Zero-Knowledge Protocols for Search Problems

Ben Berger Zvika Brakerski
Weizmann Institute of Science, Israel[*]

Abstract

We consider natural ways to extend the notion of Zero-Knowledge (ZK) Proofs beyond decision problems. Specifically, we consider *search problems*, and define zero-knowledge proofs in this context as interactive protocols in which the prover can establish the correctness of a solution to a given instance without the verifier learning anything beyond the intended solution, even if it deviates from the protocol.

The goal of this work is to initiate a study of Search Zero-Knowledge (search-ZK), the class of search problems for which such systems exist. This class trivially contains search problems where the validity of a solution can be efficiently verified (using a single message proof containing only the solution). A slightly less obvious, but still straightforward, way to obtain zero-knowledge proofs for search problems is to let the prover send a solution and prove in zero-knowledge that the instance-solution pair is valid. However, there may be other ways to obtain such zero-knowledge proofs, and they may be more advantageous.

In fact, we prove that there are search problems for which the aforementioned approach fails, but still search zero-knowledge protocols exist. On the other hand, we show sufficient conditions for search problems under which some form of zero-knowledge can be obtained using the straightforward way.

1 Introduction

The notion of Zero-Knowledge Proofs (ZK-Proofs) introduced by Goldwasser, Micali and Rackoff [20] is one of the most insightful and influential in the theory of computing. Its tremendous impact came not only from having numerous applications (e.g. [12]) but maybe more importantly from challenging the way we think about proofs and the communication between the prover and the verifier. In a nutshell, a ZK-Proof is an interactive proof of some statement, i.e. an interaction between a prover P and a verifier V with the prover attempting to convince the verifier that some instance x belongs to a language L. In addition to the

[*]{ben.berger,zvika.brakerski}@weizmann.ac.il. Supported by the Israel Science Foundation (Grant No. 468/14), Binational Science Foundation (Grants No. 2016726, 2014276), and by the European Union Horizon 2020 Research and Innovation Program via ERC Project REACT (Grant 756482) and via Project PROMETHEUS (Grant 780701).

usual completeness and soundness, in the ZK scenario the prover wants to protect itself from revealing "too much information" to the verifier. Surely the verifier needs to learn that indeed $x \in L$, but nothing else beyond this fact should be revealed. Furthermore, even a *malicious* verifier that does not follow the prescribed protocol should not be able to trick the prover into revealing more information than intended. This intuitive statement is formalized using the *simulation paradigm*, the existence of a simulator machine S that takes an input $x \in L$ and a possibly cheating verifier V^* and samples from the view of V^* in the interaction (P, V^*) (up to negligible statistical or computational distance). Since the view of the verifier can essentially be produced (up to negligible distance) knowing only that $x \in L$, it clearly does not reveal anything beyond this fact.

Our Results. In this work we consider a setting where again the prover is concerned about revealing too much information to the verifier, but now in the context of *search problems*. That is, the prover would like to assist the verifier in learning a solution y to an instance x of some search problem, but would like to limit the verifier's ability to learn anything beyond the intended solution (or distribution of solutions).

While one's first intuition of a search problem is of one where it is efficient to verify a solution (i.e. searching for an NP witness), this is actually not the interesting setting here. In fact, in this case the prover can just send the witness, and the verifier verifies locally, so no additional information beyond the solution is revealed. One example one could consider is the "isomorphic vertex problem": given two graphs (G_1, G_2) and a vertex v_1 in G_1, find a vertex v_2 in G_2 that corresponds to v_1 under *some* isomorphism.

Our first contribution is to formalize the notion of zero knowledge interactive protocols for search problems using the simulation paradigm, as follows. We require that the prover for the interactive protocol is associated with a family of distributions $\{Y_x\}_x$ over solutions for each input x, intuitively corresponding to the distribution V is allowed to learn. We require that the view of any verifier can be simulated given only a sample y drawn from Y_x. To reduce the number of free parameters in the definition we propose to associate Y_x with the distribution of solutions output by an interaction of an honest prover with an honest verifier (note that importantly this refers to the distribution of solutions y output by the honest verifier and not to the honest verifier's entire view). Thus the zero-knowledge task becomes to ensure that no verifier (including the honest verifier) learns anything except the honest verifier's prescribed output. In terms of soundness, we require that V either outputs some valid solution for the search problem (if such exists), or rejects, except perhaps with small probability, even when interacting with a malicious prover. The definition, a discussion and an example protocol are provided in Section 3.

Intuitively one could think that in order to achieve search-ZK, the prover should first sample a solution from Y_x, send it to the verifier and then prove in decision-ZK the validity of the solution (that is, that in a sense search-ZK is reducible to decision-ZK). In section 4 we formally define the class prefix-ZK of search problems that have search-ZK protocols of the form described above. We show that prefix-ZK has a complete problem (which we are unable to show for general search-ZK) and we explore whether it is possible to provide a prefix-ZK protocol for any language in search-ZK or whether there are some cases where other methods can achieve search-ZK but the aforementioned outline cannot. We show conditions under which a given search-ZK protocol can be transformed into a prefix-ZK one (for the same underlying search problem), but we also show that in general, perhaps counter-intuitively, search-ZK contains problems that are not in prefix-ZK, so at least in that sense the study of search-ZK may not be a derivative of the study of decision-ZK. Interestingly, this separation follows from showing that search-PSPACE, the class of search problems solvable by a deterministic poly-space Turing machine, *does not contain* search-IP, the class of search problems solvable by an interactive protocol. This result may be of independent interest, in light of the famous IP=PSPACE theorem [25].

Lastly, in Section 5, we discuss search-ZK protocols that are also *pseudo-deterministic*, meaning that each instance is associated with some canonical solution that is output by the prescribed protocol and the soundness requirement is that a malicious prover cannot make the honest verifier output a solution that is not the canonical one (except with small probability). The notion of pseudo-deterministic algorithms and protocols was presented by Gat and Goldwasser [7] and further explored by Goldreich, Goldwasser, Grossman, Holden and Ron [10, 17–19, 21]. One of the advantages of pseudo-deterministic protocols is that they allow for soundness amplification for search problems. We show that the isomorphic vertex problem has a pseudo-deterministic search-ZK protocol, suggesting that achieving strong soundness together with strong privacy is possible in some interesting cases.

Related Notions. The first related notion is that of secure multiparty computation (MPC) by Yao [26] and Goldreich, Micali and Wigderson [11]. For the purpose of this work, the relevant setting is of secure *two-party* computation where two parties A, B with inputs x_A, x_B wish to compute values y_A, y_B which depend on both inputs. The privacy requirement is that each party does not learn more than its intended output. It would appear that setting $A = P$, $B = V$, and defining F_B appropriately to output what the verifier is allowed to learn, should result in a search-ZK protocol. However, looking more closely, the complexity of an MPC protocol scales with the complexity of the function F_B, which in general scales with the complexity of the prover's functionality. If the prover's functionality is not in NP, then MPC cannot be used. MPC appears to be useful in the restricted case of computational search-ZK for search problems that can be computed as a function of an NP witness. Our isomorphic vertex problem falls into that category (with the NP witness being an isomorphism), however for isomorphic vertex we have a *statistical* search-ZK protocol. For statistical search-ZK, the MPC methodology does not seem to be useful, since information theoretically secure *two-party* computation is not possible [3, 4].

Another related like of work is concerned with privacy of approximation algorithms, initiated by Feigenbaum et al. [6] and Halevi et al. [22], and further studied by Beimel et al. [1]. The setting in these works is quite different from ours. Their ideal setting is where a solution to some search problem is posted without revealing the input (e.g. output a vertex cover for some graph without revealing the edges of the graphs). The problem arises when solving exactly is hard and an approximation algorithm is used instead. Their goal is to show that the approximate solution does not reveal more information than the exact solution. Note that in this setting there is no soundness requirement (in fact, a client cannot be convinced that a solution is correct since it does not have the input).

Future Directions. Our work is far from being an exhaustive study of search-ZK, and we hope to open the door for additional study. One direction of research is designing search-ZK protocols for other problems of interest, and more importantly general approaches for search-ZK for classes of problems. The question of whether search-ZK has complete problems in the computational and statistical setting remains open. Another intriguing line of inquiry, which may also be helpful for resolving the above, is whether we can translate the extensive body of work on statistical ZK protocols [5, 14–16, 23, 24] into the search regime.

2 Preliminaries

In this section we go over standard definitions.

Definition 1 (Relations and Promise Problems)**.**

- A *relation* is a subset $R \subseteq \{0,1\}^* \times \{0,1\}^*$. Given a relation R we denote by $L_R := \{x \mid \exists y,\ (x,y) \in R\}$ the set of *instances* of R. Given an instance x, we denote $R(x) = \{y \mid (x,y) \in R\}$.
- A *promise decision problem* is a pair $\mathcal{L} = (L_{YES}, L_{NO})$ of sets $L_{YES}, L_{NO} \subseteq \{0,1\}^*$ where $L_{YES} \cap L_{NO} = \emptyset$. An element $x \in L_{YES}$ is called a *YES instance* of $\mathcal{L}$ and an element $x \in L_{NO}$ is called a *NO instance* of $\mathcal{L}$. The goal of a solving procedure:
  - If $x \in L_{YES}$ output 1.
  - If $x \in L_{NO}$ output 0.
- A *promise search problem* is a pair $\mathcal{R} = (R_{YES}, L_{NO})$ where $R_{YES} \subseteq \{0,1\}^* \times \{0,1\}^*$ is a relation, $L_{NO} \subseteq \{0,1\}^*$ is some set and $L_{R_{YES}} \cap L_{NO} = \emptyset$. An element $x \in L_{R_{YES}}$ is called a *YES instance* of $\mathcal{R}$ and an element $x \in L_{NO}$ is called a *NO instance* of $\mathcal{R}$. The goal of a solving procedure:
  - If $x \in L_{R_{YES}}$ output some $y \in R_{YES}(x)$.
  - If $x \in L_{NO}$ output $\perp$.

Remark 2. We stress again that in a search problem, an instance can have many legal solutions, and the solving procedure is required to output one of them.

Definition 3 (Karp and Levin Reductions for Promise Problems)**.**

- Let $\mathcal{L}^1 = (L^1_{YES}, L^1_{NO})$, $\mathcal{L}^2 = (L^2_{YES}, L^2_{NO})$ be two promise decision problems. We say that $\mathcal{L}^1$ is *Karp-reducible* to $\mathcal{L}^2$ if there is a deterministic polynomial time Turing machine M such that:

  - $\forall x \in L^1_{YES} \Rightarrow M\left(x\right) \in L^2_{YES}$.
  - $\forall x \in L^1_{NO} \Rightarrow M\left(x\right) \in L^2_{NO}$.

- Let $\mathcal{R}^1 = \left(R^1_{YES}, L^1_{NO}\right), \mathcal{R}^2 = \left(R^2_{YES}, L^2_{NO}\right)$ be two promise search problems. We say that $\mathcal{R}^1$ is *Levin-reducible* to $\mathcal{R}^2$ if there are two deterministic polynomial time Turing machines A, B such that:

  - $\forall x \in L_{R^1_{YES}} \Rightarrow A\left(x\right) \in L_{R^2_{YES}}$.
  - $\forall x \in L^1_{NO} \Rightarrow A\left(x\right) \in L^2_{NO}$.
  - $\forall x \in L_{R^1_{YES}},\ z \in R^2_{YES}\left(A\left(x\right)\right) \Rightarrow B\left(x, z\right) \in R^1_{YES}\left(x\right)$.

  The pair (A, B) is called the Levin-reduction between $\mathcal{R}^1$ and $\mathcal{R}^2$.

- We say that a Levin reducion (A, B) from $\mathcal{R}^1$ to $\mathcal{R}^2$ is *solution preserving*, if for any $x \in L_{R^1_{YES}}$, $R^1_{YES}\left(x\right) = R^2_{YES}\left(A\left(x\right)\right)$, i.e. x and $A\left(x\right)$ have the same set of solutions.

2.1 Interactive Protocols

We will omit the rigorous formal definition of an interactive protocol between two parties, which can be found in standard text books (e.g. [8]), but will just go over conventional notation. In these protocols we will have two interacting Turing machines, usually denoted P and V, which perform a joint computation on a shared input, usually denoted x. The following notation will be used:

- $\left(P, V\right)\left(x\right)$. This can have two meanings - depending on the context:

  - It can be written in a description of an implementation of some interactive protocol, in which case it means that the interacting parties execute a sub-protocol on joint input x according to the strategies P and V.
  - In any other context, this denotes the **output** of V after its interaction with P on the shared input x. That is, if the underlying problem is a decision problem then this output is a bit, and if the underlying problem is a search problem then this is the purported solution to the instance. Note that if V and/or P are randomized machines, and if furthermore x is taken from some distribution, then $\left(P, V\right)\left(x\right)$ is a random variable, which is essentially a function of all the randomness involved.

- $\mathsf{view}^{(P,V)}_V\left(x\right)$ denotes the view of V of the interaction with P on input x. More precisely, this is a concatenation of the input, the randomness of V and all the messages passed between the machines in the interaction. For example, if there are k messages that are passed in the interaction, and m_i denotes the i'th message, then

$$\mathsf{view}^{(P,V)}_V\left(x\right) = \left(x, r_V, m_1, \ldots, m_k\right)$$

  where r_V is the randomness of V.

Interactive proof systems for decision problems were introduced in [20]. In these protocols, an efficient verifier interacts with a computationally unbounded prover that tries to convince him of the validity of some statement. Altough these were originally defined for languages, they can also be defined for promise problems:

Definition 4 (The class IP)**.** We say that the promise decision problem $\mathcal{L} = (L_{YES}, L_{NO}) \in IP$ if there is an interactive protocol (P, V) where V is a PPTM and P is (possibly) computationally unbounded such that:

- Completeness: For every $x \in L_{YES}$, $\Pr\left[(P, V)(x) = 1\right] = 1$.
- Soundness: For every $x \in L_{NO}$ and prover strategy P^*, $\Pr\left[(P^*, V)(x) = 1\right] \leq \frac{1}{2}$.

We will now define zero-knowledge protocols for decision problems, also introduced in [20].

Notational Convention In this work three types of zero knowledge classes are considered: computational, statistical and perfect zero knowledge, denoted CZK, SZK and PZK respectively. Most of the claims in this text apply to all three of these classes, and therefore we use the abbreviation ZK whenever the statement or definition it appears in refers to all three types of zero knowledge at the same time. For example, the statement "any ZK protocol p_1 admits a ZK protocol p_2" expands to three different statements for each of the zero-knowledge types considered.

Definition 5. We say a promise decision problem $\mathcal{L} = (L_{YES}, L_{NO}) \in ZK$ if there is a triplet (P, V, S) such that:

- (P, V) is an IP protocol for $\mathcal{L}$.
- Zero knowledge: S is an expected PPTM and for any PPTM V^* and $x \in L_{YES}$,

$$\mathsf{view}_{V^*}^{(P,V^*)}(x) \approx S(x, V^*).$$

Remark 6. The meaning of '$\mathsf{view}_{V^*}^{(P,V^*)}(x) \approx S(x, V^*)$' depends on the type of zero knowledge considered, as explained below:

- CZK: For any PPTM D,

$$\left|\Pr\left[D\left(\mathsf{view}_{V^*}^{(P,V^*)}(x)\right) = 1\right] - \Pr\left[D\left(S(x, V^*)\right) = 1\right]\right| = negl(|x|).$$

- SZK:

$$\left\|\mathsf{view}_{V^*}^{(P,V^*)}(x) - S(x, V^*)\right\| = negl(|x|)$$

  where $\|\cdot\|$ denotes statistical distance.

- PZK:

$$\left\|\mathsf{view}_{V^*}^{(P,V^*)}(x) - S(x, V^*)\right\| = 0$$

Remark 7. In the original definition of zero-knowledge proofs, given in [20], the zero knowledge property required that for any V^* there exists a (possibly different) S^* that simulates the view of V^* in the original protocol. Here we chose to adopt the notion of a *universal simulator*: a single algorithm that simulates the view of V^* when given as input a description of V^*.

Remark 8. A stronger definition of zero-knowledge proofs was given in [13] where the verifier is required to learn nothing from the protocol even when it has access to some external auxilliary input. This stronger notion is particularly important when one wants to use a zero-knowledge protocol as a subprotocol of another zero-knowledge protocol. Since the results of this work do not require this stronger property we chose to work with the simpler definition in order to facilitate the presentation.

Remark 9. Note that $PZK \subseteq SZK \subseteq CZK$.

3 Zero Knowledge Protocols for Search Problems

We start by defining the search counterpart of IP, which we call $Search-IP$. Similarly to IP protocols, $Search-IP$ protocols also feature an efficient verifier that interacts with a computationally unbounded prover, but unlike the decisional case where the verifier outputs a bit that represents his "decision", here the verifier outputs a solution to the instance of the search problem. The titles "verifier" and "prover" are slightly misleading since here, as opposed to the decisional case, there is no statement that needs to be proved and verified but rather an interactive computation in which the two parties come up with some solution together (in a way that the verifier can not be fooled to output a solution that is not correct but with small probability). Still we choose to refer to the two parties by these names as their roles closely resemble the roles of the verifier and prover in IP protocols.

Definition 10. We say that the promise search problem $\mathcal{R} = (R_{YES}, L_{NO}) \in Search-IP$ if there is a pair (P, V) where V is a PPTM and P is (possibly) computationally unbounded such that:

- "Completeness": For any $x \in L_{R_{YES}}$, $\Pr\left[(P,V)(x) \in R_{YES}(x)\right] = 1$.
- "Soundness":
  - For any $x \in L_{NO}$ and any P^*, $\Pr\left[(P^*,V)(x) = \perp\right] \geq \frac{1}{2}$.
  - For any $x \in L_{R_{YES}}$ and any P^*, $\Pr\left[(P^*,V)(x) \in R_{YES}(x) \cup \{\perp\}\right] \geq \frac{1}{2}$.

A few comments about this definition are due. Note that we require zero probability of error when the two honest parties interact. That is, we require perfect completeness, a requirement that we also have in the decisional definition. This choice complies with the defintion of interactive proofs as formulated in [9] and is not very significant: we could have also allowed small completeness error and most of the results of this thesis would still hold. Furthermore, note that the soundness condition refers also to the case where $x \in L_{R_{YES}}$: even if the instance has a solution, no prover strategy can make the verifier output a wrong solution but with small probability. Another thing to note is that it is not clear how one can reduce the soundness error. Repetition of the protocol can yield different solutions, and it is not clear why one of these solutions should be favored more than another. Later in this work we consider different ways to deal with this problem. On the other hand, note that a $search - IP$ protocol immediately gives rise to an IP protocol for the task of deciding whether the given instance has a solution or not: the parties just need to run the original protocol, in the end of which the verifier accepts if and only if the output is not '$\perp$'. This protocol can be repeated (sequentially or in parallel) to reduce the soundness error.

We now turn to present zero knowledge protocols for search problems. These are $Search - IP$ protocols that apart from simply solving the search problem they also have a 'zero knowledge' property: it is guaranteed that the verifier does not learn anything other than the obtained solution, in the sense that given a solution, he could have simulated the entire interaction with the prover. This is analogous to the decisional version of zero knowedge, where the verifier is guaranteed not the learn anything but the validity of the proven statement in the sense that **given the bit** 1 (i.e. given that the statement is true), he could have simulated the entire interaction with the prover. What exactly do we mean by "given a solution"? Our interpretation is given in the formal definition below:

Definition 11. We say that the promise search problem $\mathcal{R} = (R_{YES}, L_{NO}) \in Search - ZK$ if there is a triplet (P, V, S) such that

- (P, V) is a $Search - IP$ protocol for $\mathcal{R}$.
- Zero-knowledge: S is an expected PPTM and for any PPTM adversary V^* and $x \in L_{R_{YES}}$:
$$\mathsf{view}_{V^*}^{(P,V^*)}(x) \approx S\left(x, V^*, (P,V)(x)\right)$$

Our interpretation of the zero knowledge property is that whatever any (possibly malicious) verifier can learn from the protocol (when run on an instance that has a solution) he could have also learned when presented with a **sample** from the distribution of legal solutions that corresponds to the protocol (when run by the honest parties). This is indeed a distribution since a yes instance x can have many possible solutions, and the one that is output can depend on the randomness. This is in contrast to the decisional version of zero knowledge protocols, where any instance has only one possible solution - either yes or no. Later we will show that if some search problem has a $Search - ZK$ protocol that always outputs the same solution for every yes instance x (i.e. $|supp\left((P,V)(x)\right)| = 1$) then in some sense the problem has a decisional zero-knowledge protocol.

Remark 12. The distribution on solutions from which the simulator gets a sample in Definition 11 depends on the honest parties executing it. That is, the simulator gets a sample from $(P,V)(x)$ rather than from some other distribution on $R_{YES}(x)$, which could be independent from the protocol. We could have defined an implementation-independent notion, where the search problem $\mathcal{R}$ is associated with a collection of distributions - each yes instance $x \in L_{R_{YES}}$ is paired with a distribution Y_x of legal solutions for x, and the protocol would have to satisfy the requirement that any poly-time verifier could have simulated its interaction with the honest prover when given a sample from Y_x. In this work we investigate the notion of zero-knowledge protocols for search problems as defined in Definition 11, but it is interesting to understand how things change when considering the other variant.

Remark 13. Like in the decisional case, the zero knowledge property can be either perfect, statistical or computational so we really have three classes: $Search-PZK$, $Search-SZK$ and $Search-CZK$.

3.1 Perfect Zero Knowledge Protocols For Any Search Problem With Efficiently Verifiable Solutions

The first thing to notice about $Search-ZK$ protocols is that any search problem for which the solutions can be verified by a deterministic poly-time Turing machine admits such a protocol. In fact, any such search problem admits a $Search-PZK$ protocol with a single message that is sent by the prover - namely some arbitrary solution to the given instance. Upon receiving the alleged solution the verifier checks its validity and then outputs it (or outputs $\perp$ if it was a fake solution). Completeness and soundness are clear, as the verifier only accepts legal solutions, and zero knowledge is achieved by having the simulator output the solution it is given (along with the input and the randomness for the verifier). As an important example, $Search-NP \subseteq Search-PZK$, i.e. given any language $L \in NP$ and NP-relation for it R_L, it holds that $R_L \in Search-PZK$. This example can be extended to MA which is the randomized counterpart of NP: formally, a language $L \in MA$ if there exists a randomized poly-time verifier V such that if $x \in L$ then there is some polynomially bounded witness $w = w(x)$ such that $V(x, w)$ accepts with probability 1, and if $x \notin L$ then for any w^*, $V(x, w) = 1$ with probability at most $\frac{1}{2}$. For any such verifier, the search problem of finding a witness w that makes the verifier accept with probability at least $\frac{1}{2}$ is in $Search-PZK$ with respect to the same protocol as described above, where the honest prover sends an arbitrary witness that makes the verifier accept with probability 1.

3.2 An Example of a $Search-PZK$ Protocol

Given two (undirected) graphs G_1, G_2 and two vertices $v_1 \in G_1, v_2 \in G_2$ we say that v_1 is isomorphic to v_2 if there is an isomorphism between G_1 and G_2 that maps v_1 to v_2. Consider the following search problem which we call $Find-Isomorphic-Vertex$: given two undirected graphs G_1, G_2 on n vertices and some vertex $v_1 \in G_1$, find a vertex $v_2 \in G_2$ that is isomorphic to v_1, or output $\perp$ if there is no such isomorphism. Note that it is not clear that this problem has efficiently verifiable solutions. That is, given $v_2 \in G_2$, it is not clear how to verify that v_2 corresponds to v_1 via some isomorphism between G_1 and G_2 without knowing what the isomorphism is. We propose the following protocol for this problem that is inspired by the original decisional zero knowledge protocol for the Graph-Isomorphism problem ([12]): given input (G_1, G_2, v_1):

P		V
Choose some arbitrary isomorphism π:$G_1 \to G_2$		
	$\xrightarrow{v_2 := \pi(v_1)}$	
Pick a random permutation $\varphi : [n] \to [n]$		
	$\xrightarrow{H = \varphi(G_1)}$	
	$\xrightarrow{u = \varphi(v_1)}$	
		randomly choose $j \in \{1, 2\}$
	$\xleftarrow{j}$	

if $j = 1,\ \phi := \varphi$
if $j = 2,\ \phi := \varphi \circ \pi^{-1}$		

	$\xrightarrow{\phi}$	
		verify that ϕ is an isomorphism, verify that $\phi(G_j) = H, \phi(v_j) = u$ if verification succeeded, output v_2

Lemma 14. *The above protocol is a $Search-PZK$ protocol for $Find-Isomorphic-Vertex$.*

Proof. First, note that V can be implemented by a polynomial time machine, as required. We will show completeness, soundness and perfect zero knowledge:

- Completeness: if G_1 is isomorphic to G_2 , and π is an isomorphism between them, then indeed $\varphi(G_1) = \varphi \circ \pi^{-1}(G_2) = H$ and $\varphi(v_1) = \varphi \circ \pi^{-1}(v_2) = u$. Hence w.p 1 V outputs v_2 as desired.

- Soundness: if G_1 is not isomorphic to G_2 then the graph H that the prover sends can not be isomorphic to both G_1 and G_2 (since isomorphism between graphs is an equivalence relation), and therefore with probability $\frac{1}{2}$ V rejects. If on the other hand, G_1 is isomorphic to G_2 but v_2 is not isomorphic to v_1, then, either H is not isomorphic to one of G_1, G_2 in which case w.p $\frac{1}{2}$ V rejects, or H is isomorphic to both, in which case it can not be that both v_1 and v_2 are isomorphic to u (since this would imply that v_1 and v_2 are isomorphic) and again V rejects w.p $\frac{1}{2}$.

- Zero-knowledge: Note that for any input (G_1, G_2, v_1) that is a yes instance it holds that $|supp((P, V)(x))| = |\{v_2\}| = 1$. We need to construct a simulator S for which $S((G_1, G_2, v_1), V^*, v_2)$ and $\mathsf{view}_{V^*}^{(P,V^*)}(G_1, G_2, v_1)$ are identically distributed for any verifier V^*. The simulator S, given $((G_1, G_2, v_1), V^*, v_2)$ proceeds as follows:
  - Choose random $j \in \{1, 2\}$.
  - Chooses a random permutation $\varphi : [n] \to [n]$.

– Choose randomness r for V^* and feed $H := \varphi(G_j)$ and $u := \varphi(v_j)$ into V^*.
– V^* outputs $j^* \in \{1, 2\}$.
– If $j^* \neq j$ halt.
– otherwise output $((G_1, G_2, v_1), r_{V^*}, v_2, H, u, j^*, \varphi)$.

S is indeed polynomial time, and conditioned on $j^* = j$ (which occurs with probability $\frac{1}{2}$), the output distribution of S is exactly $\mathsf{view}_{V^*}^{(P,V^*)}(x)$.

Hence, $Find - Isomorphic - Vertex \in Search - PZK$. $\square$

Now, consider the language of all tuples (G_1, G_2, v_1, v_2) such that $v_1 \in G_1, v_2 \in G_2$ and v_1 and v_2 are isomorphic. The protocol that we showed above (discarding the first message of the prover) is a decisional perfect zero knowledge protocol for that language. This shows that the search problem of finding an isomorphic vertex is in fact a prefix-completion problem of a language that has a (decisional) zero-knowledge protocol. Is this example a coincidence or is it a general phenomenon? Is it true that every search problem that has a $Search - ZK$ protocol is in fact a prefix-completion problem of a ZK decision problem? This question is addressed in the next section.

4 Prefix-Completion Problems

In this section we introduce a special sub-class of $Search - ZK$ which we call $Prefix - ZK$ and investigate its properties. Loosely speaking, these are search problems that can be solved in the following way: given an instance x, the prover sends a solution y and then proves in zero knowledge that the pair (x, y) satisfies some predetermined property. It need not necessarily be that (x, y) satisfies the property for **any** possible legal solution y, but there must exist at least one such y. Furthermore, if y is not a legal solution then (x, y) should not satisfy this property.

Definition 15 (The class $Prefix - ZK$)**.** We say that a promise search problem $\mathcal{R} = (R_{YES}, L_{NO}) \in Prefix - ZK$ if there exists a promise decision problem $\widehat{\mathcal{L}} = \left(\widehat{L}_{YES}, \widehat{L}_{NO}\right) \in ZK$ such that

- $\widehat{L}_{YES} \subseteq R_{YES}$.
- $\widehat{L}_{NO} = ((L_{R_{YES}} \cup L_{NO}) \times \{0,1\}^*) \setminus R_{YES}$.
- For every $x \in L_{R_{YES}}$ there is some $y \in R_{YES}(x)$ such that $(x, y) \in \widehat{L}_{YES}$.
- $\widehat{\mathcal{L}}_{YES}$ is polynomially bounded - there is some polynomial $p(\cdot)$ such that if $(x, y) \in \widehat{L}_{YES}$ then $|y| \leq p(|x|)$.

Remark 16. The condition bounding $|y|$ is necessary since the verifier is a polynomial time Turing machine and it should be able to read the solution y that the prover sends him.

Remark 17. $Find - Isomorphic - Vertex \in Prefix - PZK$.

The next lemma formalizes the intuition behind the definition of $Prefix - ZK$.

Lemma 18. $Prefix - ZK \subseteq Search - ZK$.

Proof. Let $\mathcal{R} = (R_{YES}, L_{NO}) \in Prefix - ZK$ with respect to the promise decision problem $\widehat{\mathcal{L}} = \left(\widehat{L}_{YES}, \widehat{L}_{NO}\right) \in ZK$, and let $\left(V_{\widehat{\mathcal{L}}}, P_{\widehat{\mathcal{L}}}, S_{\widehat{\mathcal{L}}}\right)$ be the ZK protocol for $\widehat{\mathcal{L}}$. We construct the following protocol $(P_{\mathcal{R}}, V_{\mathcal{R}}, S_{\mathcal{R}})$ for $\mathcal{R}$ and we will prove that it is a $Search - ZK$ protocol. Given input x:

$P_{\mathcal{R}}$		$V_{\mathcal{R}}$
Choose an arbitrary y such that $(x, y) \in \widehat{L}_{YES}$.		
	$\xrightarrow{y}$	
	$\left(P_{\widehat{\mathcal{L}}}, V_{\widehat{\mathcal{L}}}\right)((x, y))$	
		If previous step accepted, output y. Otherwise output $\perp$.

The simulator $S_{\mathcal{R}}$ works as follows: given (x, V^*, y), print y and run $S_{\widehat{\mathcal{L}}}((x, y), V^*)$. We claim that $(P_{\mathcal{R}}, V_{\mathcal{R}}, S_{\mathcal{R}})$ is a $Search - ZK$ protocol for $\mathcal{R}$. We need to show completeness, soundness and zero-knowledge.

- Completeness: suppose x is a yes instance. Then, by definition of $\widehat{\mathcal{L}}$, there exists some y such that $(x, y) \in \widehat{L}_{YES}$, which $P_{\mathcal{R}}$ sends. By the completeness of the ZK protocol for $\widehat{\mathcal{L}}$ it is guaranteed w.p 1 that the $\left(P_{\widehat{\mathcal{L}}}, V_{\widehat{\mathcal{L}}}\right)((x, y))$ sub protocol makes $V_{\mathcal{R}}$ accept y.
- Soundness:
  - Let x be a no instance and let $P^*_{\mathcal{R}}$ be any prover. The fact that it is a no instance implies that for any string y, $(x, y) \in \widehat{\mathcal{L}}_{NO}$. Therefore, no matter what string y $P^*_{\mathcal{R}}$ sends, we are guaranteed by the soundness of the $\left(P_{\widehat{\mathcal{L}}}, V_{\widehat{\mathcal{L}}}\right)$ sub protocol that $V_{\mathcal{R}}$ will output $\perp$ w.p at least $\frac{1}{2}$.
  - Let x be a yes instance and let $P^*_{\mathcal{R}}$ be any prover. If $P^*_{\mathcal{R}}$ sends a non-solution y then $(x, y) \in \widehat{L}_{NO}$ and the soundness of the $\left(P_{\widehat{\mathcal{L}}}, V_{\widehat{\mathcal{L}}}\right)$ protocol will cause $V_{\mathcal{R}}$ to output $\perp$ w.p at least $\frac{1}{2}$. If on the other hand he sends a correct solution y, then there is no problem if $V_{\mathcal{R}}$ accepts (note that in this case, the pair (x, y) is not necessarily in the promise of $\widehat{\mathcal{L}}$, i.e. it does not necessarily belong to $\widehat{L}_{YES} \cup \widehat{L}_{NO}$).
- Zero-Knowledge: for any $x \in L_{R_{YES}}$ and verifier V^* the fact that $\mathsf{view}_{V^*}^{(P_{\mathcal{R}}, V^*)}(x) \approx S_{\mathcal{R}}(x, V^*, (P_{\mathcal{R}}, V_{\mathcal{R}})(x))$ follows immediately from the fact that $\mathsf{view}_{V^*}^{\left(P_{\widehat{\mathcal{L}}}, V^*\right)}((x, y)) \approx S_{\widehat{\mathcal{L}}}((x, y), V^*)$.

□

Remark 19. Note that the solution y that $P_{\mathcal{R}}$ sends in the first message can be taken from any distribution on $\widehat{L}_{YES}(x)$ and the proof would still work. Thus Lemma 18 also holds if $Search-ZK$ is defined in an implementation-independent manner as described in Remark 12 (as long as for any $x \in L_{R_{YES}}$ the corresponding solution distribution is supported only on elements from $\widehat{L}_{YES}(x)$).

4.1 A Complete Problem for $Prefix-ZK$ Based on Any Complete Problem for ZK

In this section we show that the existence of a complete problem for ZK, with respect to Karp reductions, implies the existence of a complete problem for $Prefix-ZK$ with respect to solution-preserving Levin reductions. In particular, since SZK has a complete problem ([24]) so does $Prefix-SZK$. Let $\mathcal{L}^* = (L^*_{YES}, L^*_{NO})$ be a complete problem for ZK with respect to Karp reductions. We define the 'General Prefix-Completion Problem' $\widehat{\mathcal{R}} = \left(\widehat{R}_{YES}, \widehat{L}_{NO}\right)$, as follows: the legal inputs consist of all tuples $(M, 1^t, x)$ where M is a deterministic Turing machine and it is promised that for any string y, $M(x,y)$ outputs after at most t steps a string z which is guaranteed to belong either to L^*_{YES} or to L^*_{NO}. A string y is a solution to the instance $(M, 1^t, x)$ if the output string z belongs to L^*_{YES}. An instance $(M, 1^t, x)$ is a no instance if for every string y, $M(x,y)$ outputs after at most t steps a string $z \in L^*_{NO}$. We claim that $\mathcal{R}^* \in Prefix-ZK$. Indeed, the promise decision problem $\mathcal{L}' = \left(L'_{YES}, L'_{NO}\right)$ where

$$L'_{YES} = \left\{\left(\left(M, 1^t, x\right), y\right) \mid M(x,y) \text{ outputs a string } z \in L^*_{YES} \text{ after at most } t \text{ steps}\right\}$$
$$L'_{NO} = \left\{\left(\left(M, 1^t, x\right), y\right) \mid M(x,y) \text{ outputs a string } z \in L^*_{NO} \text{ after at most } t \text{ steps}\right\}$$

satisifes the bullets in Definition 15, and it is in ZK, with the protocol for it consisting of computing z from $(M, 1^t, x, y)$ and then running the original protocol for $\mathcal{L}^*$ on z.

Lemma 20. *Let $\mathcal{R} = (R_{YES}, L_{NO})$ be a promise search problem. The following are equivalent:*

- *$\mathcal{R} \in Prefix-ZK$.*
- *There is a solution preserving Levin reduction from $\mathcal{R}$ to $\widehat{\mathcal{R}}$.*

Proof. Suppose $\mathcal{R} = (R_{YES}, L_{NO}) \in Prefix-ZK$ with respect to $\tilde{\mathcal{L}} = \left(\tilde{L}_{YES}, \tilde{L}_{NO}\right) \in ZK$, and let p be the polynomial bounding $\tilde{L}_{YES}$ (i.e, $(x,y) \in \tilde{L}_{YES}$ implies $|y| \le p(|x|)$). Let M be the Karp reduction machine from $\tilde{\mathcal{L}}$ to $\mathcal{L}^*$ and let's denote its polynomial running time by $p^*(\cdot)$ (we know that such a reduction exists since $\tilde{\mathcal{L}} \in ZK$ and $\mathcal{L}^*$ is ZK-complete). Then the Levin-reduction consists of the transformation algorithms A, B such that

$$A(x) = \left(M, 1^{p^*(|x|+p(|x|))}, x\right)$$
$$B(x,y) = y$$

Note that if $x \in L_{R_{YES}}$ then there is some y such that $(x, y) \in \tilde{L}_{YES}$ which implies that $M(x, y) \in L^*_{YES}$ and therefore $A(x) = \left(M, 1^{p^*(|x|+p(|x|))}, x\right) \in L_{\widehat{R}_{YES}}$ (note that the running time of M on (x, y) is bounded by $p^*(|x| + p(|x|))$. On the other hand, if $x \in L_{NO}$ then for any y, $(x, y) \in \tilde{L}_{NO}$ which implies that $M(x, y) \in L^*_{NO}$ and so $A(x) \in \widehat{L}_{NO}$. Finally, this reduction is clearly solution preserving. We now turn to prove the other direction. We showed before that $\widehat{\mathcal{R}} \in Prefix - ZK$ with respect to $\mathcal{L}' = (L'_{YES}, L'_{NO}) \in ZK$. Let's assume that (A, B) is a solution preserving Levin reduction from $\mathcal{R}$ to $\widehat{\mathcal{R}}$ and we need to show that $\mathcal{R} \in Prefix - ZK$. We denote $A(x) = (M_x, 1^{t_x}, x')$ and

$$\tilde{L}_{YES} = \{(x, y) \in R_{YES} \mid (A(x), y) \in L'_{YES}\}$$
$$\tilde{L}_{NO} = ((L_{R_{YES}} \cup L_{NO}) \times \{0,1\}^*) \setminus R_{YES}$$

$\tilde{\mathcal{L}} = \left(\tilde{L}_{YES}, \tilde{L}_{NO}\right)$ fulfills the requirements from Definition 15. It is left to prove that $\tilde{\mathcal{L}} \in ZK$: on input (x, y) the parties simply execute the L'_{YES} protocol on $(A(x), y)$, and the simulator on input (x, y) simply runs the simulator of $\mathcal{L}'$ on $(A(x), y)$.

- Completeness: follows immediately from the definition of $\tilde{L}_{YES}$ and the completeness of the ZK protocol for $\mathcal{L}'$.
- Soundness: let's assume that $(x, y) \in \tilde{L}_{NO}$. There are two cases:
  - $x \in L_{NO}$: in this case $A(x) = (M_x, 1^{t_x}, x') \in \widehat{L}_{NO}$ which implies $(A(x), y) \in L'_{NO}$. Hence the protocol rejects w.p $\frac{1}{2}$ due to the soundness of the $\mathcal{L}'$ protocol.
  - $x \in L_{R_{YES}}$ but $y \notin R_{YES}(x)$. Therefore, since the Levin reduction is solution preserving, $y \notin R^*_{YES}(A(x))$, which implies $(A(x), y) \in L'_{NO}$, and again this implies that the protocol rejects w.p $\frac{1}{2}$.
- Zero knowledge: Follows immediately from the zero-knowledge of the ZK protocol for $\mathcal{L}'$.

□

4.2 The Relationship Between $Search - ZK$ and $Prefix - ZK$

Does every $Search - ZK$ protocol essentially amount to having the prover send the solution and prove in zero knowledge that it really is a solution? Is it true that $Search - ZK \subseteq Prefix - ZK$? As we will see, it turns out that in general the answer to this question is no. Nevertheless there are some conditions under which a $Search - ZK$ protocol can be transformed into a $Prefix - ZK$ protocol. In the first part of this section we show results that hold simultaneously for all three types of zero-knowledge (CZK, SZK, PZK). After that we will give specific treatments of the computational and statistical settings.

We start by considering a special class of $Search - ZK$ protocols, in which the prover sends the solution at the first message of the interaction. It is tempting to conclude immediately that in this case the language of instance-solution pairs is in ZK (implying that the search problem is in $Prefix-ZK$) since what else could the prover be doing if not sending a solution and proving (in zero knowledge) that it is a solution? The problem with this reasoning is that it does not take into account the fact that the prover can be randomized and in particular the solution it sends in the first message can be a random variable that is supported on more than one element. In this case, the straight-forward protocol one would come up with to show that the language of instance-solution pairs is in ZK (namely, given input (x, y), run the original protocol on x and accept if the solution is y) does not work, as there is no real reason to believe that the solution given in the input would equal the solution that the prover sends in the first message. Nevertheless, we can consider a distributional variant of zero knowledge under which the intuition above is correct.

Definition 21 (The class $Dist - ZK$)**.** We say that a promise decision problem $\mathcal{L} = (L_{YES}, L_{NO}) \in Dist - ZK$ if there is a triplet (P, V, S) such that:

- (P, V) is an IP protocol for $\mathcal{L}$.
- Distributional zero knowledge: S is an expected PPTM and there is a collection of distributions on yes instances $\{X_k\}_{k=1}^{\infty}$ such that every yes instance is in the support of one of these distributions, and for any verifier V^* and k,

$$S\left(X_k, V^*\right) \approx \mathsf{view}_{V^*}^{(P,V^*)}\left(X_k\right)$$

We define $Prefix - Dist - ZK$ like $Prefix - ZK$ only that the decision problem $\widehat{\mathcal{L}}$ from that definition is required to be in $Dist - ZK$.

Lemma 22. *Let $\mathcal{R} = (R_{YES}, L_{NO}) \in Search - ZK$ with respect to (P, V, S) in which the prover sends the solution in the first message of the protocol. Then $\mathcal{R} \in Prefix - Dist - ZK$.*

Proof. Denote $\widehat{\mathcal{L}} = \left(\widehat{L}_{YES}, \widehat{L}_{NO}\right)$ where

$$\widehat{L}_{YES} := \{(x, y) \mid x \in L_{R_{YES}}, y \in supp\left((P, V)(x)\right)\}$$
$$\widehat{L}_{NO} := \left(\left(L_{R_{YES}} \cup L_{R_{NO}}\right) \times \{0,1\}^*\right) \backslash R_{YES}$$

The four bullets of Definition 15 are clearly statisfied, and it is left to show that $\widehat{\mathcal{L}} \in Dist-ZK$. The protocol for $\widehat{\mathcal{L}}$ is as follows: given input (x, y) the parties run the given $Search - ZK$ protocol on x, with the modification that the prover sends y in the first message (and not an independent sample from $(P, V)(x)$). The verifier accepts if the $Search - ZK$ protocol succeeded. Soundness and perfect completeness follow immediately from the soundness and perfect completeness of the $Search - ZK$ protocol. We need to show distributional zero knowledge. The simulator $S_{\widehat{\mathcal{L}}}\left((x, y), V^*\right)$ runs $S(x, V^*, y)$, and for each $x \in L_{R_{YES}}$ we define the distribution D_x whose support is all the pairs (x, y) for $y \in supp\left((P, V)(x)\right)$ and

$$\Pr\left[D_x = (x, y)\right] := \Pr\left[(P, V)(x) = y\right].$$

Since the $Search - ZK$ protocol guarantees that for any $x \in L_{R_{YES}}$ and any verifier V^* we have

$$S\left(x, V^*, (P, V)(x)\right) \approx \mathsf{view}_{V^*}^{(P,V^*)}(x)$$

this implies that for any $x \in L_{R_{YES}}$ and any verifier V^* we have

$$S_{\widehat{\mathcal{L}}}\left(D_x, V^*\right) \approx \mathsf{view}_{V^*}^{\left(P_{\widehat{\mathcal{L}}}, V^*\right)}\left(D_x\right)$$

where $P_{\widehat{\mathcal{L}}}$ is the honest prover for $\widehat{\mathcal{L}}$. □

The relaxed notion of distributional zero knowledge was required for the proof of Lemma 22 since the prover could have sampled the solution from a distribution whose support contains more than one element. The next lemma shows that if, for any yes instance x, the solution $(P, V)(x)$ is supported on one element then we can assume without loss of generality that the prover sends it in the first message of the interaction and conclude that the underlying search problem is indeed in $Prefix - ZK$. Intuitively, if the solution for each instance is unique then in particular it does not depend on the verifier randomness and so the prover knows it already in the beginning of the protocol.

Lemma 23. *Let $\mathcal{R} = (R_{YES}, L_{NO}) \in Search - ZK$ with respect to the protocol (P, V, S). Assume that the protocol has the property that each yes instance has only one solution that is output when the honest parties interact. That is, for any $x \in L_{R_{YES}}$ we have $\left|supp\left((P, V)(x)\right)\right| = 1$. Then $\mathcal{R} \in Prefix - ZK$.*

Proof. We define $\widehat{\mathcal{L}} = \left(\widehat{L}_{YES}, \widehat{L}_{NO}\right)$ where

$$\widehat{L}_{YES} = \left\{(x, y) \mid x \in L_{R_{YES}}, y = (P, V)(x)\right\}$$
$$\widehat{L}_{NO} = \left(\left(L_{R_{YES}} \cup L_{NO}\right) \times \{0,1\}^*\right) \backslash R_{YES}$$

Note that the solutions $y \in supp\left((P, V)(x)\right)$ are polynomially bounded (since they are output by the poly-time verifier) and in particular $\widehat{L}_{YES}$ is polynomially bounded. $\mathcal{L}$ satisfies the rest of the bullets in Definition 15, and it is in ZK: given (x, y), the parties run the $Search - ZK$ protocol for $\mathcal{R}$ on x and the verifier accepts if the solution was y. Since $\left|supp\left((P, V)(x)\right)\right| = 1$, it is guaranteed with probability 1 that on input $(x, y) \in \widehat{L}_{YES}$ the original $Search - ZK$ protocol outputs y when run on x with the honest prover. Thus we have perfect completeness. If $(x, y) \in \widehat{L}_{NO}$ then in particular $(x, y) \notin R_{YES}$ and soundness follows from the soundness of the $Search - ZK$ protocol. As for zero knowledge - the simulator $S_{\widehat{\mathcal{L}}}$ on input $\left(V^*, (x, y)\right)$ runs $S\left(x, V^*, y\right)$. □

Next we investigate the relationship between $Search - ZK$ and $Prefix - ZK$ specifically in the computational and statistical settings.

4.2.1 The Computational Setting

Is it true that $Search-CZK \subseteq Prefix-CZK$? In this section we will investigate this question and we start by characterizing $Prefix-CZK$.

Definition 24 (The class $Search-PSPACE$)**.** We say that a promise search-problem $\mathcal{R} = (R_{YES}, L_{NO}) \in Search-PSPACE$ if there is a deterministic polynomial space Turing machine M that solves $\mathcal{R}$: for every $x \in L_{R_{YES}}$, $M(x) \in R_{YES}(x)$ and for every $x \in L_{NO}$, $M(x) = \perp$.

Definition 25 (The class $Search-PSPACE(poly)$)**.** $\mathcal{R} \in Search-PSPACE(poly)$ if $\mathcal{R} \in Search-PSPACE$ and there is a polynomial space deterministic Turing machine M and a polynomial $p(\cdot)$ such that M solves $\mathcal{R}$ and for any $x \in L_{R_{YES}}$ $|M(x)| \le p(|x|)$.

Remark 26. Note that if $\mathcal{R} \in Search-PSPACE(poly)$ then every yes instance has at least one polynomially bounded solution.

Lemma 27. *$Prefix-CZK \subseteq Search-PSPACE(poly)$.*

Proof. Let $\mathcal{R} = (R_{YES}, L_{NO}) \in Prefix-CZK$ with respect to $\widehat{\mathcal{L}} = \left(\widehat{L}_{YES}, \widehat{L}_{NO}\right) \in CZK$. In particular, $\widehat{\mathcal{L}} \in IP$ and therefore $\widehat{\mathcal{L}} \in PSPACE$ ([25]). We can therefore solve $\mathcal{R}$ in polynomial space in the following way: given input x, we go over all the strings y of length at most $p(|x|)$ where $p(\cdot)$ is the polynomial from Definition 15, and decide in polynomial space if $(x,y) \in \widehat{L}_{YES}$. If there is some y for which $(x,y) \in \widehat{L}_{YES}$, we output that y. Otherwise we output $\perp$. The algorithm we described indeed solves $\mathcal{R}$: if $x \in L_{R_{YES}}$ then there is some y of length at most $p(|x|)$ for which $(x,y) \in \widehat{L}_{YES} \subseteq R_{YES}$ and one of these strings y is output by the algorithm . If on the other hand $x \in L_{NO}$ then for any y $(x,y) \in \widehat{L}_{NO}$, and thus the algorithm on x outputs $\perp$. □

Lemma 28. *If one-way functions exist then $Search-PSPACE(poly) \subseteq Prefix-CZK$.*

Proof. Let $\mathcal{R} = (R_{YES}, L_{NO}) \in Search-PSPACE(poly)$ with respect to the deterministic poly-space Turing machine M and polynomial $p(\cdot)$. Denote $\widehat{\mathcal{L}} = \left(\widehat{L}_{YES}, \widehat{L}_{NO}\right)$ for $\widehat{L}_{YES} = \{(x, M(x)) \in R_{YES}\}$, $\widehat{L}_{NO} = ((L_{R_{YES}} \cup L_{NO}) \times \{0,1\}^*) \setminus R_{YES}$. $\mathcal{L}$ clearly satisfies the requirements in the four bullets of Definition 15. We need to show that $\mathcal{L} \in CZK$ - assuming the existence of one-way functions, it is enough to show that $\mathcal{L} \in IP$ ([2]), or equivalently that $\mathcal{L} \in PSPACE$. The following deterministic and poly-space algorithm solves $\mathcal{L}$: Given (x,y), run M on x and accept if $M(x) = y$. □

The combination of Lemmas 27 and 28 give us

Theorem 29. *If one-way functions exist then $Prefix-CZK = Search-PSPACE(poly)$.*

The characterization of $Prefix-CZK$ hints at a possible path for proving that $Search-CZK \subseteq Prefix-CZK$ - show that any problem in $Search-CZK$ has a deterministic poly-space machine that solves it with polynomially bounded solutions. Now, it is tempting to claim that $Search-CZK \subseteq Search-PSPACE\left(poly\right)$ in the spirit of the proof that $IP \subseteq PSPACE$, but it turns out that the idea behind that proof does not translate to the realm of search problems.

Following is an informal sketch of the proof of $IP \subseteq PSPACE$: given an input x, the computation tree that corresponds to the given protocol execution on x is considered. Each node in the i'th level of the tree corresponds to the party whose turn it is to send the i'th message, and each out-edge corresponds to a possible message that can be sent by that party. A polynomial space machine can determine the prover strategy that maximizes the verifier's acceptance probability in the following recursive manner: first assign value 0 or 1 to each leaf depending on whether the computation path that leads to that leaf makes the verifier accept or reject. Then assign each verifier node the average of the values of its out-neighbors, according to the probability of choosing each out-message (a poly-space machine can compute this probability by going over all possible coin tosses for the verifier), and assign each prover node the maximum value among the values of its out-neighbors. The value in the root corresponds to the maximum acceptance probability and so if that value is 1 the algorithm can conclude that the input was a yes instance, and if that value is at most $\frac{1}{2}$ then the algorithm can conclude that the input was a no instance. In the case of interactive protocols for search problems it is not clear how to mimic this procedure. Each computation path (leaf) corresponds to some solution, but it is not possible to verify if that solution is valid or not unless assuming a priory that a solution can be verified in polynomial space. The soundness condition assures us that the output of the interaction between any prover $P^*$and the honest verifier on some $x \in L_{R_{YES}}$ is either a solution or $\perp$ w.p at least $\frac{1}{2}$, but nothing more. Imagine the following scenario for example: take some yes intance x, a solution y_1 for it, a non-solution y_2 and some prover P^* for which

$$\Pr\left[\left(P^*,V\right)(x)=y_1\right]=\frac{1}{4}$$
$$\Pr\left[\left(P^*,V\right)(x)=y_2\right]=\frac{1}{2}-\varepsilon$$
$$\Pr\left[\left(P^*,V\right)(x)=\perp\right]=\frac{1}{4}+\varepsilon$$

this scenario complies with the soundness requirement, but why should a deterministic machine choose y_1 over y_2 upon computing these probabilities?

Theorem 30. *$Search-IP \nsubseteq Search-PSPACE\left(poly\right)$.*

The theorem is proven using a counterexample based on [10]. The version presented here contains a modification due to a discussion with Grossman which allows to generalize the counterexample to protocols with perfect completeness.

Proof. Consider the search problem $R=\left\{(x,y) \mid |y|=5\,|x| \text{ and } K\left(y\right)>2\,|x|\right\}$ where all instances are legal (i.e. the promise is trivial) and $K\left(z\right)$ denotes the Kolmogorov complexity of z. Then $R \notin Search - PSPACE\,(poly)$ since otherwise, if M is the respective machine solving R, each solution y would satisfy $K\,(y) = |M| + |x| = O\,(1) + |x|$ which is a contradiction to $K\,(y) > 2\,|x|$ (when $|x|$ is large enough). We claim that $R \in Search - IP$. To show this, we note that the the number of strings whose Kolmogorov complexity is at most $2\,|x|$ is upper bounded by $2^{2|x|+1} - 1$ (which is a bound on the number of Turing machines whose encoding is of size at most $2\,|x|$). In particular, There exists a prefix $a \in \{0,1\}^{2|x|+1}$ such that for any suffix $b \in \{0,1\}^{3|x|-1}$, the concatenation y=$a \parallel b \in \{0,1\}^{5|x|}$ satisfies $K\,(y) > 2\,|x|$ and so y is a solution to the instance x. We call such a prefix a 'good'. On the other hand, For any prefix $a \in \{0,1\}^{2|x|+1}$, the fraction of such suffixes b for which $K\,(a \parallel b) \leq 2\,|x|$ is upper bounded by $\frac{2^{2|x|+1}}{2^{3|x|-1}} < \frac{1}{2}$ (for $|x| > 2$). This suggests the following $Search - IP$ protocol for R. Given input x:

P		V
choose a good $a \in \{0,1\}^{2|x|+1}$		
	$\xrightarrow{a}$	
		sample a uniform $b \leftarrow \{0,1\}^{3|x|-1}$ and output $a \parallel b$

The discussion above shows that this protocol has perfect completeness and soundness error $\frac{1}{2}$, and the theorem follows. $\square$

The $Search - IP$ protocol presented above is also perfect zero-knowledge. Note that the only information revealed by the prover is a part of the solution, hence it is clear that no (possibly malicious) verifier can learn anything other than the solution when interacting with the honest prover. A simulator for this protocol, upon receiving $(x, V^*, a \parallel b)$, simply prints x, the randomness for V^* and a. Thus $R \in Search - PZK$ and together with $R \notin Search - PSPACE\,(poly)$ and $Prefix - CZK \subseteq Search - PSPACE\,(poly)$ (Lemma 27) we get

Theorem 31. *$Search - PZK \nsubseteq Prefix - CZK$.*

Strengthening the Requirements - Zero Knowledge Protocols for Search Problems with Zero Error. The foregoing counter-example shows us that the definition of $Search - ZK$ as presented captures even uncomputable problems. Indeed, the analysis of the example shows that it can not be solved by any deterministic Turing machine. We would like to modify the definition of search zero knowledge so that such a phenomenon could not be possible. As we will see next, one option is to not allow any soundness error in the protocol. In other words, the soundness requirement from the honest verifier is:

1. For any $x \in L_{NO}$ and any P^*, $\Pr\left[(P^*, V)\,(x) = \perp\right] = 1$.

2. For any $x \in L_{R_{YES}}$ and any P^*, $\Pr\left[(P^*, V)\,(x) \in R_{YES}\,(x) \cup \{\perp\}\right] = 1$.

This is very similar to the requirement in ZPP algorithms, which are randomized poly-time algorithms for decision problems with the guarantee that **whenever they output a solution it is always correct**, but they are allowed to output a 'don't know' symbol with some low probability ($\perp$). In the same manner, a $Search - IP$ protocol with perfect completeness and perfect soundness guarantees that whenever the honest verifier outputs a solution, it really is a legal solution. We denote the class of search problems having $Search - IP$ protocols with perfect soundness by $ZP - Search - IP$. The subclass of search problems that have $ZP - Search - IP$ protocols which also satisfy the zero-knowledge property (as in definition 11) is denoted by $ZP - Search - ZK$ (as usual, ZK is replaced by CZK, SZK and PZK depending on the quality of simulation). The following claim shows that $ZP - Search - ZK$ contains only computable problems.

Lemma 32. *$ZP - Search - IP \subseteq Search - PSPACE\,(poly)$.*

Proof. Let $\mathcal{R} \in ZP - Search - IP$ with respect to the protocol (P, V). Given input x, a $Search - PSPACE(poly)$ machine can simulate the execution of the given protocol on x between the honest verifier and any deterministic prover - it can iterate over all possible coins for the honest verifier and all possible prover responses. The completeness property guarantees that in one of the iterations some valid solution is output. Furthermore, the soundness property guarantees that whenever a solution is output (in contrast to $\perp$) then it is valid. Hence, the machine can simply output the first solution it encounters. $\square$

Corollary 33. *$ZP - Search - CZK \subseteq Search - PSPACE\,(poly)$.*

If we could prove the converse inclusion of Corollary 33 then we could conclude that $ZP - Search - CZK = Prefix - CZK$, showing exactly which $Search - CZK$ protocols are in fact prefix-completion problems of CZK problems. Unfortunately, as we will show next, it is very likely that the inclusion $ZP - Search - IP \subseteq Search - PSPACE\,(poly)$ is strict (which implies that the inclusion $ZP - Search - CZK \subseteq Search - PSPACE\,(poly)$ is strict), indicating that the zero error requirement is too strong. We will need the following lemma:

Lemma 34. *Let $R \in ZP - Search - IP$ where the promise is trivial and every yes instance has exactly one solution, i.e. $|R(x)| = 1$ for any $x \in L_R$. Then the set R is an NP language.*

Proof. Let R be as above and suppose that (P, V) is the $ZP - Search - IP$ protocol for R. The perfect soundness and completeness conditions imply that we can assume without loss of generality that V is deterministic (by fixing its random tape to the all zero string for example) - we are still guaranteed that if x is a yes instance then the legal solution will be output when interacting with the honest prover P, and no prover P^* can make V output something other than the solution, or $\perp$. Since V is deterministic there is no need for interaction at all, since the prover can anticipate all of the messages of V. That is, there is another protocol for R with perfect completeness and soundness where, given input x, the prover sends the entire transcript that would have been produced in the original protocol and the verifier simply needs to verify consistency. Hence the set of pairs R is in NP: given (x, y), the prover can send the message that corresponds to x in the protocol for R, and the verifier accepts if the solution was y. If $(x, y) \in R$ then y is the only legal solution for x and therefore the message that the honest prover sends corresponds to the solution y. On the other hand, if $(x, y) \notin R$ then the perfect soundness of the 1-round protocol for R guarantees that the verifier always rejects regardless of the message he receives. $\square$

Theorem 35. *If $NP \subsetneq PSPACE$ then $ZP - Search - IP \subsetneq Search - PSPACE\,(poly)$.*

Proof. Assume towards contradiction that $ZP - Search - IP = Search - PSPACE\,(poly)$ and let $L \in PSPACE$. Let's assume for now that L is a language of pairs (x, y), where $|x| = |y|$ and for any $(x, y), (x, y') \in L$ it holds that $y = y'$. In other words, L is a length-preserving function on a subset of $\{0,1\}^*$. First we will show that $L \in NP$ and after that we will show that if $NP \subsetneq PSPACE$ then there is some $L' \in PSPACE \backslash NP$ that is a length preserving function, giving us the desired contradiction . To see that $L \in NP$, consider the search problem R induced from L. That is, given x the goal is to find y such that $(x, y) \in L$. The fact that $L \in PSPACE$ clearly implies that $R \in Search - PSPACE\,(poly)$, since given x, a poly-space machine can go over all y of the same length as x and check if $(x, y) \in L$. Therefore by our assumption, $R \in ZP - Search - IP$. Furthermore, observe that each yes instance of R has only one solution (since L is a function). Therefore, by Lemma 34, $L = R \in NP$. To summarize, we have shown that under the assumption $ZP - Search - IP = Search - PSPACE\,(poly)$, any $PSPACE$ language that is a length-presesrving function is also an NP language. It is left to show that if $NP \subsetneq PSPACE$ then there is some language in $PSPACE \backslash NP$ that is a length-preserving function: given $L \in PSPACE \backslash NP$, take $L' = \{(x, x) \mid x \in L\}$. $\square$

4.2.2 The Statistical Setting

In this section we show conditions under which a search problem that has a $Search - SZK$ protocol is "close to being" in $Prefix - SZK$, in the sense that the zero-knowledge property that we are able to guarantee for the decision problem $\widehat{\mathcal{L}}$ from the definition of $Prefix - SZK$ is weaker. Specifically we show that under some conditions a $Search - SZK$ protocol can be modified so as to output unique solutions for any yes instance. As we will see, this modification preserves soundness and perfect completeness, but unfortunately the zero-knowledge property gets weaker. Then we will use (a variant of) Lemma 23 to conclude that the search problem is in some class which is a relaxation of $Prefix - SZK$. We define the classes $HV - ZK$, $Search - HV - ZK$ and $Prefix - HV - ZK$ as follows (where HV stands for 'honest verifier'):

Definition 36 ($HV - ZK$)**.** We say a promise decision problem $\mathcal{L} = (L_{YES}, L_{NO}) \in HV - ZK$ if there is a triplet (P, V, S) such that:

- (P, V) is an IP protocol for $\mathcal{L}$.
- Honest verifier zero knowledge: S is an expected PPTM and for any $x \in L_{YES}$,

$$\mathsf{view}_V^{(P,V)}(x) \approx S(x, V).$$

Definition 37 ($Search - HV - ZK$)**.** We say that the promise search problem $\mathcal{R} = (R_{YES}, L_{NO}) \in Search - HV - ZK$ if there is a triplet (P, V, S) such that

- (P, V) is a $Search - IP$ protocol for $\mathcal{R}$.
- Honest verifier zero knowledge: S is an expected PPTM and for any $x \in L_{R_{YES}}$:

$$\mathsf{view}_V^{(P,V)}(x) \approx S(x, V, (P, V)(x))$$

Definition 38 ($Prefix - HV - ZK$)**.** We say that $\mathcal{R} = (R_{YES}, L_{NO}) \in Prefix - HV - ZK$ if there exists a promise decision problem $\widehat{\mathcal{L}} = \left(\widehat{L}_{YES}, \widehat{L}_{NO}\right) \in HV - ZK$ such that

- $\widehat{L}_{YES} \subseteq R_{YES}$.
- $\widehat{L}_{NO} = ((L_{R_{YES}} \cup L_{NO}) \times \{0,1\}^*) \setminus R_{YES}$.
- For every $x \in L_{R_{YES}}$ there is some $y \in R_{YES}(x)$ such that $(x, y) \in \widehat{L}_{YES}$.
- $\widehat{\mathcal{L}}_{YES}$ is polynomially bounded.

First we notice that Lemmas 18 and 23 still hold when switching every appearance of ZK with $HV - ZK$: this can be readily seen by noting that the simulators constructed in the proofs are essentially the same simulators which are guaranteed by the assumptions in the lemmas. Thus, if we show conditions under which a search problem $\mathcal{R}$ that has a $Search - SZK$ protocol also admits a $Search - HV - SZK$ protocol that outputs unique solutions for any yes instance, then in particular we can conclude that $\mathcal{R} \in Prefix - HV - SZK$.

We will focus on **public-coin** $Search - SZK$ protocols. These are protocols where every verifier message is a uniformly-distributed string that is independent from the rest of the messages, and the output of the protocol is a deterministic function of the transcript. That is, the verifier does not utilize any randomness that was not sent to the prover in one of the messages. Public-coin $Search - SZK$ protocols that have perfect completeness have the following property: any transcript that is generated from an interaction with the honest prover corresponds to some legal solution, even if the verifier is not honest. To be more formal, Let (P, V, S) be a public-coin $Search - SZK$ protocol for some search problem $\mathcal{R}$. Let's assume that the number of rounds in the protocol is $m(|x|)$ and each message has length $q(|x|)$. Let's also assume that the prover sends the first message. Hence each possible transcript in the protocol has the following form:

$$(p_1, v_1, \ldots, p_m, v_m)$$

where v_i and p_i are the i'th verifier and prover message (respectively), and for any i, v_i and p_i are binary strings of length $q(|x|)$. Each of the messages is a random variable that depends on the verifier and prover randomness. When the honest verifier runs the protocol, all the v_i's are uniform and independent, and the perfect completness property asserts that for each of the $2^{m(|x|)q(|x|)}$ possible message tuples that the verifier can send, and for each of the possible honest prover message tuples (the amount of which can be much smaller than $2^{m(|x|)\cdot q(|x|)}$), the resulting transcript corresponds to some solution that the honest verifier outputs. In particular, even if a cheating verifier runs the protocol and sends messages that are not uniform or independent, the resulting transcript still corresponds to some solution as long as the prover runs his predescribed strategy. The cheating verifier can of course decide to output something other than that solution. Now, recall that for any instance x and verifier V^*, $\mathsf{view}_{V^*}^{(P,V^*)}(x)$ is the concatenation of the input x, the randomness of V^* and the transcript produced by the interaction. We denote by sol the function that takes such a view and outputs its corresponding solution. This function, as explained above, is well defined when the view corresponds to a yes instance, and it is essentially the function which is computed by the honest verifier after the final message in the transcript is transmitted. Note that sol can be extended also to views that do not correspond to an interaction with the honest prover: this is defined as the output computed by the honest verifier from this view. Now, given some distribution on views D (such as the output of a simulator, or the real view of the verifier) and a solution $y \in supp\left((P,V)(x)\right)$, we denote by $[D \mid y]$ the induced distribution conditioned on $\mathsf{sol}(D) = y$. Formally, for any t

$$\Pr\left[[D \mid y] = t\right] := \Pr\left[D = t \mid \mathsf{sol}(D) = y\right].$$

The main result of this section is presented next:

Theorem 39. *Let $\mathcal{R} = (R_{YES}, L_{NO}) \in Search - SZK$ with respect to a public-coin protocol (P,V,S) that has the following properties:*

- *For any $x \in L_{R_{YES}}$, $\mathsf{sol}\left(S\left(x, V, (P,V)(x)\right)\right) = (P,V)(x)$. I.e., the transcript output by the simulator when run on the honest verifier always corresponds to the solution that it started with.*

- *The first message in the protocol is sent by the prover, and that message contains the solution that the honest verifier outputs when the prover is honest.*

Then $\mathcal{R} \in Prefix - HV - SZK$.

To prove this we will need to following lemma.

Lemma 40. *Let (P,V,S) be a public coin $Search - SZK$ protocol for a search problem $\mathcal{R} = (R_{YES}, L_{NO})$, and let V^* be some verifier strategy for which:*

- *For any $x \in L_{R_{YES}}$, $\mathsf{sol}\left(S\left(x, V^*, (P,V)(x)\right)\right)$ and $(P,V)(x)$ are identically distributed.*

- *For any $x \in L_{R_{YES}}$, $\mathsf{sol}\left(\mathsf{view}_{V^*}^{(P,V^*)}(x)\right)$ and $(P,V)(x)$ are identically distributed.*

Then for any $x \in L_{R_{YES}}$:

$$\begin{aligned}\left\|S\left(x, V^*, (P,V)(x)\right) - \mathsf{view}_{V^*}^{(P,V^*)}(x)\right\| \\ = E_{y\leftarrow(P,V)(x)}\left[\left\|\left[S\left(x, V^*, (P,V)(x)\right) \mid y\right] - \left[\mathsf{view}_{V^*}^{(P,V^*)}(x) \mid y\right]\right\|\right]\end{aligned}$$

Proof. Let $x \in L_{R_{YES}}$. Denote

$$\begin{aligned} A &:= S\left(x, V^*, (P,V)(x)\right) \\ B &:= \mathsf{view}_{V^*}^{(P,V^*)}(x) \\ C &:= y_x \end{aligned}$$

We have:

$$\begin{aligned} \|A - B\| &= \sum_t \left|\Pr[A = t] - \Pr[B = t]\right| \\ &= \sum_{y \in supp(y_x)} \Pr[y_x = y] \sum_{t:\mathsf{sol}(t)=y} \left| \frac{\Pr[A=t]}{\Pr[y_x = y]} - \frac{\Pr[B=t]}{\Pr[y_x=y]} \right| \qquad (1) \end{aligned}$$

Now, if $\mathsf{sol}(t) = y$ and $A = t$ then

$$\mathsf{sol}(A) = y$$

and therefore, recalling that $\mathsf{sol}(A)$ is distributed exactly like y_x, we get

$$\begin{aligned} \frac{\Pr[A=t]}{\Pr[y_x=y]} &= \frac{\Pr[A = t \wedge \mathsf{sol}(A) = y]}{\Pr[\mathsf{sol}(A) = y]} \\ &= \Pr[A = t \mid \mathsf{sol}(A) = y] \\ &= \Pr[[A \mid y] = t] \end{aligned}$$

In a very similar manner we get

$$\frac{\Pr[B=t]}{\Pr[y_x=y]} = \Pr[[B \mid y] = t]$$

Plugging in 1 we get

$$\begin{aligned} \|A - B\| &= \sum_y \Pr[y_x = y] \sum_{t:\mathsf{sol}(t)=y} \left|\Pr[[A \mid y] = t] - \Pr[[B \mid y] = t]\right| \\ &= E_{y \leftarrow (P,V)(x)}\left[\|[A \mid y] - [B \mid y]\|\right] \end{aligned}$$

as required.

□

We now turn to prove Theorem 39.

Proof. Let $x \in L_{R_{YES}}$. The first bullet in the theorem statement implies in particular that $\mathsf{sol}\left(S\left(x, V, (P,V)(x)\right)\right)$ is distributed exactly like $(P,V)(x)$. Furthermore, by definition $\mathsf{sol}\left(\mathsf{view}_V^{(P,V)}(x)\right)$ is also distributed exactly like $(P,V)(x)$. Hence, by Lemma 40,

$$\begin{aligned} &\left\| S\left(x, V, (P,V)(x)\right) - \mathsf{view}_V^{(P,V)}(x) \right\| \\ &\qquad = E_{y \leftarrow (P,V)(x)}\left[\left\| \left[S\left(x, V, (P,V)(x)\right) \mid y\right] - \left[\mathsf{view}_V^{(P,V)}(x) \mid y\right] \right\| \right] \qquad (2) \end{aligned}$$

Let $y = y(x)$ be any string and denote by P_y the prover that sends y in the first message (a specific deterministic y, in contrast to a sample from $(P, V)(x)$ as in the original protocol) and then acts exactly as P in the rest of the protocol. Clearly it holds that

$$\left[\mathsf{view}_V^{(P,V)}(x) \mid y\right] = \mathsf{view}_V^{(P_y,V)}(x)$$

Furthermore, the first bullet in the theorem statement implies that

$$[S(x, V, (P, V)(x)) \mid y] = S(x, V, y)$$

Plugging into (2) we get

$$\left\|S(x, V, (P, V)(x)) - \mathsf{view}_V^{(P,V)}(x)\right\| = E_{y \leftarrow (P,V)(x)}\left[\left\|S(x, V, y) - \mathsf{view}_V^{(P_y,V)}(x)\right\|\right]$$

Denote

$$y^x := \arg \min_{y \in supp(P,V)(x)} \left\|S(x, V, y) - \mathsf{view}_V^{(P_y,V)}(x)\right\|$$

Now, since $\left\|S(x, V, (P, V)(x)) - \mathsf{view}_V^{(P,V)}(x)\right\|$ is negligible then by an averaging argument so is $\left\|[S(x, V, y^x)] - \mathsf{view}_V^{(P_{y^x},V)}(x)\right\|$. Consider the new protocol (P_{y^x}, V, S). We have shown that it is zero-knowledge against the honest verifier. Furthermore, the soundness and perfect completeness of the original protocol clearly imply the soundness and perfect completeness of the new protocol. Thus (P_{y^x}, V, S) is a public-coin $Search - HV - SZK$ protocol for $\mathcal{R}$ that outputs a unique solution for any yes instance. Hence, by (the variant of) Lemma 23, $\mathcal{R} \in Prefix - HV - SZK$. $\square$

5 Pseudo-deterministic Zero-Knowledge Protocols

Pseudo-deterministic protocols for search problems were introduced by [19]. Loosely speaking, these are protocols for search problems between an efficient verifier and a computationally unbounded prover that output some predetermined canonical solution with high probability. Moreover, the soundness condition in these protocols requires that no malicious prover can cause the verifier to output a solution other than the canonical one but with small probability. Consequently these protocols can be repeated to reduce the soundness error, potentially making them more suitable for applications. In this section we define pseudo-deterministic zero knowledge protocols for search problems, where the verifier is guaranteed not to learn anything else apart from the canonical solution. We show how these protocols relate to previous notions introduced in this work and give an example of such a protocol for the $Find{-}Isomorphic{-}Vertex$ problem that was introduced in a previous section. We start with the definition of pseudo-deterministic protocols as given in [19], with our usual modification of requiring perfect completeness.

Definition 41 (The class $PSD - IP$)**.** We say that the promise search problem $\mathcal{R} = (R_{YES}, L_{NO}) \in PSD - IP$ if there is a triplet (P, V, c) where V is a PPTM, P is computationally unbounded and $c : L_{R_{YES}} \to \{0,1\}^*$ is a function such that:

- "Completeness": For any $x \in L_{R_{YES}}$, $\Pr\left[(P,V)(x) = c(x)\right] = 1$.
- "Soundness":
  - For any $x \in L_{NO}$ and any P^*, $\Pr\left[(P^*,V)(x) = \perp\right] \geq \frac{1}{2}$.
  - For any $x \in L_{R_{YES}}$ and any P^*, $\Pr\left[(P^*,V)(x) \in \{c(x), \perp\}\right] \geq \frac{1}{2}$.

Remark 42. $PSD - IP \subseteq Search - IP$.

We turn to our definition of pseudo-deterministic zero knowledge protocols.

Definition 43 (The class $PSD - ZK$)**.** We say that the promise search problem $\mathcal{R} = (R_{YES}, L_{NO}) \in PSD - ZK$ if there is a tuple (P, V, S, c) such that:

- (P, V, c) is a $PSD - IP$ protocol for $\mathcal{R}$.
- (P, V, S) is a $Search - ZK$ protocol for $\mathcal{R}$.

Remark 44. We define the class $PSD - HV - ZK$ analagously.

Note that a $PSD-IP$ protocol (P, V) satisifes $|supp\left((P,V)(x)\right)| = 1$ for any yes instance x. Thus, by Lemma 23 we have:

Lemma 45. *$PSD - ZK \subseteq Prefix - ZK$.*

There is more to be said regarding the inclusion of Lemma 45. Recall that in the proof of Lemma 23, the protocol for the promise problem $\widehat{\mathcal{L}} = \left(\widehat{L}_{YES}, \widehat{L}_{NO}\right)$ from the definition of $Prefix - ZK$ consisted of, given input (x, y), running the given $Search - ZK$ protocol on x and accepting if its output was y. It was important for the soundness argument that $\widehat{L}_{NO}$ contained no element $(x', y') \in R_{YES}$, since otherwise, some other prover could have, in principle, caused the verifer to output y' when running the given $Search - ZK$ protocol on x', causing the protocol to accept a no instance with high probability. In the case that the given $Search - ZK$ protocol is pseudo-deterministic, no prover can cause the verifier to output a solution that is not the canonical one and therefore soundness also holds when $\widehat{L}_{NO}$ contains elements $(x, y) \in R_{YES}$ such that $y \neq c(x)$.

Definition 46 (The class $Single - Prefix - ZK$)**.** We say that a promise search problem $\mathcal{R} = (R_{YES}, L_{NO}) \in Single - Prefix - ZK$ if there exists a promise decision problem $\widehat{\mathcal{L}} = \left(\widehat{L}_{YES}, \widehat{L}_{NO}\right) \in ZK$ such that

- $\widehat{L}_{YES} \subseteq R_{YES}$.
- $\widehat{L}_{NO} = \left((L_{R_{YES}} \cup L_{NO}) \times \{0,1\}^*\right) \backslash L_{YES}$.

- For every $x \in L_{R_{YES}}$ there is a **unique** $y \in R_{YES}(x)$ such that $(x, y) \in \widehat{L}_{YES}$.
- $\widehat{\mathcal{L}}_{YES}$ is polynomially bounded - there is some polynomial $p(\cdot)$ such that if $(x, y) \in \widehat{L}_{YES}$ then $|y| \leq p(|x|)$.

Remark 47. We define $Single - Prefix - HV - ZK$ accordingly.

The discussion above shows that $PSD - ZK \subseteq Single - Prefix - ZK$. In fact, the reverse inclusion also holds and our proof of this fact is essentially identical to the proof of 18 (the only difference is that in the soundness argument here, one only needs to consider the case that $y \neq c(x)$) and therefore we choose to omit it. Finally, we note that the entire discussion above also holds when only considering honest verifier zero knowledge classes:

Theorem 48.

- *$PSD - ZK = Single - Prefix - ZK$.*
- *$PSD - HV - ZK = Single - Prefix - HV - ZK$.*

5.1 A Pseudo-Deterministic Statistical Zero Knowledge Protocol for $Find - Isomorphic - Vertex$

Recall the decision problem $Isomorphic - Vertex$ which is the language of all the tuples (G_1, G_2, v_1, v_2) where G_1, G_2 are graphs on n vertices, v_1, v_2 are some vertices in G_1, G_2 respsectively and there is some isomorphism between G_1 and G_2 that matches v_1 to v_2. We have shown previously that $Find - Isomorphic - Vertex \in Search - PZK$ and $Isomorphic - Vertex \in PZK$. The complement $\overline{Isomorphic - Vertex}$ is the language of all tuples (G_1, G_2, v_1, v_2) where either G_1, G_2 are not isomorphic or they are but v_1, v_2 are not.

Lemma 49. *$\overline{Isomorphic - Vertex} \in HV - PZK$.*

Proof. The following protocol is a mild modification of the classic interactive proof for the $Graph - Non - Isomorphism$ problem from [12]: on input (G_1, G_2, v_1, v_2):

P		V
		randomly choose $j \in \{1, 2\}$ choose a random permutation $\varphi : [n] \to [n]$
	$\xleftarrow{G' = \varphi(G_j),\, v' = \varphi(v_j)}$	
	$\xrightarrow{j'}$	
		accept iff $j = j'$

If G_1 and G_2 are not isomorphic then G' can be isomorphic to only one of the input graphs, and P can determine which one since he is computationally unbounded. Similarly he can find out j from G', v' in the case that G_1, G_2 are isormorphic but v_1 and v_2 are not. Thus we have shown that the above protocol has perfect completeness. If, on the other hand, G_1, G_2 are isomorphic and v_1, v_2 are also isomorphic, then the pair G', v' does not contain any information about j, implying that P can not do better than guessing. Hence the protocol has soundness error $\frac{1}{2}$. To show honest verifier perfect zero knowledge, the simulator $S\left(\left(G_1, G_2, v_1, v_2\right), V\right)$ randomly chooses j, a random perumatation φ, and outputs

$$\left(\left(G_1, G_2, v_1, v_2\right), j, \varphi, \varphi\left(G_j\right), \varphi\left(v_j\right), j\right)$$

and it is clear that this output is identical to the view of the honest verifier in the protocol when interacting with the honest prover on a yes instance. □

In the next theorem we give an example of a pseudo-deterministic honest verifier perfect zero knowledge protocol for $Find - Isomorphic - Vertex$. Intuitively, the solution to the yes instance (G_1, G_2, v_1) will be the lexicographically first vertex in G_2 that is isomorphic to v_1, and the protocol will contain as sub-protocols the proofs of the facts that the solution vertex is indeed isomorphic to v_1 and that all the vertices that are lexicographically smaller than the solution are not isomorphic to v_1. We will see that the fact that each sub-protocol is honest verifier perfect zero knowledge implies that the entire protocol is honest verifier perfect zero knowledge.

Theorem 50. *$Find - Isomorphic - Vertex \in PSD - HV - PZK$*

Proof. Let (P_I, V_I, S_I) and $\left(\overline{P_I}, \overline{V_I}, \overline{S_I}\right)$ be the $HV - PZK$ protocols for $Isomorphic - Vertex$ and $\overline{Isomorphic - Vertex}$ respectively. We construct the following protocol for $Find - Isomorphic - Vertex$: on input (G_1, G_2, v_1), where $\{u_1, \ldots, u_n\}$ is the lexicographical ordering of the vertices of G_2:

P		V
Determine u_k, the lexicographically first vertex in G_2 that is isomorphic to v_1.		
	$\xrightarrow{u_k}$	
	$\left(\overline{P_I}, \overline{V_I}\right)(G_1, G_2, v_1, u_1)$	
	$\vdots$	
	$\left(\overline{P_I}, \overline{V_I}\right)(G_1, G_2, v_1, u_{k-1})$	
	$(P_I, V_I)(G_1, G_2, v_1, u_k)$	
		output u_k if all the sub-protocols succeeded and otherwise output $\perp$.

We will now prove that the protocol above is indeed a $PSD - HV - PZK$ protocol. Perfect completeness follows from the perfect completeness of the protocols (P_I, V_I) and

$\left(\overline{P_I}, \overline{V_I}\right)$. As for soundness: if (G_1, G_2, v_1) is a no instance then G_1 and G_2 are not isomorphic, in which case (G_1, G_2, v_1, u_k) is a no instance of $Isomorphic - Vertex$ implying that the last sub-protocol fails w.p at least $\frac{1}{2}$. If (G_1, G_2, v_1) is a yes instance but u_k is not the lexicographically first vertex in G_2 that is isomorphic to v_1 then clearly at least one of the inputs of the sub-protocols is a no instance of the respective sub-protocol, and soundness again follows from the soundness of that sub-protocol. It is left to prove honest verifier perfect zero knowledge. Formally, we defined the view of the honest verifier as the concatenation of the shared input, the verifier's randomness and all messages exchanged. Note that for the protocol above, this view is exactly

$$\left(\mathsf{view}_{\overline{V}_I}^{\left(\overline{P_I},\overline{V_I}\right)}\left(G_1, G_2, v_1, u_1\right), \ldots, \mathsf{view}_{\overline{V}_I}^{\left(\overline{P_I},\overline{V_I}\right)}\left(G_1, G_2, v_1, u_{k-1}\right), \mathsf{view}_{V_I}^{(P_I,V_I)}\left(G_1, G_2, v_1, u_k\right)\right)$$

up to some permutation of the locations within the tuple. We will treat the tuple above as the real view to facilitate the presentation. We construct the simulator S, that on input (G_1, G_2, v_1, V, u_k) should produce a tuple which is identical to the above. Not suprisingly, S simply outputs:

$$\left(\overline{S_I}\left(G_1, G_2, v_1, u_1, \overline{V_I}\right), \ldots, \overline{S_I}\left(G_1, G_2, v_1, u_{k-1}, \overline{V_I}\right), S_I\left(G_1, G_2, v_1, u_k, V_I\right)\right)$$

Now, each sub-view is independent from the rest and each sub-simulator output is independent from the rest. Hence we can use Fact 2.3 from [24] to conclude that

$$\begin{aligned}
&\left\|\mathsf{view}_V^{(P,V)}\left(G_1, G_2, v_1\right) - S\left(\left(G_1, G_2, v_1, V, u_k\right)\right)\right\| \\
&\qquad\qquad \le \sum_{i=1}^{k-1}\left\|\mathsf{view}_{\overline{V}_I}^{\left(\overline{P_I},\overline{V_I}\right)}\left(G_1, G_2, v_1, u_i\right) - \overline{S_I}\left(G_1, G_2, v_1, u_i, \overline{V_I}\right)\right\| \\
&\qquad\qquad\qquad + \left\|\mathsf{view}_{V_I}^{(P_I,V_I)}\left(G_1, G_2, v_1, u_k\right) - S_I\left(G_1, G_2, v_1, u_k, V_I\right)\right\| = 0
\end{aligned}$$

as required. □

In [14] it was proved that any honest verifier statistical zero knowledge proof can be transformed into a statistical zero knowledge proof (that is zero knowledge against any possible verifier), i.e. $HV - SZK = SZK$. In particular, the characterization $PSD - HV - PZK = Single - Prefix - HV - PZK$ implies that $PSD - HV - PZK \subseteq PSD - SZK$, giving us:

Theorem 51. *$Find - Isomorphic - Vertex \in PSD - SZK$.*

Acknowledgements. We thank Ofer Grossman and Oded Goldreich for helpful discussions.

References

[1] Amos Beimel, Paz Carmi, Kobbi Nissim, and Enav Weinreb. Private approximation of search problems. In Jon M. Kleinberg, editor, *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 119–128. ACM, 2006.

[2] Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In Shafi Goldwasser, editor, *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, volume 403 of *Lecture Notes in Computer Science*, pages 37–56. Springer, 1988.

[3] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10, 1988.

[4] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 11–19, 1988.

[5] Ivan Damgård, Oded Goldreich, Tatsuaki Okamoto, and Avi Wigderson. Honest verifier vs dishonest verifier in public coin zero-knowledge proofs. In Don Coppersmith, editor, *Advances in Cryptology - CRYPTO '95, 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27-31, 1995, Proceedings*, volume 963 of *Lecture Notes in Computer Science*, pages 325–338. Springer, 1995.

[6] Joan Feigenbaum, Yuval Ishai, Tal Malkin, Kobbi Nissim, Martin Strauss, and Rebecca N. Wright. Secure multiparty computation of approximations. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *Automata, Languages and Programming, 28th International Colloquium, ICALP 2001, Crete, Greece, July 8-12, 2001, Proceedings*, volume 2076 of *Lecture Notes in Computer Science*, pages 927–938. Springer, 2001.

[7] Eran Gat and Shafi Goldwasser. Probabilistic search algorithms with unique answers and their cryptographic applications. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:136, 2011.

[8] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques.* Cambridge University Press, 2001.

[9] Oded Goldreich. *Computational complexity - a conceptual perspective.* Cambridge University Press, 2008.

[10] Oded Goldreich, Shafi Goldwasser, and Dana Ron. On the possibilities and limitations of pseudodeterministic algorithms. In Robert D. Kleinberg, editor, *Innovations in*

Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013, pages 127–138. ACM, 2013.

[11] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

[12] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991.

[13] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *J. Cryptology*, 7(1):1–32, 1994.

[14] Oded Goldreich, Amit Sahai, and Salil P. Vadhan. Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In Jeffrey Scott Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 399–408. ACM, 1998.

[15] Oded Goldreich, Amit Sahai, and Salil P. Vadhan. Can statistical zero knowledge be made non-interactive? or on the relationship of SZK and NISZK. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 467–484. Springer, 1999.

[16] Oded Goldreich and Salil P. Vadhan. Comparing entropies in statistical zero knowledge with applications to the structure of SZK. In *Proceedings of the 14th Annual IEEE Conference on Computational Complexity, Atlanta, Georgia, USA, May 4-6, 1999*, page 54. IEEE Computer Society, 1999.

[17] Shafi Goldwasser and Ofer Grossman. Perfect bipartite matching in pseudo-deterministic RNC. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:208, 2015.

[18] Shafi Goldwasser and Ofer Grossman. Bipartite perfect matching in pseudo-deterministic NC. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 87:1–87:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[19] Shafi Goldwasser, Ofer Grossman, and Dhiraj Holden. Pseudo-deterministic proofs. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 17:1–17:18. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

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

[21] Ofer Grossman. Finding primitive roots pseudo-deterministically. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:207, 2015.

[22] Shai Halevi, Robert Krauthgamer, Eyal Kushilevitz, and Kobbi Nissim. Private approximation of np-hard functions. In Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis, editors, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 550–559. ACM, 2001.

[23] Tatsuaki Okamoto. On relationships between statistical zero-knowledge proofs. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 649–658. ACM, 1996.

[24] Amit Sahai and Salil P. Vadhan. A complete problem for statistical zero knowledge. *J. ACM*, 50(2):196–249, 2003.

[25] Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992.

[26] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164, 1982.