$ in Exp_4. it is not hard to see that Epx_5 is the same as Exp_4. So, D_5 and D_4 are identical.

Overall, D_5 and D_0 are indistinguishable. That is, $\left\{\mathcal{S}^{L^n_z(\cdot)}(G, desc(V^*))\right\}_{x \in L}$ is indistinguishable from $\{View_{V^*}(G)\}_{x \in L}$.

Proof of Knowledge: By Definition 3, we need to construct a knowledge extractor $\mathcal{K}$ with knowledge error κ, such that $\mathcal{K}$, given access to the prover-strategy oracle P^*, can output a witness at least with the probability $p' = p - \kappa$ if P^* is accepted with the probability $p > \kappa$. Here, $\kappa = 2^{-n}$.

On input G, $\mathcal{K}^{P^*}$ first interacts with P^* to execute the protocol as an honest verifier. Let ch the challenge generated in Stage 3. If the proof of P^* is rejected, $\mathcal{K}^{P^*}$ output $\perp$ and aborts. Otherwise, $\mathcal{K}$ rewinds P^* to the beginning of Stage 3 to rerun the residual protocol with a random ch'_V. $\mathcal{K}^{P^*}$ repeats this until another acceptable proof occurs. Assume the second accepting proof occurs in jth iteration and the corresponding challenge is denoted by ch'. Thus, for the same commitments

$C = (C_1, \cdots, C_n)$, $\mathcal{K}^{P^*}$ obtains two accepting proofs corresponding to ch and ch' respectively . If $ch = ch'$, $\mathcal{K}^{P^*}$ outputs $\perp$ and aborts. Otherwise, there exists $j \in [n]$, such that $ch_j \neq ch'_j$, and $\mathcal{K}^{P^*}$ obtains π_i and Hamiltonian cycle H_i of $G_i = \pi_i(G)$. Finally, $\mathcal{K}^{P^*}$ outputs $H = \pi^{-1}(H_i)$ and stops. The details are in Figure 7.

Next, we show that $\mathcal{K}^{P^*}$ runs in the expected polynomial time.

In fact, Let p the probability that the proof of the first execution of the protocol is accepted by V, p' be the probability that P^* successfully responds to the query in Stage 4. From that fact that the ch, ch' are uniformly distributed if V is honest, it is easy to see that P correctly responds to ch and ch' with the same probability. Therefore, it follows that $p' \geq p$. (Because the fact the proof is accepted means that P^* correctly replies ch.). Notice that $\mathcal{K}^{\mathcal{P}^*}$ clearly runs in strict polynomial time when $p = 0$. So we assume that $p > 0$ in what follows. Therefore, the expected running time of $\mathcal{K}^{P^*}$ is given by

$$(1-p) \cdot poly(n) + p \cdot \frac{1}{p'} \cdot poly(n) = poly(n)$$

That is, $\mathcal{K}^{P^*}$ an expected polynomial-time algorithm.

Step 1	$\mathcal{K}$, playing the role of the honest verifier, interacts with P^* to perform the first three stages. Then, $\mathcal{K}$ obtains the commitment C_i to the adjacency of G_i ($i \in [n]$) from P^*.
Step 2	$\mathcal{K}$ honestly executes Stage 3. Concretly, $\mathcal{K}$ first randomly selects $ch_V \in_R \{0,1\}^n$) to run Π_{cd}. After receiving ch_P from P^*, $\mathcal{S}$ computes $ch = ch_P \oplus ch_V$.
Step 3	$\mathcal{K}$ honestly executes Stage 4 with P^*. After receiving the response to ch from P^*, $\mathcal{K}$ verifies the response. If the verification fails, $\mathcal{K}$ outputs $\perp$ and aborts. Otherwise, $\mathcal{K}$ proceeds next step.
Step 4	$\mathcal{K}$ rewinds P^* to run Stage 3, and obtains ch' at the end of Stage 3.
Step 5	$\mathcal{K}$ executes Stage 4 with P^*. After receiving the response to ch' from P^*, $\mathcal{K}$ verifies this response. If the verification fails, $\mathcal{K}$ returns to the point of the beginning of Step 4. Otherwise, $\mathcal{K}$ proceeds next step.
Step 6	If $ch = ch'$, $\mathcal{K}$ outputs $\perp$ and aborts. Otherwise, let i be such that $ch_i \neq ch'_i$, $\mathcal{K}$ obtains two accepting response corresponding to two different queries ch_i and ch'_i respectively, e. g. $ch_i = 0, ch'_i = 1$. Thus, for given commitment C_i, $\mathcal{K}$ obtains a Hamiltonian cycle $H_i$of G_i from the response to ch_i and a permutation π_i meeting $G_i = \pi_i(G)$ fron the response to ch'_i. Finally, $\mathcal{K}$ outputs $H = \pi^{-1}(H_i)$ and stops.

Figure 7: Knowledge extractor

Next, we show that the probability for $\mathcal{K}$ to output a Hamiltonian cycle is at least $p - 2^{-n}$ when $p > 2^{-n}$.

Let $ErrorFail$ be the event that the first proof is reject. Then, we have

$$p = \Pr[\neg ErrorFail] = 1 - \Pr[ErrorFail]$$

Let $CollisionFail$ denote the event that $\mathcal{K}^{P^*}$ outputs $\perp$ becasue of $ch = ch'$ in Step 6. Thus, $\mathcal{K}^{P^*}$ fails if $ErrorFail$ or $CollisionFail$ takes palce. We have

$$\Pr[\mathcal{K}^{P^*}(G) = \perp] = \Pr[ErrorFail] + \Pr[\neg ErrorFail \vee CollisionFail]$$

We denote by $CollisionFail_j$ the event that $\mathcal{K}^{P^*}$ outputs $\perp$ at jth repetition, that is, P^* fails to reply the correponding challenge in the first $j-1$ times repetition and correctly responds to ch'in jth iteration but $ch = ch'$. Thus, using $\Pr[ch = ch'] = 2^{-n}$, we have the following

$$\begin{aligned}
\Pr[\neg ErrorFail \vee CollisionFail] &= \Pr[\neg ErrorFail \vee (\vee_{j=1} CollisionFail_j)] \\
&= \Pr[\neg ErrorFail] \Pr[(\vee_{j=1} CollisionFail_j) | \neg ErrorFail] \\
&= \Pr[\neg ErrorFail] \sum_{j=1} \Pr[CollisionFail_j | \neg ErrorFail] \\
&= p \sum_{j=1} (1-p')^{j-1} \cdot 2^{-n} \\
&= 2^{-n} p \cdot \tfrac{1}{p'} \leq 2^{-n}
\end{aligned}$$

Therefore, the probability that $\mathcal{K}^{P^*}$ succeeds in computing a Hamiltonian cycle of G is

$$\begin{aligned}
Pr[\mathcal{K}^{P^*}(G) = H : (G,H) \in R_{HC}] &= 1 - \Pr[\neg ErrorFail \vee Collision] \\
&= 1 - \Pr[ErrorFail] - \Pr[\neg ErroFail \vee CollisionFail] \\
&\geq p - 2^{-n}
\end{aligned}$$

The proof is completed. ■

Acknowledgements
This work was partially supported the National Natural Science Foundation of China (Grant No. 60970139,61003276), Strategic Priority Program of Chinese Academy of Sciences (Grant No. XDA06010702).

References

1. M. Bellare, O. Goldreich. On defining proofs of knowledge. Advances in Cryptology-CRYPT'92, LNCS, vol.740, Springer-Verlag,1992, pages 390-420.
2. M. Bellare, O. Goldreich. On probabilistic versus deterministic provers in the definition of proofs of knowledge. Electronic Colloquium on Computational Complexity Report TR06-136.
3. Barak B., Goldreich O., Universal arguments and their applications. In Proc. of 17th IEEE Annual Conference on Computational Complexity, 2002, page 162-171.
4. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. SIAM Journal on computing, 1989, 18(16):186-208.
5. O. Goldreich. Foundations of Cryptography - Basic Tools. Cambridge University Press, 2001.
6. S. Halevi, S. Micali. More on proofs of knowledge. http://eprint.iacr.org/1998/015.
7. Hongda Li, Qihua Niu, Bei Liang. Leakage-Resilient Zero-Knowledge Proofs of Knowledge for NP. (to appear)
8. Itoh Toshiya, Sakurai Kouichi. On the Complexity of Constant Round ZKIP of Possession of Knowledge. IEICE TRANS. FUNDAMENTALS, 1993, VOL. E76-A, NO, 1:31-39.
9. Nir Bitansky, Ran Canetti, and Shai Halevi. Leakage tolerant interactive protocols. Cryptology ePrint Archive, Report 2011/204, 2011.
10. E.Boyle, S. Goldwasser, A. Jain and Yael T. Kalai. Multiparty computation secure against continual memory leakage. In proceedings of the 44th symposium on Theory of Computing, STOC '12, Pages 1235-1254.

11. Elette Boyle, Shafi Goldwasser, and Yael Tauman Kalai. Leakage-resilient coin tossing. In DISC, 2011.
12. Ivan Damgard, Carmit Hazay, and Arpita Patra. Leakage resilient two-party computation. Cryptology ePrint Archive, Report 2011/256, 2011.
13. Sanjam Garg, Abhishek Jain, and Amit Sahai. Leakage-resilient zero knowledge. In CRYPTO, pages 297-315, 2011.
14. O. Goldreich, A. Kahan. How to construct constant-round zero-knowledge proof system for NP. Journal of Cryptology, 1996,9(3):167-189.
15. Li Hongda, Feng Dengguo, Li Bao, Xu Haixia. Round-optimal zero-knowledge proofs of knowledge for NP. Science China: Information Science, 2012, 55(11):2417-2662.
16. Y. Lindell. Constant-Round Zero-Knowledge Proofs of Knowledge. http://eprint.iacr.org/2010/656.
17. Y. Lindell, B.Pinkas. A proof of security of Yao's protocol for two-party compuation. Journal of Cryptology, 22(2):161-188, 2009.
18. Moni Naor. Bit commitment using pseudo-randomness (extended abstract). In CRYPTO, pages 128-136, 1989.
19. Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In SODA, pages 448C457, 2001.
20. Omkant Pandey. Achieving constant round leakage-resilient zero knowledge. eprint.iacr.org/2012/362.pdf.
21. Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In FOCS, 2002.