

Efficient Zero-Knowledge for NP from Secure Two-Party Computation

Hongda Li^{1,2}, Dongxue Pan^{1,2}, Peifang Ni^{1,2}

¹ The Data Assurance and Communication Security Research Center,
Chinese Academy of Sciences, Beijing 100093, China

² State Key Lab of Information Security, Institute of Information Engineering,
School of Cyber Security, Chinese Academy of Sciences, Beijing 100093, China
pandongxue@iie.ac.cn, lihongda@iie.ac.cn, nipeifang@iie.ac.cn

Abstract. Ishai et al. [28, 29] introduced a powerful technique that provided a general transformation from secure multiparty computation (MPC) protocols to zero-knowledge (ZK) proofs in a black-box way, called “MPC-in-the-head”. A recent work [27] extends this technique and shows two ZK proof protocols from a secure two-party computation (2PC) protocol. The works [28, 27] both show a basic three-round ZK proof protocol which can be made negligibly sound by standard sequential repetition [19]. Under general black-box zero knowledge notion, neither ZK proofs nor arguments with negligible soundness error can be achieved in less than four rounds without additional assumptions [15].

In this paper, we address this problem under the notion of augmented black-box zero knowledge [26], which is defined with a new simulation method, called augmented black-box simulation. It is presented by permitting the simulator to have access to the verifier’s current private state (i.e. “random coins” used to compute the current message) in a special manner. We first show a three-round augmented black-box ZK proof for the language graph 3-colorability, denoted G3C. And then we generalize the construction to a three-round augmented black-box ZK proof for any NP relation $\mathcal{R}(x, w)$ without relying on expensive Karp reductions. The two constructions are based on a family of claw-free permutations and the general construction is additionally based on a black-box use of a secure 2PC for a related two-party functionality. Besides, we show our protocols can be made negligibly sound by directly parallel repetition.

Keywords: Zero-knowledge, claw-free permutation, two-party computation, parallel repetition.

1 Introduction

Zero-knowledge proofs were first introduced by Goldwasser, Micali, and Rackoff in [19]. Zero-knowledge proofs allow a prover to convince a verifier about the validity of a statement, while giving no information beyond the truth of the statement. Informally, an honest prover should always be able to convince an honest verifier about a true statement (completeness) while a malicious prover cannot convince a verifier of a false statement (soundness). Besides, a malicious verifier cannot learn anything beyond

the validity of the statement (zero-knowledge). The zero-knowledge property requires that for every probabilistic polynomial-time (PPT) verifier there exists a PPT algorithm (simulator), the output of which is indistinguishable from the view of the verifier in the real protocol.

Simulation techniques. Traditionally, zero-knowledge property is formalized by simulation and uses black-box (BB) simulator or non-black-box (NBB) simulator to simulate the verifier’s view. The BB simulator only has oracle access to the verifier’s strategy program while the NBB simulator can utilize the verifier’s strategy description as in [1] or [10]. [9, 12] promoted the simulator’s ability by using the “distinguisher-dependent” simulator, which permits the simulator to know the possibly cheating verifier’s program and the distinguisher. The simulator thus simulates the view of the verifier with respect to the distinguisher. One another NBB simulation is based on the knowledge assumptions [8, 23], which require that any algorithm that produces a DDH tuple, must have “knowledge” of the corresponding exponents specified by the DDH tuple. In this simulation technique, the simulator is given access to the specified exponent (secret coins) of the verifier by an efficient algorithm (extractor) and can complete the simulation.

Very recently, [26] proposed a new NBB simulation method, called augmented black-box simulation, which extends the idea of knowledge assumption. It is presented by permitting the simulator to have access to the verifier’s current private state (such as secret coins) in a special manner. The simulator simulates the prover P to interact with the verifier V . Upon receiving a message from V , the simulator computes P ’s response to it. Unlike the prover, the simulator receives not only V ’s message but also V ’s current private state. In essential, the simulation describes a way to make use of some internal data of the verifier’s computing process in order to say that whatever the verifier might have learned from the interaction with an honest prover, the verifier could have actually obtained by himself. Thus it requires that the power of the simulator is, in fact, inferior to that of the verifier. Although this simulation technique allows the simulator to receive the the verifier’s private state, it can only do what the verifier can do. This satisfies the requirement of zero-knowledge property in essential.

Three-round protocols. [22] first shows that three-round BB ZK for NP is not possible. Later, [15] shows that neither ZK proofs nor arguments for languages outside BPP with negligible soundness error can be achieved in less than four rounds without additional assumptions. However, there are still works [28, 27, 21, 2] on three-round BB ZK proofs with constant soundness error. [2] shows the classic three-round ZK proof of the NP-complete hamilton circle problem. [21] shows the classic three-round ZK proof of the NP-complete graph 3-colorability (G3C) problem. Ishai et al. [28] obtain a ZK proof for any NP relation with an multi-party computation (MPC) protocol in the commitment-hybrid model. They show the “MPC-in-the-head” approach [28], which provides a powerful tool to obtain black-box ZK proofs for generic statements that does not rely on expensive Karp reductions. Hazay et al. [27] extend this idea and give a generic transformation from a two-party computation (2PC) protocol to a ZK proof. The works [28, 27] both show a basic three-round BB ZK proof protocol which can be made negligibly sound by standard sequential repetition [19]. Our ZK proof for NP also

uses secure 2PC protocol to avoid the expensive Karp reduction of an NP language to an NP-complete language.

There are works [23, 25, 8, 9, 4, 3, 14, 7] on three-round NBB ZK for NP. The works [23, 8] provide a three-round NBB ZK argument from the knowledge assumptions, which allow the simulator to extract the verifier’s private coins and complete the simulation. Extending the idea of [23], [25] presents a three-round NBB ZK proof with Blum’s three-round ZK proof and the proof of knowledge assumption (POKA). The POKA allows the verifier to prove the knowledge of required private coins and thus the simulator can extract the verifier’s private coins. With the “distinguisher-dependent” simulation technique, [9] shows a three-round weak ZK argument from obfuscators for multibit point circuits. Three-round ZK argument protocols have been shown in restricted adversarial models where either the prover or the verifier is assumed to be uniform [4, 3]. Recently, [14] gives a negative result of three-round NBB ZK proofs with negligible soundness error against non-uniform verifiers under certain assumptions on program obfuscation [5]. The similar way has been used in [24] to rule out constant round public-coin ZK proof systems for NP. [14] also shows that the negative result applies to the protocol achieved by parallel repetitions of the protocol (executed by non-uniform verifiers) in [25]. Bitansky et al. [7] shows a three-round NBB ZK argument from keyless multi-collision resistant hash functions (MCRH). They construct the protocol with a three-round witness indistinguishable proof of knowledge (WIPOK) protocol, and thus that the language they defined in the protocol (from the proving statement and transcripts) must have a WIPOK, otherwise it needs a reduction process.

In this paper, we address this problem under the notion of augmented black-box zero knowledge [26]. We focus on the construction of three-round ZK proof protocols that can be made negligibly sound by directly parallel repetition.

1.1 Our Results

To the best of our knowledge, previous works on three-round ZK proofs under general black-box zero knowledge notion all formed ZK proofs in a similar way: the prover first make a “commitment”, which can later be opened to corresponding values according to the verifier’s different challenges. The simulator for the zero-knowledge property needs to guess the verifier’s challenge ch first, and aborts if the verifier’s algorithm oracle does not output ch as its challenge. This is the problem which breaks the zero-knowledge property of polynomial times parallel repetitions of previous three-round BB ZK proofs since the simulator can guess the verifier’s challenge with negligible probability.

In this paper, we address this problem under the notion of augmented black-box zero knowledge [26], which allows the simulator to have access to the verifier’s current private state in a special manner. We first show a three-round augmented black-box ZK proof for the language graph 3-colorability, denoted G3C. And then we generalize the construction to a three-round augmented black-box ZK proof for any NP relation $\mathcal{R}(x, w)$ without relying on expensive Karp reductions. The first construction is based on a family of claw-free permutations, a statistically binding commitment scheme, a pseudorandom generator, and a hard-core function for the claw-free permutations. The second construction needs a black-box use of a secure 2PC for a related two-party

functionality, where by black-box we mean that the functionality can be programmed to make only black-box (oracle) access to the relation \mathcal{R} . Besides claw-free permutations and 2PC, the general construction still needs hard-core functions and hash functions. The constructions will inherit the hardness assumptions required for the constructions of 2PC, which in case of [31, 11] only require trapdoor permutations while in case of [33, 20] will require the existence of an oblivious-transfer (OT) protocol.

Our three-round augmented black-box ZK proofs first let the verifier’s challenge be perfectly hiding from the prover and then the prover’s “commitment” and opened value can be sent together after the verifier’s “challenge” phase. Since the challenge is hiding from the prover, the prover cannot obtain the challenge, which guarantees the soundness of the proof system. However, the augmented black-box simulator can obtain the verifier’s challenge after the “challenge” phase and complete the simulation according to the obtained challenge. Our simulator does not need to guess the verifier’s challenge first, which helps show that our ZK proofs can be made negligibly sound by directly parallel repetition.

1.2 Related Works

ZK and MPC. Before ZK was introduced, Yao introduced secure 2PC and garbled circuits (GC) [34]. The general condition of 2PC is the problem of MPC [33, 20, 6], which requires a set of parties holding private inputs to compute a joint function while preserving the secrecy of each private input. A line of recent works [31, 32, 18, 11] address the long standing open problem of obtaining round-optimal secure computation. Recently, a line of works [28, 30, 27, 16] studies ZK with MPC. There are two main streams of works using the techniques of 2PC or MPC to obtain real efficient ZK protocols, one relies on “MPC-in-the-head” approach [28, 29] while the other relies on garbled circuit based approach [30].

Three-round NBB ZK. The works [23, 8] provide a ZK argument from the knowledge assumptions. And [25] presents a ZK proof with Blum’s three-round ZK proof and the proof of knowledge assumption (POKA). With the “distinguisher-dependent” simulation technique, [9] shows a weak ZK argument from obfuscators for multibit point circuits. Three-round ZK argument protocols also have been shown in restricted adversarial models [4, 3]. Recently, Bitansky et al. [7] shows a ZK argument from keyless multi-collision resistant hash functions (MCRH).

Negative result on three-round private coin ZK proof. Recently, [14] shows that three-round private coin (verifier keeps random coins private) ZK proofs against non-uniform verifiers does not exist under certain assumptions on program obfuscation [5]. The authors define an auxiliary input for the verifier, which is an obfuscation of the following program: upon receiving a message α from the prover, it computes a message β of the honest verifier using randomness $r = PRF_k(\alpha)$. With this obfuscated program, they implement a compiler for compressing any such ZK proof into a two-round argument, and they show that a simulator for the three-round proof can be used to construct a cheating prover for the two-round argument, which breaks the soundness of the two-round argument. The relation between our result and the result in [14] is unknown (see the details in section 3).

1.3 Outline

In section 2, we define the notations and definitions that are used through the paper. In section 3, we present our three-round augmented black-box ZK proof protocol for G3C. In section 4, we generalize the construction for G3C to a three-round augmented black-box ZK proof for any NP relation $\mathcal{R}(x, w)$ without relying on expensive Karp reductions.

2 Preliminaries

2.1 Notations

We use n to denote the security parameter. We use $[k]$ for any $k \in \mathbb{N}$ to denote the set $\{1, \dots, k\}$. For any probabilistic algorithm $A(\cdot)$, $A(x)$ is the result of executing A with input x and uniformly chosen randomness. We use $y = A(x)$ (or $y \leftarrow A(x)$) to denote that y is set to $A(x)$. For a set S , we use $y \in_R S$ to denote that y is uniformly chosen from S . For any language L and any instance $x \in L$, we denote by \mathcal{R}_L the efficiently computable binary NP relation for L . And for any witnesses w of $x \in L$, $\mathcal{R}_L(x, w) = 1$. We write $\text{negl}(\cdot)$ to denote an unspecified negligible function, $\text{poly}(\cdot)$ an unspecified polynomial. We denote by $a||b$ the concatenation of two bit strings a and b and $|a|$ the length of a . We use “ $X \stackrel{c}{\approx} Y$ ” to denote that probabilistic distributions X and Y are computationally indistinguishable. See the definition of hard-core functions, commitment schemes and two-party computation in Appendix A.

2.2 Claw-Free Permutation

Definition 2.1 Claw-Free Permutation [13]. For an index set $I \subseteq \{0, 1\}^*$, a collection of pairs of functions $C = \{(f_i : D_i \rightarrow D_i, g_i : D_i \rightarrow D_i) | i \in I\}$ is a family of claw-free permutations if:

- There is an efficient sampling algorithm $CFGen(1^n)$ which outputs a random index $i \in \{0, 1\}^n \cap I$ and a pair of trapdoors TK, TK' .
- There are efficient sampling algorithms which, on input i , output a random $x \in D_i$ and a random $z \in D_i$. We write $x \leftarrow D_i, z \leftarrow D_i$ as a shorthand.
- Each f_i (resp. g_i) is efficiently computable given index i and input $x \in D_i$ (resp. $z \in D_i$).
- Each f_i (resp. g_i) is a permutation which is efficiently invertible given the trapdoor information TK (resp. TK') and output $y \in D_i$. Namely, using TK (resp. TK') one can efficiently compute (unique) $x = f_i^{-1}(y)$ (resp. $z = g_i^{-1}(y)$).
- For any probabilistic algorithm B , define the advantage of B as

$$\text{Adv}_B^C(n) = \Pr[f_i(x) = g_i(z) | (i, TK, TK') \leftarrow CFGen(1^n), (x, z) \leftarrow B(i)].$$

If B runs in time at most $t(n)$ and $\text{Adv}_B^C(n) \geq \varepsilon(n)$, then B is said to $(t(n), \varepsilon(n))$ -break C . C is said to be $(t(n), \varepsilon(n))$ -secure if no adversary B can $(t(n), \varepsilon(n))$ -break it. In the asymptotic setting, we require that the the advantage of any PPT B is negligible in n . Put differently, it is hard to find a “claw” (x, z) (meaning $f_i(x) = g_i(z)$) without the trapdoors TK, TK' .

We show one example of claw-free permutation based on RSA modified from [13]. $CFGen(1^n)$ picks two random $n/4$ -bit primes p and q , sets $n_{RSA} = pq$, $\phi(n_{RSA}) = (p-1)(q-1)$, picks random $e \in Z_{\phi(n_{RSA})}^*$, sets $d = e^{-1} \bmod \phi(n_{RSA})$, picks a random $y^* \in Z_{n_{RSA}}^*$ and outputs $i = (n_{RSA}, e, y^*)$, $TK = d$, $TK' = d$. Here $D_i = Z_{n_{RSA}}^*$, $f_i(x) = x^e \bmod n_{RSA}$, $f^{-1}(y) = y^d \bmod n_{RSA}$, $g_i(z) = y^* z^e \bmod n_{RSA}$, $g^{-1}(y) = (y/y^*)^d \bmod n_{RSA}$. Finding a claw (x, z) implies $y^* = (x/z)^e \bmod n_{RSA}$, which implies inverting RSA on a random input y . [13] also shows a construction of claw-free permutations from homomorphic trapdoor permutations, which is a generalization of the RSA -based construction.

2.3 Augmented Black-box Zero knowledge

Definition 2.2 Interactive Proof System. 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 such that for every $x \in L$,

$$\Pr[\langle P, V \rangle(x) = 1] > 1 - c(|x|)$$

- *Soundness:* There exists a negligible function s such that for every $x \notin L$ and every interactive machine B , it holds that

$$\Pr[\langle B, V \rangle(x) = 1] < s(|x|)$$

$c(\cdot)$ is called the completeness error, and $s(\cdot)$ the soundness error.

Let $\langle P, V \rangle$ be an interactive proof system for a language L . We use $Next_V(x; \cdot; \cdot)$ to denote the next message function of an honest V with common input x . Then the function $Next_V(x; \cdot; \cdot)$ is a public algorithm. Define the message sent by V in the i -th round as

$$\alpha_i = Next_V(x; i, \beta; R_i),$$

where β is the collection of all received messages and R_i is the collection of all random coins used to produce α_i . If the computation of α_i involves the random coins chosen by V before round i , they would be in R_i . Then the current private state $state_V^{(i)}$ here for an honest V consists of R_i and a bit ($\Delta_i = 1$, which indicates that there is a data collection R_i satisfying $\alpha_i = Next_V(x; i, \beta; R_i)$). Notice that if α_i contains multiple parts as $\alpha_i = (\alpha_{i,1}, \dots, \alpha_{i,poly(n)})$, then each part $\alpha_{i,j}$ ($j \in [poly(n)]$) can correspond to a state as $state_V^{(i,j)} = (R_{i,j}, \Delta_{i,j} = 1)$ and thus

$$state_V^{(i)} = state_V^{(i,1)}, \dots, state_V^{(i,poly(n))}.$$

We recall the definition of the extended verifier \widehat{V} [26] (an imaginary verifier) for any honest V . \widehat{V} does the same as V when it interacts with P except that it records its secret state on the private output tape, the tape V does not have. The next message function of \widehat{V} , denoted by $Next_{\widehat{V}}(x; \cdot; \cdot)$, is defined as follows:

$$\left(\alpha_i, state_V^{(i)} \right) \leftarrow Next_{\widehat{V}}(x; i, \beta; R_i),$$

where $\alpha_i = \text{Next}_V(x; i, \beta; R_i)$ is written on \widehat{V} 's communication output tape, and V 's current private state $\text{state}_V^{(i)} = (R_i, \Delta = 1)$ is written on \widehat{V} 's private output tape.

For any malicious verifier V^* , the corresponding extended verifier \widehat{V}^* computes α_i with the secret next message function

$$\alpha_i = \text{Next}_{V^*}(x, z, i, \beta; R_{V^*})$$

selected by V^* and outputs

$$\left(\alpha_i, \text{state}_{V^*}^{(i)} = (R'_i, \Delta_i = 1) \right)$$

if V^* knows a data collection R'_i such that the public next message function holds: $\alpha_i = \text{Next}_V(x; i, \beta; R'_i)$; otherwise \widehat{V}^* outputs $(\alpha_i, \text{state}_{V^*}^{(i)} = (\Delta_i = 0))$. For the condition $\alpha_i = (\alpha_{i,1}, \dots, \alpha_{i,\text{poly}(n)})$, each part $\alpha_{i,j}$ ($j \in [\text{poly}(n)]$) corresponds to a state $\text{state}_V^{(i,j)} = (R_{i,j}, \Delta_{i,j} = 1)$ or $\text{state}_V^{(i,j)} = (\Delta_{i,j} = 0)$.

Let $\mathcal{O}_{\widehat{V}^*}$ be the oracle of the next message function of \widehat{V}^* . Then, when being queried, $\mathcal{O}_{\widehat{V}^*}(x, z, \cdot; \cdot)$ first computes

$$\alpha_i = \text{Next}_{V^*}(x, z, i, \beta; R_{V^*}),$$

where the function $\text{Next}_{V^*}(x, z, \cdot; \cdot)$ is the secret next message function selected by V^* and may be different from the public function, and then answers with $(\alpha_i, \text{state}_{V^*}^{(i)})$.

Definition 2.3 Augmented Black-box Zero-knowledge Proof. Let $\langle P, V \rangle$ be an interactive proof system for a language L . $\langle P, V \rangle$ is called an augmented black-box computational zero-knowledge proof if for every PPT V^* , the corresponding extended verifier is denoted by \widehat{V}^* , there exists an augmented black-box simulator S with oracle access to $\mathcal{O}_{\widehat{V}^*}$, such that for any auxiliary input z , it holds for all statement $x \in L$:

- 1) The probability that $S^{\mathcal{O}_{\widehat{V}^*}}(x, z)$ fails is at most $\frac{1}{2}$.
- 2) Under the condition that the augmented black-box simulator does not fail, the view of V^* in the real protocol $\text{View}_{V^*}^P(x)$ and $S^{\mathcal{O}_{\widehat{V}^*}}(x, z)$ are computationally indistinguishable.

The above definition of zero-knowledge property is reasonable, since although this simulation technique allows the simulator to receive the the verifier's private state, the power of the simulator is inferior to that of the verifier. With this kind of simulator, we can see that the interaction with an honest prover will not improve the verifier's computing power. This satisfies the requirement of zero-knowledge property in essential.

3 Three-round Zero-knowledge Proof for G3C

The language G3C consists of all simple (finite) graphs that can be vertex-colored using three colors such that no two adjacent vertices are given the same color. Formally, a graph $G = (V, E)$ (V is the vertices collection and E is the edges collection) is 3-colorable if there exists a mapping $\phi : V \rightarrow \{1, 2, 3\}$ such that $\phi(u) \neq \phi(v)$ for every $(u, v) \in E$. In this section, we aim to construct a three-round augmented black-box ZK proof for G3C, which is an NP-complete problem. Our proof system can be made negligibly sound by polynomial times parallel repetitions.

3.1 Construction

Let Π be a collection of permutations over $\{1, 2, 3\}$. Let $C = \{(f_i : D_i \rightarrow D_i, g_i : D_i \rightarrow D_i) | i \in I\}$ with $D_i = \{0, 1\}^n$ be a family of claw-free permutations. Let Com be any statistically binding non-interactive commitment scheme and let $Com_r(m)$ denote a commitment to m with random coins r . Let $h : \{0, 1\}^n \rightarrow \{0, 1\}^{l(n)}$ be any hard-core function for all functions in C . Let $\mathcal{G} : \{0, 1\}^{l(n)} \rightarrow \{0, 1\}^{poly(n)}$ be a pseudorandom generator, where $poly(n) > l(n)$ is fixed according to the construction.

We now describe our zero-knowledge protocol $\Pi_{G3C} = \langle P, V \rangle$ for the NP relation \mathcal{R}_{G3C} . The prover and verifier are both given an input $G = (V, E)$ with $V = \{1, \dots, n\}$, $|E| = 2^k$ (a polynomial in n). The prover is also given a witness ϕ (a mapping $\phi : V \rightarrow \{1, 2, 3\}$) for $G \in G3C$.

In first step, the prover samples $i_1, \dots, i_k \in_R I$ and obtains the corresponding k pairs of claw-free permutations ($f^j = f_{i_j}, g^j = g_{i_j}$) and k pairs of trapdoors (TK_{f^j}, TK_{g^j}) for (f^j, g^j) respectively. And the prover sends (f^j, g^j) for $j \in [k]$.

In the challenge phase, the verifier V then chooses $a_1, \dots, a_k \in_R \{0, 1\}^n$ independently, and chooses k bits $b_1, \dots, b_k \in \{0, 1\}$ at random. And the verifier computes

$$y_j = \begin{cases} f^j(a_j), & \text{if } b_j = 0 \\ g^j(a_j), & \text{if } b_j = 1 \end{cases}$$

and sends y_1, \dots, y_k .

To responds, the prover computes $a_j^0 = (f^j)^{-1}(y_j), a_j^1 = (g^j)^{-1}(y_j)$ with trapdoors (TK_{f^j}, TK_{g^j}) respectively for $j \in [k]$, and then computes

$$h_j^b = h(a_j^b), b \in \{0, 1\}, j \in [k].$$

Then, for any $\alpha = (\alpha_1, \dots, \alpha_k) \in \{0, 1\}^k$, the prover computes

$$\beta_\alpha = \bigoplus_{j \in [k]} h_j^{\alpha_j} = h_1^{\alpha_1} \oplus \dots \oplus h_k^{\alpha_k},$$

where \oplus denotes the exclusive OR operation on the binary coded values. And then, the prover selects a random permutation $\pi \in \Pi$ on $\{1, 2, 3\}$, sets $col_s = \pi(\phi(s))$ for all $s \in V$, and computes $c_s = Com_{r_s}(col_s)$ with uniformly randomness r_s . Finally, for every edge in $E = \{(u_\alpha, v_\alpha)\}_{\alpha \in \{0, 1\}^k}$ ($u_\alpha, v_\alpha \in V$ are the corresponding vertices of that edge) the prover computes $e_\alpha = \mathcal{G}(\beta_\alpha) \oplus (col_{u_\alpha}, r_{u_\alpha}, col_{v_\alpha}, r_{v_\alpha})$ and sends $\{c_s\}_{s \in V}, \{e_\alpha\}_{\alpha \in \{0, 1\}^k}$.

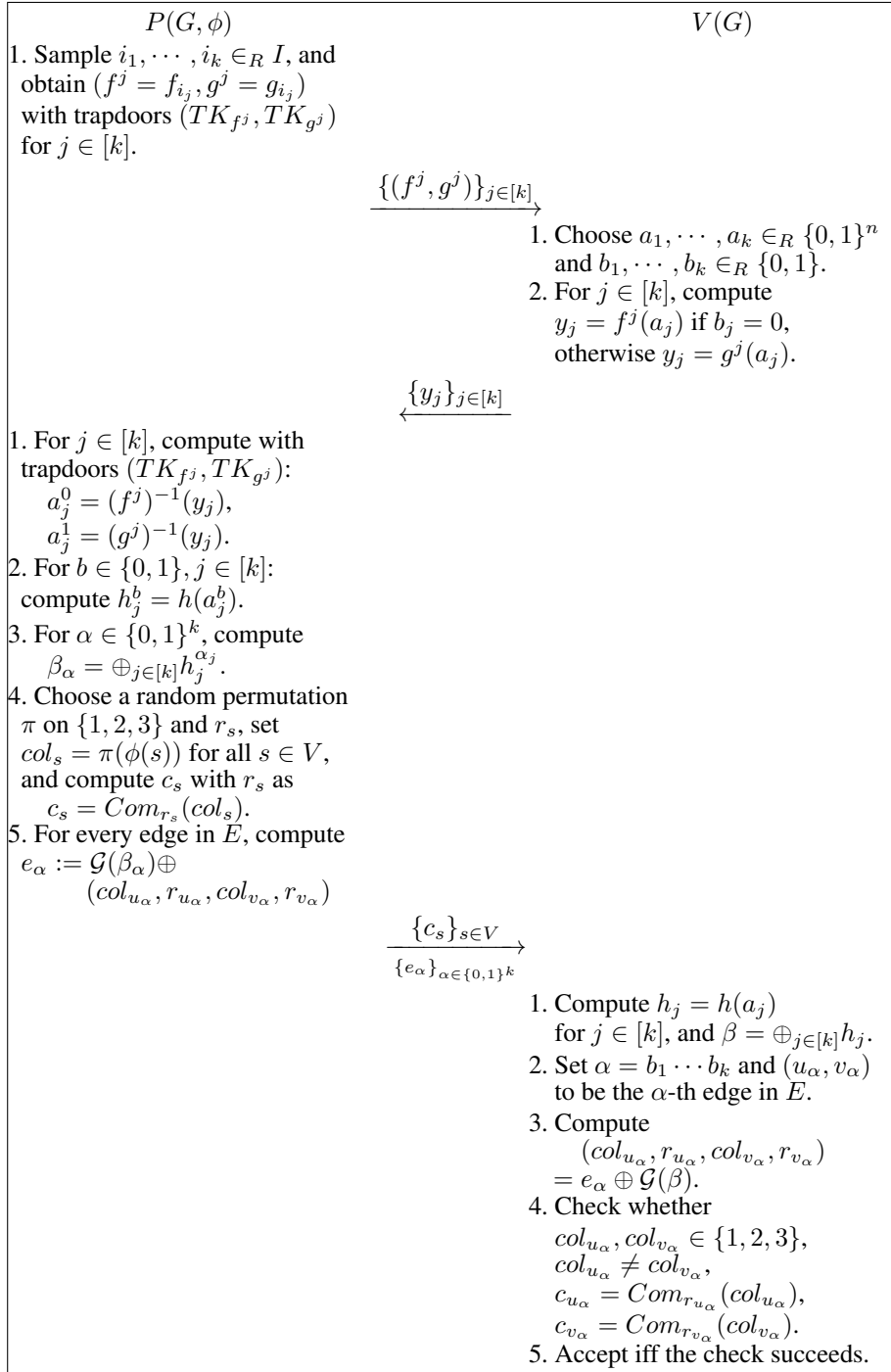
In the verification phase, the verifier computes $\{h_j = h(a_j)\}_{j \in [k]}, \beta = \bigoplus_{j \in [k]} h_j$, sets $\alpha = b_1 \dots b_k$, and sets (u_α, v_α) to be the α -th edge in E , where $u_\alpha, v_\alpha \in V$ are the corresponding vertices of that edge. And then, the verifier computes

$$(col_{u_\alpha}, r_{u_\alpha}, col_{v_\alpha}, r_{v_\alpha}) = e_\alpha \oplus \mathcal{G}(\beta)$$

and checks that whether $col_{u_\alpha}, col_{v_\alpha} \in \{1, 2, 3\}$ are distinct and

$$c_{u_\alpha} = Com_{r_{u_\alpha}}(col_{u_\alpha}), c_{v_\alpha} = Com_{r_{v_\alpha}}(col_{v_\alpha}).$$

The verifier accepts the proof only if the check succeeds. The protocol is depicted in Figure 1.

Figure 1. Three-round ZK protocol Π_{G3C}

Theorem 3.1 *Assuming the existences of pseudorandom generators, a family C of claw-free permutations on $\{0, 1\}^n$, statistically binding non-interactive commitment schemes, and hard-core functions for all functions in C , the construction of Π_{G3C} in Figure 1 is an augmented black-box zero-knowledge proof.*

Proof. Completeness. If the prover holding witness and the verifier execute the protocol honestly, the verifier will obtain $(col_{u_\alpha}, r_{u_\alpha}, col_{v_\alpha}, r_{v_\alpha})$ hiding in e_α , which are the committed colors and randomness used in $c_{u_\alpha}, c_{v_\alpha}$. Since the prover holding witness behaves honestly, col_{u_α} and col_{v_α} must be distinct. Hence then, the verifier will accept the proof.

Soundness. From the property of permutations on $\{0, 1\}^n$, for any received $y_j \in \{0, 1\}^n$ ($j \in [k]$), a malicious prover can invert both f^j and g^j on y_j with corresponding trapdoors and obtain a_j^0, a_j^1 . Hence then, the prover cannot predict that which value in $\{a_j^0, a_j^1\}$ is the one used by the verifier to compute y_j . This perfectly hides the challenge $\alpha = (b_1, \dots, b_k)$ -th edge chosen by the verifier from the prover.

And since $G = (V, E) \notin G3C$, for any mapping $\phi : V \rightarrow \{1, 2, 3\}$, there would be at least one edge, of which the two vertices are in same color. And then no matter how the prover behaves, from the statistically binding property of the commitment scheme, for the colors of the vertices committed in $\{col_s\}_{s \in V}$, there would still be at least one edge, of which the two vertices are in same color. Therefore, the soundness error is no more than $1 - |E|^{-1}$.

Augmented black-box zero-knowledge. Since zero-knowledge property is required against any PPT verifier, we consider that any PPT (distinguishing) algorithm providing auxiliary input to the verifier is a part of the verifier algorithm. I.e., if a PPT algorithm V_1 provides auxiliary input z to a verifier V , we view the composition of V and V_1 as a new PPT verifier V^* . Then our simulator is given access to V^* 's current private state. Hence then, we assume that the auxiliary input given to the verifier cannot be computed by any PPT algorithm (this assumption keeps for the proofs of other protocols).

Fix any statement $G \in G3C$, for any PPT verifier V^* , the corresponding extended verifier is denoted by \widehat{V}^* (see subsection 2.3), and let $\mathcal{O}_{\widehat{V}^*}$ denote the oracle of the next message function of \widehat{V}^* . Then, for any auxiliary input z that either is an empty string or cannot be computed by any PPT algorithm, the augmented black-box simulator $S^{\mathcal{O}_{\widehat{V}^*}}(G, z, 1^n)$ proceeds as follows:

1. On input $G = (V, E)$ where $V = \{1, \dots, n\}$, $|E| = 2^k$, sample $i_1, \dots, i_k \in_R I$ and obtain the corresponding k pairs of claw-free permutations $(f^j = f_{i_j}, g^j = g_{i_j})$ and k pairs of trapdoors (TK_{f^j}, TK_{g^j}) for (f^j, g^j) respectively.
2. Choose random coins r_{V^*} , invoke $\mathcal{O}_{\widehat{V}^*}(G, z; \cdot; \cdot)$ with $(2, r_{V^*}, \{(f^j, g^j)\}_{j \in [k]})$ and obtain $\{y_j\}_{j \in [k]}$ and V^* 's current private state $\{(b_j, a_j, \Delta_j)\}_{j \in [k]}$.
I.e., if $\Delta_j = 1$, $y_j = f^j(a_j)$ when $b_j = 0$ and $y_j = g^j(a_j)$ when $b_j = 1$; if $\Delta_j = 0$, $a_j = \perp$, $b_j = 0$ if y_j is computed from f^j and $b_j = 1$ if y_j is computed from g^j , otherwise $b_j = \perp$.
3. If $\perp \notin \{b_j\}_{j \in [k]}$:
 - compute $h_j = h(a_j)$ for $j \in [k]$, and then compute $\beta_{b_1, \dots, b_k} = \bigoplus_{j \in [k]} h_j$;
 - let $u, v \in V$ denote the two vertices corresponding to the (b_1, \dots, b_k) -th edge in E , select $col_u \neq col_v \in \{1, 2, 3\}$, and set $col_s = 1$ for $s \in V$ and $s \neq u, v$;

- compute $c_s = Com_{r_s}(col_s)$ with randomness r_s for $s \in V$;
 - compute $e_{b_1, \dots, b_k} = \mathcal{G}(\beta_{b_1, \dots, b_k}) \oplus (col_u, r_u, col_v, r_v)$ and select random $e_\alpha \in \{0, 1\}^{l'(n)}$ ($l'(n) = 2(|r_s| + |col_s|)$) for $\alpha \in \{0, 1\}^k$ and $\alpha \neq (b_1, \dots, b_k)$.
- If $\perp \in \{b_j\}_{j \in [k]}$: set $col_s = 1$ for $s \in V$, compute $c_s = Com_{r_s}(col_s)$ with randomness r_s for $s \in V$, and select random $e_\alpha \in \{0, 1\}^{l'(n)}$ ($l'(n) = 2(|r_s| + |col_s|)$) for $\alpha \in \{0, 1\}^k$.
4. Output $(G, r_{V^*}, \{(f^j, g^j)\}_{j \in [k]}, \{c_s\}_{s \in V}, \{e_\alpha\}_{\alpha \in \{0, 1\}^k})$.

Now we show that the output of $S^{\mathcal{O}_{\widehat{V^*}}}(G, z, 1^n)$ is computational indistinguishable with the real protocol view $View_{V^*(z)}^P(G)$ of V^* . This follows from the hiding property of the commitment scheme Com and the properties of the claw-free permutations, the hard-core function h and the pseudorandom generator \mathcal{G} .

We analyze the computations of a prover as follows. Firstly, since it is hard to find a ‘‘claw’’ of (f^j, g^j) for every $j \in [k]$, the verifier V^* knows at most one of a_j^0, a_j^1 such that $y_j = f^j(a_j^0) = g^j(a_j^1)$. For any unknown a_j^b ($b \in \{0, 1\}, j \in [k]$), the value $h(a_j^b) \in \{0, 1\}^{l(n)}$ is indistinguishable with a random value in $\{0, 1\}^{l(n)}$, which is guaranteed by the property of the hard-core function h . Then the β_α computed with a such $h(a_j^b)$ is indistinguishable with a random value in $\{0, 1\}^{l(n)}$. And thus from the property of the pseudorandom generator \mathcal{G} , the e_α computed with a such β_α is indistinguishable with a random value in $\{0, 1\}^{l'(n)}$, where $l'(n) = 2(|r_s| + |col_s|)$. From the hiding property of the commitment scheme Com , the commitments of $\{col_s\}_{s \in V}$ made by the prover are indistinguishable with those made by the simulator.

We analyze the computations of the simulator as follows. If the verifier V^* knows a preimage for each y_j ($j \in [k]$), V^* does choose an challenge edge and wants to check whether the colors of the two vertices corresponding to that edge are different. In this case, the simulator would obtain the challenge (b_1, \dots, b_k) -th edge in E and commit the corresponding two vertices to different colors. This would convince V^* as the prover does. Notice that if the private coins of computing each y_j is hiding from the verifier, it is similar to the above condition of unknown a_j^b ($b \in \{0, 1\}, j \in [k]$).

Therefore, we have $View_{V^*(z)}^P(G) \stackrel{c}{=} S^{\mathcal{O}_{\widehat{V^*}}}(G, z, 1^n)$. □

The result in [14] proves that three-round NBB ZK proof cannot exist for language beyond BPP under the assumptions of sub-exponentially secure one-way functions, sub-exponentially secure indistinguishability obfuscation for circuits, and exponentially secure input-hiding obfuscation for multi-bit point functions. The augmented black-box simulation technique uses different simulation models (see the above proving process as an example), which makes our result not need to assume or deny the existences of the assumptions of [14]. Hence then, the relation between our result and the assumptions of [14] is unknown, which implies that the relation between our result and the result in [14] is unknown.

3.2 Construction with Negligible Soundness Error

We show our three-round ZK proof protocol constructed in Figure 1 can be made negligibly sound by parallel repetitions.

Theorem 3.2 *Assuming the existences of pseudorandom generators, a family C of claw-free permutations on $\{0, 1\}^n$, statistically binding non-interactive commitment schemes, and hard-core functions for all functions in C , then construction of Π_{G3C} in Figure 1 is an augmented black-box zero-knowledge proof. Furthermore, the protocol obtained from $p(n) = n \cdot 2^k$ (2^k is the number of edges in the graph) parallel repetitions of $\Pi_{\mathcal{R}_L}$ is an augmented black-box zero-knowledge proof with soundness error $2^{-\Omega(n)}$.*

Proof. The completeness follows from the completeness of each execution of the basic three-round protocol Π_{G3C} . The verifier accepts the proof if and only if each execution of the basic three-round protocol Π_{G3C} is accepted. Then the soundness error is $(1 - 2^{-k})^{n \cdot 2^k}$.

Augmented black-box zero-knowledge. The proof follows from the proof of Theorem 3.1, which implies that the indistinguishability is obtained from the hiding property of the commitment scheme Com and the properties of the claw-free permutations, the hard-core function h and the pseudorandom generator \mathcal{G} . See the full proof in Appendix B. \square

4 General Construction for Any NP Relation

As previously stated, the purpose of this section is to obtain a three-round augmented black-box ZK proof for any NP relation $\mathcal{R}_L(x, w)$, where L is any language in NP. Besides, we show our ZK proof can be made negligibly sound by directly parallel repetition.

Loosely speaking, the strategy of our augmented black-box ZK proof is to let the verifier's challenge be perfectly hiding from the prover and computationally binding for the verifier. After the challenge phase, the verifier obtain the prover's "commitment" and the opened value corresponding to the chosen challenge from the last message sent by the prover. The verifier accepts the proof if the opened value corresponding to the chosen challenge is consistent with the "commitment". However, the augmented black-box simulator can obtain the verifier's challenge after the "challenge" phase and complete the simulation according to the obtained challenge. Our simulator does not need to guess the verifier's challenge first, which helps show that our ZK proof can be made negligibly sound by directly parallel repetition.

Concretely, the "commitment" is the transcript between two parties in a secure semi-honest 2PC protocol for $F(x, w_0, w_1) = \mathcal{R}_L(x, w_0 \oplus w_1)$, where the w_b ($b \in \{0, 1\}$) holding by party P_b is chosen by the prover with w for $x \in L$ such that $w_0 \oplus w_1 = w$. The verifier obtains the opened value w_b, r_b corresponding to the challenge b , where r_b is the random value sued by P_b .

4.1 Construction

Let x denote a statement in an NP language L , associated with relation \mathcal{R}_L , and let F be the following function corresponding to \mathcal{R}_L : $F(x, w_0, w_1) = \mathcal{R}_L(x, w_0 \oplus w_1)$, where \oplus here denotes bitwise exclusive-or of two strings (both of the same length). We

view F as an two-party functionality specified by x , where x is a public input known to both players, w_b ($b \in \{0, 1\}$) is a private input of player P_b . Then we let $\Pi_F = (P_0, P_1)$ denote a two-party protocol that privately realizes F with perfect correctness.

Let $C = \{(f_i : D_i \rightarrow D_i, g_i : D_i \rightarrow D_i) \mid i \in I\}$ with $D_i = \{0, 1\}^n$ be a family of claw-free permutations. Let $\mathcal{H} = \{H_s : \{0, 1\}^* \rightarrow \{0, 1\}^{k(n)}\}_{s \in \{0, 1\}^n}$ be a family of hash functions. For an arbitrary one-way function f on $\{0, 1\}^n$, let $B_f(a, r) = \langle a, r \rangle$ be the Goldreich-Levin hard-core predicate for the function $g(a, r) = (f(a), r)$. Then, for $l = l(n)$ independent one-way functions f^1, \dots, f^l on $\{0, 1\}^n$, the function $H_f(a, r)$ defined as follows is a hard-core function for $g(a, r) = (f(a), r)$:

$$H_f(a_1, \dots, a_l, r_1, \dots, r_l) = (B_{f^1}(a_1, r_1), \dots, B_{f^l}(a_l, r_l)),$$

where $a = a_1, \dots, a_l$, $r = r_1, \dots, r_l$ are uniformly distributed over $\{0, 1\}^{nl}$, and $f(a) = f^1(a_1), \dots, f^l(a_l)$. This can be proven with the following sequence of hybrid distributions.

$$h_0 = f(a), H_f(a, r) = f(a), (B_{f^1}(a_1, r_1), \dots, B_{f^l}(a_l, r_l)),$$

$$h_i = f(a), (B_{f^1}(a_1, r_1), \dots, B_{f^{l-i}}(a_{l-i}, r_{l-i}), U_i), i \in [l],$$

where U_i is uniformly distributed over $\{0, 1\}^i$ and $h_l = f(a), U_l$. Let $|f(a)| = k$, then h_{i-1} and h_i only differ in the $(k + l - i + 1)$ -th bit. The $(k + l - i + 1)$ -th bit of h_{i-1} is a hard-core predicate while the $(k + l - i + 1)$ -th bit of h_i is a random bit. Then, to prove that the function $H_f(a, r)$ is a hard-core function for $g(a, r) = (f(a), r)$ is to prove that $h_0 \stackrel{c}{=} h_l$. On the contrary, if h_0 and h_l are distinguishable, there must exist an $i \in [l]$ such that h_{i-1} and h_i are distinguishable, which breaks the unpredictability (see definition A.1) of the Goldreich-Levin hard-core predicate $B_{f^{l-i+1}}(a_{l-i+1}, r_{l-i+1})$ for the function

$$g^{l-i+1}(a_{l-i+1}, r_{l-i+1}) = (f^{l-i+1}(a_{l-i+1}), r_{l-i+1}).$$

Hence then, the function $H_f(a, r)$ is a hard-core function for $g(x, r) = (f(x), r)$.

We now describe our zero-knowledge protocol $\Pi_{\mathcal{R}_L} = \langle P, V \rangle$ for the NP-relation \mathcal{R}_L . The prover and verifier are both given an input x and they both have a black-box access to the 2PC protocol Π_F . The prover is also given a witness w for $x \in L$.

Our first step in constructing a ZK proof involves the prover P emulating ‘‘in-her-head’’ the execution of Π_F on input (x, w_0, w_1) by first sampling w_0 and w_1 at random such that $w_0 \oplus w_1 = w$, and this involves choosing randomness r_0, r_1 for the two players respectively and running the protocol. We assume that the shorter one in $\{r_0, r_1\}$ is padded to the same length as the other with 0. Based on this execution, the prover prepares the transcript τ between the two players and two values $m_0 = w_0 || r_0, m_1 = w_1 || r_1$, and sets $k = |m_0| = |m_1|, l = 2k$. Then, the prover samples $i_1, \dots, i_l \in_R I$ and obtain the corresponding l pairs of claw-free permutations ($f^j = f_{i_j}, g^j = g_{i_j}$) and l pairs of trapdoors (TK_{f^j}, TK_{g^j}) for (f^j, g^j) respectively. And the prover sends (f^j, g^j) for $j \in [l]$.

In the challenge phase, the verifier V then chooses $a_1, \dots, a_l \in_R \{0, 1\}^n$ independently, sets $a = (a_1, \dots, a_l)$, and chooses $b \in \{0, 1\}$ at random. And the verifier computes $y_j = f^j(a_j)$ if $b = 0$, otherwise computes $y_j = g^j(a_j)$, and sends y_1, \dots, y_l .

To respond to the challenge, the prover computes $a_j^0 = (f^j)^{-1}(y_j)$ and $a_j^1 = (g^j)^{-1}(y_j)$ with trapdoors (TK_{f^j}, TK_{g^j}) respectively for $j \in [l]$. Then, the prover chooses $r_f^1, \dots, r_f^l, r_g^1, \dots, r_g^l \in_R \{0, 1\}^n$ independently, sets the values

$$a^0 = (a_1^0, \dots, a_l^0), a^1 = (a_1^1, \dots, a_l^1), r_f = (r_f^1, \dots, r_f^l), r_g = (r_g^1, \dots, r_g^l),$$

and defines

$$f(a^0) = (f^1(a_1^0), \dots, f^l(a_l^0)), f'(a^0, r_f) = (f(a^0), r_f),$$

$$g(a^1) = (g^1(a_1^1), \dots, g^l(a_l^1)), g'(a^1, r_g) = (g(a^1), r_g).$$

Then, the prover obtains $H_f(a^0, r_f)$ for $f'(a^0, r_f)$ and $H_g(a^1, r_g)$ for $g'(a^1, r_g)$ as defined above, where $|H_f(a^0, r_f)| = |H_g(a^1, r_g)| = l = 2k$. The prover chooses two random hash functions H_{s_0}, H_{s_1} from $\mathcal{H} = \{H_s : \{0, 1\}^* \rightarrow \{0, 1\}^{k(n)}\}_{s \in \{0, 1\}^n}$, which map an arbitrary string in $\{0, 1\}^{2k}$ to a string in $\{0, 1\}^k$, and sets

$$c_0 = H_{s_0}(H_f(a^0, r_f)) \oplus m_0 = H_{s_0}(\langle a_1^0, r_f^1 \rangle, \dots, \langle a_l^0, r_f^l \rangle) \oplus m_0,$$

$$c_1 = H_{s_1}(H_g(a^1, r_g)) \oplus m_1 = H_{s_1}(\langle a_1^1, r_g^1 \rangle, \dots, \langle a_l^1, r_g^l \rangle) \oplus m_1,$$

and sends $\tau, c_0, c_1, r_f, r_g, H_{s_0}, H_{s_1}$, where τ is obtained in the first step.

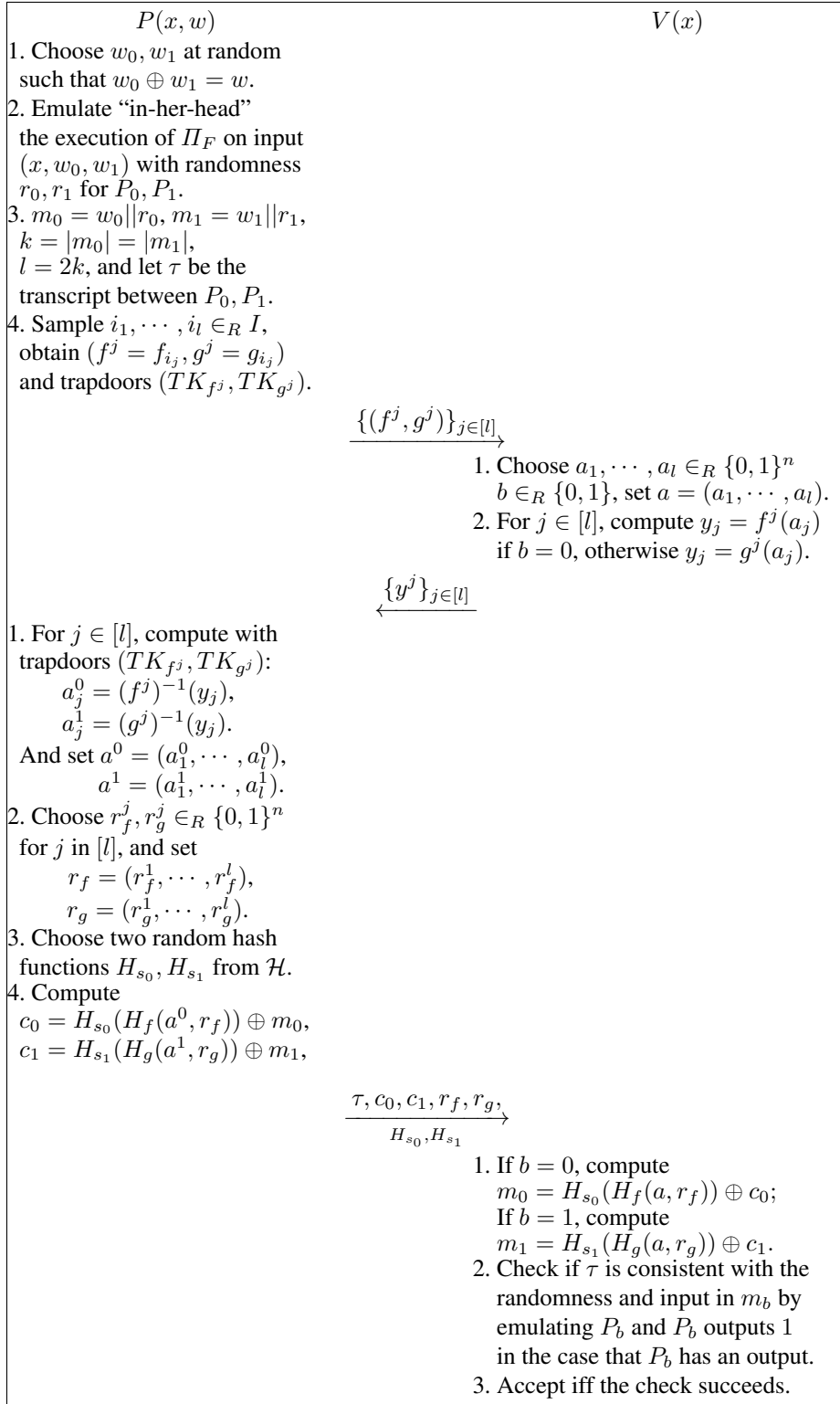
In the verification phase, the verifier computes $m_0 = H_{s_0}(H_f(a, r_f)) \oplus c_0$ if $b = 0$, otherwise the verifier computes $m_1 = H_{s_1}(H_g(a, r_g)) \oplus c_1$, and then checks if the transcript τ is consistent with the randomness and input in m_b by emulating the corresponding party P_b . In the case that P_b has an output, the verifier checks that P_b outputs 1. The verifier accepts if the check succeeds. The protocol $\Pi_{\mathcal{R}_L} = \langle P, V \rangle$ is depicted in Figure 2.

Theorem 4.1 *Assuming the existences of claw-free permutations on $\{0, 1\}^n$, hash functions, and secure semi-honest two-party computation protocols, the construction of $\Pi_{\mathcal{R}_L}$ in Figure 2 is an augmented black-box zero-knowledge proof.*

Proof. Completeness. If the prover holding witness and the verifier execute the protocol honestly, the verifier will obtain $m_b = w_b || r_b$, the check of consistency between τ and m_b will succeed, and P_b will output 1 in the case that P_b has an output. Hence then, the verifier will accept the proof.

Soundness. From the property of permutations on $\{0, 1\}^n$, for any received $y_j \in \{0, 1\}^n$ ($j \in [l]$), a malicious prover can invert both f^j and g^j on y_j with corresponding trapdoors and obtain a_j^0, a_j^1 . Hence then, the prover cannot predict that which value in $\{a^0 = (a_1^0, \dots, a_l^0), a^1 = (a_1^1, \dots, a_l^1)\}$ is the one used by the verifier to compute $\{y_j\}_{j \in [l]}$. This perfectly hides the challenge bit $b \in \{0, 1\}$ chosen by the verifier from the prover.

From the perfect correctness of the secure semi-honest 2PC protocol Π_F , a malicious prover cannot provide randomness and input for both parties that are consistent with τ since that would imply that Π_F computes an incorrect output for a false statement $x \notin L$, and violates the perfect correctness of Π_F . Therefore, the soundness error is no more than $1/2$.

Figure 2. Three-round ZK protocol $\Pi_{\mathcal{R}_L}$

Augmented black-box zero-knowledge. Let (S_0, S_1) be the simulation algorithms for the views of two parties in Π_F . Fix any statement $x \in L$, for any verifier V^* , the corresponding extended verifier is denoted by \widehat{V}^* (see subsection 2.3), and let $\mathcal{O}_{\widehat{V}^*}$ denote the oracle of the next message function of \widehat{V}^* . Then, for any auxiliary input z that either is an empty string or cannot be computed by any PPT algorithm, the augmented black-box simulator $S^{\mathcal{O}_{\widehat{V}^*}}(x, z, 1^n, 1^l)$ proceeds as follows:

1. On input 1^l , choose $i_1, \dots, i_l \in_R I$, and obtain the corresponding l pairs of claw-free permutations $(f^j = f_{i_j}, g^j = g_{i_j})$ and l pairs of trapdoors (TK_{f^j}, TK_{g^j}) for (f^j, g^j) respectively.
2. Choose random coins r_{V^*} , invoke $\mathcal{O}_{\widehat{V}^*}(x, z; \cdot; \cdot)$ with $(2, r_{V^*}, \{(f^j, g^j)\}_{j \in [l]})$ and obtain $\{y_j\}_{j \in [l]}$ and V^* 's current private state $\{b, \{(b_j, a_j, \Delta_j)\}_{j \in [l]}\}$.
I.e., $b_j = 0$ if $y_j = f^j(a_j)$ or y_j is computed from f^j and $b_j = 1$ if $y_j = g^j(a_j)$ or y_j is computed from g^j , otherwise $b_j = \perp$; if $\{b_j\}_{j \in [l]} \neq \{\perp\}$, $b \in \{0, 1\}$ is the majority of $\{b_j\}_{j \in [l]} - \{\perp\}$, otherwise $b = \perp$; if $b_j = b$ and y_j is computed from a_j with the function according to b , $\Delta_j = 1$; otherwise $\Delta_j = 0$ and $a_j = \perp$.
3. If $b = \perp$, reset b to be a random bit in $\{0, 1\}$. Compute $a_j^0 = (f^j)^{-1}(y_j)$ and $a_j^1 = (g^j)^{-1}(y_j)$ with trapdoors (TK_{f^j}, TK_{g^j}) respectively for $j \in [l]$. And set $a^0 = (a_1^0, \dots, a_l^0)$, $a^1 = (a_1^1, \dots, a_l^1)$.
4. Choose w_b at random, invoke the simulation algorithm S_b for the view of P_b with $(w_b, 1)$, and obtain $View_b^{\Pi_F} = \{w_b, r_b, m^1, \dots, m^t\}$.
5. Compute the whole transcript τ between P_0, P_1 by emulating P_b with $View_b^{\Pi_F}$.
Choose w_{1-b}, r_{1-b} at random and set $m_0 = w_0 || r_0, m_1 = w_1 || r_1$, where $|m_0| = |m_1| = k$ and $k = l/2$.
6. Choose $r_f^j, r_g^j \in_R \{0, 1\}^n$ for j in $[l]$, and set $r_f = (r_f^1, \dots, r_f^l), r_g = (r_g^1, \dots, r_g^l)$.
Choose two random hash functions H_{s_0}, H_{s_1} from \mathcal{H} and compute

$$c_0 = H_{s_0}(H_f(a^0, r_f)) \oplus m_0, \quad c_1 = H_{s_1}(H_g(a^1, r_g)) \oplus m_1,$$

where the functions f, g, H_f, H_g are defined as above.

7. Output $(x, r_{V^*}, \{(f^j, g^j)\}_{j \in [l]}, \tau, c_0, c_1, r_f, r_g, H_{s_0}, H_{s_1})$.

Now we show that the output of $S^{\mathcal{O}_{\widehat{V}^*}}(x, z, 1^n, 1^l)$ is computational indistinguishable with the real protocol view $View_{V^*(z)}^P(x)$ of V^* . Firstly, since it is hard to find a ‘‘claw’’ of (f^j, g^j) for every $j \in [l]$, the verifier V^* knows at most one of a_j^0, a_j^1 , then the inner product mod 2 of the other one and a random value can form a hard-core of the corresponding function. As a result, the verifier knows at most one of $H_f(a^0, r_f), H_g(a^1, r_g)$. According to the unpredictability of a hard-core predicate, if V^* does not know a_j^0 or a_j^1 for $j \in [l]$, the bit of $H_f(a^0, r_f), H_g(a^1, r_g)$ produced from a_j^0 or a_j^1 looks random. And then the verifier knows at most one of $v_0 = H_{s_0}(H_f(a^0, r_f)), v_1 = H_{s_1}(H_g(a^1, r_g))$, while the other one is indistinguishable from a random value for the verifier according to the property of the hash function.

The simulator and the prover differ in the way to choose w_0, w_1 and the way to produce the transcript τ of the 2PC protocol. However, no matter interacting with a

prover or a simulator, V^* can only know v_b while v_{1-b} is indistinguishable from a random value, where $b \in \{0, 1\}$ is actually chosen by V^* , then V^* can only obtain $m_b = v_b \oplus c_b$ while c_{1-b} is still indistinguishable from a random value. Besides, (τ, m_b) of the 2PC protocol Π_F produced by the prover and that produced by the simulator are indistinguishable while no PPT algorithm can obtain m_{1-b} from (τ, m_b) , which follows from the privacy of Π_F . Notice that if the private coins of computing each y_j is hiding from the verifier, it is similar to the above condition of unknown a_j^b ($b \in \{0, 1\}, j \in [l]$).

In a word, the indistinguishability of $S^{\mathcal{O}_{\tilde{V}^*}}(x, z, 1^n, 1^l)$ and $View_{V^*(z)}^P(x)$ follows from the privacy of the 2PC protocol Π_F and the properties of the hash functions, the hard-core functions and the claw-free permutations. \square

4.2 Construction with Negligible Soundness Error

We show our three-round ZK proof protocol constructed in Figure 2 can be made negligibly sound by any polynomial times parallel repetitions. See the proof in Appendix C.

Theorem 4.2 *Assuming the existences of claw-free permutations on $\{0, 1\}^n$, hash functions, and secure semi-honest two-party computation protocols, then $\Pi_{\mathcal{R}_L}$ described above is an augmented black-box zero-knowledge proof. Furthermore, the protocol obtained from $p(n)$ parallel repetitions of $\Pi_{\mathcal{R}_L}$ is an augmented black-box zero-knowledge proof with soundness error $2^{-p(n)}$.*

References

1. Barak, B.: How to go beyond the black-box simulation barrier. In FOCS, pages 106-115, 2001.
2. Blum, M.: How to prove a theorem so no one else can claim it. In Proceedings of the International Congress of Mathematicians, 1986, pages 1444-1451.
3. Bitansky, N., Brakerski, Z., Kalai, Y., Paneth, O., Vaikuntanathan, V.: 3-message zero knowledge against human ignorance. In: TCC 2016-B, pp. 57-83.
4. Bitansky, N., Canetti, R., Paneth, O., Rosen, A.: On the existence of extractable one-way functions. In: STOC 2014, pp. 505-514.
5. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1-18. Springer, Heidelberg (2001).
6. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In Proc. of 20th STOC, pages 1-10, 1988.
7. Bitansky, N., Kalai, Y.T., Paneth, O.: Multi-collision resistance: A paradigm for keyless hash functions. In: STOC 2018: 671-684.
8. Bellare, M., Palacio, A.: The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In: Franklin, M. (ed.) CRYPTO 2004, pp. 273-289.
9. Bitansky, N., Paneth, O.: Point obfuscation and 3-round zero knowledge. In: TCC 2012. pages 189-207.
10. Bitansky, N., Paneth, O.: On the impossibility of approximate obfuscation and application to resettable cryptography. In STOC 2013, pages 241-250.
11. Ciampi, M., Ostrovsky, R., Siniscalchi, L., Visconti, I.: Round-optimal secure two-party computation from trapdoor permutations. In: Kalai, Y., Reyzin, L. (eds.): TCC 2017, pp. 678-710.

12. Dwork, C., Naor, M., Reingold, O., Stockmeyer, L.: Magic functions. In *Memoriam: Bernard M. Dwork 1923-1998*. *Journal of the ACM*, 50(6):852-921, 2003.
13. Dodis, Y., Reyzin, L.: On the power of claw-free permutations. In: Cimato, S. et al. (eds.): *SCN 2002*, LNCS 2576, pp. 55-73, 2003.
14. Fleischhacker, N., Goyal, V., Jain, A.: On the existence of three round zero-knowledge proofs. In: Nielsen, J.B., Rijmen, V. (eds.) *EUROCRYPT 2018*, LNCS 10822, pp. 3-33, 2018.
15. Goldreich, O., Krawczyk, H.: On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169-192, 1996.
16. Ganesh, C., Kondi, Y., Patra, A., Sarkar, P.: Efficient adaptively secure zero-knowledge from garbled circuits. In: Abdalla, M., Dahab, R. (eds.): *PKC 2018*, LNCS 10770, pp. 499-529, 2018.
17. Goldreich, O., Levin, L.A.: Hard-core predicates for any one-way function. In *21st ACM Symposium on the Theory of Computing*, pages 25-32, 1989.
18. Garg, S., Mukherjee, P., Pandey, O., Polychroniadou, A.: The exact round complexity of secure computation. In: Fischlin, M., Coron, J-S. (eds) *EUROCRYPT 2016*, pages 448-476.
19. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186-208, 1989.
20. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A. (ed.) *19th ACM STOC*, pages 218-229. ACM Press, May 1987.
21. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *J. ACM*, 38(3):691-729, 1991. (FOCS 1986, pages 174-187)
22. Goldreich, O., Oren, Y.: Definitions and properties of zero-knowledge proof systems. *J. Cryptology*, 7(1):1-32, 1994.
23. Hada, S., Tanaka, T.: On the existence of 3-round zero-knowledge protocols. In *CRYPTO 1998*, pages 408-423.
24. Kalai, Y.T., Rothblum, G.N., Rothblum, R.D.: From obfuscation to the security of Fiat-Shamir for proofs. In: Katz, J., Shacham, H. (eds.) *CRYPTO 2017, Part II*. LNCS, vol. 10402, pp. 224-251. Springer, Cham (2017).
25. Lepinski, M.: On the Existence of 3-Round Zero-Knowledge Proofs. Ph.D. thesis, Massachusetts Institute of Technology (2002)
26. Hongda, L., Dongxue, P., Peifang, N.: Augmented black-box simulation and zero knowledge argument for NP. <https://eprint.iacr.org/eprint-bin/search.pl>
27. Hazay, C., Venkatasubramanian, M.: On the power of secure two-party computation. In: Robshaw, M., Katz, J. (eds.) *CRYPTO 2016*, pages 397-429.
28. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Johnson, D.S., Feige, U. (eds.) *39th ACM STOC*, pp. 21-30. ACM Press, June 2007
29. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121-1152, 2009.
30. Jawurek, M., Kerschbaum, F., Orlandi, C.: Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In: *CCS 2013*, pp. 955-966 (2013)
31. Katz, J., Ostrovsky, R.: Round-optimal secure two-party computation. In *CRYPTO 2004*, pages 335-354.
32. Ostrovsky, R., Richelson, S., Scafuro, A.: Round-optimal black-box two-party computation. In: Gennaro, R., Robshaw, M. (eds) *CRYPTO 2015*, pages 339-358.
33. Yao, A. C-C.: How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162-167. IEEE Computer Society Press, October 1986.
34. Yao, A. C-C.: Protocols for secure computations (extended abstract). In *FOCS*, pages 160-164. IEEE Computer Society, 1982.

A Definitions.

We show the definitions used in our constructions.

A.1 Hard-Core Functions

Definition A.1 Hard-Core Predicate. A polynomial-time-computable predicate function $b : \{0, 1\}^* \rightarrow \{0, 1\}$ is called a hard-core of a function f if for every probabilistic polynomial-time algorithm A , every positive polynomial $p(\cdot)$, and all sufficiently large n :

$$\Pr[A(F(U_n) = B(U_n))] < 1/2 + 1/p(n),$$

where U_n is uniformly distributed over $\{0, 1\}^n$. We call it as the unpredictability of a hard-core predicate.

Theorem A.1 Goldreich-Levin Hard-Core Predicate. Let f be an arbitrary one-way function, and let g be defined by $g(x, r) = (f(x), r)$, where $|x| = |r|$. Let $b(x, r) = \langle x, r \rangle$ denote the inner product mod 2 of the binary vectors x and r . Then the predicate b is a hard-core of the function g [17].

Definition A.2 Hard-Core Function. Let $h : \{0, 1\}^* \rightarrow \{0, 1\}^*$ be a polynomial-time-computable function satisfying $|h(x)| = |h(y)|$ for all $|x| = |y|$, and let $l(n) = |h(1^n)|$. The function h is called a hard-core of a function f if for every PPT algorithm D , every positive polynomial $p(\cdot)$, and all sufficiently large n :

$$|\Pr[D((F(X_n), h(X_n)) = 1)] - \Pr[D((F(X_n), R_{l(n)}) = 1)]| < 1/p(n),$$

where X_n and $R_{l(n)}$ are two independent random variables, the first uniformly distributed over $\{0, 1\}^n$ and the second uniformly distributed over $\{0, 1\}^{l(n)}$.

A.2 Commitment Schemes

Definition A.3 Commitment Schemes. A commitment scheme $(Com, Open)$ executed between two parties (a committer and a receiver) consists of two phases, the commit phase and the open phase.

- **Commit Phase.** The committer takes as input a message $m \in \{0, 1\}^*$ and chooses a random $r \in \{0, 1\}^*$. The committer then computes $c := Com_r(m)$ and sends it to the receiver.
- **Open Phase.** The committer reveals the message m committed and the randomness r used in the commit phase. And the receiver verifies $Open(c, m, r) = m$.

A commitment scheme $(Com, Open)$ is secure if it is binding and hiding. A **statistically binding commitment** scheme requires the following properties.

- **Statistically Binding.** For any (possibly unbounded) committer, he can reveal a commitment of m to another message $m' \neq m$ with negligible probability.
- **Computationally Hiding.** For every $m_0, m_1 \in \{0, 1\}^*$, $Com(m_0)$ and $Com(m_1)$ are computationally indistinguishable.

A.3 Two-Party Computation

Definition A.4 Semi-honest Two-Party Computation. A secure semi-honest two-party computation protocol $\Pi_f(P_1, P_2)$ is a protocol (executed between P_1 and P_2) securely realizing an two-party functionality $f : \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}^* \times \{0, 1\}^*$. Concretely, P_1 holding input w_1 and P_2 holding input w_2 ($|w_1| = |w_2|$), both wish to obtain the output of $f(w_1, w_2)$ by executing the protocol $\Pi_f(P_1, P_2)$ with the following property.

- **Privacy.** The view of P_1 (resp., P_2) during an execution of $\Pi_f(P_1, P_2)$ on (w_1, w_2) , $View_1^\Pi(w_1, w_2)$ (resp., $View_2^\Pi(w_1, w_2)$), consists of $(w_1, r, m_1, \dots, m_t)$ (resp., $(w_2, r, m_1, \dots, m_t)$), where r represents the outcome of P_1 's (resp., P_2 's) internal coin tosses, and m_i represents the i -th message it has received. For a deterministic functionality f , we say that $\Pi_f(P_1, P_2)$ privately computes f if there exist PPT algorithms, denoted S_1 and S_2 , such that

$$\{S_1(w_1, f(w_1, w_2))\}_{w_1, w_2 \in \{0, 1\}^*} \stackrel{c}{=} \{View_1^\Pi(w_1, w_2)\}_{w_1, w_2 \in \{0, 1\}^*},$$

$$\{S_2(w_2, f(w_1, w_2))\}_{w_1, w_2 \in \{0, 1\}^*} \stackrel{c}{=} \{View_2^\Pi(w_1, w_2)\}_{w_1, w_2 \in \{0, 1\}^*},$$

where $|w_1| = |w_2|$.

B Proof of Theorem 3.2.

Theorem 3.2. Assuming the existences of pseudorandom generators, a family C of claw-free permutations on $\{0, 1\}^n$, statistically binding non-interactive commitment schemes, and hard-core functions for all functions in C , then construction of Π_{G3C} in Figure 1 is an augmented black-box zero-knowledge proof. Furthermore, the protocol obtained from $p(n) = n \cdot 2^k$ (2^k is the number of edges in the graph) parallel repetitions of $\Pi_{\mathcal{R}_L}$ is an augmented black-box zero-knowledge proof with soundness error $2^{-\Omega(n)}$.

Proof. The completeness follows from the completeness of each execution of the basic three-round protocol Π_{G3C} . The verifier accepts the proof if and only if each execution of the basic three-round protocol Π_{G3C} is accepted. Then the soundness error is $(1 - 2^{-k})^{n \cdot 2^k}$.

Augmented black-box zero-knowledge. Fix any statement $G \in G3C$, for any PPT verifier V^* , the corresponding extended verifier is denoted by \widehat{V}^* (see subsection 2.3), and let $\mathcal{O}_{\widehat{V}^*}$ denote the oracle of the next message function of \widehat{V}^* . Then, for any auxiliary input z that either is an empty string or cannot be computed by any PPT algorithm, the augmented black-box simulator $S^{\mathcal{O}_{\widehat{V}^*}}(G, z, 1^n)$ proceeds as follows:

1. On input $G = (V, E)$ where $V = \{1, \dots, n\}$, $|E| = 2^k$, for $d \in [p(n)]$, sample $i_{d,1}, \dots, i_{d,k} \in_R I$ and obtain the corresponding $p(n) \cdot k$ pairs of claw-free permutations $(f^{d,j} = f_{i_{d,j}}, g^{d,j} = g_{i_{d,j}})$ and k pairs of trapdoors $(TK_{f^{d,j}}, TK_{g^{d,j}})$ for $(f^{d,j}, g^{d,j})$ respectively.

2. Choose random coins r_{V^*} , invoke $\mathcal{O}_{\widehat{V^*}}(G, z; \cdot; \cdot)$ with

$$(2, r_{V^*}, \{(f^{d,j}, g^{d,j})\}_{d \in [p(n)], j \in [k]})$$

and obtain $\{y_{d,j}\}_{d \in [p(n)], j \in [k]}$ and V^* 's current private state

$$\{(b_{d,j}, a_{d,j}, \Delta_{d,j})\}_{j \in [k], d \in [p(n)]}.$$

I.e., if $\Delta_{d,j} = 1$, $y_{d,j} = f^{d,j}(a_{d,j})$ when $b_{d,j} = 0$ and $y_{d,j} = g^{d,j}(a_{d,j})$ when $b_{d,j} = 1$; if $\Delta_{d,j} = 0$, $a_{d,j} = \perp$, $b_{d,j} = 0$ if $y_{d,j}$ is computed from $f^{d,j}$ and $b_{d,j} = 1$ if $y_{d,j}$ is computed from $g^{d,j}$, otherwise $b_{d,j} = \perp$.

3. For $d \in [p(n)]$:

if $\perp \notin \{b_{d,j}\}_{j \in [k]}$,

- compute $h_{d,j} = h(a_{d,j})$ for $j \in [k]$, and then compute

$$\beta_{b_{d,1}, \dots, b_{d,k}} = \bigoplus_{j \in [k]} h_{d,j};$$

- let $u, v \in V$ denote the two vertices corresponding to the $(b_{d,1}, \dots, b_{d,k})$ -th edge in E , select $col_{u,d} \neq col_{v,d} \in \{1, 2, 3\}$, and set $col_{s,d} = 1$ for $s \in V$ and $s \neq u, v$;

- compute $c_{s,d} = Com_{r_{s,d}}(col_{s,d})$ with randomness $r_{s,d}$ for $s \in V$;

- compute $e_{b_{d,1}, \dots, b_{d,k}} = \mathcal{G}(\beta_{b_{d,1}, \dots, b_{d,k}}) \oplus (col_{u,d}, r_{u,d}, col_{v,d}, r_{v,d})$ and select random $e_{\alpha,d} \in \{0, 1\}^{l'(n)}$ ($l'(n) = 2(|r_{s,d}| + |col_{s,d}|)$) for $\alpha \in \{0, 1\}^k$ and $\alpha \neq (b_{d,1}, \dots, b_{d,k})$.

If $\perp \in \{b_{d,j}\}_{j \in [k]}$: set $col_{s,d} = 1$ for $s \in V$, compute $c_{s,d} = Com_{r_{s,d}}(col_{s,d})$ with randomness $r_{s,d}$ for $s \in V$, and select random $e_{\alpha \in d} \in \{0, 1\}^{l'(n)}$ ($l'(n) = 2(|r_{s,d}| + |col_{s,d}|)$) for $\alpha \in \{0, 1\}^k$.

4. Output

$$(G, r_{V^*}, \{(f^{d,j}, g^{d,j})\}_{d \in [p(n)], j \in [k]}, \{c_{s,d}\}_{s \in V, d \in [p(n)]}, \{e_{\alpha,d}\}_{\alpha \in \{0,1\}^k, d \in [p(n)]}).$$

The proof of the indistinguishability between $S^{\mathcal{O}_{\widehat{V^*}}}(G, z, 1^n)$ and $View_{V^*(z)}^P(G)$ follows from the proof of Theorem 3.1, which implies that the indistinguishability is obtained from the hiding property of the commitment scheme Com and the properties of the claw-free permutations, the hard-core function h and the pseudorandom generator \mathcal{G} . \square

C Proof of Theorem 4.2.

Theorem 4.2. Assuming the existences of claw-free permutations on $\{0, 1\}^n$, hash functions, and secure semi-honest two-party computation protocols, then $\Pi_{\mathcal{R}_L}$ described above is an augmented black-box zero-knowledge proof. Furthermore, the protocol obtained from $p(n)$ parallel repetitions of $\Pi_{\mathcal{R}_L}$ is an augmented black-box zero-knowledge proof with soundness error $2^{-p(n)}$.

Proof. The completeness follows from the completeness of each execution of the basic three-round protocol $\Pi_{\mathcal{R}_L}$. The verifier accepts the proof if and only if each execution of the basic three-round protocol $\Pi_{\mathcal{R}_L}$ is accepted. Then the soundness error is $2^{-p(n)}$.

Augmented black-box zero-knowledge. Let (S_0, S_1) be the simulation algorithms for the views of two parties in Π_F . Fix any statement $x \in L$, for any PPT verifier V^* , the corresponding extended verifier is denoted by \widehat{V}^* (see subsection 2.3), and let $\mathcal{O}_{\widehat{V}^*}$ denote the oracle of the next message function of \widehat{V}^* . Then, for any auxiliary input z that either is an empty string or cannot be computed by any PPT algorithm, the augmented black-box simulator $S^{\mathcal{O}_{\widehat{V}^*}}(x, z, 1^n, 1^l)$ proceeds as follows:

1. On input 1^l , for $d \in [p(n)]$, choose $i_1^d, \dots, i_l^d \in_R I$, and obtain the corresponding $p(n) \cdot l$ pairs of claw-free permutations $(f^{d,j} = f_{i_j^d}, g^{d,j} = g_{i_j^d})$ and $p(n) \cdot l$ pairs of trapdoors $(TK_{f^{d,j}}, TK_{g^{d,j}})$ for $(f^{d,j}, g^{d,j})$ respectively.
2. Choose random coins r_{V^*} , invoke $\mathcal{O}_{\widehat{V}^*}(x, z; \cdot)$ with

$$(2, r_{V^*}, \{(f^{d,j}, g^{d,j})\}_{d \in [p(n)], j \in [l]})$$

and obtain $\{y_{d,j}\}_{d \in [p(n)], j \in [l]}$ and V^* 's current private state

$$\{b_d, \{(b_{d,j}, a_{d,j}, \Delta_{d,j})\}_{j \in [l]}\}_{d \in [p(n)]}.$$

I.e., $b_{d,j} = 0$ if $y_{d,j} = f^{d,j}(a_{d,j})$ or $y_{d,j}$ is computed from $f^{d,j}$ and $b_{d,j} = 1$ if $y_{d,j} = g^{d,j}(a_{d,j})$ or $y_{d,j}$ is computed from $g^{d,j}$, otherwise $b_{d,j} = \perp$; for $d \in [p(n)]$, if $\{b_{d,j}\}_{j \in [l]} \neq \{\perp\}$, $b_d \in \{0, 1\}$ is the majority of $\{b_{d,j}\}_{j \in [l]} - \{\perp\}$, otherwise $b_d = \perp$; if $b_{d,j} = b_d$ and $y_{d,j}$ is computed from $a_{d,j}$ with the function according to b_d , $\Delta_{d,j} = 1$; otherwise $\Delta_{d,j} = 0$ and $a_{d,j} = \perp$.

3. For $d \in [p(n)]$, if $b_d = \perp$, reset b_d to be a random bit in $\{0, 1\}$. And then compute $a_{d,j}^0 = (f^{d,j})^{-1}(y_{d,j})$ and $a_{d,j}^1 = (g^{d,j})^{-1}(y_{d,j})$ with trapdoors $(TK_{f^{d,j}}, TK_{g^{d,j}})$ respectively for $d \in [p(n)]$, $j \in [l]$. Set $a_d^0 = (a_{d,1}^0, \dots, a_{d,l}^0)$, $a_d^1 = (a_{d,1}^1, \dots, a_{d,l}^1)$.
4. For $d \in [p(n)]$, choose $w_{b_d}^d$ at random, invoke the simulation algorithm S_{b_d} for the view of P_{b_d} with $(w_{b_d}^d, 1)$, and obtain $View_{b_d}^{\Pi_F, d} = \{w_{b_d}^d, r_{b_d}^d, m^{d,1}, \dots, m^{d,t}\}$.
5. Compute the whole transcript τ_d between P_0, P_1 by emulating P_{b_d} with $View_{b_d}^{\Pi_F, d}$. Choose $w_{1-b_d}^d, r_{1-b_d}^d$ at random and set $m_0^d = w_0^d || r_0^d, m_1^d = w_1^d || r_1^d$, where $|m_0^d| = |m_1^d| = k$ and $k = l/2$.
6. Choose $r_f^{d,j}, r_g^{d,j} \in_R \{0, 1\}^n$ for $d \in [p(n)]$, j in $[l]$, and set

$$r_f^d = (r_f^{d,1}, \dots, r_f^{d,l}), \quad r_g^d = (r_g^{d,1}, \dots, r_g^{d,l}).$$

For $d \in [p(n)]$, choose two random hash functions $H_{s_{d,0}}^d, H_{s_{d,1}}^d$, compute

$$c_0^d = H_{s_{d,0}}^d(H_{f^d}(a_d^0, r_f^d)) \oplus m_0^d, \quad c_1^d = H_{s_{d,1}}^d(H_{g^d}(a_d^1, r_g^d)) \oplus m_1^d$$

The functions $f^d, g^d, H_{f^d}, H_{g^d}$ are defined in a similar way as f, g, H_f, H_g defined above.

7. Output

$$(x, r_{V^*}, \{(f^{d,j}, g^{d,j})\}_{d \in [p(n)], j \in [l]}, \{\tau_d, c_0^d, c_1^d, r_f^d, r_g^d, H_{s_{d,0}}^d, H_{s_{d,1}}^d\}_{d \in [p(n)]}).$$

The proof of the indistinguishability between $S^{\mathcal{O}_{V^*}}(x, z, 1^n, 1^l)$ and $View_{V^*}^P(z)(x)$ follows from the proof of Theorem 4.1, which implies that the indistinguishability is obtained from the privacy of the 2PC protocol Π_F and the properties of the hash functions, the hard-core functions and the claw-free permutations.