Cryptographic Characterization of Quantum Advantage

Tomoyuki Morimae[1], Yuki Shirakawa[1], and Takashi Yamakawa[2,3,1]

[1]Yukawa Institute for Theoretical Physics, Kyoto University, Kyoto, Japan
tomoyuki.morimae@yukawa.kyoto-u.ac.jp
yuki.shirakawa@yukawa.kyoto-u.ac.jp
[2]NTT Social Informatics Laboratories, Tokyo, Japan
takashi.yamakawa@ntt.com
[3]NTT Research Center for Theoretical Quantum Information, Atsugi, Japan

Abstract

Quantum computational advantage refers to an existence of computational tasks that are easy for quantum computing but hard for classical one. Unconditionally showing quantum advantage is beyond our current understanding of complexity theory, and therefore some computational assumptions are needed. Which complexity assumption is necessary and sufficient for quantum advantage? In this paper, we show that inefficient-verifier proofs of quantumness (IV-PoQ) exist if and only if classically-secure one-way puzzles (OWPuzzs) exist. As far as we know, this is the first time that a complete cryptographic characterization of quantum advantage is obtained. IV-PoQ [Morimae and Yamakawa 2024] are a variant of proofs of quantumness (PoQ) [Brakerski et al. 2021] where the verifier is efficient during the interaction but may use unbounded time afterward. IV-PoQ capture various types of quantum advantage previously studied, such as sampling advantage and searching advantage. Previous work [Morimae and Yamakawa 2024] showed that IV-PoQ can be constructed from OWFs, but a construction of IV-PoQ from weaker assumptions was left open. Our result solves the open problem, because OWPuzzs are believed to be weaker than OWFs. OWPuzzs [Khurana and Tomer 2024] are one of the most fundamental quantum cryptographic primitives implied by many quantum cryptographic primitives weaker than one-way functions (OWFs), such as pseudorandom unitaries (PRUs), pseudorandom state generators (PRSGs), and one-way state generators (OWSGs). The equivalence between IV-PoQ and classically-secure OWPuzzs therefore highlights that if there is no quantum advantage, then these fundamental primitives do not exist. The equivalence also means that quantum advantage is an example of the applications of OWPuzzs. Except for commitments, no application of OWPuzzs was known before. Our result shows that quantum advantage is another application of OWPuzzs. Moreover, it is the first quantum computation classical communication (QCCC) application of OWPuzzs. To show the main result, we introduce several new concepts and show some results that will be of independent interest. In particular, we introduce an interactive (and average-case) version of sampling problems where the task is to sample the transcript obtained by a classical interaction between two quantum polynomial-time algorithms. We show that quantum advantage in interactive sampling problems is equivalent to the existence of IV-PoQ, which is considered as an interactive (and average-case) version of Aaronson's result [Aaronson 2014], $\mathbf{SampBQP} \neq \mathbf{SampBPP} \Leftrightarrow \mathbf{FBQP} \neq \mathbf{FBPP}$. Finally, we also introduce zero-knowledge IV-PoQ and study sufficient and necessary conditions for their existence.

Contents

1 Introduction

Quantum computational advantage refers to the existence of computational tasks that are easy for quantum computing but hard for classical one. Unconditionally showing quantum advantage is extremely hard, and is beyond our current understanding of complexity theory.[1] Some computational assumptions are therefore required. Which complexity assumption is necessary and sufficient for quantum advantage? As far as we know, no complete characterization of quantum advantage was obtained before.

In this paper, we identify a cryptographic assumption that is necessary and sufficient for quantum advantage. Our main result is the following one:[2][3]

Theorem 1.1. *Inefficient-verifier proofs of quantumness (IV-PoQ) exist if and only if classically-secure one-way puzzles (OWPuzzs) exist.*

As far as we know, this is the first time that a complete cryptographic characterization of quantum advantage is obtained.

What are IV-PoQ? IV-PoQ are a generalization of proofs of quantumness (PoQ) [BCM$^+$21]. A PoQ is an interactive protocol between a prover and a classical probabilistic polynomial-time (PPT) verifier over a classical channel. There exists a quantum polynomial-time (QPT) prover such that the verifier accepts with high probability (*completeness*), but for any PPT prover the verifier rejects with high probability (*soundness*). PoQ can be constructed from several cryptographic assumptions, such as (noisy) trapdoor claw-free functions with the adaptive-hardcore-bit property [BCM$^+$21], trapdoor 2-to-1 collision-resistant hash functions [KMCVY22], (full-domain) trapdoor permutations [MY23], quantum homomorphic encryptions [KLVY23], or knowledge assumptions [AGGM24]. Non-interactive PoQ are possible based on the hardness of factoring [Sho94] or random oracles [YZ24].

IV-PoQ [MY24] are the same as PoQ except that the verifier's final computation to make the decision can be unbounded. IV-PoQ are a generalization of PoQ, and as we will explain later, IV-PoQ capture various types of quantum advantage studied so far, such as sampling advantage and searching advantage.

Completely identifying a necessary and sufficient assumption for the existence of (IV-)PoQ remained open. In particular, in [MY24], IV-PoQ were constructed from classically-secure OWFs, but the problem of constructing IV-PoQ from weaker assumptions was left open in that paper. Our main result Theorem 1.1 solves the open problem, because as we will explain later, OWPuzzs are believed to be weaker than OWFs [Kre21, MY22, KT24a]. Moreover, a known necessary condition for the existence of IV-PoQ was only the almost trivial one: $\mathbf{BPP} \neq \mathbf{PP}$[4] [MY24]. Our main result Theorem 1.1 improves this to a highly non-trivial necessary condition, namely, the existence of classically-secure OWPuzzs.[5]

What are OWPuzzs? In classical cryptography, the existence of OWFs is the minimum assumption [IL89], because many primitives exist if and only if OWFs exist, such as pseudorandom generators (PRGs),

[1] There are several interesting results (such as [BGK18]) that show unconditional quantum advantage by restricting classical computing. In this paper, we consider any polynomial-time classical computing.

[2] In this paper, all classically-secure OWPuzzs are ones with $(1 - \mathsf{negl}(\lambda))$-correctness and $(1 - 1/\mathrm{poly}(\lambda))$-security. Unlike quantumly-secure OWPuzzs, we do not know how to amplify the gap for classically-secure OWPuzzs.

[3] In this paper, we consider the uniform adversarial model (i.e., adversaries are modeled as Turing machines), and some steps of the proofs of Theorem 1.1 crucially rely on the uniformity of the adversary. (See Appendix A for the detail.) We leave it open to prove (or disprove) the non-uniform variant of Theorem 1.1.

[4] The output probability distribution of any QPT algorithm can be computed by a classical polynomial-time deterministic algorithm that queries the $\mathbf{PP}$ oracle [FR99]. Therefore, if $\mathbf{BPP} = \mathbf{PP}$, a PPT prover can cheat the verifier.

[5] This is an improvement, because if classically-secure OWPuzzs exist then $\mathbf{BPP} \neq \mathbf{PP}$.

pseudorandom functions (PRFs), zero-knowledge, commitments, digital signatures, and secret-key encryptions (SKE), and almost all primitives imply OWFs. On the other hand, recent active studies have revealed that in quantum cryptography OWFs are not necessarily the minimum assumption. Many fundamental primitives have been introduced, such as pseudorandom unitaries (PRUs) [JLS18], pseudorandom function-like state generators (PRFSGs) [AQY22], unpredictable state generators (UPSGs) [MYY24], pseudorandom state generators (PRSGs) [JLS18], one-way state generators (OWSGs) [MY22], EFI pairs [BCQ23], and one-way puzzles (OWPuzzs) [KT24a]. They could exist even if OWFs do not exist [Kre21, KQST23, LMW24], but still imply several useful applications such as message authentication codes [AQY22], commitments [MY22, AQY22], multi-party computations [MY22, AQY22], secret-key encryptions [AQY22], private-key quantum money [JLS18], digital signatures [MY22], etc.

In particular, one-way puzzles (OWPuzzs) are one of the most fundamental primitives in this "cryptographic world below OWFs". A OWPuzz is a pair $(\mathsf{Samp}, \mathsf{Ver})$ of two algorithms. Samp is a QPT algorithm that takes 1^λ as input and outputs two classical bit strings puzz and ans. Ver is an unbounded algorithm that takes puzz and ans' as input, and outputs $\top$ or $\bot$. We require two properties, correctness and security. Correctness requires that Ver accepts $(\mathsf{puzz}, \mathsf{ans})$ sampled by Samp with large probability. Security requires that for any QPT algorithm $\mathcal{A}$ that takes puzz as input and outputs ans', Ver accepts $(\mathsf{puzz}, \mathsf{ans}')$ with only small probability. In particular, when the security is required only for all PPT adversaries, we say that a OWPuzz is classically-secure. (Note that the Samp algorithm of classically-secure OWPuzzs is QPT, not PPT, even when we consider classical security.)

As is shown in Figure 1, OWPuzzs (and therefore classically-secure OWPuzzs) are implied by many primitives such as

- PRUs, PRFSGs, UPSGs, PRSGs, (pure) OWSGs,
- (pure) private-key quantum money, secret-key encryption schemes, digital signatures,
- several quantum computation classical communication (QCCC) primitives [CGG24a, KT24a],
- quantum EFID pairs.[6]

Our Theorem 1.1 therefore highlights that if there is no quantum advantage, then all of these quantum cryptographic primitives do not exist.

On the other hand, although many primitives imply OWPuzzs, no application of OWPuzzs is known except for commitments [KT24a]. Finding more applications of OWPuzzs is one of the most important goals in this field. Our result Theorem 1.1 shows that OWPuzzs imply quantum advantage, which demonstrates that quantum advantage is another application of OWPuzzs. Moreover, we emphasize that this is the first application of OWPuzzs in the QCCC setting: IV-PoQ are a QCCC primitive because the communication between the verifier and the prover is classical, while commitments [KT24a] constructed from OWPuzzs are those over quantum channels.

Why IV-PoQ? In addition to (IV-)PoQ, there are mainly two other approaches to demonstrate quantum advantage, namely searching and sampling. Here we argue that IV-PoQ capture both of them, and therefore identifying a necessary and sufficient assumption for the existence of IV-PoQ is significant.

[6]It is a pair (G_0, G_1) of two QPT algorithms that output classical bit strings such that the output distributions are statistically far but computationally indistinguishable. EFID pairs imply OWPuzzs by defining $(\mathsf{Samp}, \mathsf{Ver})$ as follows. Samp : On input 1^λ, choose $(b_1, ..., b_{\ell(\lambda)}) \leftarrow \{0,1\}^{\ell(\lambda)}$ and set $\mathsf{ans} := (b_1, ..., b_{\ell(\lambda)})$, where ℓ is a certain polynomial. Run $x_{b_i} \leftarrow G_{b_i}(1^\lambda)$ for each $i \in [\ell(\lambda)]$. Set $\mathsf{puzz} := (x_{b_1}, ..., x_{b_\ell})$. Ver : Given $\mathsf{puzz} = (x_{b_1}, ..., x_{b_\ell})$ and $\mathsf{ans}' = (b'_1, ..., b'_\ell)$, output $\top$ if and only if x_{b_i} is in the set of all outputs of $G_{b'_i}(1^\lambda)$ for each $i \in [\ell(\lambda)]$.

A sampling problem is a task of sampling from some distributions. There are several distributions that are easy to sample with QPT algorithms but hard with PPT algorithms, such as output ditributions of random quantum circuits [BFNV19], Boson Sampling circuits [AA11], constant-depth circuits [TD04], IQP circuits [BJS11, BMS16], and one-clean-qubit circuits [FKM^{+}18]. Several assumptions are known to be sufficient for quantum advantage in sampling problems, but these assumptions are newly-introduced assumptions that were not studied before such as an average-case $\#\mathbf{P}$-hardness of approximating some functions. Moreover, quantum advantage in sampling problems is in general not known to be verifiable (even inefficiently). On the other hand, one advantage of sampling problems (and other problems relying on newly-introduced assumptions) is that experimental realizations with NISQ machines seem to be easier.[7] As we will explain later, we introduce an average-case version of $\mathbf{SampBQP} \neq \mathbf{SampBPP}$, and show that such an average-case version of sampling advantage is equivalent to the existence of non-interactive IV-PoQ. IV-PoQ therefore capture sampling advantage.

A search problem is a task of finding an element z that satisfies a relation $R(z) = 1$. Several search problems have been shown to be easy for QPT algorithms but hard for PPT algorithms. Their classical hardness, however, relies on newly-introduced assumptions that were not studied before, such as QUATH [AC17] and XQUATH [AG19], or relies on random oracles [Aar10, ACC^{+}23]. One advantage of these search problems over sampling problems is that quantum advantage can be verified at least inefficiently when R is computable. (We can check $R(z) = 1$ or not by computing $R(z)$.) Because such inefficiently-verifiable search advantage is equivalent to the existence of non-interactive IV-PoQ, IV-PoQ capture inefficiently-verifiable search problems. There are some search problems that are efficiently verifiable such as Factoring [Sho94] and Yamakawa-Zhandry problem [YZ24] but the former is based on the hardness of a specific problem, and the latter relies on the random oracle model. Quantum advantage in efficiently-verifiable search problems is captured by non-interactive PoQ, and therefore by IV-PoQ.

Summary. In summary, we have shown that IV-PoQ exist if and only if classically-secure OWPuzzs exist. We believe that this result is significant mainly because of the following five reasons.

1. As far as we know, this is the first time to give a complete cryptographic characterization of quantum advantage.

2. IV-PoQ capture various types of quantum advantage studied so far including sampling advantage and searching advantage.

3. The previous result [MY24] constructed IV-PoQ from classically-secure OWFs, but the problem of constructing IV-PoQ from weaker assumptions was left open. We solve the open problem.

4. OWPuzzs are implied by many important primitives, such as PRUs, PRSGs, and OWSGs. Therefore, our main result shows that if there is no quantum advantage, then these quantum cryptographic primitives do not exist.

5. No application of OWPuzzs was known before except for commitments (and therefore multiparty computations). We show that quantum advantage is another application of OWPuzzs. Moreover, it is the first QCCC application of OWPuzzs.

[7]Although NISQ experimental realizations of quantum advantage are very important goals, in this paper, we focus on theoretical upper and lower bounds by assuming that any polynomial-time quantum computing is possible.

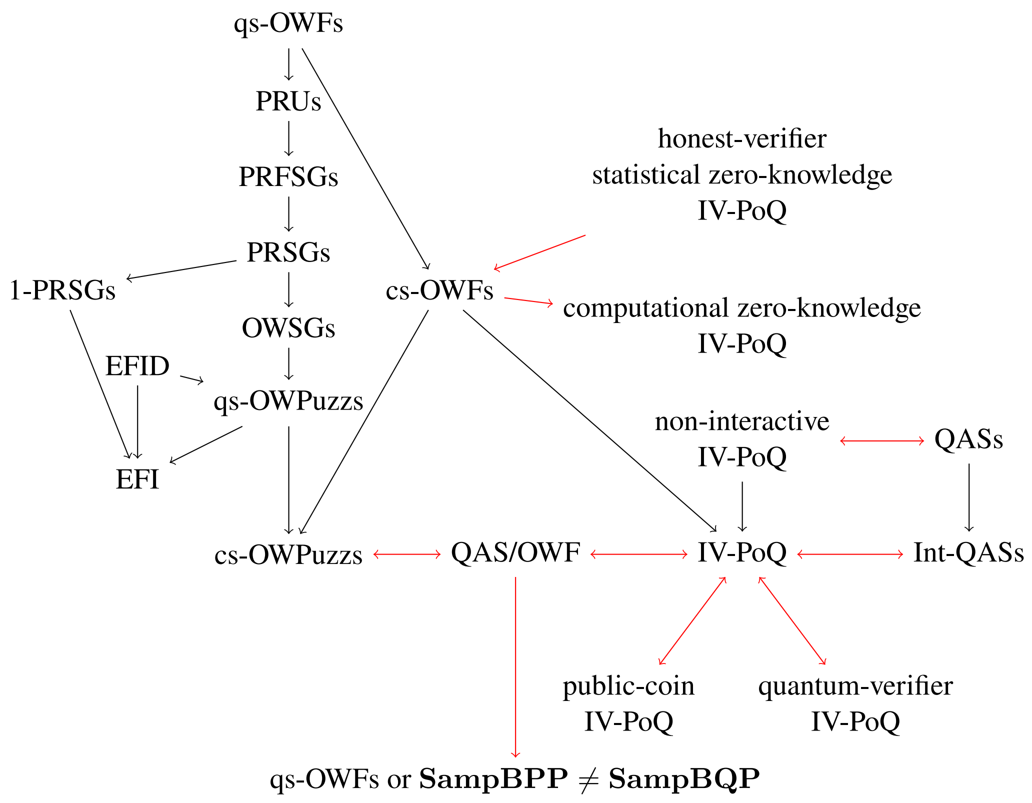

Figure 1: Summary of results. Black arrows are known or trivial implications. Red arrows are our results. “qs” stands for quantumly-secure and “cs” stands for classically-secure. 1-PRSGs are single-copy-secure PRSGs [MY22].

1.1 Additional Results

In addition to the main result, Theorem 1.1, we obtain several important results. In the following, we explain them. All our results are also summarized in Figure 1.

Relations to the sampling complexity. We show the following result.[8]

Theorem 1.2. *If IV-PoQ exist, then quantumly-secure OWFs exist or* $\mathbf{SampBPP} \neq \mathbf{SampBQP}$.

Because the existence of quantumly-secure OWFs implies $\mathbf{NP} \not\subseteq \mathbf{BQP}$, Theorem 1.2 also means that if IV-PoQ exist, then $\mathbf{NP} \not\subseteq \mathbf{BQP}$ or $\mathbf{SampBPP} \neq \mathbf{SampBQP}$. This characterizes a lower bound of IV-PoQ in terms of worst-case complexity class assumptions. Note that this lower bound improves the previous known bound, $\mathbf{BPP} \neq \mathbf{PP}$, of [MY24].[9]

[8]Note that “quantumly-secure” in the theorem is not a typo. The reason why we can get quantumly-secure OWFs from classically-secure OWPuzzs is, roughly speaking, that if $\mathbf{SampBPP} = \mathbf{SampBQP}$, then classically-secure OWFs means quantumly-secure OWFs. For details, see Section 4.

[9]First, $\mathbf{NP} \not\subseteq \mathbf{BQP}$ implies $\mathbf{BPP} \neq \mathbf{PP}$, because if $\mathbf{BPP} = \mathbf{PP}$, then $\mathbf{NP} \subseteq \mathbf{PP} = \mathbf{BPP} \subseteq \mathbf{BQP}$. Second, $\mathbf{SampBPP} \neq \mathbf{SampBQP}$ implies $\mathbf{BPP} \neq \mathbf{PP}$, because a classical deterministic polynomia-time algorithm that queries the $\mathbf{PP}$ oracle can compute the output distribution of any QPT algorithm [FR99].

Quantum advantage samplers (QASs). To show the main result, we introduce a new concept, which we call quantum advantage samplers (QASs). The existence of QASs is an average-case version of $\mathbf{SampBQP} \neq \mathbf{SampBPP}$.

Let $\mathcal{A}$ be a QPT algorithm that takes 1^λ as input and outputs classical bit strings. $\mathcal{A}$ is a quantum advantage sampler (QAS) if there exists a polynomial p such that for any PPT algorithm $\mathcal{B}$,

$$\mathsf{SD}(\mathcal{A}(1^\lambda), \mathcal{B}(1^\lambda)) > \frac{1}{p(\lambda)} \tag{1}$$

holds for all sufficiently large $\lambda \in \mathbb{N}$, where SD is the statistical distance, and $\mathcal{A}(1^\lambda)$ (resp. $\mathcal{B}(1^\lambda)$) is the output probability distribution of $\mathcal{A}$ (resp. $\mathcal{B}$) on input 1^λ.

Intuitively, Equation (1) means that the output distribution of the QPT algorithm $\mathcal{A}$ cannot be classically efficiently sampled. How is this different from the more studied notion, $\mathbf{SampBPP} \neq \mathbf{SampBQP}$? The existence of QASs can be considered as an average-case version of $\mathbf{SampBPP} \neq \mathbf{SampBQP}$.[10] In fact, the existence of QASs implies $\mathbf{SampBPP} \neq \mathbf{SampBQP}$ (Lemma 3.4), but the inverse does not seem to hold. In order to show our main result, the worst-case notion of $\mathbf{SampBPP} \neq \mathbf{SampBQP}$ is not enough for our cryptographic (and therefore average-case) argument, and therefore we introduce QASs. We believe that the new concept, QASs, will be useful for other future studies of quantum sampling advantage in the context of quantum cryptography.

We show the following result.

Theorem 1.3. *QASs exist if and only if non-interactive IV-PoQ exist.*

Because the existence of QASs is an average-case version of $\mathbf{SampBQP} \neq \mathbf{SampBPP}$ while the existence of non-interactive IV-PoQ is an average-case version of $\mathbf{FBQP} \neq \mathbf{FBPP}$, Theorem 1.3 can be considered as an average-case version of [Aar14]'s result $\mathbf{SampBPP} \neq \mathbf{SampBQP} \Leftrightarrow \mathbf{FBPP} \neq \mathbf{FBQP}$.

Interactive quantum advantage samplers (Int-QASs). We also introduce an interactive version of QASs, which we call interactive QASs (Int-QASs). Int-QASs are a generalization of QASs to interactive settings. An Int-QAS is a pair $(\mathcal{A}, \mathcal{C})$ of two interactive QPT algorithms $\mathcal{A}$ and $\mathcal{C}$ that communicate over a classical channel. Its security roughly says that no PPT algorithm $\mathcal{B}$ that interacts with $\mathcal{C}$ can sample the transcript of $(\mathcal{A}, \mathcal{C})$. (Here, the transcript is the sequence of all classical messages exchanged between $\mathcal{A}$ and $\mathcal{C}$.) More precisely, $(\mathcal{A}, \mathcal{C})$ is an Int-QAS if there exists a polynomial p such that for any PPT algorithm $\mathcal{B}$ that interacts with $\mathcal{C}$,

$$\mathsf{SD}(\langle \mathcal{A}, \mathcal{C} \rangle (1^\lambda), \langle \mathcal{B}, \mathcal{C} \rangle (1^\lambda)) > \frac{1}{p(\lambda)} \tag{2}$$

holds for all sufficiently large $\lambda \in \mathbb{N}$, where SD is the statistical distance, and $\langle \mathcal{A}, \mathcal{C} \rangle (1^\lambda)$ (resp. $\langle \mathcal{B}, \mathcal{C} \rangle (1^\lambda)$) is the probability distribution over the transcript of the interaction between $\mathcal{A}$ (resp. $\mathcal{B}$) and $\mathcal{C}$ on input 1^λ.

It is easy to see that IV-PoQ imply Int-QASs: for any IV-PoQ $(\mathcal{P}, \mathcal{V}_1, \mathcal{V}_2)$, where $\mathcal{P}$ is the prover, $\mathcal{V}_1$ is the efficient verifier, and $\mathcal{V}_2$ is the inefficient verifier, we have only to take $(\mathcal{A}, \mathcal{C}) = (\mathcal{P}, \mathcal{V}_1)$. However, the opposite direction is not immediately clear. We show that the opposite direction can be also shown. We hence have the following result.

[10] $\mathbf{SampBPP}$ and $\mathbf{SampBQP}$ are worst-case complexity classes, and therefore $\{\mathcal{A}(1^\lambda)\}_\lambda \notin \mathbf{SampBPP}$ means that there is at least one $\lambda \in \mathbb{N}$ such that $\mathcal{A}(1^\lambda)$ cannot be classically efficiently sampled, while the definition of QASs requires that $\{\mathcal{A}(1^\lambda)\}_\lambda$ cannot be classically efficiently sampled for all sufficiently large $\lambda \in \mathbb{N}$. In addition, there is a subtle technical difference: $\{\mathcal{A}_\lambda\}_\lambda \notin \mathbf{SampBPP}$ means that $\mathcal{A}(1^\lambda)$ cannot be classically efficiently sampled for a precision ϵ, but this ϵ could be $\mathsf{negl}(\lambda)$, while QASs requires that the precision is $1/\mathrm{poly}(\lambda)$. For more details, see Section 3.

Theorem 1.4. *Int-QASs exist if and only if IV-PoQ exist.*

This theorem is considered as an interactive (and average-case) version of $\mathbf{SampBPP} \neq \mathbf{SampBQP} \Leftrightarrow \mathbf{FBPP} \neq \mathbf{FBQP}$, because the existence of Int-QASs is an interactive (and average-case) version of $\mathbf{SampBQP} \neq \mathbf{SampBPP}$ while the existence of IV-PoQ is an interactive (and average-case) version of $\mathbf{FBQP} \neq \mathbf{FBPP}$.

QAS/OWF condition. In addition to QASs and Int-QASs, we also introduce another new concept, the QAS/OWF condition, which is inspired by the SZK/OWF condition [Vad06]. As we will explain later, the QAS/OWF condition plays a pivotal role to show our main result. Roughly speaking, the QAS/OWF condition is satisfied if there is a pair of candidates of a QAS and a classically-secure OWF such that for all sufficiently large security parameters, either of them is secure.[11] If a QAS exists or a classically-secure OWF exists, then the QAS/OWF is satisfied, but the converse is unlikely. For example, if there are (candidates of) a QAS that is secure for all odd security parameters and a OWF that is secure for all even security parameters, then the QAS/OWF is satisfied, but it does not necessarily imply either of a QAS or a OWF.

We show that the QAS/OWF condition is equivalent to both the existence of IV-PoQ and the existence of classically-secure OWPuzzs:

Theorem 1.5. *IV-PoQ exist if and only if the QAS/OWF condition holds.*

Theorem 1.6. *Classically-secure OWPuzzs exist if and only if the QAS/OWF condition holds.*

By combining these two results, we obtain our main result, Theorem 1.1.

Variants of IV-PoQ. Recall that the verifier of IV-PoQ must be PPT during the interaction. We consider the following two variants of IV-PoQ, *public-coin IV-PoQ*, where all the verifier's messages must be uniformly random strings, and *quantum-verifier IV-PoQ*, where the verifier is allowed to be QPT instead of PPT during the interaction. Clearly, public-coin IV-PoQ is a special case of IV-PoQ (since uniformy random strings can be sampled in PPT), and IV-PoQ is a special case of quantum-verifier IV-PoQ (since PPT computations can be simulated in QPT). We show implications in the other direction, making them equivalent in terms of existence.

Theorem 1.7. *The existence of public-coin IV-PoQ, IV-PoQ, and quantum-verifier IV-PoQ are equivalent.*

This theorem suggests that the power of IV-PoQ is robust to the choice of the computational power of the verifier during the interaction.

Zero-knowledge IV-PoQ. We define the zero-knowledge property for IV-PoQ, which roughly requires that the verifier's view can be simulated by a PPT simulator. Intuitively, this ensures that the verifier learns nothing from the prover beyond what could have been computed in PPT. We say that an IV-PoQ satisfies statistical (resp. computational) zero-knowledge if for any PPT malicious verifier, there is a PPT simulator that statistically (resp. computationally) simulates the verifier's view. We say that an IV-PoQ satisfies honest-verifier statistical zero-knowledge if there is a PPT simulator that statistically simulates the honest verifier's view. We prove the following results.

Theorem 1.8. *If honest-verifier statistical zero-knowledge IV-PoQ exist, then classically-secure OWFs exist.*

[11] See Definition 4.1 for the precise definition.

Theorem 1.9. *If classically-secure OWFs exist, then computational zero-knowledge IV-PoQ exist.*

The above theorems establish a loose equivalence between zero-knowledge IV-PoQ and classically-secure OWFs. However, there is still a gap between them, and filling it is left as an open problem.

1.2 Related Work

Khurana and Tomer [KT24b] have recently shown that *quantumly-secure* OWPuzzs can be constructed from some assumptions that imply sampling-based quantum advantage (if a mild complexity assumption, $\mathbf{P}^{\sharp\mathbf{P}} \not\subseteq (io)\mathbf{BQP}/\mathbf{qpoly}$, is additionally introduced). There is no technical overlap between their paper and the present paper. However, we here clarify relations and differences, because in a broad perspective, their paper and the present paper share several important motivations, including the goal of connecting quantum advantage and "Microcrypt" primitives.

Firstly, what they actually show is not that quantum advantage implies OWPuzzs, but that some assumptions that imply quantum advantage also imply OWPuzzs if the additional assumption, $\mathbf{P}^{\sharp\mathbf{P}} \not\subseteq (io)\mathbf{BQP}/\mathbf{qpoly}$, is introduced. On the other hand, we show that quantum advantage (in the sense of the existence of IV-PoQ) implies OWPuzzs. Secondly, they construct quantumly-secure OWPuzzs, while we construct only classically-secure ones.

These differences comes from the difference of main goals. Their goal is to construct quantum cryptographic primitives from some well-founded assumptions that will not imply OWFs. Therefore, the constructed primitives should be quantumly-secure. However, in that case, as they also mention in their paper, some additional assumptions that limit quantum power should be introduced, because quantum advantage limits only classical power. On the other hand, the goal of the present paper is to characterize quantum advantage from cryptographic assumptions, and therefore we have to consider quantum advantage itself, not assumptions that imply quantum advantage. Moreover, we want to avoid introducing any additional assumptions that are not related to quantum advantage. In that case, it is likely that we have to be satisfied with classically-secure OWPuzzs.

It is an interesting open problem whether several notions of quantum advantage studied in this paper imply quantumly-secure OWPuzzs (possibly introducing some additional assumptions that limit quantum power).

1.3 Technical Overview

Our main result is Theorem 1.1, which shows that IV-PoQ exist if and only if classically-secure OWPuzzs exist. In this technical overview, we provide intuitive explanations for the result. To show it, the QAS/OWF condition plays a pivotal role. For ease of presentation, we think of the QAS/OWF condition as just the condition that "a QAS exists or a classically-secure OWF exists" in this overview. Though this is stronger than the actual definition, this rough description is enough for understanding our ideas.

We first show IV-PoQ $\Leftrightarrow$ QAS/OWF condition. We next show classically-secure OWPuzzs $\Leftrightarrow$ QAS/OWF condition. By combining them, we finally obtain the main result. In the following, we explain each step.

Step 1: IV-PoQ $\Rightarrow$ QAS/OWF condition. Its proof is inspired by [Ost91]. However, we emphasize that our proof is not a trivial application of [Ost91]. As we will explain later, the same proof of [Ost91] does not work in our setting, and therefore we had to overcome several technical challenges to show the result.

Let $(\mathcal{P}, \mathcal{V}_1, \mathcal{V}_2)$ be an ℓ-round IV-PoQ, where $\mathcal{P}$ is the prover, $\mathcal{V}_1$ is the efficient verifier, and $\mathcal{V}_2$ is the inefficient verifier. (We count two messages as a single round.) Without loss of generality, we can assume that $\mathcal{V}_1$ first sends a message. Let $(c_1, a_1, ..., c_\ell, a_\ell)$ be the transcript (i.e., the sequence of all messages exchanged) of the interaction between $\mathcal{P}$ and $\mathcal{V}_1$, where c_i is $\mathcal{V}_1$'s i-th message and a_i is $\mathcal{P}$'s i-th message. Our goal is, by assuming that the QAS/OWF condition is not satisfied, to construct a classical PPT adversary $\mathcal{P}^*$ that breaks the soundness of the IV-PoQ.

If the QAS/OWF condition is not satisfied, then, roughly speaking, both of the following two conditions are satisfied:

(a) QASs do not exist. In other words, the output probability distribution of any QPT algorithm can be approximately sampled with a PPT algorithm.

(b) Classically-secure OWFs do not exist.

From (a), the distribution of the transcript generated by the interaction between $\mathcal{P}$ and $\mathcal{V}_1$ can be approximately sampled with a PPT algorithm $\mathcal{S}$. One might think that if a malicious PPT prover of the IV-PoQ just runs $\mathcal{S}$, the soundness of the IV-PoQ is broken. However, it is not correct: The ability to classically efficiently sample from the distribution $\langle\mathcal{P},\mathcal{V}_1\rangle(1^\lambda)$ is not enough to break the soundness of the IV-PoQ, because what the PPT adversary $\mathcal{P}^*$ has to do is not to sample $(c_1,a_1,...,c_\ell,a_\ell)$ but to sample "correct" a_k given the transcript $(c_1,a_1,...,c_{k-1},a_{k-1},c_k)$ obtained so far for every $k\in[\ell]$.

We use (b) to solve it. From $\mathcal{S}$, we define a function f as follows.

1. Get an input (k,r).
2. Run $(c_1,a_1,...,c_\ell,a_\ell)=\mathcal{S}(1^\lambda;r)$.[12]
3. Output $(k,c_1,a_1,...,c_{k-1},a_{k-1},c_k)$.

From (b), OWFs do not exist. Then, distributional OWFs do not exist as well [IL89].[13] This means that there exists a PPT algorithm $\mathcal{R}$ such that the statistical distance between $(x,f(x))$ and $(\mathcal{R}(f(x)),f(x))$ is small for random x. Therefore, for each $k\in[\ell]$, the following PPT adversary $\mathcal{P}^*$ can return "correct" a_k given $(c_1,a_1,...,c_{k-1},a_{k-1},c_k)$.

1. Take $(c_1,a_1,...,c_{k-1},a_{k-1},c_k)$ as input.
2. Run $(k',r')\leftarrow\mathcal{R}(k,c_1,a_1,...,c_{k-1},a_{k-1},c_k)$.
3. Run $(c'_1,a'_1,...,c'_\ell,a'_\ell)=\mathcal{S}(1^\lambda;r')$.
4. Output $a'_{k'}$.

In this way, we can break the soundness of the IV-PoQ. Hence we have shown that IV-PoQ $\Rightarrow$ the QAS/OWF condition.

The idea underlying this proof is similar to that of [Ost91]. In [Ost91], it was shown that if SZK is average hard then OWFs exist. To show it, [Ost91] used the zero-knowledge property to guarantee the existence of a PPT simulator that can sample the transcript between the verifier and the prover. From that simulator, [Ost91] constructed a OWF. Very roughly speaking, our $\mathcal{S}$ that comes from (a) corresponds to the zero-knowledge simulator of [Ost91]: the transcript of [Ost91] can be PPT sampled because of the zero-knowledge property while our transcript can be PPT sampled because QASs do not exist. However, there are several crucial differences between our setting and [Ost91]'s. In particular, in the setting of [Ost91] the constructed OWF

[12]For a PPT algorithm $\mathcal{A}$, $y=\mathcal{A}(x;r)$ means that $\mathcal{A}$'s output is y when the input is x and the random seed is r.

[13]An efficiently-computable function $f:\{0,1\}^*\to\{0,1\}^*$ is called a classically-secure (resp. quantumly-secure) distributional OWF if for any PPT (resp. QPT) adversary $\mathcal{A}$, the statistical distance between $(x,f(x))$ and $(\mathcal{A}(f(x)),f(x))$ is large for random x.

can depend on the simulator. On the other hand, in our setting, we finally want to construct a OWF that is independent of $\mathcal{S}$.[14] In order to solve the issue, we use the universal construction of OWFs [Lev85, HKN$^+$05]. We first construct a OWF $f_{\mathcal{S}}$ from each $\mathcal{S}$ as we have explained above, and next construct a OWF g that is independent of $\mathcal{S}$ by using the universal construction. In addition to this issue, there are several other points where the direct application of [Ost91] does not work, but for details, see the main body of the paper.

Note that in the actual proof, we do not directly show IV-PoQ $\Rightarrow$ the QAS/OWF condition. We first show IV-PoQ $\Rightarrow$ Int-QAS, and then show Int-QAS $\Rightarrow$ the QAS/OWF condition in order to obtain stronger results and to avoid repeating similar proofs twice. However, the proof of Int-QAS $\Rightarrow$ the QAS/OWF condition is essentially the same as that explained above.

Step 2: Classically-secure OWPuzzs $\Rightarrow$ QAS/OWF condition. Its proof is similar to that of step 1. Let $(\mathsf{Samp}, \mathsf{Ver})$ be a classically-secure OWPuzz. Assume that the QAS/OWF condition is not satisfied. This roughly means that both of the following two conditions are satisfied.

(a) QASs do not exist.

(b) Classically-secure OWFs do not exist.

From (a), there exists a PPT algorithm $\mathcal{S}$ such that the output probability distribution of $\mathcal{S}(1^\lambda)$ is close to that of $\mathsf{Samp}(1^\lambda)$ in the statistical distance. From such $\mathcal{S}$, we construct a function f as follows.

1. Get a bit string r as input.
2. Compute $(\mathsf{puzz}, \mathsf{ans}) = \mathcal{S}(1^\lambda; r)$.
3. Output puzz.

From (b), OWFs do not exist. This means that distributional OWFs do not exist as well. Therefore, there exists a PPT algorithm $\mathcal{R}$ such that the statistical distance between $(x, f(x))$ and $(\mathcal{R}(f(x)), f(x))$ is small for random x. From $\mathcal{S}$ and $\mathcal{R}$, we can construct a PPT adversary $\mathcal{A}$ that breaks the security of the OWPuzz as follows:

1. Take puzz as input.
2. Run $r \leftarrow \mathcal{R}(\mathsf{puzz})$.
3. Run $(\mathsf{puzz}', \mathsf{ans}') \leftarrow \mathcal{S}(1^\lambda; r)$.
4. Output ans'.

As in step 1, we actually need a OWF that is independent of $\mathcal{S}$, and therefore we have to use the universal construction [Lev85, HKN$^+$05].

[14] In the precise definition of QAS/OWF condition, the OWFs should be independent of $\mathcal{S}$. Otherwise, we do not know how to show the other direction, namely, the QAS/OWF condition $\Rightarrow$ IV-PoQ.

Step 3: QAS/OWF condition $\Rightarrow$ IV-PoQ. Assume that the QAS/OWF condition is satisfied. Then, roughly speaking, a QAS $\mathcal{Q}$ exists or a classically-secure OWF f exists. From f, we can construct an IV-PoQ by using the result of [MY24]. The non-trivial part is to construct an IV-PoQ from $\mathcal{Q}$. For that goal, we use the idea of [Aar14]. However, as we will explain later, a direct application of [Aar14] does not work for our goal, and some new technical contributions were needed.

We construct a non-interactive IV-PoQ from $\mathcal{Q}$ as follows:

1. The QPT prover runs $\mathcal{Q}(1^\lambda)$ N times, where N is a certain polynomial, and sends the result $(y_1, ..., y_N)$ to the verifier, where y_i is the output of the i-th run of $\mathcal{Q}(1^\lambda)$.

2. The unbounded verifier computes the Kolmogorov complexity $K(y_1, ..., y_N)$[15], and accepts if it is larger than a certain value.

We can show that thus constructed non-interactive IV-PoQ satisfies completeness and soundness. For completeness, we use Markov's inequality to show that $K(y_1, ..., y_N)$, where $y_i \leftarrow \mathcal{Q}(1^\lambda)$ for each $i \in [N]$, is large with high probability and therefore the verifier accepts. To evaluate the bound of Markov's inequality, we use Kraft's inequality for the prefix Kolmogorov complexity, which says that $\sum_x 2^{-K(x)} \leq 1$. For soundness, assume that there exists a PPT adversary $\mathcal{P}^*$ that outputs $(y'_1, ..., y'_N)$ that is accepted by the verifier with high probability, which means that $\log \frac{1}{\Pr[(y'_1,...,y'_N) \leftarrow \mathcal{Q}(1^\lambda)^{\otimes N}]} \lesssim K(y'_1, ..., y'_N)$. Because of the property of K, we have $K(y'_1, ..., y'_N) \lesssim \log \frac{1}{\Pr[(y'_1,...,y'_N) \leftarrow \mathcal{P}^*(1^\lambda)]}$. By combining them, we have $\log \frac{\Pr[(y'_1..,y'_N) \leftarrow \mathcal{P}^*(1^\lambda)]}{\Pr[(y'_1,...,y'_N) \leftarrow \mathcal{Q}(1^\lambda)^{\otimes N}]} \approx 0$, which roughly means that the output probability distribution of $\mathcal{Q}(1^\lambda)^{\otimes N}$ is close to that of $\mathcal{P}^*(1^\lambda)$.

From such $\mathcal{P}^*$, we can construct a PPT algorithm whose output probability distribution is close to that of $\mathcal{Q}(1^\lambda)$ in the statistical distance as follows:

1. Run $(y'_1, ..., y'_N) \leftarrow \mathcal{P}^*(1^\lambda)$.

2. Choose a random $i \in [N]$, and output y'_i.

However, this means that $\mathcal{Q}$ is not a QAS, which contradicts the assumption. In this way, we can show the QAS/OWF condition $\Rightarrow$ IV-PoQ.

The idea underlying this proof is similar to that of [Aar14]. In fact, the completeness part is exactly the same. However, for the soundness part, the direct application of [Aar14] does not work, because of several reasons. Here we explain main two issues. First, the search problem constructed in [Aar14] was not necessarily verifiable even in unbounded time since Kolmogorov complexity is uncomputable in general. This is problematic for our goal, because what we want to construct is a non-interactive IV-PoQ where the prover's message should be verified *at least inefficiently*. [Aar14] slightly mentioned an extension of the result to the time-bounded case, but there was no proof. Second, [Aar14] constructed a search advantange from $\mathbf{SampBQP} \neq \mathbf{SampBPP}$, which is a worst-case notion. However, what we need is a search advantage from the existence of QASs, namely, the *average-case version* of sampling advantage. Hence the proof of [Aar14] cannot be directly used in our setting.

Step 4: QAS/OWF condition $\Rightarrow$ classically-secure OWPuzzs. The proof uses a similar technique as that of step 3. If the QAS/OWF condition is satisfied, then, roughly speaking, a classically-secure OWF f exists or a QAS $\mathcal{Q}$ exists. From the OWF f, we can construct a classically-secure OWPuzz easily as follows:

[15]More precisely, this is time-bounded prefix Kolmogorov complexity $K_U^T(y_1, ..., y_N)$ with time bound $T(n) = 2^{2^n}$ and the universal self-delimiting machine U.

- $\mathsf{Samp}(1^\lambda) \to (\mathsf{puzz}, \mathsf{ans})$: Choose $x \leftarrow \{0,1\}^\lambda$, and output $\mathsf{puzz} := f(x)$ and $\mathsf{ans} := x$.
- $\mathsf{Ver}(\mathsf{puzz}, \mathsf{ans}') \to \top/\bot$: Accept if and only if $f(\mathsf{ans}') = \mathsf{puzz}$.

From $\mathcal{Q}$, we can construct a non-interactive IV-PoQ as in step 3. From such a non-interactive IV-PoQ, we can easily construct a classically-secure OWPuzz as follows.

- $\mathsf{Samp}(1^\lambda) \to (\mathsf{puzz}, \mathsf{ans})$: Run $\tau \leftarrow \mathcal{P}(1^\lambda)$, and output $\mathsf{puzz} := 1^\lambda$ and $\mathsf{ans} := \tau$.
- $\mathsf{Ver}(\mathsf{puzz}, \mathsf{ans}') \to \top/\bot$: Accept if and only if $\top \leftarrow \mathcal{V}_2(1^\lambda, \mathsf{ans}')$.

2 Preliminaries

2.1 Basic Notations

$\log x$ means $\log_2 x$ and $\ln x$ means $\log_e x$. We use standard notations of quantum computing and cryptography. For a bit string x, $|x|$ is its length. $\mathbb{N}$ is the set of natural numbers. We use λ as the security parameter. $[n]$ means the set $\{1, 2, ..., n\}$. For a finite set S, $x \leftarrow S$ means that an element x is sampled uniformly at random from the set S. negl is a negligible function, and poly is a polynomial. All polynomials appear in this paper are positive, but for simplicity we do not explicitly mention it. PPT stands for (classical) probabilistic polynomial-time and QPT stands for quantum polynomial-time. For an algorithm $\mathcal{A}$, $y \leftarrow \mathcal{A}(x)$ means that the algorithm $\mathcal{A}$ outputs y on input x. If $\mathcal{A}$ is a classical probabilistic or quantum algorithm that takes x as input and outputs bit strings, we often mean $\mathcal{A}(x)$ by the output probability distribution of $\mathcal{A}$ on input x. When $\mathcal{A}$ is a classical probabilistic algorithm, $y = \mathcal{A}(x; r)$ means that the output of $\mathcal{A}$ is y if it runs on input x and with the random seed r. For two interactive algorithms $\mathcal{A}$ and $\mathcal{B}$ that interact over a classical channel, $\tau \leftarrow \langle \mathcal{A}(x), \mathcal{B}(y) \rangle$ means that the transcript τ (i.e., the sequence of all messages exchanged) is generated by the interactive protocol between $\mathcal{A}$ and $\mathcal{B}$ where $\mathcal{A}$ takes x as input and $\mathcal{B}$ takes y as input. If both $\mathcal{A}$ and $\mathcal{B}$ take the same input x, we also write it as $\tau \leftarrow \langle \mathcal{A}, \mathcal{B} \rangle(x)$. For two quantum states ρ and σ, $\mathsf{TD}(\rho, \sigma) := \frac{1}{2}\|\rho - \sigma\|_1$ means their trace distance, where $\|X\|_1 := \mathrm{Tr}\sqrt{X^\dagger X}$ is the trace norm. For two probability distributions $P := \{p_i\}_i$ and $Q := \{q_i\}_i$, $\mathsf{SD}(Q, P) := \frac{1}{2}\sum_i |p_i - q_i|$ is their statistical distance. If $\rho = \sum_i p_i |\phi_i\rangle\langle\phi_i|$ and $\sigma = \sum_i q_i |\phi_i\rangle\langle\phi_i|$ for some orthonormal basis $\{|\phi_i\rangle\}_i$, we have $\mathsf{TD}(\rho, \sigma) = \mathsf{SD}(\{p_i\}_i, \{q_i\}_i)$.

2.2 One-Way Functions and Distributional One-Way Functions

We first review the definition of one-way functions (OWFs).

Definition 2.1 (One-Way Functions (OWFs)). *A function $f : \{0,1\}^* \to \{0,1\}^*$ that is computable in classical deterministic polynomial-time is a classically-secure (resp. quantumly-secure) one-way function (OWF) if for any PPT (resp. QPT) adversary $\mathcal{A}$ and any polynomial p,*

$$\Pr[f(x') = f(x) : x \leftarrow \{0,1\}^\lambda, x' \leftarrow \mathcal{A}(1^\lambda, f(x))] \le \frac{1}{p(\lambda)} \tag{3}$$

holds for all sufficiently large $\lambda \in \mathbb{N}$.

We define a variant of OWFs, which we call OWFs on a subset $\Sigma \subseteq \mathbb{N}$. The difference from the standard OWFs is that security holds only when the security parameter belongs to the subset Σ of $\mathbb{N}$.

Definition 2.2 (OWFs on Σ). *Let $\Sigma \subseteq \mathbb{N}$ be a set. A function $f : \{0,1\}^* \to \{0,1\}^*$ that is computable in classical deterministic polynomial-time is a classically-secure (resp. quantumly-secure) OWF on Σ if there exists an efficiently-computable polynomial n such that for any PPT (resp. QPT) adversary $\mathcal{A}$ and any polynomial p there exists $\lambda^* \in \mathbb{N}$ such that*

$$\Pr[f(x') = f(x) : x \leftarrow \{0,1\}^{n(\lambda)}, x' \leftarrow \mathcal{A}(1^{n(\lambda)}, f(x))] \leq \frac{1}{p(\lambda)} \tag{4}$$

holds for all $\lambda \geq \lambda^*$ in Σ.

Remark 2.3. In the definition of OWFs (Definition 2.1) the input length is treated as the security parameter, but in OWFs on Σ (Definition 2.2), we allow the input length to be an arbitrary polynomial in the security parameter.

Remark 2.4. For any finite Σ, OWFs on Σ always exist because the definition is trivially satisfied. (We have only to take $\lambda^* = \lambda_{max} + 1$, where λ_{max} is the largest element of Σ.) However, we include the case when Σ is finite in the definition for convenience.

The existence of OWFs on $\mathbb{N} \setminus \Sigma$ for a finite subset Σ is actually equivalent to that of the standard OWFs. Its proof is given in Appendix B.

Lemma 2.5. *Let $\Sigma \subseteq \mathbb{N}$ be a finite set. Classically-secure (resp. quantumly-secure) OWFs exist if and only if classically-secure (resp. quantumly-secure) OWFs on $\mathbb{N} \setminus \Sigma$ exist.*

We also review the definition of distributional one-way functions (DistOWFs).

Definition 2.6 (Distributional One-Way Functions (DistOWFs) [IL89]). *A function $f : \{0,1\}^* \to \{0,1\}^*$ that is computable in classical deterministic polynomial-time is a classically-secure (resp. quantumly-secure) distributional one-way function (DistOWF) if there exists a polynomial p such that*

$$\mathsf{SD}(\{(x, f(x))\}_{x \leftarrow \{0,1\}^\lambda}, \{(\mathcal{A}(1^\lambda, f(x)), f(x))\}_{x \leftarrow \{0,1\}^\lambda}) \geq \frac{1}{p(\lambda)} \tag{5}$$

holds for any PPT (resp. QPT) adversary $\mathcal{A}$ and for all sufficiently large $\lambda \in \mathbb{N}$.

Similarly to OWFs on Σ, we define DistOWFs on Σ for a subset $\Sigma \subseteq \mathbb{N}$.

Definition 2.7 (DistOWFs on Σ). *Let $\Sigma \subseteq \mathbb{N}$ be a set. A function $f : \{0,1\}^* \to \{0,1\}^*$ that is computable in classical deterministic polynomial-time is a classically-secure (resp. quantumly-secure) DistOWF on Σ if there exist an efficiently-computable polynomial n and a polynomial p such that for any PPT (resp. QPT) adversary $\mathcal{A}$ there exists $\lambda^* \in \mathbb{N}$ such that*

$$\mathsf{SD}(\{(x, f(x))\}_{x \leftarrow \{0,1\}^{n(\lambda)}}, \{(\mathcal{A}(1^{n(\lambda)}, f(x)), f(x))\}_{x \leftarrow \{0,1\}^{n(\lambda)}}) \geq \frac{1}{p(\lambda)} \tag{6}$$

holds for all $\lambda \geq \lambda^*$ in Σ.

Remark 2.8. Again, for any finite Σ, DistOWFs on Σ always exist because the definition is trivially satisfied. However, we include the case when Σ is finite in the definition for convenience.

It is well-known that OWFs exist if and only if DistOWFs exist [IL89, Lemma 1].[16] By inspecting its proof, one can see that the proof gives a "security-parameter-wise" reduction, i.e., if the base DistOWF is secure on inputs of length λ, then the resulting OWF is secure on inputs of length $n(\lambda)$ for some efficiently computable polynomial n.[17] This implies equivalence between OWFs on Σ and DistOWFs on Σ on any subset $\Sigma \subseteq \mathbb{N}$.

Lemma 2.9 (Based on [IL89, Lemma 1]). *For any subset $\Sigma \subseteq \mathbb{N}$, classically-secure DistOWFs on Σ exist if and only if classically-secure OWFs on Σ exist.*

It is known that there exists a *universal construction* for OWFs [Lev85], a concrete function that is a OWF if and only if OWFs exist. We observe that a similar technique yields a universal construction for OWFs on Σ as well. Its proof is given in Appendix C.

Lemma 2.10. *There exists a function $g : \{0,1\}^* \to \{0,1\}^*$ that is computable in classical deterministic polynomial-time such that for any subset $\Sigma \subseteq \mathbb{N}$, if there exist classically-secure OWFs on Σ, then g is a classically-secure OWF on Σ.*

By combining Lemmata 2.9 and 2.10, we obtain the following corollary, which will be used later.

Corollary 2.11. *There exists a function $g : \{0,1\}^* \to \{0,1\}^*$ that is computable in classical deterministic polynomial-time such that for any subset $\Sigma \subseteq \mathbb{N}$, if there exist classically-secure DistOWFs on Σ, then g is a classically-secure OWF on Σ.*

2.3 One-Way Puzzles

We also define one-way puzzles (OWPuzzs) on a subset $\Sigma \subseteq \mathbb{N}$, which are a generalization of OWPuzzs defined in [KT24a]. If $\Sigma = \mathbb{N}$, the definition becomes the standard one [KT24a], and in that case we call them just OWPuzzs.

Definition 2.12 (One-Way Puzzles (OWPuzzs) on Σ). *Let $\Sigma \subseteq \mathbb{N}$ be a set. A one-way puzzle (OWPuzz) on Σ is a pair $(\mathsf{Samp}, \mathsf{Ver})$ of algorithms such that*

- $\mathsf{Samp}(1^\lambda) \to (\mathsf{puzz}, \mathsf{ans})$: *It is a QPT algorithm that, on input the security parameter λ, outputs a pair $(\mathsf{puzz}, \mathsf{ans})$ of classical strings.*
- $\mathsf{Ver}(\mathsf{puzz}, \mathsf{ans}') \to \top/\bot$: *It is an unbounded algorithm that, on input $(\mathsf{puzz}, \mathsf{ans}')$, outputs either $\top/\bot$.*

They satisfy the following properties for some functions c and s such that $c(\lambda) - s(\lambda) \geq \frac{1}{\mathrm{poly}(\lambda)}$.

[16] [IL89] only provides a proof sketch. Its full proof can be found in [Imp92, Theorem 4.2.2]

[17] More precisely, for any efficiently computable function $f : \{0,1\}^* \to \{0,1\}^*$ and polynomial p, there are an efficiently computable function $g : \{0,1\}^* \to \{0,1\}^*$ and efficiently computable polynomial n that satisfy the following: For any PPT adversary $\mathcal{A}$ and a polynomial q, there is a PPT adversary $\mathcal{B}$ such that for any $\lambda \in \mathbb{N}$, if

$$\Pr[g(x') = g(x) : x \leftarrow \{0,1\}^{n(\lambda)}, x' \leftarrow \mathcal{A}(1^{n(\lambda)}, g(x))] > \frac{1}{q(\lambda)}, \tag{7}$$

then

$$\mathsf{SD}(\{(x, f(x))\}_{x \leftarrow \{0,1\}^\lambda}, \{(\mathcal{B}(1^\lambda, f(x)), f(x))\}_{x \leftarrow \{0,1\}^\lambda}) < \frac{1}{p(\lambda)}. \tag{8}$$

- c**-correctness:** *There exists* $\lambda^* \in \mathbb{N}$ *such that*

$$\Pr[\top \leftarrow \mathsf{Ver}(\mathsf{puzz},\mathsf{ans}) : (\mathsf{puzz},\mathsf{ans}) \leftarrow \mathsf{Samp}(1^\lambda)] \geq c(\lambda) \tag{9}$$

holds for all $\lambda \geq \lambda^*$.

- s**-security on** Σ**:** *For any QPT adversary* $\mathcal{A}$ *there exists* $\lambda^{**} \in \mathbb{N}$ *such that*

$$\Pr[\top \leftarrow \mathsf{Ver}(\mathsf{puzz},\mathcal{A}(1^\lambda,\mathsf{puzz})) : (\mathsf{puzz},\mathsf{ans}) \leftarrow \mathsf{Samp}(1^\lambda)] \leq s(\lambda) \tag{10}$$

holds for all $\lambda \geq \lambda^{**}$ *in* Σ.

Definition 2.13 (Classically-Secure OWPuzzs on Σ). *A OWPuzz on Σ is called a classically-secure OWPuzz on Σ if the security is required against PPT adversaries.*

Remark 2.14. Again, if Σ is a finite set, OWPuzzs on Σ trivially exist, but we include such a case in the definition for the convenience.

Remark 2.15. All classically-secure OWPuzzs appearing in this paper are ones with $(1-\mathsf{negl}(\lambda))$-correctness and $(1-1/\mathrm{poly}(\lambda))$-security.

Remark 2.16. It is known that c-correct and s-secure OWPuzzs with $c(\lambda)-s(\lambda) \geq 1/\mathrm{poly}(\lambda)$ can be amplified to $(1-\mathsf{negl}(\lambda))$-correct and $\mathsf{negl}(\lambda)$-secure OWPuzzs [CGG24b]. On the other hand, we do not know how to amplify the gap of classically-secure OWPuzzs.

OWPuzzs can be constructed from OWFs. We can show that this is also the case for the variants on Σ. Its proof is given in Appendix D.

Lemma 2.17. *Let $\Sigma \subseteq \mathbb{N}$ be a subset. If classically-secure OWFs on Σ exist, then classically-secure OWPuzzs on Σ with 1-correctness and* negl*-security exist.*

2.4 Inefficient-Verifier Proofs of Quantumness

In this subsection, we define inefficient-verifier proofs of quantumness (IV-PoQ) on a subset $\Sigma \subseteq \mathbb{N}$. IV-PoQ defined in [MY24] are special cases when $\Sigma = \mathbb{N}$.

Definition 2.18 (Inefficient-Verifier Proofs of Quantumness (IV-PoQ) on Σ). *Let $\Sigma \subseteq \mathbb{N}$ be a set. An IV-PoQ on Σ is a tuple $(\mathcal{P},\mathcal{V}_1,\mathcal{V}_2)$ of interactive algorithms. $\mathcal{P}$ (prover) is QPT, $\mathcal{V}_1$ (first verifier) is PPT, and $\mathcal{V}_2$ (second verifier) is unbounded. The protocol is divided into two phases. In the first phase, $\mathcal{P}$ and $\mathcal{V}_1$ take the security parameter 1^λ as input and interact with each other over a classical channel. Let τ be the transcript, i.e., the sequence of all classical messages exchanged between $\mathcal{P}$ and $\mathcal{V}_1$. In the second phase, $\mathcal{V}_2$ takes 1^λ and τ as input and outputs $\top$ (accept) or $\bot$ (reject). We require the following two properties for some functions c and s such that $c(\lambda)-s(\lambda) \geq 1/\mathrm{poly}(\lambda)$.*

- c**-completeness:** *There exists* $\lambda^* \in \mathbb{N}$ *such that*

$$\Pr[\top \leftarrow \mathcal{V}_2(1^\lambda,\tau) : \tau \leftarrow \langle \mathcal{P},\mathcal{V}_1\rangle(1^\lambda)] \geq c(\lambda) \tag{11}$$

holds for all $\lambda \geq \lambda^*$.

- **s-soundness on Σ:** *For any PPT prover $\mathcal{P}^*$ there exists $\lambda^{**} \in \mathbb{N}$ such that*

$$\Pr[\top \leftarrow \mathcal{V}_2(1^\lambda, \tau) : \tau \leftarrow \langle \mathcal{P}^*, \mathcal{V}_1 \rangle (1^\lambda)] \leq s(\lambda) \tag{12}$$

holds for all $\lambda \geq \lambda^{**}$ in Σ.

Moreover, if all the messages sent from $\mathcal{V}_1$ are uniformly random strings, we say that the IV-PoQ is public-coin.

Remark 2.19. IV-PoQ on Σ always exist for any finite set Σ, but we include the case in the definition for the convenience.

Remark 2.20. In the previous definition of IV-PoQ [MY24], $\mathcal{V}_2$ does not take 1^λ as input. However, this does not change the definition for interactive IV-PoQ, because 1^λ can be added to the first $\mathcal{V}_1$'s message. We explicitly include 1^λ in the input of $\mathcal{V}_2$ since we also consider non-interactive IV-PoQ in this paper.

[MY24] showed that classically-secure OWFs imply IV-PoQ by constructing IV-PoQ from statistically-hiding and computationally-binding commitment schemes that are implied by OWFs [HNO+09]. By inspecting its proof, one can see that the proof gives a "security-parameter-wise" reduction, i.e., for any efficiently computable polynomial n, we can construct IV-PoQ from classically-secure OWFs such that that if the base OWF is secure on inputs of length $n(\lambda)$, then the resulting IV-PoQ is sound on the security parameter λ.[18] Thus, we have the following lemma.

Lemma 2.21 (Based on [HNO+09, MY24]). *Let $\Sigma \subseteq \mathbb{N}$ be a set. If classically-secure OWFs on Σ exist, then IV-PoQ on Σ exist. Moreover, the constructed IV-PoQ is public-coin and satisfies $(1 - \mathsf{negl})$-completeness and negl-soundness on Σ.*

Remark 2.22. In [MY24], they do not explicitly state that the protocol is public-coin. To see that it is indeed public-coin, observe that the verifier's messages of the IV-PoQ of [MY24] consist of the receiver's messages of a statistically hiding commitment scheme of [HNO+09], descriptions of pairwise independent hash functions, and uniformly random strings from the verifier of [KMCVY22]. As mentioned in [HNO+09, Section 8], their commitment scheme is public-coin. Moreover, we can assume that a description of a pairwise independent hash function is public-coin without loss of generality since we can treat the randomness for choosing the function as its description. Thus, the IV-PoQ of [MY24] is public-coin.

2.5 Sampling Complexity

Definition 2.23 (Sampling Problems [Aar14, ABK24]). *A (polynomially-bounded) sampling problem S is a collection of probability distributions $\{D_x\}_{x\in\{0,1\}^*}$, where D_x is a distribution over $\{0,1\}^{p(|x|)}$, for some fixed polynomial p.*

[18] More precisely, for any efficiently computable function $f : \{0,1\}^* \to \{0,1\}^*$ and efficiently computable polynomial n, there is an IV-PoQ $(\mathcal{P}, \mathcal{V}_1, \mathcal{V}_2)$ such that for any PPT algorithm $\mathcal{P}^*$ and any polynomial p, there are a PPT algorithm $\mathcal{A}$ and a polynomial q such that for any $\lambda \in \mathbb{N}$, if

$$\Pr[\top \leftarrow \mathcal{V}_2(1^\lambda, \tau) : \tau \leftarrow \langle \mathcal{P}^*, \mathcal{V}_1 \rangle (1^\lambda)] > \frac{1}{p(\lambda)}, \tag{13}$$

then

$$\Pr[f(x') = f(x) : x \leftarrow \{0,1\}^{n(\lambda)}, x' \leftarrow \mathcal{A}(1^{n(\lambda)}, f(x))] > \frac{1}{q(\lambda)}. \tag{14}$$

Definition 2.24 (SampBPP and SampBQP [Aar14, ABK24]). SampBPP *is the class of (polynomially-bounded) sampling problems $S = \{D_x\}_{x\in\{0,1\}^*}$ for which there exists a PPT algorithm $\mathcal{B}$ such that for all x and all $\epsilon > 0$, $\mathsf{SD}(\mathcal{B}(x, 1^{\lfloor 1/\epsilon \rfloor}), D_x) \le \epsilon$, where $\mathcal{B}(x, 1^{\lfloor 1/\epsilon \rfloor})$ is the output probability distribution of $\mathcal{B}$ on input $(x, 1^{\lfloor 1/\epsilon \rfloor})$.* **SampBQP** *is defined the same way, except that $\mathcal{B}$ is a QPT algorithm rather than a PPT one.*

2.6 Kolmogorov Complexity

In this subsection, we introduce the definition of time-bounded prefix Kolmogorov complexity and prove lemmas we use in the proof of Lemma 3.8.

Time-bounded prefix Kolmogorov complexity is defined using a special type of Turing machines called *self-delimiting machines*.[19] Let M be a self-delimiting machine, p be a program, and x be a bit string. We denote $M(p) = x$ (resp. in at most t steps) if M takes as input a program p in the input tape, M halts (resp. in at most t steps), and the output tape contains x.[20] For a self-delimiting machine M, we say that $p \in \{0,1\}^*$ is a valid program for M if there exists $x \in \{0,1\}^*$ such that $M(p) = x$. For any self-delimiting machine M, the set of all valid programs $p \in \{0,1\}^*$ for M is prefix-free. Here, a set $S \subseteq \{0,1\}^*$ of strings is called prefix-free, if any $x \in S$ is not a prefix of any other $y \in S$.[21] A function $t : \mathbb{N} \to \mathbb{N}$ is called time-constructible if $t(n) \ge n$ for all $n \in \mathbb{N}$ and there exists a Turing machine that, on input 1^n, halts and outputs the binary representation of $t(n)$ in time $O(t(n))$ for all $n \in \mathbb{N}$.

Definition 2.25 (Time-Bounded Prefix Kolmogorov Complexity [JL00, LV19]). *Let M be a self-delimiting machine. Let $t : \mathbb{N} \to \mathbb{N}$ be a time-constructible function and $x \in \{0,1\}^*$ be a bit string. Let*

$$G_M^t(x) := \{p \in \{0,1\}^* : M(p) = x \text{ in at most } t(|x|) \text{ steps}\}. \tag{15}$$

The t-time bounded prefix Kolmogorov complexity $K_M^t(x)$ is defined as

$$K_M^t(x) := \begin{cases} \min\{|p| : p \in G_M^t(x)\} & \text{If } G_M^t(x) \text{ is not empty.} \\ \infty & \text{If } G_M^t(x) \text{ is empty.} \end{cases} \tag{16}$$

Remark 2.26. There is another variant called time-bounded *plain* Kolmogorov complexity where M is a normal Turing machine rather than a self-delimiting one. Though the definition of the plain version is simpler than that of the above prefix version, we choose to use the prefix version since it is unclear if Lemmata 2.28 and 2.29 hold for the plain version (though we believe that we can prove variants of Lemmata 2.28 and 2.29 for the plain version with some additional error terms, which would be also sufficient for the purpose of this paper).

The following lemma guarantees the existence of a special self-delimiting machine U. We call it *the universal self-delimiting machine*.

Lemma 2.27 (Example 7.1.1 of [LV19]). *There exists a self-delimiting machine U such that for any self-delimiting machine M, there exists a constant $c > 0$ such that*

$$K_U^{ct\log t}(x) \le K_M^t(x) + c \tag{17}$$

for all $x \in \{0,1\}^*$ and any time-constructible function $t : \mathbb{N} \to \mathbb{N}$.

[19] For the purpose of this paper, their detailed definition is not necessary (for which we refer to [LV19, Chapter 3]). The important point is, as we will explain, that the set of their valid programs is prefex-free.

[20] Actually, another additional criteria is required: when it halts, the head of the input tape is on the rightmost bit of p. This in particular means that M never scans the next bit after p, because the input tape of a self-delimiting machine is one-way.

[21] x is a prefix of y if there exists a bit string z such that $y = x\|z$. For example, if $x = 101$ and $y = 101111$, then x is a prefix of y.

We will use the following lemma to show Lemma 3.8.[22]

Lemma 2.28. *Let $m : \mathbb{N} \to \mathbb{N}$ be a function such that $m(\lambda) \geq \lambda$ for all $\lambda \in \mathbb{N}$. For each $\lambda \in \mathbb{N}$, let $\mathcal{D}_\lambda$ be a distribution over $m(\lambda)$ bits. For any $k > 0$, any time-constructible function $t : \mathbb{N} \to \mathbb{N}$ such that $t(n) = \Omega(n^2)$, and for all sufficiently large $\lambda \in \mathbb{N}$,*

$$\Pr_{x \leftarrow \mathcal{D}_\lambda}\left[K_U^t(x) \geq \log \frac{1}{\Pr[x \leftarrow \mathcal{D}_\lambda]} - k\right] \geq 1 - \frac{1}{2^k}, \tag{18}$$

where U is the universal self-delimiting machine.

Proof of Lemma 2.28. Let $m : \mathbb{N} \to \mathbb{N}$ be a function such that $m(\lambda) \geq \lambda$ for all $\lambda \in \mathbb{N}$. For each $\lambda \in \mathbb{N}$, let $\mathcal{D}_\lambda$ be a distribution over $m(\lambda)$ bits. Let $t : \mathbb{N} \to \mathbb{N}$ be a time-constructible function such that $t(n) = \Omega(n^2)$. We will show later that for all sufficiently large $\lambda \in \mathbb{N}$ and for all $x \in \{0,1\}^{m(\lambda)}$, there exists a program P_x such that $|P_x| = K_U^t(x)$ and $U(P_x) = x$ in at most $t(|x|)$ steps, where U is the universal self-delimiting machine. The set $\{P_x\}_{x \in \{0,1\}^{m(\lambda)}}$ is prefix-free because U is a self-delimiting machine and P_x is valid for all $x \in \{0,1\}^{m(\lambda)}$. It is known that $\sum_{x \in \{0,1\}^{m(\lambda)}} 2^{-|P_x|} \leq 1$ holds for prefix-free $\{P_x\}_{x \in \{0,1\}^{m(\lambda)}}$. (It is called Kraft's inequality (Section 1.11 of [LV19]).) By Markov's inequality, for any $k > 0$,

$$\Pr_{x \leftarrow \mathcal{D}_\lambda}\left[\frac{2^{-K_U^t(x)}}{\Pr[x \leftarrow \mathcal{D}_\lambda]} \geq 2^k\right] \leq \frac{1}{2^k} \sum_{x \in \{0,1\}^{m(\lambda)}} \Pr[x \leftarrow \mathcal{D}_\lambda] \frac{2^{-K_U^t(x)}}{\Pr[x \leftarrow \mathcal{D}_\lambda]} \tag{19}$$

$$= \frac{1}{2^k} \sum_{x \in \{0,1\}^{m(\lambda)}} 2^{-K_U^t(x)} = \frac{1}{2^k} \sum_{x \in \{0,1\}^{m(\lambda)}} 2^{-|P_x|} \leq \frac{1}{2^k}. \tag{20}$$

Therefore, we obtain

$$\Pr_{x \leftarrow \mathcal{D}_\lambda}\left[K_U^t(x) \geq \log \frac{1}{\Pr[x \leftarrow \mathcal{D}_\lambda]} - k\right] \geq 1 - \frac{1}{2^k} \tag{21}$$

for any $k > 0$ and for all sufficiently large $\lambda \in \mathbb{N}$.

In the remaining part, we show that for all sufficiently large $\lambda \in \mathbb{N}$ and for all $x \in \{0,1\}^{m(\lambda)}$, there exists a program P_x such that $|P_x| = K_U^t(x)$ and $U(P_x) = x$ in at most $t(|x|)$ steps. To show it, it is sufficient to show $K_U^t(x) < \infty$ for all sufficiently large $\lambda \in \mathbb{N}$ and for all $x \in \{0,1\}^{m(\lambda)}$, because in that case we have only to take P_x as the program p that achieves the minimum in Equation (16). In fact, in the following we show that for all sufficiently large $\lambda \in \mathbb{N}$ and all $x \in \{0,1\}^{m(\lambda)}$, $K_U^t(x) = O(m(\lambda))$.

Later we will show that there exist a self-delimiting machine M and a time-constructible function $T : \mathbb{N} \to \mathbb{N}$ that satisfy the following: $T(n) = O(n)$ and for any $x \in \{0,1\}^*$, $K_M^T(x) \leq O(|x|)$. Then by Lemma 2.27, there exists a constant $c > 0$ such that

$$K_U^{cT \log T}(x) \leq K_M^T(x) + c \leq O(|x|) \tag{22}$$

for all $x \in \{0,1\}^*$. Because $t(n) = \Omega(n^2)$, $t(n) \geq cT(n) \log T(n)$ for all sufficiently large n. Thus, for all sufficiently large $\ell \in \mathbb{N}$ and for all $x \in \{0,1\}^\ell$,

$$K_U^t(x) \leq K_U^{cT \log T}(x) \leq O(\ell). \tag{23}$$

[22]The lemma was shown in [LV19] for the prefix Kolmogorov complexity. Here we modify its proof for time-bounded prefix Kolmogorov complexity.

Here, in the first inequality, we have used the fact that $K^{\alpha}_U(x) \leq K^{\beta}_U(x)$ for any time-constructible functions α and β such that $\alpha(|x|) \geq \beta(|x|)$. Because $m(\lambda) \geq \lambda$, this means that $K^t_U(x) \leq O(m(\lambda))$ for all sufficiently large $\lambda \in \mathbb{N}$ and for all $x \in \{0,1\}^{m(\lambda)}$.

Finally, we show that there exist a self-delimiting machine M and a time-constructible function $T : \mathbb{N} \to \mathbb{N}$ such that $T(n) = O(n)$ and for all $x \in \{0,1\}^*$, $K^T_M(x) \leq O(|x|)$. We construct M as follows:

1. M reads the input tape cell one by one. If the input is in the form of $1^{\ell}\|0\|x\|y$, where $\ell \geq 1, x \in \{0,1\}^*$, $|x| \geq 1$, and y is the string of all blanks, then go to the next step.[23] Otherwise, M continues reading the input tape, which means that M never halts.

2. Copy the first ℓ bits of $x\|y$ into the output tape.

Consider the case where the bit string $1^{|x|}\|0\|x$ with some $x \in \{0,1\}^*$ is written from the leftmost cell of the input tape and all cells to the right of $1^{|x|}\|0\|x$ are blanks. Then, for all $x \in \{0,1\}^*$, $M(1^{|x|}\|0\|x) = x$ in at most $O(|x|)$ steps. This means that there exists a time-constructible function $T : \mathbb{N} \to \mathbb{N}$ such that $T(n) = O(n)$ and

$$K^T_M(x) \leq |1^{|x|}\|0\|x| = O(|x|) \tag{24}$$

for all $x \in \{0,1\}^*$. Therefore, we complete the proof.

□

We will also use the following lemma which will be used to show Lemma 3.8.[24]

Lemma 2.29. *Let $\mathcal{A}$ be a PPT algorithm that, on input 1^{λ}, outputs an $m(\lambda)$-bit string for all $\lambda \in \mathbb{N}$. Here $m : \mathbb{N} \to \mathbb{N}$ is a polynomial such that $m(\lambda) \geq \lambda$ for all $\lambda \in \mathbb{N}$. Let $p_{\lambda,x} := \Pr[x \leftarrow \mathcal{A}(1^{\lambda})]$ for all $\lambda \in \mathbb{N}$ and for all $x \in \{0,1\}^{m(\lambda)}$. Then, for any time-constructible function $t : \mathbb{N} \to \mathbb{N}$ such that $t(n) = 2^{\omega(\mathrm{poly}(n))}$,*

$$K^t_U(x) \leq \log \frac{1}{p_{\lambda,x}} + O(\log \lambda) \tag{25}$$

for all sufficiently large $\lambda \in \mathbb{N}$ and for all $x \in \mathrm{Supp}(\mathcal{A}(1^{\lambda}))$. Here, $\mathrm{Supp}(\mathcal{A}(1^{\lambda}))$ is the support of the distribution $\mathcal{A}(1^{\lambda})$ over the outputs of $\mathcal{A}(1^{\lambda})$, and U is the universal self-delimiting machine.

Proof of Lemma 2.29. For all $\lambda \in \mathbb{N}$ and for all $x \in \mathrm{Supp}(\mathcal{A}(1^{\lambda}))$, let $c(x)$ be the code word of x obtained by Shannon-Fano coding (Section 1.11 of [LV19]). By the property of Shannon-Fano coding, the set $\{c(x)\}_{x \in \mathrm{Supp}(\mathcal{A}(1^{\lambda}))}$ is prefix-free for all $\lambda \in \mathbb{N}$, and

$$\log \frac{1}{p_{\lambda,x}} < |c(x)| \leq \log \frac{1}{p_{\lambda,x}} + 1 \tag{26}$$

is satisfied for all $\lambda \in \mathbb{N}$ and for all $x \in \mathrm{Supp}(\mathcal{A}(1^{\lambda}))$.

Later we will show that there exist a self-delimiting machine M and a time-constructible function T that satisfy the following: $T(n) = 2^{O(n^a)}$ for some constant $a \geq 1$ and for all $\lambda \in \mathbb{N}$ and for all $x \in \mathrm{Supp}(\mathcal{A}(1^{\lambda}))$,

$$K^T_M(x) \leq |c(x)| + O(\log \lambda). \tag{27}$$

[23]It is possible to check whether the input is in this form by scanning the input tape in the one-way.

[24]This is essentially shown in [LV19]. However in its proof, the $O(\log \lambda)$ factor in Equation (25) is not explicitly given. Here we reconstruct the proof to obtain the $O(\log \lambda)$ factor, because we need it for our purpose.

By Equation (26), we have

$$K_M^T(x) \leq \log \frac{1}{p_{\lambda,x}} + O(\log \lambda). \tag{28}$$

By Lemma 2.27, there exists a constant $c > 0$ such that

$$K_U^{cT \log T}(x) \leq K_M^T(x) + c \leq \log \frac{1}{p_{\lambda,x}} + O(\log \lambda), \tag{29}$$

where U is the self-delimiting machine.

Let $t : \mathbb{N} \to \mathbb{N}$ be a time-constructible function such that $t(n) = 2^{\omega(\mathrm{poly}(n))}$. Then we have $t(n) \geq cT(n) \log T(n)$ for all sufficiently large $n \in \mathbb{N}$. Because $m(\lambda) \geq \lambda$ for all $\lambda \in \mathbb{N}$, this means $t(m(\lambda)) \geq cT(m(\lambda)) \log T(m(\lambda))$ for all sufficiently large $\lambda \in \mathbb{N}$. Thus, for all sufficiently large $\lambda \in \mathbb{N}$ and for all $x \in \mathrm{Supp}(\mathcal{A}(1^\lambda))$,

$$K_U^t(x) \leq K_U^{cT \log T}(x) \leq \log \frac{1}{p_{\lambda,x}} + O(\log \lambda). \tag{30}$$

Here, in the first inequality, we have used the fact that $K_U^\alpha(x) \leq K_U^\beta(x)$ for any time-constructible functions α and β such that $\alpha(|x|) \geq \beta(|x|)$.

In the remaining part, we show that there exist a self-delimiting machine M and a time-constructible function $T : \mathbb{N} \to \mathbb{N}$ such that $T(n) = 2^{O(n^a)}$ for some constant $a \geq 1$ and $K_M^T(x) \leq |c(x)| + O(\log \lambda)$ for all $\lambda \in \mathbb{N}$ and for all $x \in \mathrm{Supp}(\mathcal{A}(1^\lambda))$. We construct the self-delimiting machine M as follows:

1. M reads the input tape cell one by one. If the input is in the form of $1^\ell \| 0 \| \hat{\lambda} \| y \| z$, where $\ell \geq 1$, $\hat{\lambda} \in \{0,1\}^\ell$, $y \in \{0,1\}^*$, $|y| \geq 1$, and z is the string of all blanks, then go to the next step.[25] Otherwise, M continues reading the input tape, which means that M never halts.

2. Let $\lambda \in \mathbb{N}$ be the integer whose binary representation is $\hat{\lambda} \in \{0,1\}^\ell$.

3. By the brute-force computation, M computes the binary representation of $p_{\lambda,x} = \Pr[x \leftarrow \mathcal{A}(1^\lambda)]$ for all $x \in \{0,1\}^{m(\lambda)}$ and stores them in the work tape.

4. For all $x \in \mathrm{Supp}(\mathcal{A}(1^\lambda))$, M computes $c(x)$ by using Shannon-Fano coding and stores the list $\{c(x)\}_{x \in \mathrm{Supp}(\mathcal{A}(1^\lambda))}$ in the work tape.

5. M reads the input tape from the $(2\ell + 2)$-th cell one by one. If there exists $x' \in \mathrm{Supp}(\mathcal{A}(1^\lambda))$ such that $y = c(x') \| z'$, where $z' = \{0, 1, \square\}$, then go to the next step. Otherwise, M continues reading the input tape, which means that M never halts.

6. M halts and outputs x'.

We evaluate the running time of M. Let $r : \mathbb{N} \to \mathbb{N}$ be the running time of $\mathcal{A}$. Because $\mathcal{A}$ is a PPT algorithm, r is a polynomial and therefore there exists a constant $a \geq 1$ such that $r(n) = O(n^a)$. For all $\lambda \in \mathbb{N}$ and for all $x \in \{0,1\}^{m(\lambda)}$, the binary expansion of p_x is computed in time $2^{O(r(\lambda))} = 2^{O(\lambda^a)}$ by the brute-force computation because the running time of $\mathcal{A}(1^\lambda)$ is $r(\lambda)$ and the number of random bits used is at most $O(r(\lambda))$. Thus for all $\lambda \in \mathbb{N}$, M can obtain $\{p_x\}_{x \in \{0,1\}^{m(\lambda)}}$ in time $2^{O(\lambda^a + m(\lambda))}$. Moreover in step 4, M obtains $\{c(x)\}_{x \in \mathrm{Supp}(\mathcal{A}(1^\lambda))}$ as follows:

[25]It is possible to check whether the input is in this form by scanning the input tape by the $(2\ell + 2)$-th cell in the one-way.

i. Sort $\{p_{\lambda,x}\}_{x\in \mathrm{Supp}(\mathcal{A}(1^\lambda))}$ in the descending order from left to right. For each $x \in \mathrm{Supp}(\mathcal{A}(1^\lambda))$, let N_x be the integer such that $p_{\lambda,x}$ appears in the N_x-th position of the sorted list.

ii. Let $P_1 := 0$. For all $2 \le i \le |\mathrm{Supp}(\mathcal{A}(1^\lambda))|$, let

$$P_i := \sum_{x' \in \mathrm{Supp}(\mathcal{A}(1^\lambda)) : N_{x'} < i} p_{\lambda,x'}. \tag{31}$$

iii. For all $x \in \mathrm{Supp}(\mathcal{A}(1^\lambda))$, the code word $c(x)$ of x is obtained by truncating the binary expansion of P_{N_x} such that its length $|c(x)|$ satisfies

$$\log \frac{1}{p_x} < |c(x)| \le \log \frac{1}{p_x} + 1. \tag{32}$$

Then, M can obtain the list $\{c(x)\}_{x\in \mathrm{Supp}(\mathcal{A}(1^\lambda))}$ in time $2^{O(m(\lambda))}$. Totally in step 3 and 4, M runs in time $2^{O(\lambda^a + m(\lambda))}$. Because $m(\lambda) \ge \lambda$ and $a \ge 1$, $\lambda^a + m(\lambda) = O(m(\lambda)^a + m(\lambda)) = O(m(\lambda)^a)$. This means M runs in time $2^{O(m(\lambda)^a)}$ in step 3 and 4.

Consider the case where the bit string $1^{|\hat{\lambda}|}\|0\|\hat{\lambda}\|c(x)$ with some $\lambda \in \mathbb{N}$ and some $x \in \mathrm{Supp}(\mathcal{A}(1^\lambda))$ is written from the leftmost cell of the input tape. Let all cells to the right of $1^{|\hat{\lambda}|}\|0\|\hat{\lambda}\|c(x)$ be blanks. Then, for all $\lambda \in \mathbb{N}$ and for all $x \in \mathrm{Supp}(\mathcal{A}(1^\lambda))$, $M(1^{|\hat{\lambda}|}\|0\|\hat{\lambda}\|c(x)) = x$ in at most $2^{O(|x|^a)}$ steps. This means that there exists a time-constructible function $T : \mathbb{N} \to \mathbb{N}$ such that $T(n) = 2^{O(n^a)}$ for some constant $a \ge 1$ and for all $\lambda \in \mathbb{N}$ and for all $x \in \mathrm{Supp}(\mathcal{A}(1^\lambda))$,

$$K_M^T(x) \le |1^{|\hat{\lambda}|}\|0\|\hat{\lambda}\|c(x)| \le |c(x)| + 2\lceil \log \lambda \rceil + 1 = |c(x)| + O(\log \lambda). \tag{33}$$

We complete the proof.

□

3 QASs and Int-QASs

In this section, we introduce two new concepts, quantum advantage samplers (QASs) and interactive quantum advantage samplers (Int-QASs). We also show some results on them.

3.1 Definitions of QASs and Int-QASs

We first define QASs on a set $\Sigma \subseteq \mathbb{N}$. If $\Sigma = \mathbb{N}$, we call them just QASs.

Definition 3.1 (Quantum Advantage Samplers (QASs) on Σ). *Let $\Sigma \subseteq \mathbb{N}$ be a set. Let $\mathcal{A}$ be a QPT algorithm that takes 1^λ as input and outputs a classical string. $\mathcal{A}$ is a quantum advantage sampler (QAS) on Σ if the following is satisfied: There exists a polynomial p such that for any PPT algorithm $\mathcal{B}$ (that takes 1^λ as input and outputs a classical string) there exists $\lambda^* \in \mathbb{N}$ such that*

$$\mathsf{SD}(\mathcal{A}(1^\lambda), \mathcal{B}(1^\lambda)) > \frac{1}{p(\lambda)} \tag{34}$$

holds for all $\lambda \ge \lambda^*$ in Σ.

Remark 3.2. For any finite set Σ, QASs on Σ always exist, but we include the case in the definition for the convenience.

We also define interactive versions of QASs, which we call Int-QASs, as follows.

Definition 3.3 (Interactive Quantum Advantage Samplers (Int-QASs)). *Let $(\mathcal{A},\mathcal{C})$ be a tuple of two interactive QPT algorithms $\mathcal{A}$ and $\mathcal{C}$ that communicate over a classical channel. $(\mathcal{A},\mathcal{C})$ is an interactive quantum advantage sampler (Int-QAS) if the following is satisfied: There exists a polynomial p such that for any PPT algorithm $\mathcal{B}$ that interacts with $\mathcal{C}$,*

$$\mathsf{SD}(\langle\mathcal{A},\mathcal{C}\rangle(1^\lambda),\langle\mathcal{B},\mathcal{C}\rangle(1^\lambda)) > \frac{1}{p(\lambda)} \tag{35}$$

holds for all sufficiently large $\lambda\in\mathbb{N}$. Here, $\langle\mathcal{A},\mathcal{C}\rangle(1^\lambda)$ (resp. $\langle\mathcal{B},\mathcal{C}\rangle(1^\lambda)$) is the probability distribution over the transcript of the interaction between $\mathcal{A}$ (resp. $\mathcal{B}$) and $\mathcal{C}$.

3.2 Relation Between QASs and Sampling Complexity Classes

We can show that the existence of QASs implies $\mathbf{SampBPP}\neq\mathbf{SampBQP}$.

Lemma 3.4. *Let $\Sigma\subseteq\mathbb{N}$ be an infinite subset. If QASs on Σ exist, then* $\mathbf{SampBPP}\neq\mathbf{SampBQP}$.

Proof of Lemma 3.4. Let $\mathcal{A}$ be a QAS on an infinite subset $\Sigma\subseteq\mathbb{N}$ and let $\mathcal{A}(1^\lambda)$ be the output distribution of $\mathcal{A}$ on input 1^λ. Consider the collection $\{\mathcal{A}(1^\lambda)\}_{\lambda\in\Sigma}$ of distributions. Then clearly $\{\mathcal{A}(1^\lambda)\}_{\lambda\in\Sigma}\in\mathbf{SampBQP}$. What we need to show is that $\{\mathcal{A}(1^\lambda)\}_{\lambda\in\Sigma}\notin\mathbf{SampBPP}$. Because $\mathcal{A}$ is a QAS on Σ, there exists a polynomial p such that for any PPT algorithm $\mathcal{B}$, there exists $\lambda^*\in\mathbb{N}$ such that $\mathsf{SD}(\mathcal{A}(1^\lambda),\mathcal{B}(1^\lambda))>\frac{1}{p(\lambda)}$ for all $\lambda\geq\lambda^*$ in Σ. Because Σ is an infinite set, there exists $x\in\Sigma$ such that $x\geq\lambda^*$. Then for any PPT algorithm $\mathcal{B}$, there exists x and $1/p(x)$ such that $\mathsf{SD}(\mathcal{A}(1^x),\mathcal{B}(1^x))>\frac{1}{p(x)}$, which means that $\{\mathcal{A}(1^\lambda)\}_{\lambda\in\Sigma}\notin\mathbf{SampBPP}$. □

Remark 3.5. Note that the other direction, namely, $\mathbf{SampBPP}\neq\mathbf{SampBQP}$ implies the existence of QASs, does not seem to hold, because of the following reason: Assume that $\mathbf{SampBPP}\neq\mathbf{SampBQP}$. Then there exists a sampling problem $\{D_x\}_x$ that is in $\mathbf{SampBQP}$ but not in $\mathbf{SampBPP}$. The fact that $\{D_x\}_x\notin\mathbf{SampBPP}$ means that for any PPT algorithm $\mathcal{B}$, there exist x and $\epsilon>0$ such that

$$\mathsf{SD}(D_x,\mathcal{B}(x,1^{\lfloor 1/\epsilon\rfloor}))>\epsilon. \tag{36}$$

This does not necessarily mean that a QPT algorithm $\mathcal{A}$ that samples $\{D_x\}_x$ is a QAS. For example, ϵ in Equation (36) could be $2^{-|x|}$.

3.3 Non-Interactive IV-PoQ Imply QASs

In this subsection, we show the following.

Lemma 3.6. *Let $\Sigma\subseteq\mathbb{N}$ be an infinite subset. If non-interactive IV-PoQ on Σ exist, then QASs on Σ exist.*

Proof of Lemma 3.6. Let $(\mathcal{P},\mathcal{V}_1,\mathcal{V}_2)$ be a non-interactive IV-PoQ on Σ with c-completeness and s-soundness such that $c(\lambda)-s(\lambda)\geq\frac{1}{q(\lambda)}$ for some polynomial q. $\mathcal{P}$ takes 1^λ as input and outputs τ. $\mathcal{V}_1$ does nothing. $\mathcal{V}_2$ takes 1^λ and τ as input and outputs $\top/\bot$. Then, we can show that $\mathcal{P}$ is a QAS on Σ. For the sake of contradiction, assume that $\mathcal{P}$ is not a QAS on Σ. Then, for any polynomial p, there exists a PPT algorithm $\mathcal{P}_p^*$ such that

$$\mathsf{SD}(\mathcal{P}(1^\lambda),\mathcal{P}_p^*(1^\lambda)) \le \frac{1}{p(\lambda)} \tag{37}$$

holds for infinitely many $\lambda \in \Sigma$. $\mathcal{P}_p^*$ can break the s-soundness of the non-interactive IV-PoQ. In fact, if we take $p = 2q$,

$$\Pr[\top \leftarrow \mathcal{V}_2(1^\lambda,\tau) : \tau \leftarrow \mathcal{P}_{2q}^*(1^\lambda)] \ge \Pr[\top \leftarrow \mathcal{V}_2(1^\lambda,\tau) : \tau \leftarrow \mathcal{P}(1^\lambda)] - \mathsf{SD}(\mathcal{P}(1^\lambda),\mathcal{P}_{2q}^*(1^\lambda)) \tag{38}$$

$$\ge c(\lambda) - \frac{1}{2q(\lambda)} \tag{39}$$

$$> s(\lambda) \tag{40}$$

holds for infinitely many $\lambda \in \Sigma$, which breaks s-soundness. □

3.4 QASs Imply Non-Interactive IV-PoQ

The goal of this subsection is to show the following.

Lemma 3.7. *Let $\Sigma \subseteq \mathbb{N}$ be an infinite subset. If QASs on Σ exist, then non-interactive IV-PoQ on Σ exist.*

We can directly show Lemma 3.7. However, we first show the following lemma, Lemma 3.8, because it is more convenient for our later proofs. We then show that if $\mathcal{A}$ in Lemma 3.8 is a QAS, then $(\mathcal{P},\mathcal{V})$ in Lemma 3.8 is actually a non-interactive IV-PoQ, which shows Lemma 3.7.

Lemma 3.8. *Let $\mathcal{A}$ be a QPT algorithm that takes 1^λ as input and outputs a classical string. For any polynomial q, there exists the following non-interactive protocol $(\mathcal{P},\mathcal{V})$:*

- *$\mathcal{P}$ is a QPT algorithm that, on input 1^λ, outputs a bit string τ. $\mathcal{V}$ is an unbounded algorithm that, on input 1^λ and a bit string τ, outputs $\top/\bot$.*
- *For all sufficiently large $\lambda \in \mathbb{N}$,*

$$\Pr[\top \leftarrow \mathcal{V}(1^\lambda,\tau) : \tau \leftarrow \mathcal{P}(1^\lambda)] \ge 1 - \frac{1}{\lambda^{\log\lambda}}. \tag{41}$$

- *If there exist a PPT algorithm $\mathcal{P}^*$ and an infinite subset $\Lambda \subseteq \mathbb{N}$ such that*

$$\Pr[\top \leftarrow \mathcal{V}(1^\lambda,\tau) : \tau \leftarrow \mathcal{P}^*(1^\lambda)] \ge 1 - \frac{1}{\lambda^{\log\lambda}} - \frac{1}{q(\lambda)^2} \tag{42}$$

  holds for all $\lambda \in \Lambda$, then there exists a PPT algorithm $\mathcal{B}$ such that

$$\mathsf{SD}(\mathcal{A}(1^\lambda),\mathcal{B}(1^\lambda)) \le \frac{1}{q(\lambda)} \tag{43}$$

  holds for all sufficiently large $\lambda \in \Lambda$.

Proof of Lemma 3.8. We use the technique [Aar14] of constructing a search problem from a sampling problem. Let $\mathcal{A}(1^\lambda) \to \{0,1\}^{m(\lambda)}$ be a QPT algorithm and $p_y := \Pr[y \leftarrow \mathcal{A}(1^\lambda)]$ be the probability that $\mathcal{A}(1^\lambda)$ outputs $y \in \{0,1\}^{m(\lambda)}$, where m is a polynomial.

For any polynomial q, we construct a non-interactive protocol $(\mathcal{P},\mathcal{V})$ as follows.

1. If $q(\lambda)^4 \geq \lambda$, set $N(\lambda) := q(\lambda)^4$. Otherwise, set $N(\lambda) := \lambda$. $\mathcal{P}$ runs $y_i \leftarrow \mathcal{A}(1^\lambda)$ for each $i \in [N]$, and sends $(y_1, ..., y_N)$ to $\mathcal{V}$.

2. $\mathcal{V}$ outputs $\top$ if
$$K_U^T(y_1, ..., y_N) \geq \log \frac{1}{p_{y_1} \cdots p_{y_N}} - \log^2 \lambda, \tag{44}$$
where K_U^T denotes time-bounded prefix Kolmogorov complexity with time-bound $T(mN) := 2^{2^{mN}}$ and the universal self-delimiting machine U. Otherwise $\mathcal{V}$ outputs $\perp$. Note that given $(y_1, ..., y_N)$, whether $(y_1, ..., y_N)$ satisfies Equation (44) or not can be checked in an unbounded time, because K_U^T is computable.

We first show the second item of the lemma. From Lemma 2.28,
$$\Pr_{(y_1,...,y_N)\leftarrow\mathcal{A}(1^\lambda)^{\otimes N}}\left[K_U^T(y_1, ..., y_N) \geq \log \frac{1}{p_{y_1} \cdots p_{y_N}} - \log^2 \lambda\right] \geq 1 - \frac{1}{\lambda^{\log \lambda}} \tag{45}$$
for all sufficiently large $\lambda \in \mathbb{N}$, where $\mathcal{A}(1^\lambda)^{\otimes N}$ means that $\mathcal{A}(1^\lambda)$ is executed N times. Here we have used the facts that $m(\lambda)N(\lambda) \geq \lambda$ for all $\lambda \in \mathbb{N}$ and $T(n) = 2^{2^n} = \Omega(n^2)$ to apply Lemma 2.28. Thus,
$$\Pr[\top \leftarrow \mathcal{V}(1^\lambda, y_1, ..., y_N) : (y_1, ..., y_N) \leftarrow \mathcal{P}(1^\lambda)] \tag{46}$$
$$= \Pr_{(y_1,...,y_N)\leftarrow\mathcal{A}(1^\lambda)^{\otimes N}}\left[K_U^T(y_1, ..., y_N) \geq \log \frac{1}{p_{y_1} \cdots p_{y_N}} - \log^2 \lambda\right] \tag{47}$$
$$\geq 1 - \frac{1}{\lambda^{\log \lambda}} \tag{48}$$
for all sufficiently large $\lambda \in \mathbb{N}$.

We next show the third item of the lemma. We assume that there exists a PPT algorithm $\mathcal{P}^*$ and an infinite subset $\Lambda \subseteq \mathbb{N}$ such that
$$\Pr[\top \leftarrow \mathcal{V}(1^\lambda, y_1, ..., y_N) : (y_1, ..., y_N) \leftarrow \mathcal{P}^*(1^\lambda)] \geq 1 - \frac{1}{\lambda^{\log \lambda}} - \frac{1}{q(\lambda)^2} \tag{49}$$
holds for all $\lambda \in \Lambda$. From this $\mathcal{P}^*$, we construct the following PPT algorithm $\mathcal{B}$.

- $\mathcal{B}(1^\lambda) \to y$:
    1. Take 1^λ as input.
    2. Run $(y_1, ..., y_N) \leftarrow \mathcal{P}^*(1^\lambda)$.
    3. Sample $i \leftarrow [N]$.
    4. Output $y := y_i$.

Our goal is to show
$$\mathsf{SD}(\mathcal{A}(1^\lambda), \mathcal{B}(1^\lambda)) \leq \frac{1}{N^{1/4}(\lambda)} \leq \frac{1}{q(\lambda)} \tag{50}$$

for all sufficiently large $\lambda \in \Lambda$. We introduce the following projection

$$\Pi := \sum_{(y_1,...,y_N):\top \leftarrow \mathcal{V}(1^\lambda,y_1,...,y_N)} |y_1,...,y_N\rangle\langle y_1,...,y_N| \tag{51}$$

and we define the following quantum states:

$$C := \sum_{y_1,...,y_N} \Pr[(y_1,...,y_N) \leftarrow \mathcal{P}^*(1^\lambda)]|y_1,...,y_N\rangle\langle y_1,...,y_N| \tag{52}$$

$$C' := \frac{\Pi C}{\mathrm{Tr}(\Pi C)} \tag{53}$$

$$C_i := \mathrm{Tr}_{\neg i}(C) = \sum_{y_1,...,y_i,...,y_N} \Pr[(y_1,...,y_i,...,y_N) \leftarrow \mathcal{P}^*(1^\lambda)]|y_i\rangle\langle y_i| \tag{54}$$

$$C_i' := \mathrm{Tr}_{\neg i}(C') \tag{55}$$

$$B := \mathbb{E}_{i\leftarrow[N]}C_i \tag{56}$$

$$B' := \mathbb{E}_{i\leftarrow[N]}C_i' \tag{57}$$

$$A := \sum_y \Pr[y \leftarrow \mathcal{A}(1^\lambda)]|y\rangle\langle y|. \tag{58}$$

Here, $\mathrm{Tr}_{\neg i}$ refers to the operation of tracing out all coordinates except for the i-th coordinate. Then, by the triangle inequality,

$$\mathsf{SD}(\mathcal{A}(1^\lambda), \mathcal{B}(1^\lambda)) = \mathsf{TD}(A,B) \leq \mathsf{TD}(B,B') + \mathsf{TD}(A,B'). \tag{59}$$

In the following, we bound $\mathsf{TD}(B,B')$ and $\mathsf{TD}(A,B')$, respectively.

First, we can bound $\mathsf{TD}(B,B')$ as follows:

$$\begin{aligned}
\mathsf{TD}(B,B') &= \mathsf{TD}(\mathbb{E}_iC_i,\mathbb{E}_iC_i') && (60)\\
&\le \mathbb{E}_i\mathsf{TD}(C_i,C_i') \quad \text{(By the strong convexity of TD.)} && (61)\\
&= \mathbb{E}_i\mathsf{TD}(\mathrm{Tr}_{\neg i}(C),\mathrm{Tr}_{\neg i}(C')) && (62)\\
&\le \mathbb{E}_i\mathsf{TD}(C,C') && (63)\\
&= \mathsf{TD}(C,C') && (64)\\
&= \frac{1}{2}\left\|C-\frac{\Pi C}{\mathrm{Tr}(\Pi C)}\right\|_1 && (65)\\
&\le \frac{1}{2}\|(I-\Pi)C\|_1+\frac{1}{2}\left\|\Pi C-\frac{\Pi C}{\mathrm{Tr}(\Pi C)}\right\|_1 \quad \text{(By the triangle inequality.)} && (66)\\
&= \frac{1}{2}\mathrm{Tr}|(I-\Pi)C|+\frac{1}{2}\left\|(\mathrm{Tr}(\Pi C)-1)\frac{\Pi C}{\mathrm{Tr}(\Pi C)}\right\|_1 && (67)\\
&= \frac{1}{2}\mathrm{Tr}|(I-\Pi)C|+\frac{1}{2}|\mathrm{Tr}(\Pi C)-1|\left\|\frac{\Pi C}{\mathrm{Tr}(\Pi C)}\right\|_1 && (68)\\
&= \frac{1}{2}\mathrm{Tr}((I-\Pi)C)+\frac{1}{2}(1-\mathrm{Tr}(\Pi C))\left\|\frac{\Pi C}{\mathrm{Tr}(\Pi C)}\right\|_1 && (69)\\
&= 1-\mathrm{Tr}(\Pi C) && (70)\\
&= 1-\sum_{(y_1,...,y_N):\top\leftarrow\mathcal{V}(1^\lambda,y_1,...,y_N)}\Pr[(y_1,...,y_N)\leftarrow\mathcal{P}^*(1^\lambda)] && (71)\\
&\le \frac{1}{\lambda^{\log\lambda}}+\frac{1}{\sqrt{N(\lambda)}} \quad \text{(By Equation (49).)} && (72)
\end{aligned}$$

holds for all $\lambda\in\Lambda$. To derive Equation (69), we have used the fact that $(I-\Pi)C$ is positive semidefinite.

Next, we evaluate the upper bound of $\mathsf{TD}(A,B')$. Let c'_{i,y_i} be the eigenvalue of C_i' with respect to the eigenstate $|y_i\rangle$. Then,

$$\begin{aligned}
\mathsf{TD}(A,B') &\le \frac{1}{N}\sum_{i=1}^{N}\mathsf{TD}(C_i',A) && (73)\\
&= \frac{1}{N}\sum_{i=1}^{N}\mathsf{SD}(\{c'_{i,y_i}\}_{y_i},\{p_y\}_y) && (74)\\
&\le \sqrt{\frac{1}{N}\sum_{i=1}^{N}\mathsf{SD}(\{c'_{i,y_i}\}_{y_i},\{p_y\}_y)^2} \quad \text{(By Jensen's inequality.)} && (75)\\
&\le \sqrt{\frac{\ln 2}{2N}\sum_{i=1}^{N}D_{\mathrm{KL}}(\{c'_{i,y_i}\}_{y_i}\|\{p_y\}_y)}. \quad \text{(By Pinsker's inequality.)} && (76)
\end{aligned}$$

Here, D_{KL} denotes the KL-divergence defined as

$$D_{\mathrm{KL}}(\{p_x\}_x\|\{q_x\}_x) := \begin{cases} \sum_x p_x\log\frac{p_x}{q_x} & \mathrm{Supp}(\{p_x\}_x)\subseteq\mathrm{Supp}(\{q_x\}_x)\\ \infty & \text{otherwise,}\end{cases} \quad (77)$$

where Supp is the support, and Pinsker's inequality is the following one (Chapter 3 of [CK11]):

$$\mathsf{SD}(\{p_x\}_x,\{q_x\}_x)\le\sqrt{\frac{\ln 2}{2}D_{\mathrm{KL}}(\{p_x\}_x\|\{q_x\}_x)}. \tag{78}$$

Let $Y:=(y_1,...,y_N)$. Let $p_Y:=\prod_{i=1}^N p_{y_i}$. Let q_Y (resp. q'_Y) be the eigenvalue of C (resp. C') with respect to the eigenstate $|Y\rangle$. Define $G:=\left\{Y\middle|\top\leftarrow\mathcal{V}(1^\lambda,Y)\wedge\log\frac{q'_Y}{p_Y}>0\right\}$ and $G':=\left\{Y\middle|\top\leftarrow\mathcal{V}(1^\lambda,Y)\right\}$. Since $\{c'_{i,y_i}\}_{y_i}$ is a marginal distribution of $\{q'_Y\}_Y$,

$$\begin{aligned}
\sum_{i=1}^N D_{\mathrm{KL}}(\{c'_{i,y_i}\}_{y_i}\|\{p_y\}_y) &\le D_{\mathrm{KL}}(\{q'_Y\}_Y\|\{p_Y\}_Y) \quad \text{(See below.)} &(79)\\
&=\sum_Y q'_Y\log\frac{q'_Y}{p_Y} &(80)\\
&=\sum_{Y:\top\leftarrow\mathcal{V}(1^\lambda,Y)} q'_Y\log\frac{q'_Y}{p_Y} &(81)\\
&\le\sum_{Y\in G} q'_Y\log\frac{q'_Y}{p_Y} &(82)\\
&\le\frac{1}{1-\delta}\sum_{Y\in G}q_Y\left(\log\frac{q_Y}{p_Y}+\log\frac{1}{1-\delta}\right) \quad \text{(See below.)} &(83)\\
&\le\frac{1}{1-\delta}\sum_{Y\in G}q_Y\left(\log\frac{q_Y}{p_Y}+1\right) \quad \left(\text{If } \delta\le\frac{1}{2}, \text{ then } \log\frac{1}{1-\delta}\le 1.\right) &(84)\\
&\le\frac{1}{1-\delta}\left(\max_{Y\in G'}\log\frac{q_Y}{p_Y}+1\right)\sum_{Y\in G}q_Y &(85)\\
&\le\frac{1}{1-\delta}\left(\max_{Y\in G'}\log\frac{q_Y}{p_Y}+1\right) &(86)
\end{aligned}$$

holds for all sufficiently large $\lambda\in\Lambda$, where $\delta:=\frac{1}{\lambda^{\log\lambda}}+\frac{1}{\sqrt{N(\lambda)}}$, which is less than $1/2$ for sufficiently large λ. In Equation (79), we have used the following lemma.

Lemma 3.9 ([Rao08]). *Let $\mathcal{R}$ be a distribution over $[M]^N$, with the marginal distribution $\mathcal{R}_i$ on the i-th coordinate. Let $\mathcal{D}$ be a distribution over $[M]$. Then*

$$\sum_{i=1}^N D_{\mathrm{KL}}(\mathcal{R}_i\|\mathcal{D})\le D_{\mathrm{KL}}(\mathcal{R}\|\mathcal{D}^N). \tag{87}$$

In Equation (83), we have used the fact that $q'_Y\le\frac{q_Y}{1-\delta}$, which can be shown as follows. First, when

$Y \notin G'$, $q'_Y = 0$ and the inequality trivially holds. Second, when $Y \in G'$,

$$\begin{aligned} q_Y &= \langle Y|C|Y\rangle & (88)\\ &= \langle Y|\Pi C|Y\rangle \quad (\text{Because } Y \in G'.) & (89)\\ &= \mathrm{Tr}(\Pi C)\langle Y|C'|Y\rangle \quad (\text{By the definition of } C'.) & (90)\\ &= q'_Y \sum_{Y:\top\leftarrow\mathcal{V}(1^\lambda,Y)} \Pr[Y \leftarrow \mathcal{P}^*(1^\lambda)] & (91)\\ &= q'_Y \Pr[\top \leftarrow \mathcal{V}(1^\lambda, Y) : (1^\lambda, Y) \leftarrow \mathcal{P}^*(1^\lambda)] & (92)\\ &\geq q'_Y(1-\delta). & (93)\end{aligned}$$

Here the last inequality is from Equation (49).

By Lemma 2.29, for all sufficiently large λ and all $Y \in \mathrm{Supp}(\mathcal{P}^*(1^\lambda))$,

$$K_U^T(Y) \leq \log\frac{1}{q_Y} + O(\log\lambda). \tag{94}$$

Here we have used the facts that $m(\lambda)N(\lambda) \geq \lambda$ for all $\lambda \in \mathbb{N}$ and $T(n) = 2^{2^n} = 2^{\omega(\mathrm{poly}(n))}$ to apply Lemma 2.29.

By combining Equations (44) and (94), we obtain

$$\log\frac{1}{p_{y_1}\cdots p_{y_N}} \leq \log\frac{1}{q_Y} + O(\log\lambda) + \log^2\lambda \tag{95}$$

for any $Y \in G' \cap \mathrm{Supp}(\mathcal{P}^*(1^\lambda))$. Thus,

$$\max_{Y\in G'} \log\frac{q_Y}{p_Y} = \max_{Y\in G'\cap\mathrm{Supp}(\mathcal{P}^*(1^\lambda))} \log\frac{q_Y}{p_Y} \leq O(\log\lambda) + \log^2\lambda. \tag{96}$$

From Equations (76), (86) and (96),

$$\begin{aligned} \mathsf{TD}(A, B') &\leq \sqrt{\frac{\ln 2(\log^2\lambda + O(\log\lambda))}{2(1-\delta)N}} & (97)\\ &\leq \sqrt{\frac{\log^2\lambda + O(\log\lambda)}{N}} \quad \left(\text{If } \delta \leq 1 - \frac{\ln 2}{2}, \text{ then } \frac{1}{1-\delta} \leq \frac{2}{\ln 2}.\right) & (98)\\ &\leq \sqrt{\frac{2\log^2\lambda}{N}} & (99)\end{aligned}$$

holds for all sufficiently large $\lambda \in \Lambda$. Therefore, from Equations (72) and (99)

$$\mathsf{SD}(\mathcal{A}(1^\lambda), \mathcal{B}(1^\lambda)) \leq \frac{1}{\lambda^{\log\lambda}} + \frac{1}{\sqrt{N}} + \sqrt{\frac{2\log^2\lambda}{N}} \leq \frac{N^{\frac{1}{4}}}{\sqrt{N}} = \frac{1}{N^{\frac{1}{4}}} \tag{100}$$

holds for all sufficiently large $\lambda \in \Lambda$. □

Now we show Lemma 3.7 by arguing that if $\mathcal{A}$ in Lemma 3.8 is a QAS, then the $(\mathcal{P}, \mathcal{V})$ in Lemma 3.8 is actually a non-interactive IV-PoQ.

Proof of Lemma 3.7. Let $\Sigma \subseteq \mathbb{N}$ be an infinite subset. Let $\mathcal{A}$ be a QAS on Σ. Then, from the definition of QASs on Σ, there exists a polynomial q such that for any PPT algorithm $\mathcal{B}$ there exists $\lambda^* \in \mathbb{N}$ such that

$$\mathsf{SD}(\mathcal{A}(1^\lambda), \mathcal{B}(1^\lambda)) > \frac{1}{q(\lambda)} \tag{101}$$

holds for all $\lambda \geq \lambda^*$ in Σ. From Lemma 3.8, we can construct a non-interactive protocol $(\mathcal{P}, \mathcal{V})$ from the above $\mathcal{A}$ and q. We show that $(\mathcal{P}, \mathcal{V})$ is actually a non-interactive IV-PoQ on Σ with $(1 - \frac{1}{\lambda^{\log \lambda}})$-completeness and $(1 - \frac{1}{\lambda^{\log \lambda}} - \frac{1}{q(\lambda)^2})$-soundness.

First, the completeness is obtained from the second item of Lemma 3.8. Next, we show the soundness. For the sake of contradiction, we assume that the soundness is not satisfied, which means that there exist a PPT algorithm $\mathcal{P}^*$ and an infinite subset $\Lambda \subseteq \Sigma$ such that

$$\Pr[\top \leftarrow \mathcal{V}(1^\lambda, \tau) : \tau \leftarrow \mathcal{P}^*(1^\lambda)] > 1 - \frac{1}{\lambda^{\log \lambda}} - \frac{1}{q(\lambda)^2} \tag{102}$$

holds for all $\lambda \in \Lambda$. Then, from the third item of Lemma 3.8, there exists a PPT algorithm $\mathcal{B}^*$ such that

$$\mathsf{SD}(\mathcal{A}(1^\lambda), \mathcal{B}^*(1^\lambda)) \leq \frac{1}{q(\lambda)} \tag{103}$$

holds for all sufficiently large $\lambda \in \Lambda$. This contradicts Equation (101) because Λ is an infinite subset of Σ. Therefore, the soundness is satisfied. □

4 The QAS/OWF Condition

We also introduce another new concept, which we call the QAS/OWF condition.

Definition 4.1 (The QAS/OWF Condition). *The QAS/OWF condition holds if there exist a polynomial p, a QPT algorithm $\mathcal{Q}$ that takes 1^λ as input and outputs a classical string, and a function $f : \{0,1\}^* \to \{0,1\}^*$ that is computable in classical deterministic polynomial-time such that for any PPT algorithm $\mathcal{S}$, the following holds: if we define*

$$\Sigma_{\mathcal{S}} := \left\{ \lambda \in \mathbb{N} \,\middle|\, \mathsf{SD}(\mathcal{Q}(1^\lambda), \mathcal{S}(1^\lambda)) \leq \frac{1}{p(\lambda)} \right\}, \tag{104}$$

then f is a classically-secure OWF on $\Sigma_{\mathcal{S}}$.

We can show the following result:

Theorem 4.2. *If the QAS/OWF condition is satisfied, then quantumly-secure OWFs exist or* $\mathbf{SampBPP} \neq \mathbf{SampBQP}$.

Theorem 1.2 is obtained by combining this theorem and the equivalence of IV-PoQ and the QAS/OWF condition, which will be shown in Section 5.

Proof of Theorem 4.2. We first show that the QAS/OWF condition implies the existence of classically-secure OWFs or $\mathbf{SampBPP} \neq \mathbf{SampBQP}$. Let us assume that the QAS/OWF condition is satisfied. Then, by the definition of the QAS/OWF condition, there exist a polynomial p, a QPT algorithm $\mathcal{Q}$, and a function f that is computable in classical deterministic polynomial-time such that for any PPT algorithm $\mathcal{S}$, if we define

$$\Sigma_{\mathcal{S}} := \left\{ \lambda \in \mathbb{N} \,\middle|\, \mathsf{SD}(\mathcal{Q}(1^\lambda), \mathcal{S}(1^\lambda)) \leq \frac{1}{p(\lambda)} \right\}, \tag{105}$$

then f is a classically-secure OWF on $\Sigma_{\mathcal{S}}$. We divide the proof into the following two cases:

There exist a PPT algorithm $\mathcal{S}$ and a finite subset $\Lambda \subseteq \mathbb{N}$ such that $\Sigma_{\mathcal{S}} = \mathbb{N} \setminus \Lambda$. In this case, from Lemma 2.5, classically-secure OWFs exist.

For any PPT algorithm $\mathcal{S}$ and for any finite subset $\Lambda \subseteq \mathbb{N}$, $\Sigma_{\mathcal{S}} \neq \mathbb{N} \setminus \Lambda$. In this case, if we define the sampling problem $\{\mathcal{Q}(1^\lambda)\}_{\lambda\in\mathbb{N}}$, for any PPT algorithm $\mathcal{S}$, there exists an $x \in \mathbb{N} \setminus \Sigma_{\mathcal{S}}$ such that

$$\mathsf{SD}(\mathcal{Q}(1^x), \mathcal{S}(1^x)) > \frac{1}{p(x)}, \tag{106}$$

which means that $\{\mathcal{Q}(1^\lambda)\}_{\lambda\in\mathbb{N}} \notin \mathbf{SampBPP}$. On the other hand, it is clear that $\{\mathcal{Q}(1^\lambda)\}_{\lambda\in\mathbb{N}} \in \mathbf{SampBQP}$ and therefore $\mathbf{SampBPP} \neq \mathbf{SampBQP}$.

We next show that if classically-secure OWFs exist, then quantumly-secure OWFs exist or $\mathbf{SampBPP} \neq \mathbf{SampBQP}$. Let $f : \{0,1\}^* \to \{0,1\}^*$ be a classically-secure OWF. For the sake of contradiction, we assume that quantumly-secure OWFs do not exist and $\mathbf{SampBPP} = \mathbf{SampBQP}$. Then, there exists a QPT adversary $\mathcal{A}$ and a polynomial p such that

$$\Pr[f(z) = f(x) : x \leftarrow \{0,1\}^\lambda, z \leftarrow \mathcal{A}(1^\lambda, f(x))] \geq \frac{1}{p(\lambda)} \tag{107}$$

holds for infinitely many $\lambda \in \mathbb{N}$. Let $\mathcal{A}(1^\lambda, f(x))$ be the output distribution of $\mathcal{A}$ on input $(1^\lambda, f(x))$. Then, $\{\mathcal{A}(1^\lambda, f(x))\}_{\lambda, f(x)} \in \mathbf{SampBQP}$ and therefore $\{\mathcal{A}(1^\lambda, f(x))\}_{\lambda, f(x)} \in \mathbf{SampBPP}$. (Remember that we have assumed $\mathbf{SampBQP} = \mathbf{SampBPP}$.) Thus, by the definition of $\mathbf{SampBPP}$, there exists a PPT algorithm $\mathcal{B}$ such that $\mathsf{SD}(\mathcal{A}(1^\lambda, f(x)), \mathcal{B}(1^\lambda, f(x))) \leq \frac{1}{2p(\lambda)}$ for all λ and all $f(x)$. Then, for infinitely many $\lambda \in \mathbb{N}$,

$$\Pr[f(z) = f(x) : x \leftarrow \{0,1\}^\lambda, z \leftarrow \mathcal{B}(1^\lambda, f(x))] \tag{108}$$
$$\geq \Pr[f(z) = f(x) : x \leftarrow \{0,1\}^\lambda, z \leftarrow \mathcal{A}(1^\lambda, f(x))] - \mathsf{SD}(\mathcal{A}(1^\lambda, f(x)), \mathcal{B}(1^\lambda, f(x))) \tag{109}$$
$$> \frac{1}{p(\lambda)} - \frac{1}{2p(\lambda)} = \frac{1}{2p(\lambda)}. \tag{110}$$

This means that f is not classically-secure, which is the contradiction. □

We show a lemma that gives a rephrasing of the negation of the QAS/OWF condition. Before presenting the lemma, we explain its intuition and motivation. A straightforward negation of the QAS/OWF condition gives the following: For any polynomial p, QPT algorithm $\mathcal{Q}$, and a polynomial-time computable function f, there is a PPT algorithm $\mathcal{S}$ such that f is not a classically-secure OWF on $\Sigma_{\mathcal{S}}$ where $\Sigma_{\mathcal{S}}$ is as defined in Definition 4.1. In this statement, f is not allowed to depend on $\mathcal{S}$. On the other hand, in the proofs of Theorems 5.2 and 5.5, we need to allow f to depend on $\mathcal{S}$ due to a technical reason. The following lemma roughly shows that we can change the order of quantifiers of f and $\mathcal{S}$ in the above statement so that f can depend on $\mathcal{S}$. Moreover, we require that f's distributional one-wayness (rather than one-wayness) is broken on $\Sigma_{\mathcal{S}}$. The formal statement is given below.

Lemma 4.3. *If the QAS/OWF condition is not satisfied, then the following statement is satisfied: for any QPT algorthm $\mathcal{Q}$ that takes 1^λ as input and outputs a classical string and for any polynomial p, there exists a PPT algorithm $\mathcal{S}$ such that for any efficiently-computable polynomial n and any family $\{f_\lambda : \{0,1\}^{n(\lambda)} \to \{0,1\}^*\}_{\lambda\in\mathbb{N}}$ of functions that are computable in classical deterministic polynomial-time, there exists a PPT algorithm $\mathcal{R}$ such that*

$$\mathsf{SD}(\mathcal{Q}(1^\lambda), \mathcal{S}(1^\lambda)) \leq \frac{1}{p(\lambda)} \tag{111}$$

and

$$\mathsf{SD}(\{x, f_\lambda(x)\}_{x\leftarrow\{0,1\}^{n(\lambda)}}, \{\mathcal{R}(1^{n(\lambda)}, f_\lambda(x)), f_\lambda(x)\}_{x\leftarrow\{0,1\}^{n(\lambda)}}) \leq \frac{1}{p(\lambda)} \tag{112}$$

hold for infinitely many $\lambda \in \mathbb{N}$.

Proof of Lemma 4.3. For the sake of contradiction, we assume the following:

Assumption 1. There exist a QPT algorithm $\mathcal{Q}$ and a polynomial p such that for any PPT algorithm $\mathcal{S}$, there exist an efficiently-computable polynomial n and a family $\{f_\lambda^{\mathcal{S}} : \{0,1\}^{n(\lambda)} \to \{0,1\}^*\}_{\lambda\in\mathbb{N}}$ of functions that are computable in classical deterministic polynomial-time such that for any PPT algorithm $\mathcal{R}$,

$$\mathsf{SD}(\mathcal{Q}(1^\lambda), \mathcal{S}(1^\lambda)) > \frac{1}{p(\lambda)} \tag{113}$$

or

$$\mathsf{SD}(\{x, f_\lambda^{\mathcal{S}}(x)\}_{x\leftarrow\{0,1\}^{n(\lambda)}}, \{\mathcal{R}(1^{n(\lambda)}, f_\lambda^{\mathcal{S}}(x)), f_\lambda^{\mathcal{S}}(x)\}_{x\leftarrow\{0,1\}^{n(\lambda)}}) > \frac{1}{p(\lambda)} \tag{114}$$

holds for all sufficiently large $\lambda \in \mathbb{N}$.

It suffices to show that Assumption 1 implies the QAS/OWF condition. By using the above $\{f_\lambda^{\mathcal{S}}\}_{\lambda\in\mathbb{N}}$, we define a function $f_{\mathcal{S}} : \{0,1\}^* \to \{0,1\}^*$ that is computable in classical deterministic polynomial-time as follows:

- $f_{\mathcal{S}} : \{0,1\}^* \to \{0,1\}^*$:
  1. Take $x \in \{0,1\}^\ell$ as input.
  2. Let λ^* be the maximum λ such that $n(\lambda) \leq \ell$. Let x' be the $n(\lambda^*)$-bit prefix of x.
  3. Output $f_{\lambda^*}^{\mathcal{S}}(x')$.

Then, for any λ that satisfies Equation (114),

$$\mathsf{SD}(\{x, f_{\mathcal{S}}(x)\}_{x\leftarrow\{0,1\}^{n(\lambda)}}, \{\mathcal{R}(1^{n(\lambda)}, f_{\mathcal{S}}(x)), f_{\mathcal{S}}(x)\}_{x\leftarrow\{0,1\}^{n(\lambda)}}) \tag{115}$$

$$= \mathsf{SD}(\{x, f_\lambda^{\mathcal{S}}(x)\}_{x\leftarrow\{0,1\}^{n(\lambda)}}, \{\mathcal{R}(1^{n(\lambda)}, f_\lambda^{\mathcal{S}}(x)), f_\lambda^{\mathcal{S}}(x)\}_{x\leftarrow\{0,1\}^{n(\lambda)}}) > \frac{1}{p(\lambda)}. \tag{116}$$

From Assumption 1 and Equation (116), we obtain the following.

Assumption 1'. There exist a QPT algorithm $\mathcal{Q}$ and a polynomial p such that for any PPT algorithm $\mathcal{S}$, there exist a function $f_{\mathcal{S}} : \{0,1\}^* \to \{0,1\}^*$ that is computable in classical deterministic polynomial-time and an efficiently-computable polynomial n such that for any PPT algorithm $\mathcal{R}$,

$$\mathsf{SD}(\mathcal{Q}(1^\lambda), \mathcal{S}(1^\lambda)) > \frac{1}{p(\lambda)} \tag{117}$$

or

$$\mathsf{SD}(\{x, f_{\mathcal{S}}(x)\}_{x \leftarrow \{0,1\}^{n(\lambda)}}, \{\mathcal{R}(1^{n(\lambda)}, f_{\mathcal{S}}(x)), f_{\mathcal{S}}(x)\}_{x \leftarrow \{0,1\}^{n(\lambda)}}) > \frac{1}{p(\lambda)} \tag{118}$$

holds for all sufficiently large $\lambda \in \mathbb{N}$.

Assumption 1' implies the following:

Assumption 1''. There exist a QPT algorithm $\mathcal{Q}$ and a polynomial p such that for any PPT algorithm $\mathcal{S}$, there exists a function $f_{\mathcal{S}} : \{0,1\}^* \to \{0,1\}^*$ that is computable in classical deterministic polynomial-time such that the following holds: if we let

$$\Sigma_{\mathcal{S}} := \left\{ \lambda \in \mathbb{N} \;\middle|\; \mathsf{SD}(\mathcal{Q}(1^\lambda), \mathcal{S}(1^\lambda)) \leq \frac{1}{p(\lambda)} \right\}, \tag{119}$$

then $f_{\mathcal{S}}$ is a classically-secure DistOWF on $\Sigma_{\mathcal{S}}$.

By Corollary 2.11, Assumption 1'' implies the QAS/OWF condition. □

5 Equivalence of IV-PoQ and Classically-Secure OWPuzzs

Our main result, Theorem 1.1, that IV-PoQ exist if and only if classically-secure OWPuzzs exist is obtained by combining the following theorems. (For relations among these theorems, see Figure 2.)

Theorem 5.1. *If IV-PoQ exist, then Int-QASs exist.*

Theorem 5.2. *If Int-QASs exist, then the QAS/OWF condition is satisfied.*

Theorem 5.3. *If the QAS/OWF condition is satisfied, then IV-PoQ exist.*

Theorem 5.4. *If the QAS/OWF condition is satisfied, then classically-secure OWPuzzs exist.*

Theorem 5.5. *If classically-secure OWPuzzs exist, then the QAS/OWF condition is satisfied.*

Proofs of these theorems are given in the following subsections.

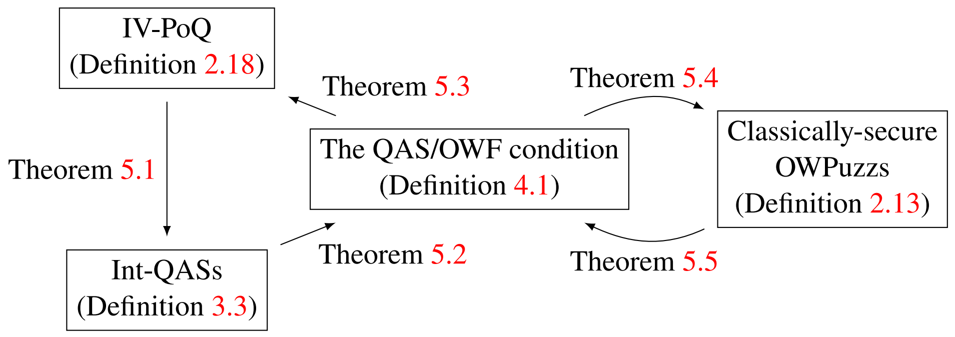

Figure 2: Relations among theorems.

5.1 Proof of Theorem 5.1

In this subsection, we show Theorem 5.1, namely, IV-PoQ $\Rightarrow$ Int-QASs.

Proof of Theorem 5.1. A proof is similar to that of Lemma 3.6. Let $(\mathcal{P},\mathcal{V}_1,\mathcal{V}_2)$ be an IV-PoQ with c-completeness and s-soundness such that $c(\lambda)-s(\lambda)\ge\frac{1}{q(\lambda)}$ for a polynomial q. Let us consider the interactive protocol $(\mathcal{P},\mathcal{V}_1)$ (i.e., the first phase of the IV-PoQ) which takes 1^λ as input and outputs a transcript τ. We claim that $(\mathcal{P},\mathcal{V}_1)$ is an Int-QAS. For the sake of contradiction, assume that it is not. Then for any polynomial p, there exists a PPT algorithm $\mathcal{P}^*_p$ such that

$$\mathsf{SD}\left(\langle\mathcal{P}^*_p,\mathcal{V}_1\rangle(1^\lambda),\langle\mathcal{P},\mathcal{V}_1\rangle(1^\lambda)\right)\le\frac{1}{p(\lambda)} \tag{120}$$

holds for infinitely many $\lambda\in\mathbb{N}$. We can prove that $\mathcal{P}^*_{2q}$ breaks the s-soundness of the IV-PoQ as follows.

$$\begin{aligned}
&\Pr[\top\leftarrow\mathcal{V}_2(1^\lambda,\tau):\tau\leftarrow\langle\mathcal{P}^*_{2q},\mathcal{V}_1\rangle(1^\lambda)] && (121)\\
&\ge\Pr[\top\leftarrow\mathcal{V}_2(1^\lambda,\tau):\tau\leftarrow\langle\mathcal{P},\mathcal{V}_1\rangle(1^\lambda)]-\mathsf{SD}\left(\langle\mathcal{P}^*_{2q},\mathcal{V}_1\rangle(1^\lambda),\langle\mathcal{P},\mathcal{V}_1\rangle(1^\lambda)\right) && (122)\\
&\ge c(\lambda)-\frac{1}{2q(\lambda)} && (123)\\
&> s(\lambda) && (124)
\end{aligned}$$

holds for infinitely many $\lambda\in\mathbb{N}$, which breaks the s-soundness of the IV-PoQ. □

5.2 Proof of Theorem 5.2

In this subsection, we show Theorem 5.2, namely, Int-QASs $\Rightarrow$ the QAS/OWF condition.

Proof of Theorem 5.2. Let $(\mathcal{A},\mathcal{C})$ be an ℓ-round Int-QAS, where ℓ is a polynomial. Without loss of generality, we can assume that in each round, $\mathcal{C}$ first sends a message to $\mathcal{A}$, and $\mathcal{A}$ returns a message to $\mathcal{C}$. This is always possible by adding some dummy messages. Let c_i be the i-th message from $\mathcal{C}$ to $\mathcal{A}$, and a_i be the i-th message from $\mathcal{A}$ to $\mathcal{C}$. We denote $(c_1,a_1,...,c_\ell,a_\ell)\leftarrow\langle\mathcal{A},\mathcal{C}\rangle(1^\lambda)$ to mean that the transcript $(c_1,a_1,...,c_\ell,a_\ell)$ is obtained by executing the interactive protocol $(\mathcal{A},\mathcal{C})$ on input 1^λ. For the notational simplicity, we denote $\tau_k:=(c_1,a_1,...,c_k,a_k)$ for $k\in[\ell]$.

For the sake of contradiction, assume that the QAS/OWF condition is not satisfied. Then, by Lemma 4.3 where we set $\mathcal{Q}=\langle\mathcal{A},\mathcal{C}\rangle$, we obtain the following statement: for any polynomial p, there exists a PPT algorithm

$\mathcal{S}_p$ such that for any efficiently-computable polynomial n and any family $\{f_\lambda : \{0,1\}^{n(\lambda)} \to \{0,1\}^*\}_{\lambda\in\mathbb{N}}$ of functions that are computable in classical deterministic polynomial-time, there exists a PPT algorithm $\mathcal{R}_p$ such that

$$\mathsf{SD}(\langle\mathcal{A},\mathcal{C}\rangle(1^\lambda), \mathcal{S}_p(1^\lambda)) \le \frac{1}{p(\lambda)} \tag{125}$$

and

$$\mathsf{SD}(\{x, f_\lambda(x)\}_{x\leftarrow\{0,1\}^{n(\lambda)}}, \{\mathcal{R}_p(1^{n(\lambda)}, f_\lambda(x)), f_\lambda(x)\}_{x\leftarrow\{0,1\}^{n(\lambda)}}) \le \frac{1}{p(\lambda)} \tag{126}$$

hold for infinitely many $\lambda \in \mathbb{N}$. Let $u(\lambda)$ be the length of the randomness used by $\mathcal{S}_p(1^\lambda)$. By using $\mathcal{S}_p$, we define a family $\{f_\lambda\}_{\lambda\in\mathbb{N}}$ of functions that are computable in classical deterministic polynomial-time as follows:

- $f_\lambda : \{0,1\}^{\lceil\log\ell(\lambda)\rceil+u(\lambda)} \to \{0,1\}^*$:
  1. Take (k,r) as input, where $k \in \{0,1\}^{\lceil\log\ell(\lambda)\rceil}$ and $r \in \{0,1\}^{u(\lambda)}$.
  2. Regard k as an encoding of an integer in $[2^{\lceil\log\ell(\lambda)\rceil}]$.[26] We use the same notation k to mean the corresponding integer. If $k \notin [\ell(\lambda)]$, output an arbitrary fixed value, say, 0.
  3. Run $\mathcal{S}_p(1^\lambda; r) = (c_1, a_1, ..., c_\ell, a_\ell)$.
  4. Output (k, τ_{k-1}, c_k).

For this specific $\{f_\lambda\}_{\lambda\in\mathbb{N}}$, there exists a PPT algorithm $\mathcal{R}_p$ such that Equations (125) and (126) hold for infinitely many $\lambda \in \mathbb{N}$. We write $\Lambda \subseteq \mathbb{N}$ to mean the set of such λ. Rewriting Equation (126), we have

$$\mathsf{SD}(\{(k,r), f_\lambda(k,r)\}_{(k,r)\leftarrow\{0,1\}^{\lceil\log\ell(\lambda)\rceil+u(\lambda)}}, \{\mathcal{R}_p(1^{n(\lambda)}, f_\lambda(k,r)), f_\lambda(k,r)\}_{(k,r)\leftarrow\{0,1\}^{\lceil\log\ell(\lambda)\rceil+u(\lambda)}}) \le \frac{1}{p(\lambda)} \tag{127}$$

for all $\lambda \in \Lambda$. For any fixed $k^* \in [\ell(\lambda)]$, we have

$$\begin{aligned}
&\mathsf{SD}(\{(k,r), f_\lambda(k,r)\}_{k,r}, \{\mathcal{R}_p(1^{n(\lambda)}, f_\lambda(k,r)), f_\lambda(k,r)\}_{k,r}) &(128)\\
&= \frac{1}{2}\sum_{k',r',\tau_{k'-1},c_{k'}} \left| \begin{array}{l} \Pr_{k,r}[((k,r), f_\lambda(k,r)) = ((k',r'),(k',\tau_{k'-1},c_{k'}))] \\ -\Pr_{k,r}[(\mathcal{R}_p(1^{n(\lambda)}, f_\lambda(k,r)), f_\lambda(k,r)) = ((k',r'),(k',\tau_{k'-1},c_{k'}))] \end{array} \right| &(129)\\
&\ge \frac{1}{2}\sum_{r',\tau_{k^*-1},c_{k^*}} \left| \begin{array}{l} \Pr_{k,r}[((k,r), f_\lambda(k,r)) = ((k^*,r'),(k^*,\tau_{k^*-1},c_{k^*}))] \\ -\Pr_{k,r}[(\mathcal{R}_p(1^{n(\lambda)}, f_\lambda(k,r)), f_\lambda(k,r)) = ((k^*,r'),(k^*,\tau_{k^*-1},c_{k^*}))] \end{array} \right| &(130)\\
&= \frac{1}{2}\Pr_k[k=k^*]\sum_{r',\tau_{k^*-1},c_{k^*}} \left| \begin{array}{l} \Pr_r[((k^*,r), f_\lambda(k^*,r)) = ((k^*,r'),(k^*,\tau_{k^*-1},c_{k^*}))] \\ -\Pr_r[(\mathcal{R}_p(1^{n(\lambda)}, f_\lambda(k^*,r)), f_\lambda(k^*,r)) = ((k^*,r'),(k^*,\tau_{k^*-1},c_{k^*}))] \end{array} \right| &(131)\\
&= \frac{1}{2^{\lceil\log\ell(\lambda)\rceil}}\mathsf{SD}(\{(k^*,r), f_\lambda(k^*,r)\}_{r\leftarrow\{0,1\}^{u(\lambda)}}, \{\mathcal{R}_p(1^{n(\lambda)}, f_\lambda(k^*,r)), f_\lambda(k^*,r)\}_{r\leftarrow\{0,1\}^{u(\lambda)}}) &(132)\\
&\ge \frac{1}{2\ell(\lambda)}\mathsf{SD}(\{(k^*,r), f_\lambda(k^*,r)\}_{r\leftarrow\{0,1\}^{u(\lambda)}}, \{\mathcal{R}_p(1^{n(\lambda)}, f_\lambda(k^*,r)), f_\lambda(k^*,r)\}_{r\leftarrow\{0,1\}^{u(\lambda)}}), &(133)
\end{aligned}$$

[26]For example, regard k as a binary encoding of an integer and then add 1.

where $(k,r) \leftarrow \{0,1\}^{\lceil \log \ell(\lambda) \rceil + u(\lambda)}$. Combining the above and Equation (127), we have

$$\mathsf{SD}(\{(k,r), f_\lambda(k,r)\}_{r \leftarrow \{0,1\}^{u(\lambda)}}, \{\mathcal{R}_p(1^{n(\lambda)}, f_\lambda(k,r)), f_\lambda(k,r)\}_{r \leftarrow \{0,1\}^{u(\lambda)}}) \leq \frac{2\ell(\lambda)}{p(\lambda)} \tag{134}$$

for all $\lambda \in \Lambda$ and all $k \in [\ell(\lambda)]$. Equivalently, we have

$$\mathsf{SD}(\{(k,r),(k,\tau_{k-1},c_k)\},\{(k',r'),(k,\tau_{k-1},c_k)\}) \leq \frac{2\ell(\lambda)}{p(\lambda)} \tag{135}$$

for all $\lambda \in \Lambda$ where $r \leftarrow \{0,1\}^{u(\lambda)}$, $\mathcal{S}_p(1^\lambda; r) = (c_1, a_1, ..., c_\ell, a_\ell)$, and $(k',r') \leftarrow \mathcal{R}_p(1^{n(\lambda)}, (k,\tau_{k-1},c_k))$. By the monotonicity of the statistical distance,[27] the above implies

$$\mathsf{SD}(\{a_k,(k,\tau_{k-1},c_k)\},\{a'_{k'},(k,\tau_{k-1},c_k)\}) \leq \frac{2\ell(\lambda)}{p(\lambda)} \tag{136}$$

for all $\lambda \in \Lambda$ where $r \leftarrow \{0,1\}^{u(\lambda)}$, $\mathcal{S}_p(1^\lambda; r) = (c_1, a_1, ..., c_\ell, a_\ell)$, $(k',r') \leftarrow \mathcal{R}_p(1^{n(\lambda)}, (k,\tau_{k-1},c_k))$, and $\mathcal{S}_p(1^\lambda; r') = (c'_1, a'_1, ..., c'_\ell, a'_\ell)$. By Equation (125), the distributions of $\{a_k,(k,\tau_{k-1},c_k)\}$ and $\{a'_{k'},(k,\tau_{k-1},c_k)\}$ in Equation (136) change only by at most $\frac{1}{p(\lambda)}$ in terms of statistical distance if we generate $(c_1, a_1, ..., c_\ell, a_\ell) \leftarrow \langle \mathcal{A}, \mathcal{C} \rangle(1^\lambda)$ instead of $r \leftarrow \{0,1\}^{u(\lambda)}$ and $\mathcal{S}_p(1^\lambda; r) = (c_1, a_1, ..., c_\ell, a_\ell)$. Thus, we have

$$\mathsf{SD}(\{a_k,(k,\tau_{k-1},c_k)\},\{a'_{k'},(k,\tau_{k-1},c_k)\}) \leq \frac{2\ell(\lambda)+2}{p(\lambda)} \tag{137}$$

for all $\lambda \in \Lambda$ where $(c_1, a_1, ..., c_\ell, a_\ell) \leftarrow \langle \mathcal{A}, \mathcal{C} \rangle(1^\lambda)$, $(k',r') \leftarrow \mathcal{R}_p(1^{n(\lambda)}, (k,\tau_{k-1},c_k))$, and $\mathcal{S}_p(1^\lambda; r') = (c'_1, a'_1, ..., c'_\ell, a'_\ell)$.

For any polynomial q, we construct a PPT algorithm $\mathcal{B}_q$ that satisfies

$$\mathsf{SD}(\langle \mathcal{A}, \mathcal{C} \rangle(1^\lambda), \langle \mathcal{B}_q, \mathcal{C} \rangle(1^\lambda)) \leq \frac{1}{q(\lambda)} \tag{138}$$

for all $\lambda \in \Lambda$. Since Λ is an infinite set, this means that $\mathcal{B}_q$ breaks the Int-QAS, whih constradicts the assumption. Thus, it suffices to prove that Equation (138) holds for all $\lambda \in \Lambda$.

Below, we give the construction of $\mathcal{B}_q$. Let $p(\lambda) := (2\ell(\lambda)+2)\ell(\lambda)q(\lambda)$. In the k-th round $\mathcal{B}_q$ interacts with $\mathcal{C}$ as follows:

1. Receive c_k from $\mathcal{C}$.
2. Run $(k',r') \leftarrow \mathcal{R}_p(1^{n(\lambda)}, (k,\tau_{k-1},c_k))$.
3. Compute $\mathcal{S}_p(1^\lambda; r') = (c'_1, a'_1, ..., c'_\ell, a'_\ell)$.
4. Return $a_k := a'_{k'}$ to $\mathcal{C}$.

For $k \in [\ell(\lambda)+1]$, let D_k be the distribution sampled by the following procedure:

1. Run $\langle \mathcal{A}, \mathcal{C} \rangle(1^\lambda)$ until the $(k-1)$-th round where we write $\tau_{k-1} = (c_1, a_1, \ldots, c_{k-1}, a_{k-1})$ to mean the partial transcript until the $(k-1)$-th round.

[27] For any random variables X and Y and any algorithm $\mathcal{A}$, $\mathsf{SD}(\mathcal{A}(X), \mathcal{A}(Y)) \leq \mathsf{SD}(X,Y)$.

2. Use $\mathcal{B}_q$ instead of $\mathcal{A}$ to complete the protocol. That is, for $i = k, k+1, \ldots, \ell$, do the following:
   (a) Receive c_i from $\mathcal{C}$.
   (b) Run $(i', r') \leftarrow \mathcal{R}_p(1^{n(\lambda)}, (i, \tau_{i-1}, c_i))$.
   (c) Compute $\mathcal{S}_p(1^\lambda; r') = (c'_1, a'_1, ..., c'_\ell, a'_\ell)$.
   (d) Return $a_i := a'_{i'}$ to $\mathcal{C}$.
3. Output the full transcript $\tau_\ell = (c_1, a_1, \ldots, c_\ell, a_\ell)$.

Clearly, we have $D_1 = \langle \mathcal{B}_q, \mathcal{C}\rangle(1^\lambda)$ and $D_{\ell+1} = \langle \mathcal{A}, \mathcal{C}\rangle(1^\lambda)$ where the equality means equivalence as distributions. Thus, it suffices to prove

$$\mathsf{SD}(D_1, D_{\ell+1}) \le \frac{1}{q(\lambda)} \tag{139}$$

for all $\lambda \in \Lambda$. By the triangle inequality, it suffices to prove

$$\mathsf{SD}(D_k, D_{k+1}) \le \frac{1}{\ell(\lambda)q(\lambda)} \tag{140}$$

for all $\lambda \in \Lambda$ and $k \in [\ell(\lambda)]$. Note that the only difference between D_k and D_{k+1} is how a_k is generated. In D_k, a_k is generated by the interaction between $\mathcal{A}$ and $\mathcal{C}$. On the other hand, in D_{k+1}, a_k is generated by running $(k', r') \leftarrow \mathcal{R}_p(k, \tau_{k-1}, c_k)$ and $\mathcal{S}_p(1^\lambda; r') = (c'_1, a'_1, ..., c'_\ell, a'_\ell)$ and then setting $a_k := a'_{k'}$ where (τ_{k-1}, c_k) is the partial transcript generated by the interaction between $\mathcal{A}$ and $\mathcal{C}$. Thus, by a straightforward reduction to Equation (137), the distributions of τ_k in D_k and D_{k+1} differ by at most $\frac{2\ell(\lambda)+2}{p(\lambda)} = \frac{1}{\ell(\lambda)q(\lambda)}$ in terms of statistical distance. Moreover, both in D_k and D_{k+1}, the partial transcript τ_k is extended to the full transcript τ_ℓ in the same manner using the interaction between $\mathcal{B}_q$ and $\mathcal{C}$. Thus, by the monotonicity of the statistical distance, Equation (140) holds for all $\lambda \in \Lambda$. This completes the proof of Theorem 5.2. □

5.3 Proof of Theorem 5.3

In this subsection, we show Theorem 5.3, namely, the QAS/OWF condition $\Rightarrow$ IV-PoQ.

Proof of Theorem 5.3. Assume that the QAS/OWF condition is satisfied. Then, from the definition of the QAS/OWF condition, there exist a polynomial p, a QPT algorithm $\mathcal{Q}$ that takes 1^λ as input and outputs a classical string, a function $f : \{0,1\}^* \to \{0,1\}^*$ that is computable in classical deterministic polynomial-time such that for any PPT algorithm $\mathcal{S}$, the following holds: Let

$$\Sigma_{\mathcal{S}} := \left\{ \lambda \in \mathbb{N} : \mathsf{SD}(\mathcal{Q}(1^\lambda), \mathcal{S}(1^\lambda)) \le \frac{1}{p(\lambda)} \right\}, \tag{141}$$

then f is a classically-secure OWF on $\Sigma_{\mathcal{S}}$.

By using such p and $\mathcal{Q}$, from Lemma 3.8, we obtain a non-interactive protocol $(\mathcal{P}, \mathcal{V})$ that satisfies the properties of Lemma 3.8. Moreover by using f, from Lemma 2.21, we obtain an IV-PoQ $(\mathcal{P}', \mathcal{V}'_1, \mathcal{V}'_2)$ on $\Sigma_{\mathcal{S}}$ with $(1 - \mathsf{negl}(\lambda))$-completeness and $\mathsf{negl}(\lambda)$-soundness. By combining $(\mathcal{P}, \mathcal{V})$ and $(\mathcal{P}', \mathcal{V}'_1, \mathcal{V}'_2)$, we construct an IV-PoQ $(\mathcal{P}'', \mathcal{V}''_1, \mathcal{V}''_2)$ with $(1 - \mathsf{negl}(\lambda))$-completeness and $(1 - \frac{1}{\lambda^{\log \lambda}} - \frac{1}{p(\lambda)^2})$-soundness as follows:

- 1st phase. $(\mathcal{P}'', \mathcal{V}''_1)(1^\lambda) \to \tau''$:

1. Take 1^λ as input.
2. Run $\tau \leftarrow \mathcal{P}(1^\lambda)$. τ is sent to $\mathcal{V}_1''$.
3. Run $\tau' \leftarrow \langle \mathcal{P}', \mathcal{V}_1' \rangle(1^\lambda)$.
4. Output $\tau'' := (\tau, \tau')$.

- 2nd phase. $\mathcal{V}_2''(1^\lambda, \tau'') \rightarrow \top/\bot$:

1. Take 1^λ and $\tau'' = (\tau, \tau')$ as input.
2. Run $\mathcal{V}(1^\lambda, \tau)$ and $\mathcal{V}_2'(1^\lambda, \tau')$.
3. If both $\mathcal{V}$ and $\mathcal{V}_2'$ output $\top$, then output $\top$. Otherwise, output $\bot$.

First, we prove $(1-\mathsf{negl})$-completeness of $(\mathcal{P}'', \mathcal{V}_1'', \mathcal{V}_2'')$. We have

$$\Pr[\top \leftarrow \mathcal{V}_2''(1^\lambda, \tau'') : \tau'' \leftarrow \langle \mathcal{P}'', \mathcal{V}_1'' \rangle(1^\lambda)] = \Pr\left[\top \leftarrow \mathcal{V}(1^\lambda, \tau) \wedge \top \leftarrow \mathcal{V}_2'(1^\lambda, \tau') : \begin{array}{r} \tau \leftarrow \mathcal{P}(1^\lambda) \\ \tau' \leftarrow \langle \mathcal{P}', \mathcal{V}_1' \rangle(1^\lambda) \end{array}\right] \tag{142}$$

$$= \Pr[\top \leftarrow \mathcal{V}(1^\lambda, \tau) : \tau \leftarrow \mathcal{P}(1^\lambda)] \tag{143}$$

$$\times \Pr[\top \leftarrow \mathcal{V}_2'(1^\lambda, \tau') : \tau' \leftarrow \langle \mathcal{P}', \mathcal{V}_1' \rangle(1^\lambda)] \tag{144}$$

$$\geq 1 - 2\mathsf{negl}(\lambda) \tag{145}$$

for all sufficiently large $\lambda \in \mathbb{N}$. Here, we have used the fact that $(\mathcal{P}', \mathcal{V}_1', \mathcal{V}_2')$ is $(1-\mathsf{negl}(\lambda))$-complete, and the second item of Lemma 3.8.

Next, we prove the soundness. For the sake of contradiction, we assume that $(\mathcal{P}'', \mathcal{V}_1'', \mathcal{V}_2'')$ does not satisfy $(1 - \frac{1}{\lambda^{\log\lambda}} - \frac{1}{p(\lambda)^2})$-soundness on $\mathbb{N}$. Then, there exist a PPT algorithm $\mathcal{P}^* = (\mathcal{P}_1^*, \mathcal{P}_2^*)$ and an infinite subset $\Lambda \subseteq \mathbb{N}$ such that

$$1 - \frac{1}{\lambda^{\log\lambda}} - \frac{1}{p(\lambda)^2} < \Pr[\top \leftarrow \mathcal{V}_2''(1^\lambda, \tau'') : \tau'' \leftarrow \langle \mathcal{P}^*, \mathcal{V}_1'' \rangle(1^\lambda)] \tag{146}$$

$$= \Pr\left[\top \leftarrow \mathcal{V}(1^\lambda, \tau) \wedge \top \leftarrow \mathcal{V}_2'(1^\lambda, \tau') : \begin{array}{r} (\tau, \mathsf{st}) \leftarrow \mathcal{P}_1^*(1^\lambda) \\ \tau' \leftarrow \langle \mathcal{P}_2^*(\mathsf{st}), \mathcal{V}_1'(1^\lambda) \rangle \end{array}\right] \tag{147}$$

holds for all $\lambda \in \Lambda$. Define a PPT adversary $\mathcal{P}_3^*$ against $(\mathcal{P}', \mathcal{V}_1', \mathcal{V}_2')$ as follows.

1. Before interacting with $\mathcal{V}_1'$, run $(\tau, \mathsf{st}) \leftarrow \mathcal{P}_1^*(1^\lambda)$.

2. Run $\tau' \leftarrow \langle \mathcal{P}_2^*(\mathsf{st}), \mathcal{V}_1'(1^\lambda) \rangle$.

Then, we have that both of

$$\Pr[\top \leftarrow \mathcal{V}(1^\lambda, \tau) : \tau \leftarrow \mathcal{P}_1^*(1^\lambda)] = \Pr\left[\top \leftarrow \mathcal{V}(1^\lambda, \tau) : \begin{array}{r} (\tau, \mathsf{st}) \leftarrow \mathcal{P}_1^*(1^\lambda) \\ \tau' \leftarrow \langle \mathcal{P}_2^*(\mathsf{st}), \mathcal{V}_1'(1^\lambda) \rangle \end{array}\right] \tag{148}$$

$$\geq \Pr\left[\top \leftarrow \mathcal{V}(1^\lambda, \tau) \wedge \top \leftarrow \mathcal{V}_2'(1^\lambda, \tau') : \begin{array}{r} (\tau, \mathsf{st}) \leftarrow \mathcal{P}_1^*(1^\lambda) \\ \tau' \leftarrow \langle \mathcal{P}_2^*(\mathsf{st}), \mathcal{V}_1'(1^\lambda) \rangle \end{array}\right] \tag{149}$$

$$> 1 - \frac{1}{\lambda^{\log\lambda}} - \frac{1}{p(\lambda)^2} \tag{150}$$

and

$$\Pr[\top \leftarrow \mathcal{V}_2'(1^\lambda,\tau') : \tau' \leftarrow \langle \mathcal{P}_3^*, \mathcal{V}_1' \rangle(1^\lambda)] = \Pr\left[\top \leftarrow \mathcal{V}_2'(1^\lambda,\tau') : \begin{array}{c}(\tau,\mathsf{st}) \leftarrow \mathcal{P}_1^*(1^\lambda) \\ \tau' \leftarrow \langle \mathcal{P}_2^*(\mathsf{st}), \mathcal{V}_1'(1^\lambda)\rangle\end{array}\right] \tag{151}$$

$$\geq \Pr\left[\top \leftarrow \mathcal{V}(1^\lambda,\tau) \wedge \top \leftarrow \mathcal{V}_2'(1^\lambda,\tau') : \begin{array}{c}(\tau,\mathsf{st}) \leftarrow \mathcal{P}_1^*(1^\lambda) \\ \tau' \leftarrow \langle \mathcal{P}_2^*(\mathsf{st}), \mathcal{V}_1'(1^\lambda)\rangle\end{array}\right] \tag{152}$$

$$> 1 - \frac{1}{\lambda^{\log\lambda}} - \frac{1}{p(\lambda)^2} \tag{153}$$

hold for all $\lambda \in \Lambda$.

By Lemma 3.8 and Equation (150), there exists a PPT algorithm $\mathcal{S}^*$ such that

$$\mathsf{SD}(\mathcal{Q}(1^\lambda), \mathcal{S}^*(1^\lambda)) \leq \frac{1}{p(\lambda)} \tag{154}$$

holds for all sufficiently large $\lambda \in \Lambda$. Let $\Theta_{\mathcal{S}^*}$ be the set of such λ. Define

$$\Sigma_{\mathcal{S}^*} := \left\{\lambda \in \mathbb{N} : \mathsf{SD}(\mathcal{Q}(1^\lambda), \mathcal{S}^*(1^\lambda)) \leq \frac{1}{p(\lambda)}\right\}. \tag{155}$$

By the QAS/OWF condition, f is a classically-secure OWF on $\Sigma_{\mathcal{S}^*}$. Because $\Theta_{\mathcal{S}^*} \subseteq \Sigma_{\mathcal{S}^*}$, this means that f is a classically-secure OWF on $\Theta_{\mathcal{S}^*}$. Then f is a classically-secure OWF on Λ. From Lemma 2.21, $(\mathcal{P}', \mathcal{V}_1', \mathcal{V}_2')$ is an IV-PoQ with $(1-\mathsf{negl}(\lambda))$-completeness and $(1 - \frac{1}{\lambda^{\log\lambda}} - \frac{1}{p(\lambda)^2})$-soundness on Λ, but it contradict Equation (153). □

5.4 Proof of Theorem 5.4

In this subsection, we show Theorem 5.4, namely, the QAS/OWF condition ⇒ classically-secure OWPuzzs.

Proof of Theorem 5.4. Assume that the QAS/OWF condition is satisfied. Then, from the definition of the QAS/OWF condition, there exist a polynomial p, a QPT algorithm $\mathcal{Q}$ that takes 1^λ as input and outputs a classical string, a function $f : \{0,1\}^* \to \{0,1\}^*$ that is computable in classical deterministic polynomial-time such that for any PPT algorithm $\mathcal{S}$, the following holds: Let

$$\Sigma_{\mathcal{S}} := \left\{\lambda \in \mathbb{N} : \mathsf{SD}(\mathcal{Q}(1^\lambda), \mathcal{S}(1^\lambda)) \leq \frac{1}{p(\lambda)}\right\}, \tag{156}$$

then f is a classically-secure OWF on $\Sigma_{\mathcal{S}}$.

By using such p and $\mathcal{Q}$, we obtain a non-interactive protocol $(\mathcal{P}, \mathcal{V})$ from Lemma 3.8. By using $(\mathcal{P}, \mathcal{V})$, we construct a pair $(\mathsf{Samp}, \mathsf{Ver})$ of algorithms as follows:

- $\mathsf{Samp}(1^\lambda) \to (\mathsf{ans}, \mathsf{puzz})$:
  1. Run $\tau \leftarrow \mathcal{P}(1^\lambda)$.
  2. Output $\mathsf{puzz} := 1^\lambda$ and $\mathsf{ans} := \tau$.
- $\mathsf{Ver}(\mathsf{ans}^*, \mathsf{puzz}) \to \top/\bot$:

1. Parse $\mathsf{puzz} = 1^\lambda$ and $\mathsf{ans}^* = \tau^*$.
2. Run $\mathcal{V}(1^\lambda, \tau^*)$.
3. Output $\top$ if $\mathcal{V}$ outputs $\top$. Output $\perp$ otherwise.

Moreover by using the classically-secure OWF f on $\Sigma_{\mathcal{S}}$, from Lemma 2.17, we obtain a classically-secure OWPuzz $(\mathsf{Samp}', \mathsf{Ver}')$ on $\Sigma_{\mathcal{S}}$ with 1-correctness and $\mathsf{negl}(\lambda)$-security. By combining $(\mathsf{Samp}, \mathsf{Ver})$ and $(\mathsf{Samp}', \mathsf{Ver}')$, we construct a classically-secure OWPuzz $(\mathsf{Samp}'', \mathsf{Ver}'')$ on $\mathbb{N}$ with $(1 - \frac{1}{\lambda^{\log\lambda}})$-correctness and $(1 - \frac{1}{\lambda^{\log\lambda}} - \frac{1}{p(\lambda)^2})$-security as follows:

- $\mathsf{Samp}''(1^\lambda) \to (\mathsf{ans}'', \mathsf{puzz}'')$:
  1. Take 1^λ as input.
  2. Run $(\mathsf{ans}, \mathsf{puzz}) \leftarrow \mathsf{Samp}(1^\lambda)$ and $(\mathsf{ans}', \mathsf{puzz}') \leftarrow \mathsf{Samp}'(1^\lambda)$.
  3. Set $\mathsf{ans}'' := (\mathsf{ans}, \mathsf{ans}')$ and $\mathsf{puzz}'' := (\mathsf{puzz}, \mathsf{puzz}')$.
  4. Output $(\mathsf{ans}'', \mathsf{puzz}'')$.
- $\mathsf{Ver}''(\mathsf{ans}''^*, \mathsf{puzz}'') \to \top/\perp$:
  1. Take $\mathsf{ans}''^* = (\mathsf{ans}^*, \mathsf{ans}'^*)$ and $\mathsf{puzz}'' = (\mathsf{puzz}, \mathsf{puzz}')$ as input.
  2. Run $\mathsf{Ver}(\mathsf{ans}^*, \mathsf{puzz})$ and $\mathsf{Ver}'(\mathsf{ans}'^*, \mathsf{puzz}')$.
  3. If both Ver and Ver' output $\top$, then output $\top$. Otherwise, output $\perp$.

First, we prove $(1 - \frac{1}{\lambda^{\log\lambda}})$-correctness of $(\mathsf{Samp}'', \mathsf{Ver}'')$. By the second item of Lemma 3.8,

$$\Pr[\top \leftarrow \mathsf{Ver}(\mathsf{ans}, \mathsf{puzz}) : (\mathsf{ans}, \mathsf{puzz}) \leftarrow \mathsf{Samp}(1^\lambda)] = \Pr[\top \leftarrow \mathcal{V}(1^\lambda, \tau) : \tau \leftarrow \mathcal{P}(1^\lambda)] \tag{157}$$
$$\geq 1 - \frac{1}{\lambda^{\log\lambda}} \tag{158}$$

holds for all sufficiently large $\lambda \in \mathbb{N}$. Thus, by using the fact that $(\mathsf{Samp}', \mathsf{Ver}')$ is 1-correct, we obtain

$$\Pr[\top \leftarrow \mathsf{Ver}''(\mathsf{ans}'', \mathsf{puzz}'') : (\mathsf{ans}'', \mathsf{puzz}'') \leftarrow \mathsf{Samp}''(1^\lambda)] \tag{159}$$
$$= \Pr\left[\top \leftarrow \mathsf{Ver}(\mathsf{ans}, \mathsf{puzz}) \wedge \top \leftarrow \mathsf{Ver}'(\mathsf{ans}', \mathsf{puzz}') : \begin{array}{c}(\mathsf{ans}, \mathsf{puzz}) \leftarrow \mathsf{Samp}(1^\lambda) \\ (\mathsf{ans}', \mathsf{puzz}') \leftarrow \mathsf{Samp}'(1^\lambda)\end{array}\right] \tag{160}$$
$$= \Pr[\top \leftarrow \mathsf{Ver}(\mathsf{ans}, \mathsf{puzz}) : (\mathsf{ans}, \mathsf{puzz}) \leftarrow \mathsf{Samp}(1^\lambda)] \tag{161}$$
$$\times \Pr[\top \leftarrow \mathsf{Ver}'(\mathsf{ans}', \mathsf{puzz}') : (\mathsf{ans}', \mathsf{puzz}') \leftarrow \mathsf{Samp}'(1^\lambda)] \tag{162}$$
$$\geq 1 - \frac{1}{\lambda^{\log\lambda}} \tag{163}$$

for all sufficiently large $\lambda \in \mathbb{N}$.

Next, we prove the security. For the sake of contradiction, we assume that $(\mathsf{Samp}'', \mathsf{Ver}'')$ does not satisfy $(1 - \frac{1}{\lambda^{\log\lambda}} - \frac{1}{p(\lambda)^2})$-security on $\mathbb{N}$. Then, there exist a PPT adversary $\mathcal{A}$ and an infinite subset $\Lambda \subseteq \mathbb{N}$ such that

$$1 - \frac{1}{\lambda^{\log\lambda}} - \frac{1}{p(\lambda)^2} < \Pr[\top \leftarrow \mathsf{Ver}''(\mathcal{A}(1^\lambda, \mathsf{puzz}''), \mathsf{puzz}'') : (\mathsf{ans}'', \mathsf{puzz}'') \leftarrow \mathsf{Samp}''(1^\lambda)] \tag{164}$$
$$= \Pr\left[\begin{array}{c}\top \leftarrow \mathsf{Ver}(\mathsf{ans}^*, \mathsf{puzz}) \\ \wedge \\ \top \leftarrow \mathsf{Ver}'(\mathsf{ans}'^*, \mathsf{puzz}')\end{array} : \begin{array}{l}(\mathsf{ans}, \mathsf{puzz}) \leftarrow \mathsf{Samp}(1^\lambda) \\ (\mathsf{ans}', \mathsf{puzz}') \leftarrow \mathsf{Samp}'(1^\lambda) \\ (\mathsf{ans}^*, \mathsf{ans}'^*) \leftarrow \mathcal{A}(1^\lambda, \mathsf{puzz}, \mathsf{puzz}')\end{array}\right] \tag{165}$$

holds for all $\lambda \in \Lambda$. We define the following two PPT adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ against $(\mathsf{Samp}, \mathsf{Ver})$ and $(\mathsf{Samp}', \mathsf{Ver}')$, respectively:

- $\mathcal{A}_1(1^\lambda, \mathsf{puzz}) \rightarrow \mathsf{ans}^*$:
  1. Take 1^λ and puzz as input.
  2. Run $(\mathsf{ans}', \mathsf{puzz}') \leftarrow \mathsf{Samp}'(1^\lambda)$.
  3. Run $(\mathsf{ans}^*, \mathsf{ans}'^*) \leftarrow \mathcal{A}(1^\lambda, \mathsf{puzz}, \mathsf{puzz}')$.
  4. Output ans^*.
- $\mathcal{A}_2(1^\lambda, \mathsf{puzz}') \rightarrow \mathsf{ans}'^*$:
  1. Take 1^λ and puzz′ as input.
  2. Run $(\mathsf{ans}, \mathsf{puzz}) \leftarrow \mathsf{Samp}(1^\lambda)$.
  3. Run $(\mathsf{ans}^*, \mathsf{ans}'^*) \leftarrow \mathcal{A}(1^\lambda, \mathsf{puzz}, \mathsf{puzz}')$.
  4. Output ans'^*.

Then, both of

$$\Pr[\top \leftarrow \mathsf{Ver}(\mathcal{A}_1(1^\lambda, \mathsf{puzz}), \mathsf{puzz}) : (\mathsf{ans}, \mathsf{puzz}) \leftarrow \mathsf{Samp}(1^\lambda)] \tag{166}$$

$$= \Pr\left[\top \leftarrow \mathsf{Ver}(\mathsf{ans}^*, \mathsf{puzz}) : \begin{array}{l} (\mathsf{ans}, \mathsf{puzz}) \leftarrow \mathsf{Samp}(1^\lambda) \\ (\mathsf{ans}', \mathsf{puzz}') \leftarrow \mathsf{Samp}'(1^\lambda) \\ (\mathsf{ans}^*, \mathsf{ans}'^*) \leftarrow \mathcal{A}(1^\lambda, \mathsf{puzz}, \mathsf{puzz}') \end{array}\right] \tag{167}$$

$$\geq \Pr\left[\begin{array}{c} \top \leftarrow \mathsf{Ver}(\mathsf{ans}^*, \mathsf{puzz}) \\ \wedge \\ \top \leftarrow \mathsf{Ver}'(\mathsf{ans}'^*, \mathsf{puzz}') \end{array} : \begin{array}{l} (\mathsf{ans}, \mathsf{puzz}) \leftarrow \mathsf{Samp}(1^\lambda) \\ (\mathsf{ans}', \mathsf{puzz}') \leftarrow \mathsf{Samp}'(1^\lambda) \\ (\mathsf{ans}^*, \mathsf{ans}'^*) \leftarrow \mathcal{A}(1^\lambda, \mathsf{puzz}, \mathsf{puzz}') \end{array}\right] > 1 - \frac{1}{\lambda^{\log \lambda}} - \frac{1}{p(\lambda)^2} \tag{168}$$

and

$$\Pr[\top \leftarrow \mathsf{Ver}'(\mathcal{A}_2(1^\lambda, \mathsf{puzz}'), \mathsf{puzz}') : (\mathsf{ans}', \mathsf{puzz}') \leftarrow \mathsf{Samp}'(1^\lambda)] \tag{169}$$

$$= \Pr\left[\top \leftarrow \mathsf{Ver}'(\mathsf{ans}'^*, \mathsf{puzz}') : \begin{array}{l} (\mathsf{ans}', \mathsf{puzz}') \leftarrow \mathsf{Samp}'(1^\lambda) \\ (\mathsf{ans}, \mathsf{puzz}) \leftarrow \mathsf{Samp}(1^\lambda) \\ (\mathsf{ans}^*, \mathsf{ans}'^*) \leftarrow \mathcal{A}(1^\lambda, \mathsf{puzz}, \mathsf{puzz}') \end{array}\right] \tag{170}$$

$$\geq \Pr\left[\begin{array}{c} \top \leftarrow \mathsf{Ver}(\mathsf{ans}^*, \mathsf{puzz}) \\ \wedge \\ \top \leftarrow \mathsf{Ver}'(\mathsf{ans}'^*, \mathsf{puzz}') \end{array} : \begin{array}{l} (\mathsf{ans}, \mathsf{puzz}) \leftarrow \mathsf{Samp}(1^\lambda) \\ (\mathsf{ans}', \mathsf{puzz}') \leftarrow \mathsf{Samp}'(1^\lambda) \\ (\mathsf{ans}^*, \mathsf{ans}'^*) \leftarrow \mathcal{A}(\mathsf{puzz}, \mathsf{puzz}') \end{array}\right] > 1 - \frac{1}{\lambda^{\log \lambda}} - \frac{1}{p(\lambda)^2} \tag{171}$$

hold for all $\lambda \in \Lambda$. From Equation (168), we obtain

$$1 - \frac{1}{\lambda^{\log \lambda}} - \frac{1}{p(\lambda)^2} < \Pr[\top \leftarrow \mathsf{Ver}(\mathcal{A}_1(1^\lambda, \mathsf{puzz}), \mathsf{puzz}) : (\mathsf{ans}, \mathsf{puzz}) \leftarrow \mathsf{Samp}(1^\lambda)] \tag{172}$$

$$= \Pr[\top \leftarrow \mathsf{Ver}(\mathcal{A}_1(1^\lambda, 1^\lambda), 1^\lambda) : (\mathsf{ans}, 1^\lambda) \leftarrow \mathsf{Samp}(1^\lambda)] \tag{173}$$

$$= \Pr[\top \leftarrow \mathsf{Ver}(\mathsf{ans}^*, 1^\lambda) : \mathsf{ans}^* \leftarrow \mathcal{A}_1(1^\lambda)] \tag{174}$$

$$= \Pr[\top \leftarrow \mathcal{V}(1^\lambda, \mathsf{ans}^*) : \mathsf{ans}^* \leftarrow \mathcal{A}_1(1^\lambda)] \tag{175}$$

$$= \Pr[\top \leftarrow \mathcal{V}(1^\lambda, \tau) : \tau \leftarrow \mathcal{A}_1(1^\lambda)] \tag{176}$$

for all $\lambda \in \Lambda$. Thus, by the third item of Lemma 3.8, there exists a PPT algorithm $\mathcal{S}^*$ such that

$$\mathsf{SD}(\mathcal{Q}(1^\lambda), \mathcal{S}^*(1^\lambda)) \leq \frac{1}{p(\lambda)} \tag{177}$$

holds for all sufficiently large $\lambda \in \Lambda$. Let $\Theta_{\mathcal{S}^*}$ be the set of such λ. Define

$$\Sigma_{\mathcal{S}^*} := \left\{ \lambda \in \mathbb{N} : \mathsf{SD}(\mathcal{Q}(1^\lambda), \mathcal{S}^*(1^\lambda)) \leq \frac{1}{p(\lambda)} \right\}. \tag{178}$$

By the QAS/OWF condition, f is a classically-secure OWF on $\Sigma_{\mathcal{S}^*}$. Because $\Theta_{\mathcal{S}^*} \subseteq \Sigma_{\mathcal{S}^*}$, this means that f is a classically-secure OWF on $\Theta_{\mathcal{S}^*}$. Then f is a classically-secure OWF on Λ. From Lemma 2.17, $(\mathsf{Samp}', \mathsf{Ver}')$ is a classically-secure OWPuzz with 1-correctness and $(1 - \frac{1}{\lambda^{\log \lambda}} - \frac{1}{p(\lambda)^2})$-security on Λ, but it contradict Equation (171).

$\square$

5.5 Proof of Theorem 5.5

In this subsection, we show Theorem 5.5, namely, classically-secure OWPuzzs $\Rightarrow$ the QAS/OWF condition.

Proof of Theorem 5.5. Let $(\mathsf{Samp}, \mathsf{Ver})$ be a classically-secure OWPuzz with c-correctness and s-security such that $c(\lambda) - s(\lambda) \geq 1/q(\lambda)$ for a polynomial q. For the sake of contradiction, assume that the QAS/OWF condition is not satisfied. Then, from Lemma 4.3 by taking $\mathcal{Q}$ of the lemma as Samp, we obtain the following statement: For any polynomial p, there exists a PPT algorithm $\mathcal{S}_p$ such that for any efficiently-computable polynomial n and any family $\{f_\lambda : \{0,1\}^{n(\lambda)} \to \{0,1\}^*\}_{\lambda \in \mathbb{N}}$ of functions that are computable in classical deterministic polynomial-time, there exists a PPT algorithm $\mathcal{R}_p$ such that

$$\mathsf{SD}(\mathsf{Samp}(1^\lambda), \mathcal{S}_p(1^\lambda)) \leq \frac{1}{p(\lambda)} \tag{179}$$

and

$$\mathsf{SD}(\{x, f_\lambda(x)\}_{x \leftarrow \{0,1\}^{n(\lambda)}}, \{\mathcal{R}_p(1^{n(\lambda)}, f_\lambda(x)), f_\lambda(x)\}_{x \leftarrow \{0,1\}^{n(\lambda)}}) \leq \frac{1}{p(\lambda)} \tag{180}$$

hold for infinitely many $\lambda \in \mathbb{N}$.[28]

Let $n(\lambda)$ be the length of the random seed for $\mathcal{S}_p(1^\lambda)$. By using $\mathcal{S}_p$, we define a function family $\{f_{p,\lambda}\}_{\lambda \in \mathbb{N}}$ as follows:

- $f_{p,\lambda} : \{0,1\}^{n(\lambda)} \to \{0,1\}^*$:
  1. Take $r \in \{0,1\}^{n(\lambda)}$ as input.
  2. Run $\mathcal{S}_p(1^\lambda; r) = (\mathsf{puzz}, \mathsf{ans})$.
  3. Output puzz.

[28]We mean that both Equation (179) and Equation (180) are satisfied for *the same* λ, and there are infinitely many such λ.

As we have stated above, for this specific $\{f_{p,\lambda}\}_{\lambda\in\mathbb{N}}$, there exists a PPT algorithm $\mathcal{R}_p^{f_{p,\lambda}}$ such that Equations (179) and (180) hold for infinitely many $\lambda \in \mathbb{N}$. From $\mathcal{S}_p$ and $\mathcal{R}_p^{f_{p,\lambda}}$, we construct a PPT algorithm $\mathcal{A}$ such that

$$\Pr[\top \leftarrow \mathsf{Ver}(\mathsf{puzz}, \mathcal{A}(1^\lambda, \mathsf{puzz})) : (\mathsf{puzz}, \mathsf{ans}) \leftarrow \mathsf{Samp}(1^\lambda)] \geq c(\lambda) - \frac{1}{2q(\lambda)} \tag{181}$$

holds for infinitely many $\lambda \in \mathbb{N}$. This means that $\mathcal{A}$ breaks s-security of the classically-secure OWPuzz $(\mathsf{Samp}, \mathsf{Ver})$. We define such $\mathcal{A}$ as follows:

- $\mathcal{A}(1^\lambda, \mathsf{puzz}) \to \mathsf{ans}'$:
  1. Take 1^λ and puzz as input.
  2. Run $r \leftarrow \mathcal{R}_{6q}^{f_{6q,\lambda}}(1^{n(\lambda)}, \mathsf{puzz})$.
  3. Run $\mathcal{S}_{6q}(1^\lambda; r) = (\mathsf{ans}', \mathsf{puzz}')$.
  4. Output ans'.

Then,

$$\begin{aligned}
&\Pr[\top \leftarrow \mathsf{Ver}(\mathsf{puzz}, \mathcal{A}(1^\lambda, \mathsf{puzz})) : (\mathsf{ans}, \mathsf{puzz}) \leftarrow \mathsf{Samp}(1^\lambda)] && (182)\\
&\geq \Pr[\top \leftarrow \mathsf{Ver}(\mathsf{puzz}, \mathcal{A}(1^\lambda, \mathsf{puzz})) : (\mathsf{ans}, \mathsf{puzz}) \leftarrow \mathcal{S}_{6q}(1^\lambda)] - \mathsf{SD}(\mathsf{Samp}(1^\lambda), \mathcal{S}_{6q}(1^\lambda)) && (183)\\
&= \Pr\left[\top \leftarrow \mathsf{Ver}(\mathsf{puzz}, \mathsf{ans}') : \begin{array}{r} r \leftarrow \{0,1\}^{n(\lambda)}, \\ \mathsf{puzz} = f_{6q,\lambda}(r), \\ r' \leftarrow \mathcal{R}_{6q}^{f_{6q,\lambda}}(1^{n(\lambda)}, \mathsf{puzz}), \\ (\mathsf{puzz}', \mathsf{ans}') = \mathcal{S}_{6q}(1^\lambda; r') \end{array}\right] - \frac{1}{6q(\lambda)} && (184)\\
&\geq \Pr\left[\top \leftarrow \mathsf{Ver}(\mathsf{puzz}, \mathsf{ans}') : \begin{array}{r} r \leftarrow \{0,1\}^{n(\lambda)}, \\ \mathsf{puzz} = f_{6q,\lambda}(r), \\ (\mathsf{puzz}', \mathsf{ans}') = \mathcal{S}_{6q}(1^\lambda; r) \end{array}\right] - \frac{2}{6q(\lambda)} && (185)\\
&= \Pr[\top \leftarrow \mathsf{Ver}(\mathsf{puzz}, \mathsf{ans}) : (\mathsf{puzz}, \mathsf{ans}) \leftarrow \mathcal{S}_{6q}(1^\lambda)] - \frac{2}{6q(\lambda)} && (186)\\
&\geq \Pr[\top \leftarrow \mathsf{Ver}(\mathsf{puzz}, \mathsf{ans}) : (\mathsf{puzz}, \mathsf{ans}) \leftarrow \mathsf{Samp}(1^\lambda)] - \frac{3}{6q(\lambda)} && (187)\\
&\geq c(\lambda) - \frac{1}{2q(\lambda)} && (188)
\end{aligned}$$

holds for infinitely many $\lambda \in \mathbb{N}$. Here, we have used Equation (179) to obtain Equations (184) and (187), and used Equation (180) to obtain Equation (185). Therefore, $\mathcal{A}$ breaks the soundness of classically-secure OWPuzz $(\mathsf{Samp}, \mathsf{Ver})$.

□

6 Variants of IV-PoQ

In Section 6.1 we show equivalence among variants of IV-PoQ. In Section 6.2, we introduce *zero-knowledge* IV-PoQ and show their relationship with OWFs.

6.1 Equivalence Among Variants of IV-PoQ

We consider the following variant of IV-PoQ.

Definition 6.1 (Quantum-Verifier IV-PoQ). *A quantum-verifier IV-PoQ $(\mathcal{P}, \mathcal{V}_1, \mathcal{V}_2)$ is defined similarly to IV-PoQ (Definition 2.18) except that $\mathcal{V}_1$ is QPT instead of PPT but still only sends classical messages.*

We show that the following equivalence theorem.

Theorem 6.2. *The following are equivalent:*

1. *Public-coin IV-PoQ exist.*
2. *IV-PoQ exist.*
3. *Quantum-verifier IV-PoQ exist.*

Proof of Theorem 6.2. It is clear that Item 1 implies Item 2 and Item 2 implies Item 3. Below, we show that Item 3 implies Item 1. To show this, we first observe that quantum-verifier IV-PoQ imply Int-QAS since the proof of Theorem 5.1 works even if $\mathcal{V}_1$ is QPT. Thus, combined with Theorem 5.2, quantum-verifier IV-PoQ imply the QAS/OWF condition. Next, we observe that the IV-PoQ constructed from the QAS/OWF condition in the proof of Theorem 5.3 is public-coin. To see this, recall that the IV-PoQ is obtained by combining the non-interactive protocol obtained by Lemma 3.8 and the IV-PoQ based on OWFs by Lemma 2.21. The former is non-interactive and in particular there is no message sent from the verifier, and thus it is public-coin. Moreover, the latter is also public-coin by Lemma 2.21. Thus, their combination (as in the proof of Theorem 5.3) also results in public-coin IV-PoQ. Thus, the IV-PoQ constructed in the proof of Theorem 5.3 is public-coin. Combining the above observations, we complete the proof that quantum-verifier IV-PoQ imply public-coin IV-PoQ. □

6.2 Zero-Knowledge IV-PoQ

We give a definition of zero-knowledge IV-PoQ below.

Definition 6.3 (Zero-Knowledge IV-PoQ). *An IV-PoQ $(\mathcal{P}, \mathcal{V}_1, \mathcal{V}_2)$ satisfies computational (resp. statistical) zero-knowledge if for any PPT malicious verifier $\mathcal{V}_1^*$, there exists a PPT simulator $\mathcal{S}$ such that for any PPT (resp. unbounded-time) distinguisher $\mathcal{D}$,*

$$\left|\Pr[\mathcal{D}(\mathsf{view}\langle \mathcal{P}, \mathcal{V}_1^* \rangle(1^\lambda)) = 1] - \Pr[\mathcal{D}(\mathcal{S}(1^\lambda)) = 1]\right| \leq \mathsf{negl}(\lambda) \tag{189}$$

where $\mathsf{view}\langle \mathcal{P}, \mathcal{V}_1^* \rangle(1^\lambda)$ means the view of $\mathcal{V}_1^*$ which consists of the transcript and the random coin of $\mathcal{V}_1^*$.

We say that an IV-PoQ $(\mathcal{P}, \mathcal{V}_1, \mathcal{V}_2)$ satisfies honest-verifier computational (resp. statistical) zero-knowledge if the above holds for the case of $\mathcal{V}_1^* = \mathcal{V}_1$.

Remark 6.4. Standard definitions of the zero-knowledge property in the literature usually consider *non-uniform* malicious verifiers and distinguishers. On the other hand, since we treat the uniform model as a default notion in this paper, we define the zero-knowledge property in the uniform-style as above. However, we remark that this choice of model of computation is not essential for the results of this subsection, and all the results of this subsection readily extend to the non-uniform setting with essentially the same proofs.

We show relationships between zero-knowledge IV-PoQ and OWFs. First, we show that honest-verifier statistical zero-knowledge IV-PoQ imply classically-secure OWFs.

Theorem 6.5. *If honest-verifier statistical zero-knowledge IV-PoQ exist, then classically-secure OWFs exist.*

Proof of Theorem 6.5. Let $(\mathcal{P},\mathcal{V}_1,\mathcal{V}_2)$ be an honest-verifier statistical zero-knowledge IV-PoQ. By the proof of Theorem 5.1, $(\mathcal{P},\mathcal{V}_1)$ is an Int-QAS, and thus by the proof of Theorem 5.2, the QAS/OWF condition holds where $\mathcal{Q}=\langle\mathcal{P},\mathcal{V}_1\rangle$. That is, there exist a polynomial p, and a function $f:\{0,1\}^*\to\{0,1\}^*$ that is computable in classical deterministic polynomial-time such that for any PPT algorithm $\mathcal{S}$, the following holds: if we define

$$\Sigma_{\mathcal{S}} := \left\{\lambda\in\mathbb{N} \;\middle|\; \mathsf{SD}(\langle\mathcal{P},\mathcal{V}_1\rangle(1^\lambda),\mathcal{S}(1^\lambda)) \le \frac{1}{p(\lambda)}\right\}, \tag{190}$$

then f is a classically-secure OWF on $\Sigma_{\mathcal{S}}$. By the honest-verifier statistical zero-knowledge property of $(\mathcal{P},\mathcal{V}_1,\mathcal{V}_2)$, there is a PPT simulator $\mathcal{S}$ such that $\mathsf{SD}(\langle\mathcal{P},\mathcal{V}_1\rangle(1^\lambda),\mathcal{S}(1^\lambda)) \le \mathsf{negl}(\lambda)$.[29] For this $\mathcal{S}$, $\Sigma_{\mathcal{S}}$ consists of all but finite elements of $\mathbb{N}$ by the definition of negligible functions. Since f is a classically-secure OWF on $\Sigma_{\mathcal{S}}$, this implies the existence of classically secure OWFs by Lemma 2.5. □

Next, we show that OWFs imply computational zero-knowledge IV-PoQ.

Theorem 6.6. *If classically-secure OWFs exist, then computational zero-knowledge IV-PoQ exist.*

To prove Theorem 6.6 we rely on (classically-secure) extractable commitments. The following definition is based on the one in [PW09].

Definition 6.7 (Classically-Secure Extractable Commitments). *A classically-secure extractable commitment scheme consists of interactive PPT algorithms $\mathcal{S}$ (sender) and $\mathcal{R}$ (receiver). An execution of the scheme is divided into two phases, the commit phase and open phase. In the commit phase, $\mathcal{S}$ takes the security parameter 1^λ and message m as input, $\mathcal{R}$ takes the security parameter 1^λ as input, and $\mathcal{S}$ and $\mathcal{R}$ interact with each other where $\mathcal{R}$ may declare rejection and abort at any point of the commit phase. We call a transcript of the commit phase a commitment and denote it by com. In the open phase, $\mathcal{S}$ sends m and a classical string d (decommitment) to $\mathcal{R}$ and $\mathcal{R}$ outputs $\top$ (accept) or $\bot$ (reject). We assume that $\mathcal{R}$ is stateless, i.e., there is no state information of $\mathcal{R}$ kept from the commit phase.*[30] *We require the following four properties.*

- **Correctness.** *For any λ and m, if $\mathcal{S}(1^\lambda,m)$ and $\mathcal{R}(1^\lambda)$ run the commit and open phases honestly, then $\mathcal{R}$ always accepts.*
- **Computational hiding.** *For any (stateful) PPT malicious receiver $\mathcal{R}^*$, we have*

$$\left|\Pr\left[\mathsf{Out}_{\mathcal{R}^*}\langle\mathcal{S}(m_b),\mathcal{R}^*\rangle(1^\lambda)=b : \begin{array}{l}(m_0,m_1)\leftarrow\mathcal{R}^*(1^\lambda)\\ b\leftarrow\{0,1\}\end{array}\right]-\frac{1}{2}\right| \le \mathsf{negl}(\lambda) \tag{191}$$

  where $\mathsf{Out}_{\mathcal{R}^*}\langle\mathcal{S}(m_b),\mathcal{R}^*\rangle(1^\lambda)$ means the output of $\mathcal{R}^*$ after interacting with $\mathcal{S}$ with the common input 1^λ and private input m_b in the commit phase.
- **Statistical binding.**[31] *For any unbounded-time cheating sender $\mathcal{S}^*$, let com be a commitment generated by $\mathcal{S}^*$ and the honest receiver $\mathcal{R}$. Then there is at most one m such that there exists d such that $\mathcal{R}$ accepts (m,d) as an opening of com except for a negligible probability.*

[29]Recall that we write $\langle\mathcal{P},\mathcal{V}_1\rangle(1^\lambda)$ to mean the machine that outputs a transcript of interaction between $\mathcal{P}$ and $\mathcal{V}_1$. Since the honest-verifier statistical zero-knowledge requires the simulator to simulate both the transcript and the verifier's randomness, it is trivial to simulate only the transcript.

[30]We can also define a commitment scheme with a stateful receiver, but we focus on the stateless receiver case for simplicity.

[31]This follows from extractability, but we state it separately for convenience.

- **Extractability.** *There is an expected PPT oracle machine (the extractor) $\mathcal{E}$ that takes the security parameter 1^λ as input, given oracle access to any deterministic unbounded-time cheating sender $\mathcal{S}^*$, and outputs a pair (com, m^*) such that:*
  - **Simulation:** com *is identically distributed to a commitment generated by $\mathcal{S}^*$ and $\mathcal{R}$.*
  - **Extraction:** *the probability that* com *is accepting (i.e., the transcript indicates that $\mathcal{R}$ never aborts in the commit phase) and $m^* = \bot$ is negligible.*
  - **Binding:** *If $m^* \neq \bot$, then there is no $m \neq m^*$ and d such that $\mathcal{R}$ accepts (m, d) as an opening of* com.

Theorem 6.8 ([PW09]). *If classically-secure OWFs exist, then classically-secure extractable commitments exist.*

We introduce a notation on extractable commitments. For a commitment com, if there is a unique m such that there is d such that $\mathcal{R}$ accepts (m, d) as an opening of com, then we define $\mathsf{val}(\mathsf{com}) = m$. If such m does not exist or not unique, we define $\mathsf{val}(\mathsf{com}) = \bot$. The following lemmas are easy to prove.

Lemma 6.9. *If* com *is generated by $\mathcal{S}(1^\lambda, m)$ and $\mathcal{R}(1^\lambda)$, then* $\mathsf{val}(\mathsf{com}) = m$ *except for a negligible probability.*

Proof. This immediately follows from correctness and statistical binding. □

Lemma 6.10. *For any deterministic unbounded-time cheating sender $\mathcal{S}^*$, if $\mathcal{E}^{\mathcal{S}^*}(1^\lambda)$ outputs* (com, m^*)*, then* $\mathsf{val}(\mathsf{com}) = m^*$ *or* $\mathsf{val}(\mathsf{com}) = \bot$ *except for a negligible probability.*

Proof. This immediately follows from the second and third items of the extractability. □

Then we prove Theorem 6.6.

Proof of Theorem 6.6. By Lemma 2.21, public-coin IV-PoQ exist if classically-secure OWFs exist. By Theorem 6.8, classically-secure extractable commitments exist if classically-secure OWFs exist. Thus, it suffices to construct computational zero-knowledge IV-PoQ from public-coin IV-PoQ and classially-secure extractable commitments. We show how to do it below.

Let $\Pi = (\mathcal{P}, \mathcal{V}_1, \mathcal{V}_2)$ be a public-coin IV-PoQ that satisfies c-completeness and s-soundness and let ExtCom be a classically-secure extractable commitment scheme. Without loss of generality, we assume that Π is an ℓ-round protocol where the first message is sent from the verifier and verifier's messages in each round are n-bit strings for some polynomials $\ell = \ell(\lambda)$ and $n = n(\lambda)$. Then we construct a computational zero-knowledge IV-PoQ $\tilde{\Pi} = (\tilde{\mathcal{P}}, \tilde{\mathcal{V}}_1, \tilde{\mathcal{V}}_2)$ that works as follows.

(The first phase)

- Upon receiving the security parameter 1^λ, $\tilde{\mathcal{P}}$ sets $\mathsf{st}_0 := |1^\lambda\rangle$.
- For $i = 1, 2, \ldots, \ell$, do the following:
  - $\tilde{\mathcal{V}}_1$ chooses $v_i \leftarrow \{0,1\}^n$ and sends v_i to $\tilde{\mathcal{P}}$.
  - $\tilde{\mathcal{P}}$ runs $\mathcal{P}$ on the state st_{i-1} and i-th verifier's message v_i to generate i-th prover's message p_i. Let st_i be the internal state of $\mathcal{P}$ at this point.
  - $\tilde{\mathcal{P}}$ commits to p_i using ExtCom where $\tilde{\mathcal{P}}$ and $\tilde{\mathcal{V}}_1$ play the roles of the sender and receiver, respectively. Let com_i be the commitment generated in this step.

(The second phase)

- Upon receiving 1^λ and a transcript $\tilde{\tau} = (v_1, \mathsf{com}_1, v_2, \mathsf{com}_2, \ldots, v_\ell, \mathsf{com}_\ell)$ of the first phase, $\tilde{\mathcal{V}}_2$ computes $p'_i = \mathsf{val}(\mathsf{com}_i)$ by brute-force for all $i \in [\ell]$. If $p'_i = \perp$ for some $i \in [\ell]$, then $\tilde{\mathcal{V}}_2$ outputs $\perp$. Otherwise $\tilde{\mathcal{V}}_2$ runs $\mathcal{V}_2$ on transcript $\tau = (v_1, p'_1, v_2, p'_2, \ldots, v_\ell, p'_\ell)$ and outputs whatever $\mathcal{V}_2$ outputs.

By Lemma 6.9, we have $p'_i = p_i$ except for a negligible probability when we run the protocol honestly. Thus, $\tilde{\Pi}$ satisfies $(c - \mathsf{negl})$-completeness by c-completeness of Π. The computational zero-knowledge property of $\tilde{\Pi}$ immediately follows from the computational hiding property of ExtCom since a simulator can simply commit to 0...0 instead of to p_i.

Below, we reduce $(s + \mathsf{negl})$-soundness of $\tilde{\Pi}$ to s-soundness of Π using the extractability of ExtCom as follows. Toward contradiction, suppose that there is a PPT cheating prover $\tilde{\mathcal{P}}^*$ such that

$$\Pr[\top \leftarrow \tilde{\mathcal{V}}_2(1^\lambda, \tau) : \tau \leftarrow \langle \tilde{\mathcal{P}}^*, \tilde{\mathcal{V}}_1 \rangle (1^\lambda)] \geq s(\lambda) + 1/q(\lambda) \tag{192}$$

for some polynomial q and infinitely many λ. For each $j \in \{0, 1, \ldots, \ell\}$, we consider a (not necessarily PPT) cheating provers $\mathcal{P}^*_j$ against Π that works as follows:

- $\mathcal{P}^*_j$ takes randomness r, which has the same length as randomness of $\tilde{\mathcal{P}}^*$.
- For $i = 1, 2, \ldots, j$, do the following:
  - Receive v_i from $\tilde{\mathcal{V}}_1$.
  - Let $\tilde{\mathcal{P}}^*[r, \tau_i]$ be the part of $\tilde{\mathcal{P}}^*$ that runs the i-th execution of ExtCom where we hardwire the randomness r and the partial transcript $\tau_i = (v_1, \mathsf{com}_1, \ldots, v_{i-1}, \mathsf{com}_{i-1}, v_i)$.
  - Run $(\mathsf{com}_i, p_i) \leftarrow \mathcal{E}^{\tilde{\mathcal{P}}^*[r,\tau_i]}(1^\lambda)$ and send p_i to $\tilde{\mathcal{V}}_1$.
- For $i = j+1, j+2, \ldots, \ell$, do the following:
  - Receive v_i from $\tilde{\mathcal{V}}_1$.
  - Run $\tilde{\mathcal{P}}^*$ on the fixed randomness r and the partial transcript $\tau_i = (v_1, \mathsf{com}_1, \ldots, v_{i-1}, \mathsf{com}_{i-1}, v_i)$ to complete the i-th execution of ExtCom. Let com_i be the commitment generated in this step.
  - Compute $p_i = \mathsf{val}(\mathsf{com}_i)$ by brute-force.
  - If $p_i = \perp$, abort. Otherwise send p_i to $\mathcal{V}_1$.

By the definition, it is easy to see that

$$\Pr[\top \leftarrow \mathcal{V}_2(1^\lambda, \tau) : \tau \leftarrow \langle \mathcal{P}^*_0, \mathcal{V}_1 \rangle (1^\lambda)] = \Pr[\top \leftarrow \tilde{\mathcal{V}}_2(1^\lambda, \tau) : \tau \leftarrow \langle \tilde{\mathcal{P}}^*, \tilde{\mathcal{V}}_1 \rangle (1^\lambda)]. \tag{193}$$

For $j \in [\ell]$, the only difference between $\mathcal{P}^*_{j-1}$ and $\mathcal{P}^*_j$ is how to generate (com_j, p_j): $\mathcal{P}^*_{j-1}$ runs $\tilde{\mathcal{P}}^*$ to generate com_j and then sets $p_j = \mathsf{val}(\mathsf{com}_j)$ by brute-force whereas $\mathcal{P}^*_j$ generates them by running the extractor. By the first item of extractability, the distributions of com_j are identical for both cases. Moreover, by Lemma 6.10, p_j takes the same value in both cases unless $\mathsf{val}(\mathsf{com}_j) = \perp$ except for a negligible probability. Note that if $\mathsf{val}(\mathsf{com}_j) = \perp$, then $\mathcal{P}^*_{j-1}$ aborts and thus never passes the verification of $\mathcal{V}_2$. Thus, we have

$$\Pr[\top \leftarrow \mathcal{V}_2(1^\lambda, \tau) : \tau \leftarrow \langle \mathcal{P}^*_j, \mathcal{V}_1 \rangle (1^\lambda)] \geq \Pr[\top \leftarrow \mathcal{V}_2(1^\lambda, \tau) : \tau \leftarrow \langle \mathcal{P}^*_{j-1}, \mathcal{V}_1 \rangle (1^\lambda)] - \mathsf{negl}(\lambda). \tag{194}$$

By combining Equations (192) to (194),

$$\Pr[\top \leftarrow \mathcal{V}_2(1^\lambda, \tau) : \tau \leftarrow \langle \mathcal{P}_\ell^*, \mathcal{V}_1\rangle(1^\lambda)] \geq s(\lambda) + 1/q(\lambda) - \mathsf{negl}(\lambda) \tag{195}$$

for infinitely many λ.

We remark that $\mathcal{P}_\ell^*$ no longer runs brute-force, but it is still not PPT since it runs the extractor $\mathcal{E}$ that runs in *expected* PPT. This can be converted into a PPT machine by a standard truncation technique. That is, let $\mathcal{P}_{\ell+1}^*$ be a cheating prover that runs similarly to $\mathcal{P}_\ell^*$ except that whenever it runs $\mathcal{E}$, if its running time is $2\ell(\lambda)q(\lambda)$ times larger than its expectation, then it aborts. Clearly, $\mathcal{P}_{\ell+1}^*$ runs in PPT. Moreover, by Markov's inequality, for each invocation of $\mathcal{E}$, the probability that $\mathcal{P}_{\ell+1}^*$ aborts is at most $1/(2\ell(\lambda)q(\lambda))$. Since $\mathcal{P}_{\ell+1}^*$ runs $\mathcal{E}$ $\ell(\lambda)$ times, by the union bound, the probability that this occurs is at most $1/(2q(\lambda))$. Thus, Equation (195), we have

$$\Pr[\top \leftarrow \mathcal{V}_2(1^\lambda, \tau) : \tau \leftarrow \langle \mathcal{P}_{\ell+1}^*, \mathcal{V}_1\rangle(1^\lambda)] \geq s + 1/(2q(\lambda)) - \mathsf{negl}(\lambda) \tag{196}$$

for infinitely many λ. This contradicts s-soundness of Π. Thus, $\tilde{\Pi}$ satisfies $(s + \mathsf{negl}(\lambda))$-soundness. □

Zero-knowledge PoQ. Though our main focus is on IV-PoQ, we briefly discuss zero-knowledge (efficiently-verifiable) PoQ. First, we observe that the conversion in the proof of Theorem 6.6 works in the efficiently-verifiable setting as well if we introduce an additional layer of zero-knowledge proofs where the prover proves that the committed transcript passes the verification. However, the conversion requires the base PoQ to be public-coin while most existing PoQ are not public-coin. Fortunately, we observe that we can relax the public-coin property to the "transcript-independent" property which means that the distribution of verifier's messages does not depend on the transcript and only depends on the verifier's randomness. At first glance, one may think that it is problematic if the verifier uses its private randomness to make a decision in which case the statement that "the committed transcript passes the verification" is not an **NP** statement. However, this issue can be resolved by letting the verifier reveal its randomness after receiving all the commitments from the prover.[32] Since the randomness is revealed after the commitments are sent, a cheating prover can no longer change the committed transcript by the binding property of the extractable commitment, and thus this does not affect the soundness. In summary, we can generically upgrade any PoQ with transcript-independent verifiers into a (computational) zero-knowledge PoQ by additionally assuming the existence of OWFs. To our knowledge, all existing PoQ [BCM+21, KMCVY22, YZ24, MY23, KLVY23, AGGM24] have transcript-independent verifiers.

Toward equivalence between OWFs and zero-knowledge IV-PoQ. Theorems 6.5 and 6.6 can be regarded as a loose equivalence between OWFs and zero-knowledge IV-PoQ. However, there is a gap between them as Theorem 6.5 assumes honest-verifier *statistical* zero-knowledge while Theorem 6.6 only gives *computational* zero-knowledge. It is an interesting open question if we can fill the gap.

There are two approaches toward solving that. One is to show that computational zero-knowledge IV-PoQ imply OWFs and the other is to show that OWFs imply honest-verifier statistical zero-knowledge IV-PoQ. For the former approach, the technique of [OW93, Vad06], which shows that computational zero-knowledge arguments for average-case-hard languages imply OWFs, might be useful, but it is unclear how to adapt their technique to the setting of IV-PoQ.

We also do not have solution for the latter approach either, but we have the following observation. We observe that we can construct statistical zero-knowledge IV-PoQ (or even efficiently-verifiable PoQ) if

[32]A similar idea is used in [BKL+22].

we additionally assume the existence of an $\mathbf{NP}$ search problem that is easy for QPT algorithms but hard for PPT algorithms (or equivalently publicly-verifiable one-round PoQ). To see this, we can consider a protocol where the honest quantum prover solves the $\mathbf{NP}$ search problem and then proves the knowledge of the solution by using statistical zero-knowledge arguments of knowledge for $\mathbf{NP}$, which exists if OWFs exist [HNO$^+$09]. Examples of classically-hard and quantumly-easy $\mathbf{NP}$ search problems are the factoring and discrete-logarithm problems (assuming classical hardness of them) [Sho94]. Another example based on a random oracle was recently found in [YZ24]. Thus, based on the random oracle heuristic [BR93], we have a candidate construction of statistical zero-knowledge PoQ from hash functions.[33] Though this is far from a construction solely based on OWFs, this can be seen as an evidence that "structured" assumptions are not necessary for statistical zero-knowledge PoQ, let alone for statistical zero-knowledge IV-PoQ.[34]

Acknowledgements. TM is supported by JST CREST JPMJCR23I3, JST Moonshot R&D JPMJMS2061-5-1-1, JST FOREST, MEXT QLEAP, the Grant-in Aid for Transformative Research Areas (A) 21H05183, and the Grant-in-Aid for Scientific Research (A) No.22H00522. YS is supported by JST SPRING, Grant Number JPMJSP2110.

A On Uniformity of Adversaries

As mentioned in Footnote 3, we consider the uniform adversarial model in this paper. The only place where the uniformity of the adversary plays a crucial role is in the proof of Lemma 3.8 where we relate the hardness of non-interactive search and sampling problems. In the proof, we make heavy use of Kolmogorov complexity, which is defined with respect to uniform Turing machines, and thus the same proof does not work when we consider non-uniform adversaries. Thus, all the results that rely on Lemma 3.8 do not extend to the non-uniform setting. They include Lemma 3.7 and Theorems 5.3, 5.4 and 6.2. On the other hand, Theorems 4.2, 5.1, 5.2, 5.5, 6.5 and 6.6 seem to extend to the non-uniform setting with appropriate adaptations.

B Proof of Lemma 2.5

Proof of Lemma 2.5. The only if part is trivial. We show the if part. Let $f : \{0,1\}^* \to \{0,1\}^*$ be a OWF on $\mathbb{N} \setminus \Sigma$ for a finite subset $\Sigma \subseteq \mathbb{N}$. Then by the definition of OWFs on $\mathbb{N} \setminus \Sigma$, there exists an efficiently-computable polynomial n such that for any PPT adversary $\mathcal{A}$ and any polynomial p there exists $\lambda^* \in \mathbb{N}$ such that

$$\Pr[f(x') = f(x) : x \leftarrow \{0,1\}^{n(\lambda)}, x' \leftarrow \mathcal{A}(1^{n(\lambda)}, f(x))] \leq \frac{1}{p(\lambda)} \tag{197}$$

for all $\lambda \geq \lambda^*$ in $\mathbb{N} \setminus \Sigma$. Since Σ is a finite set, we can assume that λ^* is larger than the largest element of Σ without loss of generality. Then, Equation (197) holds for all $\lambda \geq \lambda^*$ in $\mathbb{N}$.

From such f, we construct a OWF $g : \{0,1\}^* \to \{0,1\}^*$ as follows.

1. On input $x \in \{0,1\}^\ell$, find the maximum λ such that $n(\lambda) \leq \ell$. Set λ_ℓ to be such maximum λ.

[33] This is *not* a construction in the quantum random oracle model since we use the hash function in a non-black-box manner. Instead, we rely on the assumption that the problem considered in [YZ24] is classically hard when the random oracle is instantiated with a concrete hash function.

[34] "Structure" is a commonly used informal term that refers to problems behind constructions of existing public key encryption such as the hardness of factoring, discrete-logarithm, learning with errors, etc. On the other hand, hash functions are often regarded as "unstructured". See [Bar17] for more context.

2. Output $g(x) := f(x_{1,...,n(\lambda_\ell)})$, where $x_{1,...,n(\lambda_\ell)}$ is the first $n(\lambda_\ell)$ bits of x.

Then for any PPT $\mathcal{A}$ and any polynomial p, if we define a polynomial $q(\lambda) := p(n(\lambda+1))$, there exists $\lambda^*_{\mathcal{A},q} \in \mathbb{N}$ such that

$$\Pr[g(x') = g(x) : x \leftarrow \{0,1\}^\ell, x' \leftarrow \mathcal{A}(1^\ell, g(x))] \tag{198}$$
$$= \Pr[f(x'_{1,...,n(\lambda_\ell)}) = f(x_{1,...,n(\lambda_\ell)}) : x \leftarrow \{0,1\}^\ell, x' \leftarrow \mathcal{A}(1^\ell, f(x_{1,...,n(\lambda_\ell)}))] \tag{199}$$
$$= \Pr[f(x'_{1,...,n(\lambda_\ell)}) = f(x) : x \leftarrow \{0,1\}^{n(\lambda_\ell)}, x' \leftarrow \mathcal{A}(1^\ell, f(x))] \tag{200}$$
$$= \Pr[f(w) = f(x) : x \leftarrow \{0,1\}^{n(\lambda_\ell)}, w \leftarrow \mathcal{A}(1^{n(\lambda_\ell)}, f(x))] \tag{201}$$
$$\leq \frac{1}{q(\lambda_\ell)} \tag{202}$$
$$= \frac{1}{p(n(\lambda_\ell+1))} \tag{203}$$
$$\leq \frac{1}{p(\ell)} \tag{204}$$

if $\lambda_\ell \geq \lambda^*_{\mathcal{A},q}$. For all sufficiently large $\ell \in \mathbb{N}$, $\lambda_\ell \geq \lambda^*_{\mathcal{A},q}$, and therefore the above inequality is satisfied for all sufficiently large $\ell \in \mathbb{N}$. This means that g is a OWF. ☐

C Proof of Lemma 2.10

Proof of Lemma 2.10. The proof is similar to the universal construction technique [Lev85, HKN+05] of OWFs. Let $M_1, M_2, ...$ be an enumeration of all Turing machines. We construct a function g as follows:

- $g : \{0,1\}^* \to \{0,1\}^*$:
  1. Take $y \in \{0,1\}^\ell$ as input.
  2. Let $N \in \mathbb{N}$ be the maximum value such that $N^2 \leq \ell$.
  3. Let $y_1\|y_2\|...\|y_N$ be the N^2-bit prefix of y, where $y_i \in \{0,1\}^N$ for all $i \in [N]$.
  4. For all $i \in [N]$, run $M_i(y_i)$ for N^3 steps. If $M_i(y_i)$ halts, then set $v_i := M_i(y_i)$. Otherwise, set $v_i := \perp$.
  5. Output $v_1\|v_2\|...\|v_N$.

The time to compute $g(y)$ is $O(|y|^2)$ for all $y \in \{0,1\}^*$ and therefore g is computable in classical deterministic polynomial-time.

Fix a subset $\Sigma \subseteq \mathbb{N}$. Let $f : \{0,1\}^* \to \{0,1\}^*$ be a classically-secure OWF on Σ. From Lemma C.1 below, we can assume that the time to compute $f(x)$ is $O(|x|^2)$ for any $x \in \{0,1\}^*$. Then, by the definition of OWFs on Σ, there exists an efficiently-computable polynomial n such that for any PPT algorithm $\mathcal{R}$ and for any polynomial p, there exists $\lambda^* \in \mathbb{N}$ such that

$$\Pr[f(z) = f(x) : x \leftarrow \{0,1\}^{n(\lambda)}, z \leftarrow \mathcal{R}(1^{n(\lambda)}, f(x))] \leq \frac{1}{p(\lambda)} \tag{205}$$

holds for all $\lambda \geq \lambda^*$ in Σ. Our goal is to show that g is a classically-secure OWF on Σ.

For the sake of contradiction, we assume that g is not a classically-secure OWF on Σ. Then, Σ is an infinite subset of $\mathbb{N}$, because if Σ is finite, g is trivially a OWF on Σ. Moreover, by the definition of OWFs on Σ, for any efficiently-computable polynomial m, there exist a PPT algorithm $\mathcal{A}$, a polynomial q and an infinite subset $\Lambda \subseteq \Sigma$ such that

$$\Pr[g(w) = g(y) : y \leftarrow \{0,1\}^{m(\lambda)}, w \leftarrow \mathcal{A}(1^{m(\lambda)}, g(y))] > \frac{1}{q(\lambda)} \tag{206}$$

holds for all $\lambda \in \Lambda$. Because f is computable in classical deterministic polynomial-time, there exists a deterministic Turing machine F such that $F(x) = f(x)$ for all $x \in \{0,1\}^*$. Moreover, because as we have said the time to compute $f(x)$ is $O(|x|^2)$, the running time of F is $O(|x|^2)$. The description size of F is a constant of $|x|$, and therefore there exists $\alpha^* \in \mathbb{N}$ such that $M_{\alpha^*} = F$. We define the PPT algorithm $\mathcal{R}^*$ as follows:

- $\mathcal{R}^*(1^{n(\lambda)}, f(x)) \to z$:
  1. Take $1^{n(\lambda)}$ and $f(x)$ as input, where $x \leftarrow \{0,1\}^{n(\lambda)}$.
  2. If $n(\lambda) \geq \alpha^*$, set $j = \alpha^*$. Otherwise, set $j = 1$. Set $v_j := f(x)$.
  3. Set $N := n(\lambda)$.
  4. For all $i \in [N]$ except for j, sample $y_i \leftarrow \{0,1\}^N$ and run $M_i(y_i)$ for N^3 steps. If $M_i(y_i)$ halts, set $v_i := M_i(y_i)$. Otherwise, set $v_i := \perp$.
  5. Run $\mathcal{A}(1^{N^2}, v_1 \| ... \| v_N)$ to obtain $w \in \{0,1\}^m$.[35]
  6. Parse $w := w_1 \| ... \| w_N \| w'$, where $w_i \in \{0,1\}^N$ for all $i \in [N]$ and $w' \in \{0,1\}^{m-N^2}$.
  7. Output $z := w_j$.

We can show that for any efficiently computable polynomial n, there exists a PPT algorithm $\mathcal{R}^*$ and a polynomial q and an infinite subset $\Lambda' := \{\lambda \in \Lambda : n(\lambda) \geq \alpha^*\}$ of Σ such that

$$\Pr[f(z) = f(x) : x \leftarrow \{0,1\}^{n(\lambda)}, z \leftarrow \mathcal{R}^*(1^{n(\lambda)}, f(x))] \tag{207}$$
$$\geq \Pr[g(w) = g(y) : y \leftarrow \{0,1\}^{n(\lambda)^2}, w \leftarrow \mathcal{A}(1^{n(\lambda)^2}, g(y))] \tag{208}$$
$$> \frac{1}{q(\lambda)} \tag{209}$$

holds for all $\lambda \in \Lambda'$. Here, in Equation (208) we have used the fact that the time to compute $f(x)$ is $O(|x|^2)$, and in Equation (209), we have used Equation (206). This contradicts Equation (205), and therefore g is a classically-secure OWF on Σ. □

Lemma C.1. *Let $\Sigma \subseteq \mathbb{N}$ be a set. If there exists a OWF f on Σ, there exists a OWF f' on Σ such that the time to compute $f'(x)$ is $O(|x|^2)$ for any $x \in \{0,1\}^*$.*

Proof of Lemma C.1. The proof is based on the padding argument of [Gol04]. Let $\Sigma \subseteq \mathbb{N}$ be a set. Let f be a OWF on Σ. If the time to compute f is $O(|x|^c)$ for $c > 2$, we define a function f' as follows.

1. On input a bit string, $z \in \{0,1\}^r$, find the maximum $i \in [r]$ such that $|z| \geq |z_{1,...,i}|^c$, where $z_{1,...,i}$ is the first i bits of z.

2. Output $f'(z) := f(z_{1,...,i}) \| z_{i+1,...,r}$, where $z_{i+1,...,r}$ is the last $r - i$ bits of z.

[35]Note that the length m of the output w of $\mathcal{A}$ is not necessarily equal to the input length N^2, because $g(w) = g(y)$ could happen in Equation (206) for some w whose length is longer than N^2.

The time to compute $f'(z)$ is $O(|z|^2)$ because finding the maximum i spends $O(|z|^2)$ time[36] and computing $f(z_{1,\dots,i})$ spends $O(|z_{1,\dots,i}|^c) \le O(|z|)$ time.

Moreover, we can show that f' is a OWF on Σ. For the sake of contradiction, assume that f' is not a OWF on Σ. Then by the definition of OWFs on Σ, for any efficiently computable polynomial n, there exist a PPT adversary $\mathcal{A}$, a polynomial p, and an infinite subset $\Lambda \subseteq \Sigma$ such that

$$\Pr[f'(z) = f'(w) : z \leftarrow \{0,1\}^{n(\lambda)}, w \leftarrow \mathcal{A}(1^{n(\lambda)}, f'(z))] > \frac{1}{p(\lambda)} \tag{210}$$

holds for all $\lambda \in \Lambda$. From such $\mathcal{A}$, we can construct a PPT adversary $\mathcal{B}$ that breaks the security of f as follows:

1. Let s be a polynomial. Take $1^{s(\lambda)}$ and $f(x)$ with $x \leftarrow \{0,1\}^{s(\lambda)}$ as input.
2. Let $n(\lambda) := s(\lambda)^c$ and sample $u \leftarrow \{0,1\}^{n(\lambda)-s(\lambda)}$.
3. Run $w \leftarrow \mathcal{A}(1^{n(\lambda)}, f(x)\|u)$.
4. Output $w_{1,\dots,s(\lambda)}$.

Then, for any polynomial s, there exist a PPT adversary $\mathcal{B}$, a polynomial p, and an infinite subset $\Lambda \subseteq \Sigma$ such that

$$\Pr[f(x) = f(y) : x \leftarrow \{0,1\}^{s(\lambda)}, y \leftarrow \mathcal{B}(1^{s(\lambda)}, f(x))] \tag{211}$$
$$= \Pr[f(x) = f(w_{1,\dots,s(\lambda)}) : x \leftarrow \{0,1\}^{s(\lambda)}, u \leftarrow \{0,1\}^{n(\lambda)-s(\lambda)}, w \leftarrow \mathcal{A}(1^{n(\lambda)}, f(x)\|u)] \tag{212}$$
$$= \Pr[f'(x\|u) = f'(w_{1,\dots,s(\lambda)}\|u) : x \leftarrow \{0,1\}^{s(\lambda)}, u \leftarrow \{0,1\}^{n(\lambda)-s(\lambda)}, w \leftarrow \mathcal{A}(1^{n(\lambda)}, f'(x\|u))] \tag{213}$$
$$= \Pr[f'(z) = f'(\xi) : z \leftarrow \{0,1\}^{n(\lambda)}, \xi \leftarrow \mathcal{A}(1^{n(\lambda)}, f'(z))] \tag{214}$$
$$> \frac{1}{p(\lambda)} \tag{215}$$

holds for all $\lambda \in \Lambda$. This contradicts the assumption that f is a OWF on Σ. Therefore, f' is a OWF on Σ. $\square$

D Proof of Lemma 2.17

Proof of Lemma 2.17. Let $\Sigma \subseteq \mathbb{N}$ be a subset. Let $f : \{0,1\}^* \to \{0,1\}^*$ be a classically-secure OWF on Σ. Then by the definition of OWFs on Σ, there exists an efficiently computable polynomial n such that for any PPT algorithm $\mathcal{A}$ and a polynomial p, there exists $\lambda^* \in \mathbb{N}$ such that

$$\Pr[f(x) = f(x') : x \leftarrow \{0,1\}^{n(\lambda)}, x' \leftarrow \mathcal{A}(1^{n(\lambda)}, f(x))] \le \frac{1}{p(\lambda)} \tag{216}$$

holds for all $\lambda \ge \lambda^*$ in Σ. From f, we construct a classically-secure OWPuzz $(\mathsf{Samp}, \mathsf{Ver})$ on Σ with 1-correctness and negl-security as follows:

- $\mathsf{Samp}(1^\lambda) \to (\mathsf{ans}, \mathsf{puzz})$:
  1. Take 1^λ as input.
  2. Sample $x \leftarrow \{0,1\}^{n(\lambda)}$ and set $\mathsf{ans} := x$.

[36] A single-tape Turing machine spends $O(|z|^2)$ time to find the maximum i.

3. Compute $f(x) = y$ and set $\mathsf{puzz} := (1^{n(\lambda)}, y)$.
4. Output $(\mathsf{ans}, \mathsf{puzz})$.

- $\mathsf{Ver}(\mathsf{ans}', \mathsf{puzz}) \to \top/\bot$:
  1. Parse $\mathsf{ans}' = x'$ and $\mathsf{puzz} = (1^{n(\lambda)}, y)$.
  2. If $f(x') = y$, output $\top$. Otherwise, output $\bot$.

First we show that $(\mathsf{Samp}, \mathsf{Ver})$ satisfies 1-correctness. In fact, for all $\lambda \in \mathbb{N}$,

$$\Pr[\top \leftarrow \mathsf{Ver}(\mathsf{ans}, \mathsf{puzz}) : (\mathsf{ans}, \mathsf{puzz}) \leftarrow \mathsf{Samp}(1^\lambda)] = \Pr[f(x) = y : x \leftarrow \{0,1\}^{n(\lambda)}, y := f(x)] \tag{217}$$
$$= 1. \tag{218}$$

Next, we prove that $(\mathsf{Samp}, \mathsf{Ver})$ satisfies negl-security on Σ. From the one-wayness of f, for any PPT adversary $\mathcal{A}$, polynomial p, there exists $\lambda^* \in \mathbb{N}$ such that

$$\Pr[\top \leftarrow \mathsf{Ver}(\mathcal{A}(1^\lambda, \mathsf{puzz}), \mathsf{puzz}) : (\mathsf{ans}, \mathsf{puzz}) \leftarrow \mathsf{Samp}(1^\lambda)] \tag{219}$$
$$= \Pr[f(x') = f(x) : x \leftarrow \{0,1\}^{n(\lambda)}, x' \leftarrow \mathcal{A}(1^{n(\lambda)}, f(x))] \tag{220}$$
$$\leq \frac{1}{p(\lambda)}. \tag{221}$$

holds for for all $\lambda \geq \lambda^*$ in Σ. This means that $(\mathsf{Samp}, \mathsf{Ver})$ satisfies negl-security on Σ. ☐

References

[AA11] Scott Aaronson and Alex Arkhipov. The computational complexity of linear optics. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 333–342. ACM Press, June 2011. (Cited on page 5.)

[Aar10] Scott Aaronson. BQP and the polynomial hierarchy. In Leonard J. Schulman, editor, *42nd ACM STOC*, pages 141–150. ACM Press, June 2010. (Cited on page 5.)

[Aar14] Scott Aaronson. The equivalence of sampling and searching. *Theory of Computing Systems*, 55:281–298, 2014. (Cited on page 7, 12, 17, 18, 24.)

[ABK24] Scott Aaronson, Harry Buhrman, and William Kretschmer. A qubit, a coin, and an advice string walk into a relational problem. *ITCS*, 2024. (Cited on page 17, 18.)

[AC17] Scott Aaronson and Lijie Chen. Complexity-theoretic foundations of quantum supremacy experiments. CCC'17: Proceedings of the 32nd Computational Complexity Conference, 2017. (Cited on page 5.)

[ACC$^+$23] Atul Singh Arora, Andrea Coladangelo, Matthew Coudron, Alexandru Gheorghiu, Uttam Singh, and Hendrik Waldner. Quantum depth in the random oracle model. In Barna Saha and Rocco A. Servedio, editors, *55th ACM STOC*, pages 1111–1124. ACM Press, June 2023. (Cited on page 5.)

[AG19] Scott Aaronson and Sam Gunn. On the classical hardness of spoofing linear cross-entropy benchmarking. *arXiv:1910.12085*, 2019. (Cited on page 5.)

[AGGM24] Petia Arabadjieva, Alexandru Gheorghiu, Victor Gitton1, and Tony Metger. Single-round proofs of quantumness from knowledge assumptions. *arXiv:2405.15736*, 2024. (Cited on page 3, 48.)

[AQY22] Prabhanjan Ananth, Luowen Qian, and Henry Yuen. Cryptography from pseudorandom quantum states. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 208–236. Springer, Cham, August 2022. (Cited on page 4.)

[Bar17] Boaz Barak. The complexity of public-key cryptography. Cryptology ePrint Archive, Paper 2017/365, 2017. (Cited on page 49.)

[BCM$^+$21] Zvika Brakerski, Paul Christiano, Urmila Mahadev, Umesh Vazirani, and Thomas Vidick. A cryptographic test of quantumness and certifiable randomness from a single quantum device. *Journal of the ACM*, 68(5):31:1–31:47, 2021. (Cited on page 3, 48.)

[BCQ23] Zvika Brakerski, Ran Canetti, and Luowen Qian. On the computational hardness needed for quantum cryptography. ITCS 2023, 2023. (Cited on page 4.)

[BFNV19] Adam Bouland, Bill Fefferman, Chinmay Nirkhe, and Umesh Vazirani. On the complexity and verification of quantum random circuit sampling. *Nature Physics*, 15:159–163, 2019. (Cited on page 5.)

[BGK18] Sergey Bravyi, David Gosset, and Robert König. Quantum advantage with shallow circuits. *Science*, 2018. (Cited on page 3.)

[BJS11] Michael J. Bremner, Richard Jozsa, and Dan J. Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 467:459–472, 2011. (Cited on page 5.)

[BKL$^+$22] James Bartusek, Yael Tauman Kalai, Alex Lombardi, Fermi Ma, Giulio Malavolta, Vinod Vaikuntanathan, Thomas Vidick, and Lisa Yang. Succinct classical verification of quantum computation. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 195–211. Springer, Cham, August 2022. (Cited on page 48.)

[BMS16] Michael J. Bremner, Ashley Montanaro, and Dan J. Shepherd. Average-case complexity versus approximate simulation of commuting quantum computations. *Physical Review Letters*, 117:080501, 2016. (Cited on page 5.)

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993. (Cited on page 49.)

[CGG24a] Kai-Min Chung, Eli Goldin, and Matthew Gray. On central primitives for quantum cryptography with classical communication. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 215–248. Springer, Cham, August 2024. (Cited on page 4.)

[CGG24b] Kai-Min Chung, Eli Goldin, and Matthew Gray. On central primitives for quantum cryptography with classical communication. Cryptology ePrint Archive, Paper 2024/356, 2024. https://eprint.iacr.org/2024/356. (Cited on page 16.)

[CK11] Imre Csiszár and János Körner. *Information Theory: Coding Theorems for Discrete Memoryless Systems*. Cambridge University Press, 2 edition, 2011. (Cited on page 28.)

[FKM$^+$18] Keisuke Fujii, Hirotada Kobayashi, Tomoyuki Morimae, Harumichi Nishimura, Seiichiro Tani, and Shuhei Tamate. Impossibility of classically simulating one-clean-qubit model with multiplicative error. *Physical Review Letters*, 120:200502, 2018. (Cited on page 5.)

[FR99] Lance Fortnow and John Rogers. Complexity limitations on quantum computation. *Journal of Computer and System Sciences*, 59:240–252, 1999. (Cited on page 3, 6.)

[Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004. (Cited on page 51.)

[HKN$^+$05] Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 96–113. Springer, Berlin, Heidelberg, May 2005. (Cited on page 11, 50.)

[HNO$^+$09] Iftach Haitner, Minh-Huyen Nguyen, Shien Jin Ong, Omer Reingold, and Salil P. Vadhan. Statistically hiding commitments and statistical zero-knowledge arguments from any one-way function. *SIAM J. Comput.*, 39(3):1153–1218, 2009. (Cited on page 17, 49.)

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th FOCS*, pages 230–235. IEEE Computer Society Press, October / November 1989. (Cited on page 3, 10, 14, 15.)

[Imp92] Russell Impagliazzo. Pseudo-random generators for cryptography and for randomized algorithms, 1992. Ph.D. thesis, University of California, Berkeley. (Cited on page 15.)

[JL00] D. W. Juedes and J. H. Lutz. Modeling Time-Bounded Prefix Kolmogorov Complexity. *Theory of Computing Systems*, 33(2):111–123, April 2000. https://doi.org/10.1007/s002249910008. (Cited on page 18.)

[JLS18] Zhengfeng Ji, Yi-Kai Liu, and Fang Song. Pseudorandom quantum states. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 126–152. Springer, Cham, August 2018. (Cited on page 4.)

[KLVY23] Yael Kalai, Alex Lombardi, Vinod Vaikuntanathan, and Lisa Yang. Quantum advantage from any non-local game. In Barna Saha and Rocco A. Servedio, editors, *55th ACM STOC*, pages 1617–1628. ACM Press, June 2023. (Cited on page 3, 48.)

[KMCVY22] Gregory D. Kahanamoku-Meyer, Soonwon Choi, Umesh V. Vazirani, and Norman Y. Yao. Classically verifiable quantum advantage from a computational bell test. *Nature Physics*, 2022. (Cited on page 3, 17, 48.)

[KQST23] William Kretschmer, Luowen Qian, Makrand Sinha, and Avishay Tal. Quantum cryptography in algorithmica. In Barna Saha and Rocco A. Servedio, editors, *55th ACM STOC*, pages 1589–1602. ACM Press, June 2023. (Cited on page 4.)

[Kre21] W. Kretschmer. Quantum pseudorandomness and classical complexity. *TQC 2021*, 2021. (Cited on page 3, 4.)

[KT24a] Dakshita Khurana and Kabir Tomer. Commitments from quantum one-wayness. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *56th ACM STOC*, pages 968–978. ACM Press, June 2024. (Cited on page 3, 4, 15.)

[KT24b] Dakshita Khurana and Kabir Tomer. Founding quantum cryptography on quantum advantage, or, towards cryptography from #P-hardness. Cryptology ePrint Archive, Paper 2024/1490, 2024. (Cited on page 9.)

[Lev85] Leonid A. Levin. One-way functions and pseudorandom generators. In *17th ACM STOC*, pages 363–365. ACM Press, May 1985. (Cited on page 11, 15, 50.)

[LMW24] Alex Lombardi, Fermi Ma, and John Wright. A one-query lower bound for unitary synthesis and breaking quantum cryptography. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *56th ACM STOC*, pages 979–990. ACM Press, June 2024. (Cited on page 4.)

[LV19] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Texts in Computer Science. Springer Cham, 4th edition, 2019. (Cited on page 18, 19, 20.)

[MY22] Tomoyuki Morimae and Takashi Yamakawa. Quantum commitments and signatures without one-way functions. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 269–295. Springer, Cham, August 2022. (Cited on page 3, 4, 6.)

[MY23] Tomoyuki Morimae and Takashi Yamakawa. Proofs of quantumness from trapdoor permutations. In Yael Tauman Kalai, editor, *ITCS 2023*, volume 251, pages 87:1–87:14. LIPIcs, January 2023. (Cited on page 3, 48.)

[MY24] Tomoyuki Morimae and Takashi Yamakawa. Quantum advantage from one-way functions. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part V*, volume 14924 of *LNCS*, pages 359–392. Springer, Cham, August 2024. (Cited on page 3, 5, 6, 12, 16, 17.)

[MYY24] Tomoyuki Morimae, Shogo Yamada, and Takashi Yamakawa. Quantum unpredictability. Cryptology ePrint Archive, Paper 2024/701, 2024. (Cited on page 4.)

[Ost91] Rafail Ostrovsky. One-way functions, hard on average problems, and statistical zero-knowledge proofs. *Computational Complexity Conference*, 1991. (Cited on page 9, 10, 11.)

[OW93] Rafail Ostrovsky and Avi Wigderson. One-way fuctions are essential for non-trivial zero-knowledge. In *Second Israel Symposium on Theory of Computing Systems, ISTCS 1993, Natanya, Israel, June 7-9, 1993, Proceedings*, pages 3–17. IEEE Computer Society, 1993. (Cited on page 48.)

[PW09] Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 403–418. Springer, Berlin, Heidelberg, March 2009. (Cited on page 45, 46.)

[Rao08] Anup Rao. Parallel repetition in projection games and a concentration bound. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 1–10. ACM Press, May 2008. (Cited on page 28.)

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994. (Cited on page 3, 5, 49.)

[TD04] B. M. Terhal and D. P. DiVincenzo. Adaptive quantum computation, constant-depth circuits and arthur-merlin games. *Quant. Inf. Comput.*, 4(2):134–145, 2004. (Cited on page 5.)

[Vad06] Salil P. Vadhan. An unconditional study of computational zero knowledge. *SIAM Journal on Computing*, 2006. (Cited on page 8, 48.)

[YZ24] Takashi Yamakawa and Mark Zhandry. Verifiable quantum advantage without structure. *J. ACM*, 71(3), jun 2024. (Cited on page 3, 5, 48, 49.)