Weak Zero-Knowledge and One-Way Functions

Rohit Chatterjee
Department of Computer Science, National University of Singapore, Singapore

Yunqi Li
Department of Computer Science, National University of Singapore, Singapore

Prashant Nalini Vasudevan
Department of Computer Science, National University of Singapore, Singapore

Abstract

We study the implications of the existence of weak Zero-Knowledge (ZK) protocols for worst-case hard languages. These are protocols that have completeness, soundness, and zero-knowledge errors (denoted ϵ_c, ϵ_s, and ϵ_z, respectively) that might not be negligible. Under the assumption that there are worst-case hard languages in NP, we show the following:

1. If all languages in NP have NIZK proofs or arguments satisfying $\epsilon_c + \epsilon_s + \epsilon_z < 1$, then One-Way Functions (OWFs) exist. This covers all possible non-trivial values for these error rates. It additionally implies that if all languages in NP have such NIZK proofs and ϵ_c is negligible, then they also have NIZK proofs where all errors are negligible. Previously, these results were known under the more restrictive condition $\epsilon_c + \sqrt{\epsilon_s} + \epsilon_z < 1$ [Chakraborty et al., CRYPTO 2025].
2. If all languages in NP have k-round public-coin ZK proofs or arguments satisfying $\epsilon_c + \epsilon_s + (2k-1) \cdot \epsilon_z < 1$, then OWFs exist.
3. If, for some constant k, all languages in NP have k-round public-coin ZK proofs or arguments satisfying $\epsilon_c + \epsilon_s + k \cdot \epsilon_z < 1$, then infinitely-often OWFs exist.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases One-way functions, zero-knowledge

Digital Object Identifier 10.4230/LIPIcs.ITC.2026.5

Related Version *Full Version*: https://eprint.iacr.org/2026/299

Funding This work was supported by the National Research Foundation, Singapore, under its NRF Fellowship programme, award no. NRF-NRFF14-2022-0010.

Acknowledgements We thank the anonymous reviewers for their comments.

Declaration on GenAI/LLM use Commercial AI tools (ChatGPT, Claude) were used as typing assistants for grammar and basic editing, and for mild assistance with LaTeX formatting.

1 Introduction

The notion of Zero-Knowledge (ZK) protocols is a vital part of modern cryptography. These are interactive proof systems in which a *prover* proves to a *verifier* the validity of a given statement, with the additional guarantee that even a possibly cheating verifier cannot obtain any secrets that might have been used in the proof. More precisely, such protocols guarantee *soundness*, i.e. cheating provers cannot prove false statements, and *zero knowledge*, i.e. a cheating verifier cannot glean anything beyond the validity of the statement over the course of the protocol.

Hardness from Zero-Knowledge

One natural question arising in the context of relating zero-knowledge to other cryptographic notions is that of which other cryptographic primitives are implied by it. This was first studied in the work of Ostrovsky [11], who showed that a statistical ZK proof system for any average-case hard language implies the existence of One-Way Functions (OWFs). This was later generalized by Ostrovsky and Wigderson [12] to computational ZK proofs, and they showed in addition that such proofs for even a worst-case hard language implies a weaker form of OWFs called auxiliary-input OWFs.

Building on these, the recent work of Hirahara and Nanashima [7] showed that if all languages in NP have computational ZK proofs (or even arguments, which only have computational soundness) and NP is hard in the worst-case, then OWFs exist.

Weak Zero-Knowledge

The above results all work with proof systems where the completeness, soundness, and zero-knowledge errors are guaranteed to be negligible. There are, however, a number of natural and useful ZK protocols, such as the commonly taught protocols for 3-Coloring and Graph Non-isomorphism, that do not natively have negligible error rates. Such *Weak ZK* protocols may have completeness error ϵ_c, soundness error of ϵ_s, and zero-knowledge error ϵ_z, that may be as large as a constant number. Along the same lines as above, it is natural and important to study the power of such protocols as well.

Amplifying Weak NIZKs

The work of Goyal et al. [6] was a first step in this direction, investigating the power of weak *Non-Interactive ZK* (NIZK) arguments. NIZKs consider a non-interactive setting where the prover and verifier have access to a common random string, and the protocol only involves a single prover message, following which the verifier decides whether to accept or reject. They showed that weak NIZKs with negligible ε_c satisfying $\epsilon_s + \epsilon_z < 1 - \delta$ for any non-zero constant δ can be amplified to a standard NIZK argument with negligible errors if we additionally have access to a sub-exponentially secure Public Key Encryption (PKE) scheme.

Bitansky and Geier [2] improved this result to show that standard PKE suffices for this amplification. They also showed that if the NIZK system is a proof (i.e., has statistical soundness), then amplification is possible assuming only OWFs. Applebaum and Kachlon [1] improved on this to allow for δ to be as small as an inverse-polynomial function.

Hardness from Weak ZK

Seeking to reduce the assumptions needed for such amplification, Chakraborty et al. [4] showed that in certain settings, weak NIZKs can be used to derive OWFs. Specifically, they show that if all languages in NP have NIZK arguments satisfying $\epsilon_c + \sqrt{\epsilon_s} + \epsilon_z < 1$, then OWFs exist under just the worst-case assumption that $\mathsf{NP} \not\subseteq \mathsf{ioP/poly}$[1]. They then combined this with the amplification results of [2, 1] to show that under the additional hypotheses that these are NIZK proofs with negligible ε_c, they could get NIZK proofs with negligible errors for all of NP.

[1] That is, there are no polynomial-size circuit families for NP, even if they are only required to be correct infinitely often.

While this result helps characterize the hardness of a broad class of weak NIZKs, it is still somewhat unsatisfactory as it does not cover *all* possible non-trivial weak NIZK parameters. As noted in [4], the setting $\epsilon_c + \epsilon_s + \epsilon_z \geq 1$ is not meaningful for NIZK protocols. Thus, to complete the picture, what is required is to understand the implications of any weak NIZK protocol satisfying $\epsilon_c + \epsilon_s + \epsilon_z < 1$.

Another important question that has not been studied in this recent line of work is that of the complexity of weak interactive ZK protocols. The earlier implications of ZK shown in [12, 7] work for interactive ZK protocols with negligible errors. But the extent of their validity for weak ZK protocols was not understood.

Our Results

Our first result addresses the first question above, extending the construction of OWFs from NIZKs to the most general parameters, under the same assumptions as in prior work.

▶ **Theorem 1** (Informally, Theorem 43). *If* NP $\not\subseteq$ ioP/poly*, and every language in* NP *has an* $(\epsilon_c, \epsilon_s, \epsilon_z)$*-NIZK proof (or argument) with* $\epsilon_c + \epsilon_s + \epsilon_z < 1$*, then one-way functions exist.*

Similar to [4], we can in turn use the amplification results of [2, 1] to obtain amplification of NIZK proofs with near-perfect completeness, from the most general setting of the remaining errors.

▶ **Corollary 2** (NIZK Amplification). *If* NP $\not\subseteq$ ioP/poly*, and every language in* NP *has an* $(\epsilon_c, \epsilon_s, \epsilon_z)$*-NIZK proof with* $\epsilon_c + \epsilon_s + \epsilon_z < 1$ *and negligible* ϵ_c*, then every language in* NP *has a NIZK proof with negligible errors. Here, the soundness of the NIZK proofs is required to be adaptive.*[2]

We also address the second question raised above for *public-coin* protocols, where the verifier's messages consist solely of uniform random bits.

▶ **Theorem 3** (Informally, Theorem 44). *If* NP $\not\subseteq$ ioP/poly*, and every language in* NP *has a* t*-message* $(\epsilon_c, \epsilon_s, \epsilon_z)$*-public-coin ZK proof (or argument) with* $\epsilon_c + \epsilon_s + (t-1) \cdot \epsilon_z < 1$*, then one-way functions exist.*

In the above case, the techniques of [12] alone would have resulted in the condition being $\epsilon_c + \epsilon_s + t \cdot \epsilon_z < 1$ instead. For constant-round protocols, we improve this condition much more significantly, though in this case we only obtain infinitely-often one-way functions. Below, a *round* refers to one pair of messages in the protocol – one from the verifier and its response from the prover.

▶ **Theorem 4** (Informally, Theorem 45). *If* NP $\not\subseteq$ P/poly*, and for some constant* k*, every language in* NP *has a* k*-round* $(\epsilon_c, \epsilon_s, \epsilon_z)$*-public-coin ZK proof (or argument) with* $\epsilon_c + \epsilon_s + k \cdot \epsilon_z < 1$*, then infinitely-often one-way functions exist.*

Our results apply to protocols with computational (weak) zero-knowledge and computational (weak) soundness (i.e., arguments), and thus capture the most general class of such protocols.

[2] In the rest of the paper, we exclusively use the weaker non-adaptive definition of soundness for NIZK protocols. This only strengthens our results, as we use NIZKs to construct other things. Here, the stronger adaptive notion of soundness is needed. See, e.g., [2] for the definition of this notion.

Concurrent Work

In concurrent and independent work, Chakraborty et al. [5] obtain results similar to ours. They prove Theorem 1 and a stronger version of Theorem 4, but they do not show Theorem 3. Most remarkably, they are able to relax the condition on the errors for constant-round public-coin ZK protocols required in Theorem 4 to $\epsilon_c + \epsilon_s + \epsilon_z < 1$, which covers all non-trivial protocols. The techniques in the two papers are considerably different.

1.1 Technical Overview

In this section, we provide a high-level overview of the main ideas and techniques behind our results. We start by reviewing the construction of [12] with their analysis for non-interactive ZKs. We then introduce our improved construction and outline the ideas that enable improvement. Further, we find that our approach naturally generalizes to a broader setting – specifically that of public-coin ZK protocols, which we will discuss later.

Throughout, we use the notation $\mathcal{U}$ to denote the uniform distribution over strings whose length will be clear from the context.

Non-interactive zero-knowledge

We first consider non-interactive ZK (NIZK) arguments. A NIZK argument for a language $\mathcal{L}$ allows a prover to produce a single proof π to certify that an input $x \in \mathcal{L}$ while preserving zero-knowledge. Specifically, given a uniformly random string r, also known as the *common reference string*, a polynomial-time prover holding with a valid NP witness w for x computes a proof $\pi \leftarrow \mathsf{P}(x, w; r)$; upon receiving the prover's message, the verifier computes $a \leftarrow \mathsf{V}(x; r, \pi)$ to decide whether $x \in \mathcal{L}$.

The protocol is required to satisfy completeness and computational soundness, where the errors are correspondingly denoted by ϵ_c and ϵ_s. Besides these, it also satisfies computational zero-knowledge. In particular, there is a polynomial-time simulator Sim that on input x generates a distribution (r, π) such that no polynomial-time distinguisher can distinguish between this and the (r, π) from the actual protocol with advantage greater than the zero-knowledge error ϵ_z. For simplicity, we assume that the NIZK protocol has perfect completeness ($\epsilon_c = 0$) and we have a deterministic verification algorithm V.

The Ostrovsky-Wigderson approach

The key observation underlying [12] is that an inverter for the NIZK simulator can be used to construct a distinguisher to decide the language, contradicting its hardness.

Suppose the language $\mathcal{L}$ that has the NIZK argument $(\mathsf{P}, \mathsf{V}, \mathsf{Sim})$ is worst-case hard. Let Sim be the simulator that runs on the input x and randomness ρ, and outputs the common reference string r and a proof π. The candidate hard-to-invert function f_x is defined to be

$$\begin{aligned} f_x(\rho): \quad & (r, \pi) \leftarrow \mathsf{Sim}(x; \rho) \\ & \text{output } r \end{aligned}$$

Since the randomness is treated explicitly as part of the input, the above construction is deterministic and therefore the function is well defined.

Towards a contradiction, assume that there are no auxiliary-input one-way functions. In fact, suppose that there is an adversary $\mathcal{A}$ that perfectly inverts f_x distributionally – that is, given y, $\mathcal{A}$ samples a uniformly random pre-image $f_x^{-1}(y)$. It is known that distributional

OWFs imply OWFs [8], so the only loss of generality here is the assumption that the inverter is perfect, but this is not difficult to remove at the cost of an inverse polynomial loss in parameters.

We have that the joint distributions

$$(\mathcal{A}(f_x(\mathcal{U})), f_x(\mathcal{U})) \approx (\mathcal{U}, f_x(\mathcal{U}))$$

are close, where the left-hand side represents the distribution of first computing f_x on random inputs and applying $\mathcal{A}$, while the right-hand side denotes the distribution of sampling a random input r and then outputting $(r, f_x(r))$.

The algorithm $\mathcal{D}$ that decides $\mathcal{L}$ is as follows. Given input x, it invokes $\mathcal{A}$ to decide whether $x \in \mathcal{L}$ or not: it samples $r \leftarrow \mathcal{U}$, runs $\hat{\rho} \leftarrow \mathcal{A}(r)$, then computes $(\hat{r}, \hat{\pi}) \leftarrow \mathsf{Sim}(x; \hat{\rho})$, and accepts if and only if $\mathsf{V}(x; r, \hat{\pi}) = 1$. Note that under our assumptions, we will always have $r = \hat{r}$. We now analyze the performance of $\mathcal{D}$ in the two possible cases.

1. When $x \in \mathcal{L}$, completeness implies that when (r, π) is generated following the protocol, $\mathsf{V}(x; r, \pi) = 1$ holds with probability 1. Zero-knowledge guarantees that the probability that $\mathsf{V}(x; r, \pi) = 1$ when $(r, \pi) \leftarrow \mathsf{Sim}(x; \mathcal{U})$ is at least $1 - \epsilon_z$.
   In the algorithm $\mathcal{D}(x)$, we run V on (r, π) generated from $\mathsf{Sim}(x; \mathcal{A}(\mathcal{U}))$. When the inverter $\mathcal{A}$ is run on r sampled from $\mathsf{Sim}(x; \mathcal{U})$, the resulting inverse is uniformly random (due to the definition of f_x). Again by zero-knowledge, the uniform distribution of r is ϵ_z-indistinguishable from the distribution of r sampled by $\mathsf{Sim}(x; \mathcal{U})$. So the distribution of $\mathcal{A}(\mathcal{U})$ is also ϵ_z-indistinguishable from uniform. This implies that the distribution of $\mathsf{Sim}(x; \mathcal{A}(\mathcal{U}))$ is ϵ_z-indistinguishable from $\mathsf{Sim}(x; \mathcal{U})$. Altogether, we have

   $$\Pr[\mathcal{D}(x) = 1] \geq 1 - 2\epsilon_z.$$

2. When $x \notin \mathcal{L}$, soundness ensures that for any efficient method of generating π for random r, the probability that V accepts is bounded by ϵ_s, and so

   $$\Pr[\mathcal{D}(x) = 1] \leq \epsilon_s.$$

Consequently, if $\epsilon_s < 1 - 2\epsilon_z$, the inverter $\mathcal{A}$ can be used to determine whether x is in $\mathcal{L}$. If such an inverter works for every x, then we can decide the language, contradicting its hardness. Thus, the family of functions $\{f_x\}$ must be a family of auxiliary-input one-way functions.

Improving the condition on errors

More recently, such analysis has seen further progress. In particular, this bound was improved by [4], where it was shown that $\sqrt{\epsilon_s} + \epsilon_z < 1$ suffices. This was shown with sophisticated arguments using a one-sided version of universal approximation, where the inverter is used to estimate the probabilities of certain outputs.

Our starting point is the observation that with rather simple but careful arguments, it is feasible to relax this bound to $\epsilon_s + \epsilon_z < 1$, which is the most general it can be. We avoid paying for the zero-knowledge error twice by involving the verification procedure inside the candidate one-way function. Our one-way function is as follows

$f_x(\rho)$: $(r, \pi) \leftarrow \mathsf{Sim}(x; \rho)$
$a \leftarrow \mathsf{V}(x; r, \pi)$
output (r, a)

Suppose again that there is a near-perfect polynomial-time inverter $\mathcal{A}$ for f_x (it need not be a distributional inverter). That is,

$$\Pr_{(r,a)\leftarrow f_x(\mathcal{U})}[f_x(\mathcal{A}(r,a)) = (r,a)] \approx 1,$$

The algorithm $\mathcal{D}$ for $\mathcal{L}$ can be constructed as follows. Given input x, sample r uniformly at random, compute $\hat{\rho} \leftarrow \mathcal{A}(r,1)$, and accept if and only if this is a valid pre-image of $(r,1)$ under f_x – that is, iff $f_x(\hat{\rho}) = (r,1)$.

1. For $x \in \mathcal{L}$, consider the following procedure $g(r,\pi)$: it computes $a \leftarrow \mathsf{V}(x;r,\pi)$ and outputs (r,a). When (r,π) is sampled from the simulator $\mathsf{Sim}(x;\mathcal{U})$, the distribution of $g(r,\pi)$ is equivalent to $f_x(\mathcal{U})$. When (r,π) follows the protocol view, the distribution of $g(r,\pi)$ is the same as $(r,1)$ in $\mathcal{D}$. So by ZK and the data processing inequality, these distributions are ϵ_z-indistinguishable.
   Further, when (r,π) is sampled from the simulator, the inverter $\mathcal{A}$ will almost always find a valid pre-image of (r,a). Therefore, given $(r,1)$ as in $\mathcal{D}$, it finds a valid pre-image with overall probability at least $\approx 1-\epsilon_z$.

   $$\Pr[\mathcal{D}(x) = 1] = \Pr_{r\leftarrow\mathcal{U}}[f_x(\mathcal{A}(r,1)) = (r,1)] \geq 1-\epsilon_z.$$

2. For $x \notin \mathcal{L}$, whenever a valid pre-image $\hat{\rho}$ for $(r,1)$ is found, the corresponding $(r,\pi) \leftarrow \mathsf{Sim}(x;\hat{\rho})$ is accepted by V. As both $\mathcal{A}$ and Sim are efficient algorithms, soundness guarantees that a valid inverse cannot be found with probability more than ϵ_s. So

   $$\Pr[\mathcal{D}(x) = 1] \leq \epsilon_s.$$

Therefore, following the same remaining arguments as earlier, $\epsilon_s + \epsilon_z < 1$ suffices to imply the existence of auxiliary-input one-way functions.

The key idea behind our improvement is that we implicitly utilize the verification V while restricting the inverter to output a good pre-image corresponding to a valid proof. Thus as long as the inverter succeeds, there is no need to perform an additional explicit verification. This saves us the extra zero-knowledge error penalty.

The distinguisher: an alternate view

Our analysis reveals that in both the [12] construction and our new one, these distinguishers can be regarded as providing efficient simulations of the protocol, where the original honest prover algorithm is replaced with an efficient algorithm without the witness and V serves to certify correctness. For example, the Ostrovsky-Wigderson distinguisher admits an equivalent formulation as a protocol between a prover $\tilde{\mathsf{P}}$ and the verifier V. The prover $\tilde{\mathsf{P}}$ performs: on randomness r, find the message $\tilde{\pi}$ corresponding to the simulator randomness $\tilde{\rho}$, where $\tilde{\rho}$ is found efficiently by the inverter. Our construction yields a similar prover as well. The only difference is the modification to the inverter since the correctness of a pre-image ensures the validity of corresponding proof.

Multiple-round public-coin zero-knowledge

Our approach naturally extends to the setting of multiple-round public-coin zero-knowledge protocols. Such a protocol allows interaction between the prover and the verifier, where all of the verifier's communication consists of uniform random bits. In particular, all its randomness is public and available to the prover.

For the sake of exposition, we assume below perfect completeness with soundness error ϵ_s and zero-knowledge error ϵ_z. By adapting our approach for the NIZK case, we obtain that the existence a k-round public-coin ZK protocol for a hard language $\mathcal{L}$ with parameters $\epsilon_s + (2k-1)\epsilon_z < 1$ implies the existence of one-way functions. We use the term *round* to denote one interaction where the verifier sends a random challenge and the prover responds with a proof message.

Moreover, the bound can be further improved to $\epsilon_s + k \cdot \epsilon_z < 1$ using a more sophisticated argument when k is only a constant.

For illustration, we focus here on the simplest setting in which the language $\mathcal{L}$ admits a two-round public-coin ZK protocol. Here both parties share a common input x while the honest prover additionally holds a witness w. In the first round, the verifier first samples a random string r_1 and the prover replies with a message $\pi_1 \leftarrow \mathsf{P}_1(x, w; r_1)$. In the second round, the verifier sends another random string r_2 and the prover responds with $\pi_2 \leftarrow \mathsf{P}(x, w; r_1, \pi_1, r_2)$. Finally, the verifier checks the transcript by computing $0/1 \leftarrow \mathsf{V}(x; r_1, \pi_1, r_2, \pi_2)$, where 1 denotes acceptance.

The zero-knowledge condition implies the existence of a simulator Sim that on input x and randomness ρ, outputs a transcript (r_1, π_1, r_2, π_2); for any $x \in \mathcal{L}$ and a valid witness w, $\mathsf{Sim}(x)$ is ϵ_z-indistinguishable from the protocol view.

In the following, we will follow a recursive approach to construct our candidate one-way functions. More precisely, we begin by defining a function, and any algorithm that breaks the one-wayness of this function will be incorporated into the construction of a second function. If the second one is not one-way either, then the resulting inverters can be combined to decide the language. Accordingly, we first define a function $f_{2,x}$ as follows:

$$\begin{aligned} f_{2,x}(\rho): \quad & (r_1, \pi_1, r_2, \pi_2) \leftarrow \mathsf{Sim}(x; \rho) \\ & a \leftarrow \mathsf{V}(x; r_1, \pi_1, r_2, \pi_2) \\ & \text{output } (r_1, \pi_1, r_2, a) \end{aligned}$$

Assume that there is a poly-time algorithm $\mathcal{A}_2$ that inverts $f_{2,x}$

$$\Pr_{(r_1,\pi_1,r_2,a)\leftarrow f_{2,x}(\mathcal{U})} [f_{2,x}(\mathcal{A}_2(x; r_1, \pi_1, r_2, a)) = (r_1, \pi_1, r_2, a)] \approx 1. \tag{1}$$

As in the NIZK case, the deciding procedure is essentially equivalent to efficiently simulating the protocol by constructing an efficient prover $\tilde{\mathsf{P}}$: this queries the inverter $\mathcal{A}$ on input $(r, 1)$ and generates the prover's message π accordingly. For the second round of the protocol, we define the strategy $\tilde{\mathsf{P}}_2$ similarly.

$$\begin{aligned} \tilde{\mathsf{P}}_2(x; r_1, \pi_1, r_2): \quad & \tilde{\rho} \leftarrow \mathcal{A}_2(x; r_1, \pi_1, r_2, 1) \\ & (\tilde{r}_1, \tilde{\pi}_1, \tilde{r}_2, \tilde{\pi}_2) \leftarrow \mathsf{Sim}(x; \tilde{\rho}) \\ & \text{output } \tilde{\pi}_2 \end{aligned}$$

We measure the performance of $\tilde{\mathsf{P}}_2$ formally by the following quantity:

$$\mathsf{Succ}_3(x; r_1, \pi_1, r_2) = 1[f_{2,x}(\mathcal{A}_2(x; r_1, \pi_1, r_2, 1)) = (r_1, \pi_1, r_2, 1)]$$

where $\mathcal{A}_2$ is assumed to be deterministic for simplicity. The value of Succ_3 represents the probability that $\mathcal{A}_2$ finds a valid pre-image with respect to $f_{2,x}$. This provides a lower bound for the acceptance probability of the protocol conditioned on the first 3 messages being (r_1, π_1, r_2), since $\tilde{\pi}_2$ is valid as long as $\mathcal{A}_2$ finds a valid pre-image of $f_{2,x}$.

$$f_{2,x}(\tilde{\rho}) = (r_1, \pi_1, r_2, 1) \Rightarrow \mathsf{V}(x; r_1, \pi_1, r_2, \tilde{\pi}_2) = 1.$$

Since the verifier samples r_2 uniformly, we can equivalently lower bound the success probability of this prover by the following expression (where the first two messages are (r_1, π_1) and $\tilde{\mathsf{P}}_2$ is the second round strategy of the prover):

$$\mathsf{Succ}_2(x; r_1, \pi_1) = \mathop{\mathbb{E}}_{r_2 \leftarrow \mathcal{U}} [\mathsf{Succ}_3(x; r_1, \pi_1, r_2)]$$

Now consider the protocol $\langle \mathsf{P}, \mathsf{V} \rangle$ that replaces the prover algorithm in the second round with $\tilde{\mathsf{P}}_2$. We can establish an upper bound for the completeness error of this modified protocol.

Denote by $D_{2,x}$ the distribution that samples (r_1, π_1, r_2, π_2) from the protocol $\langle \mathsf{P}, \mathsf{V} \rangle(x, w)$, sets $a \leftarrow \mathsf{V}(x; r_1, \pi_1, r_2, \pi_2)$ and outputs (r_1, π_1, r_2, a). By the perfect completeness of $\langle \mathsf{P}_w, \mathsf{V} \rangle$, $a = 1$ always holds. It follows by the data processing inequality that, for $x \in \mathcal{L}$

$$\Delta(f_{2,x}(\mathcal{U}); D_{2,x}) \leq \Delta(\mathsf{Sim}(x); \langle \mathsf{P}, \mathsf{V} \rangle(x)) \leq \epsilon_z.$$

Combining above with our assumption (1), the acceptance probability is at least

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\substack{r_1 \leftarrow \mathcal{U} \\ \pi_1 \leftarrow \mathsf{P}_1(x,w;r_1)}} [\mathsf{Succ}_2(x; r_1, \pi_1)] \\
&= \Pr_{(r_1, \pi_1, r_2, a) \leftarrow D_{2,x}} [f_{2,x}(\mathcal{A}_2(x; r_1, \pi_1, r_2, a)) = (r_1, \pi_1, r_2, a)] \\
&\geq \Pr_{(r_1, \pi_1, r_2, a) \leftarrow f_{2,x}(\mathcal{U})} [f_{2,x}(\mathcal{A}_2(x; r_1, \pi_1, r_2, a)) = (r_1, \pi_1, r_2, a)] - \epsilon_z \\
&\geq 1 - \epsilon_z. \qquad (2)
\end{aligned}$$

The above implies that when the prover applies $\mathsf{P}_1(x, w)$ and $\tilde{\mathsf{P}}_2(x)$ in each round, respectively, the acceptance probability is at least $1 - \epsilon_z$ when $x \in \mathcal{L}$. However, because the witness for P_1 is unavailable to our deciding strategy, we need a different polynomial-time algorithm as a replacement to obtain an efficient simulation for the protocol.

We now turn to building our first-round strategy. A natural attempt is to define a new function $f_{1,x}(\rho)$ analogously to $f_{2,x}$ and apply the inverter to produce the message π_1, where $f_{1,x}(\rho)$ can be defined as follows: sample $(r_1, \pi_1, r_2, \pi_2) \leftarrow \mathsf{Sim}(x; \rho)$, $a \leftarrow \mathsf{V}(x; r_1, \pi_1, r_2, \pi_2)$, and output (r_1, a) (or only output r_1). However, our analysis of the resulting prover algorithms shows that the best achievable result is $\epsilon_s + 3\epsilon_z < 1$, which matches $\epsilon_s + (2k-1)\epsilon_z < 1$ for $k = 2$. We refer the reader to Section 3.2 for the details.

Instead, we introduce a new construction of $f_{1,x}$ that leads to an improved bound. After fixing $\tilde{\mathsf{P}}_2$, our objective is to find an efficient way $\tilde{\mathsf{P}}_1$ to generate π_1 that maximizes the value $\mathsf{Succ}_2(x; r_1, \pi_1)$ optimally for $x \in \mathcal{L}$. The value of Succ_2 can be efficiently approximated up to any arbitrary inverse-polynomial error due to our assumption on $\mathcal{A}_2$ and standard concentration arguments. For convenience, we ignore the accuracy issue here and assume that we are able to compute Succ_2 precisely in polynomial-time. The key is to instead have the new function compute Succ_2 for the relevant partial transcript. More precisely, we define:

$$\begin{aligned}
f_{1,x}(\rho): \quad & (r_1, \pi_1, r_2, \pi_2) \leftarrow \mathsf{Sim}(x; \rho) \\
& \text{output } (r_1, \mathsf{Succ}_2(x; r_1, \pi_1))
\end{aligned}$$

Suppose now that there is an efficient algorithm $\mathcal{A}_1$

$$\Pr_{(r_1, a) \leftarrow f_{1,x}(\mathcal{U})} [f_{1,x}(\mathcal{A}_1(x; r_1, a)) = (r_1, a)] \approx 1.$$

We are now ready to formally specify $\tilde{\mathsf{P}}_1$, which works as follows:

$$\begin{aligned}\tilde{\mathsf{P}}_1(x;r_1):\quad &\tilde{a}\leftarrow \arg\max_a \{a\cdot \mathbf{1}[f_{1,x}(\mathcal{A}_1(x;r_1,a))=(r_1,a)\}]\\ &\tilde{\rho}\leftarrow \mathcal{A}_1(x;r_1,\tilde{a})\\ &\tilde{\pi}_1\leftarrow \mathsf{Sim}(\tilde{\rho})\\ &\text{output } \tilde{\pi}_1\end{aligned}$$

where we take the support size of a to be polynomial, enabling us to efficiently iterate over all possible values and find the maximum $\tilde{a}$. In fact, in general one can efficiently find an approximate maximum value instead, which suffices.

On input $(x;r_1)$, the prover $\tilde{\mathsf{P}}_1$ goes through all possible values of a, and selects the highest one on which $\mathcal{A}_1$ inverts $f_{1,x}$ successfully. Note that when $\mathcal{A}_1$ finds a correct pre-image of (r_1,a), it implies that there exists a consistent π_1 with value $\mathsf{Succ}_2(x;r_1,\pi_1)=a$; so $\tilde{\mathsf{P}}_1$ takes the maximal a and outputs its associated prover message.

With the construction of $\tilde{\mathsf{P}}_1$ and $\tilde{\mathsf{P}}_2$, it remains to analyze the performance of the resulting protocol $\langle\tilde{\mathsf{P}},\mathsf{V}\rangle$. More specifically, we are interested in

$$\mathop{\mathbb{E}}_{\substack{r_1\leftarrow\mathcal{U},\pi_1\leftarrow\tilde{\mathsf{P}}_1(x;r_1)\\ r_2\leftarrow\mathcal{U},\pi_2\leftarrow\tilde{\mathsf{P}}_2(x;r_1,\pi_1,r_2)}}[\mathsf{V}(x;r_1,\pi_1,r_2,\pi_2)] \tag{3}$$

Recall that

$$(3)\geq \mathop{\mathbb{E}}_{\substack{r_1\leftarrow\mathcal{U}\\ \pi_1\leftarrow\tilde{\mathsf{P}}_1(x;r_1)}}[\mathsf{Succ}_2(x;r_1,\pi_1)] = \mathop{\mathbb{E}}_{r_1\leftarrow\mathcal{U}}\left[\max_a\{a\cdot\mathbf{1}[f_{1,x}(\mathcal{A}_1(x;r_1,a))=(r_1,a)]\}\right] \tag{4}$$

where the last equality is ensured since by definition of $\tilde{\mathsf{P}}_1(x;r_1)$, the Succ_2 estimate a associated with π_1 is the largest such value with respect to which $\mathcal{A}_1$ can succeed, i.e.,

$$\mathsf{Succ}_2(x;r_1,\pi_1)=\max_a\{a\cdot\mathbf{1}[f_{1,x}(\mathcal{A}_1(x;r_1,a))=(r_1,a)]\}.$$

Now observe that, for any r_1 and any particular a^*

$$\max_a\{a\cdot\mathbf{1}[f_{1,x}(\mathcal{A}_1(x;r_1,a))=(r_1,a)]\}\geq a^*\cdot\mathbf{1}[f_{1,x}(\mathcal{A}_1(x;r_1,a^*))=(r_1,a^*)]. \tag{5}$$

Let a^* above be sampled as $a^*\leftarrow\mathsf{Succ}_2(x;r_1,\pi_1)$ where $\pi_1\leftarrow\mathsf{P}_1(x,w;r_1)$. We now investigate the new completeness error to obtain a lower bound for (4) when $x\in\mathcal{L}$. Note that the value of Succ_2 falls in the range $[0,1]$, thus we have

$$\begin{aligned}(4)&\geq \mathop{\mathbb{E}}_{\substack{r_1\leftarrow\mathcal{U},\pi_1\leftarrow\mathsf{P}_1(x,w;r_1)\\ a\leftarrow\mathsf{Succ}_2(x;r_1,\pi_1)}}[a\cdot\mathbf{1}[f_{1,x}(\mathcal{A}_1(x;r_1,a))=(r_1,a)]]\\ &\geq \mathop{\mathbb{E}}_{\substack{r_1\leftarrow\mathcal{U},\pi_1\leftarrow\mathsf{P}_1(x,w;r_1)\\ a\leftarrow\mathsf{Succ}_2(x;r_1,\pi_1)}}[a]-\mathop{\mathbb{E}}_{\substack{r_1\leftarrow\mathcal{U},\pi_1\leftarrow\mathsf{P}_1(x,w;r_1)\\ a\leftarrow\mathsf{Succ}_2(x;r_1,\pi_1)}}[\mathbf{1}[f_{1,x}(\mathcal{A}_1(x;r_1,a))\neq(r_1,a)]]\\ &\geq 1-\epsilon_z-\Pr_{\substack{r_1\leftarrow\mathcal{U},\pi_1\leftarrow\mathsf{P}_1(x,w;r_1)\\ a\leftarrow\mathsf{Succ}_2(x;r_1,\pi_1)}}[f_{1,x}(\mathcal{A}_1(x;r_1,a))\neq(r_1,a)],\end{aligned} \tag{6}$$

where the first inequality is obtained by the observation (5) and the last inequality holds by (2). Now define the distribution $D_{1,x}$: sample $r_1\leftarrow\mathcal{U}$ and $\pi_1\leftarrow\mathsf{P}_1(x,w;r_1)$, and finally output $(r_1,\mathsf{Succ}_2(x;r_1,\pi_1))$.

For $x\in\mathcal{L}$, by the data processing inequality and zero-knowledge condition

$$\Delta(f_{1,x}(\mathcal{U});D_{1,x})\leq\Delta(\mathsf{Sim}(x);\langle\mathsf{P}_w,\mathsf{V}\rangle(x))\leq\epsilon_z.$$

Since we assume that $\mathcal{A}_1$ inverts the function almost perfectly, we get

$$\begin{aligned}
&\Pr_{\substack{r_1\leftarrow\mathcal{U},\pi_1\leftarrow\mathsf{P}_1(x,w;r_1)\\ a\leftarrow\mathsf{Succ}_2(x;r_1,\pi_1)}}[f_{1,x}(\mathcal{A}_1(x;r_1,a))\neq(r_1,a)]\\
&\leq \Pr_{(r_1,a)\leftarrow f_{1,x}(\mathcal{U})}[f_{1,x}(\mathcal{A}_1(x;r_1,a))\neq(r_1,a)]+\epsilon_z\\
&\leq \epsilon_z. \qquad (7)
\end{aligned}$$

Therefore, by combining (3), (6) and (7), we derive that for $x\in\mathcal{L}$

$$\mathop{\mathbb{E}}_{\substack{r_1\leftarrow\mathcal{U},\pi_1\leftarrow\tilde{\mathsf{P}}_1(x;r_1)\\ r_2\leftarrow\mathcal{U},\pi_2\leftarrow\tilde{\mathsf{P}}_2(x;r_1,\pi_1,r_2)}}[\mathsf{V}(x;r_1,\pi_1,r_2,\pi_2)]\geq 1-2\epsilon_z.$$

On the other hand, soundness of the original protocol $\langle\mathsf{P}_w,\mathsf{V}\rangle$ ensures that, for $x\notin\mathcal{L}$

$$\mathop{\mathbb{E}}_{\substack{r_1\leftarrow\mathcal{U},\pi_1\leftarrow\tilde{\mathsf{P}}_1(x;r_1)\\ r_2\leftarrow\mathcal{U},\pi_2\leftarrow\tilde{\mathsf{P}}_2(x;r_1,\pi_1,r_2)}}[\mathsf{V}(x;r_1,\pi_1,r_2,\pi_2)]\leq \epsilon_s.$$

When $\epsilon_s+2\epsilon_z$ is noticeably less than 1, we thus obtain an efficient algorithm that decides if $x\in\mathcal{L}$, which yields the desired result.

One catch here is that while this approach is conceptually complete, it only gives us a construction of an infinitely often one-way function. This is because in our approach, we will require that the adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ must both succeed in their inversion so that we successfully decide on an instance. This means that the sequence of input lengths on which our assumed inverters work must overlap when we aim for a contradiction - and the formal negation for this only implies infinitely-often hardness. See Section 4.2.2 and Remark 42 for details.

2 Preliminaries

Denote by $\mathcal{U}_\ell$ the uniform distribution over the length-ℓ binary strings $\{0,1\}^\ell$. For a language $\mathcal{L}$, let $\mathcal{L}_n=\mathcal{L}\cap\{0,1\}^n$ be the set of all the length-n strings in the language. For $k\in\mathbb{N}$, denote $[k]=\{1,\ldots,k\}$. We define a distribution ensemble $\mathcal{X}=\{X_n\}_{n\in\mathbb{N}}$ to be a collection of distributions where each X_n is defined over $\{0,1\}^{m(n)}$ where $m:\mathbb{N}\to\mathbb{N}$ is some arithmetic function. We say a function ν is negligible if for every polynomial p there exists an $n_0\in\mathbb{N}$ such that for all $n\geq n_0$, $\mu(n)<\frac{1}{p(n)}$. Similarly, we call a function μ noticeable if there exists a polynomial p and $n_0\in\mathbb{N}$ such that for all $n\geq n_0$, $\mu(n)\geq\frac{1}{p(n)}$.

For $\epsilon_1=\epsilon_1(n),\epsilon_2=\epsilon_2(n)$, we say $\epsilon_1<_n\epsilon_2$ (or $\epsilon_2>_n\epsilon_1$) to represent the noticeable gap between ϵ_1 and ϵ_2, if there exists a polynomial p such that for all sufficiently large $n\in\mathbb{N}$, we have $\epsilon_1(n)+\frac{1}{p(n)}<\epsilon_2(n)$, i.e. there is an inverse-polynomial gap between ϵ_1 and ϵ_2.

We use the Hoeffding bound, stated as follows.

▶ **Lemma 5** (Hoeffding's inequality). *For independent random variables $X_1,\ldots,X_q$ with support $[0,1]$, let $X=\sum_{i\in[q]}X_i$, the following holds*

$$\Pr\left[|X-\mathbb{E}X|>t\cdot q\right]\leq 2e^{-2t^2q}.$$

2.1 Indistinguishability

Assume that we have two distributions X and Y defined over a common universe U. We first consider statistical indistinguishability.

▶ **Definition 6** (Statistical Distance). *The* statistical distance *between distributions X and Y (defined over the support of X, denoted by $\mathsf{Supp}(X)$) is defined as*

$$\Delta_s(X;Y) = \frac{1}{2} \sum_{u \in \mathsf{Supp}(X)} \big|\Pr[X=u] - \Pr[Y=u]\big|.$$

The following properties hold.

▶ **Lemma 7** (Triangle Inequality). *For any three distributions X, Y and Z, we have $\Delta_s(X;Z) \le \Delta_s(X;Y) + \Delta_s(Y;Z)$.*

▶ **Lemma 8** (Data Processing Inequality). *For any two probability distributions X, Y (on a common universe U), and any (possibly randomized) process f, we have $\Delta_s(f(X);f(Y)) \le \Delta_s(X;Y)$.*

Suppose that we have a given distinguishing algorithm D to distinguish between X and Y, taking inputs in U and outputting a bit to indicate whether it identifies a given input as being sampled from X or Y. Define the *distinguishing advantage* $\mathsf{Adv}_D(X,Y)$ of D as:

$$\mathsf{Adv}_D(X;Y) = \left| \Pr_{x \leftarrow X}[D(x)=1] - \Pr_{y \leftarrow Y}[D(y)=1] \right|.$$

In fact, the statistical distance implicitly provides an upper bound on the advantage that any distinguisher can obtain, that is $\Delta_s(X;Y) = \max_D \mathsf{Adv}_D(X;Y)$, where D can be any possible algorithm. Usually, for a constant ϵ, when $\Delta_s(X;Y) < \epsilon$, X and Y are said to be ϵ-statistically indistinguishable. Next, we formally define the statistical indistinguishability asymptotically.

▶ **Definition 9** (Statistical Indistinguishability). *Consider a function $\epsilon : \mathbb{N} \to [0,1]$, and distribution ensembles $\mathcal{X} = \{X_n\}_{n\in\mathbb{N}}$ and $\mathcal{Y} = \{Y_n\}_{n\in\mathbb{N}}$. We say that $\mathcal{X}$ and $\mathcal{Y}$ are* ϵ-statistically indistinguishable *(or have statistical distance at most ϵ), denoted by $\Delta_s(\mathcal{X};\mathcal{Y}) \le \epsilon$, if for all $n \in \mathbb{N}$, we have $\Delta_s(X_n;Y_n) \le \epsilon(n)$.*

Notice that we do not impose any computational constraint on the distinguisher above, thus the statistical notion implies that even computationally unbounded algorithms cannot achieve an advantage better than ϵ. It is also natural to restrict attention to polynomial-time algorithms. Next, we define computational indistinguishability.

▶ **Definition 10** (Computational Indistinguishability). *Consider a function $\epsilon : \mathbb{N} \to [0,1]$, and distribution ensembles $\mathcal{X} = \{X_n\}_{n\in\mathbb{N}}$ and $\mathcal{Y} = \{Y_n\}_{n\in\mathbb{N}}$. If for every non-uniform probabilistic polynomial-time algorithm D, there is an $n_D \in \mathbb{N}$ such that for all $n \ge n_D$ we have $\mathsf{Adv}_D(X_n;Y_n) \le \epsilon(n)$, then we say that $\mathcal{X}$ and $\mathcal{Y}$ are* ϵ-computationally indistinguishable *(with respect to polynomial-time algorithms). We denote this by $\Delta_c(\mathcal{X};\mathcal{Y}) \le \epsilon$.*

In the course of our technical arguments, for the sake of simplicity, we often make statements of the form $\Delta_c(X_n;Y_n) \le \epsilon(n)$. These statements and their implications are to be interpreted in the asymptotic sense, as holding for all large enough n rather than for all n.

The definition ensures that $\Delta_c(\mathcal{X};\mathcal{Y}) \le \Delta_s(\mathcal{X};\mathcal{Y})$. Versions of the triangle inequality and data processing inequality hold computational indistinguishability as well. The former is easily implied by essentially a hybrid argument. We state it formally as follows.

▶ **Lemma 11** (Triangle Inequality). *For any three distribution ensembles $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{Z}$, we have*

$$\Delta_c(\mathcal{X};\mathcal{Y}) \le \epsilon_1, \Delta_c(\mathcal{Y};\mathcal{Z}) \le \epsilon_2 \Rightarrow \Delta_c(\mathcal{X};\mathcal{Z}) \le \epsilon_1 + \epsilon_2.$$

▶ **Lemma 12** (Data Processing Inequality). *For any distribution ensembles $\mathcal{X}, \mathcal{Y}$ and any probabilistic polynomial-time procedure f, we have*

$$\Delta_c(\mathcal{X}; \mathcal{Y}) \leq \epsilon \Rightarrow \Delta_c(f(\mathcal{X}); f(\mathcal{Y})) \leq \epsilon.$$

This is a simple consequence of the observation that any such efficient function f can simply be run on top of samples from $\mathcal{X}$ or $\mathcal{Y}$ and then fed into a distinguisher for $f(\mathcal{X})$ and $f(\mathcal{Y})$.

We slightly abuse the notation by writing $\Delta_c(\mathcal{X}; \mathcal{Z}) \leq \Delta_c(\mathcal{X}; \mathcal{Y}) + \Delta_c(\mathcal{Y}; \mathcal{Z})$ and $\Delta_c(f(\mathcal{X}); f(\mathcal{Y})) \leq \Delta_c(\mathcal{X}; \mathcal{Y})$, by which we mean the properties defined above.

2.2 Circuits and Oracles

We define notions of circuits and functions computed with respect to certain oracles.

▶ **Definition 13** (Oracle-Aided Circuits and Algorithms). *Let $\mathcal{O} : \{0,1\}^* \to \{0,1\}^*$ be an oracle (an arbitrary function). An oracle circuit (or algorithm) C with respect to $\mathcal{O}$, denoted by $C^{\mathcal{O}}$ is a circuit (or algorithm) where in addition to standard operations, C also has oracle gates (or oracle operations) where it can make a query to $\mathcal{O}$, and expect its output as response.*

▶ **Definition 14** (Oracle-Aided Functions). *Let $\mathcal{O} : \{0,1\}^* \to \{0,1\}^*$ be an oracle (an arbitrary function). An oracle aided function f with respect to $\mathcal{O}$, denoted $f^{\mathcal{O}}$, is a function computable by a deterministic oracle-aided algorithm $A^{\mathcal{O}}$.*

▶ Remark 15. When $\mathcal{O}$ is a randomized algorithm, we view the oracle gates for $\mathcal{O}$ as deterministic ones that take an additional randomness as input.

2.3 One-Way Functions

In the following, we present the definition of one-way functions, along with several weaker variants, which will serve as intermediate steps in our later proofs.

▶ **Definition 16** (One-Way Function, OWF). *For m_1, m_2 being polynomials, a function family $\mathcal{F} = \{f_n : \{0,1\}^{m_1(n)} \to \{0,1\}^{m_2(n)}\}_{n\in\mathbb{N}}$ is said to be a* One-Way Function (OWF) *if $\mathcal{F}$ is efficiently computable and for every non-uniform PPT algorithm $\mathcal{A}$ there is a negligible function $\nu(\cdot)$ such that for all large enough $n \in \mathbb{N}$,*

$$\Pr_{x \leftarrow \mathcal{U}_{m_1(n)}} \left[f_n(\mathcal{A}(f_n(x))) = f_n(x) \right] \leq \nu(n).$$

▶ **Definition 17** (Weak One-Way Function). *For m_1, m_2 being polynomials, a function family $\mathcal{F} = \{f_n : \{0,1\}^{m_1(n)} \to \{0,1\}^{m_2(n)}\}_{n\in\mathbb{N}}$ is said to be a* Weak One-Way Function *if $\mathcal{F}$ is efficiently computable and there is a polynomial p such that for every non-uniform PPT algorithm $\mathcal{A}$, for all large enough $n \in \mathbb{N}$,*

$$\Pr_{x \leftarrow \mathcal{U}_{m_1(n)}} \left[f_n(\mathcal{A}(f_n(x))) = f_n(x) \right] \leq 1 - \frac{1}{p(n)}.$$

▶ **Definition 18** (Distributional One-Way Function, dOWF). *For m_1, m_2 being polynomials, a function family $\mathcal{F} = \{f_n : \{0,1\}^{m_1(n)} \to \{0,1\}^{m_2(n)}\}_{n\in\mathbb{N}}$ is a* Distributional One-Way Function (dOWF) *if $\mathcal{F}$ is efficiently computable and there is a polynomial p such that for every non-uniform PPT algorithm $\mathcal{A}$, for all large enough n, $\Delta_s(X_n; Y_n) > \frac{1}{p(n)}$ holds where X_n and Y_n are defined as:*

1. $X_n : \{(x, f(x)) : x \leftarrow \mathcal{U}_{m_1(n)}\}$
2. $Y_n : \{(\mathcal{A}(f(x)), f(x)) : x \leftarrow \mathcal{U}_{m_1(n)}\}$

▶ Remark 19. In our arguments, we will consider adversaries against distributional one-way functions. We will refer to such an adversary $\mathcal{A}$ as inverting the function in a distributional sense (or distributionally inverting it as shorthand), with deviation say $\frac{1}{q(n)}$ (where $q(\cdot)$ is a polynomial), to mean that $\Delta_s\big((x, f(x)); (\mathcal{A}(f(x), f(x)))\big) \leq \frac{1}{q(n)}$ for uniformly sampled $x \leftarrow \mathcal{U}_{m_1(n)}$.

▶ **Definition 20** (Auxiliary-Input One-Way Functions, ai-OWF). *For m_1, m_2 being polynomials, a function family $\mathcal{F} = \{f_a : \{0,1\}^{m_1(|a|)} \to \{0,1\}^{m_2(|a|)}\}_{a\in\{0,1\}^*}$ is said to be an* Auxiliary-Input One-Way Function (ai-OWF) *if $\mathcal{F}$ is efficiently computable and for every non-uniform PPT machine $\mathcal{A}$ there is a negligible function $\nu(\cdot)$ such that for all large enough $n \in \mathbb{N}$, there exists some $a \in \{0,1\}^n$ such that we have*

$$\Pr_{x\leftarrow\mathcal{U}_{m_1(n)}}\big[f_a(A(a, f_a(x))) = f_a(x)\big] \leq \nu(n).$$

▶ Remark 21. If in the above definition the string a (for a given n) is always fixed to be 0^n, then the definition collapses to that of a standard one-way function.

▶ Remark 22. Similar to the above variant of (standard) OWFs, we can also extend the definitions of weak and distributional one-way functions to the auxiliary input setting in the natural manner.

▶ **Definition 23** (Infinitely-Often One-Way Functions, ioOWF). *For m_1, m_2 being polynomials, a function family $\mathcal{F} = \{f_n : \{0,1\}^{m_1(n)} \to \{0,1\}^{m_2(n)}\}$ is an* Infinitely Often One-Way Function (ioOWF) *if $\mathcal{F}$ is efficiently computable and if for every non-uniform PPT algorithm A there is a negligible function $\nu(\cdot)$ and an infinite set $S_A \subseteq \mathbb{N}$ such that we have*

$$\Pr_{r\leftarrow\{0,1\}^{m_1(n)}}\big[f_n(A(f_n(r))) = f_n(r)\big] \leq \nu(n)$$

for all $n \in S_A$.

▶ Remark 24. When the properties of any of the primitives above hold only for infinitely many $n \in \mathbb{N}$, we refer to them as infinitely-often versions. Similarly, infinitely-often versions of complexity classes also be defined – e.g., ioP is the set of languages that have deterministic polynomial-time algorithms that are correct on some infinite set of input lengths.

▶ Remark 25. Similar to above, we can also define weak and distributional infinitely often one-way functions with straightforward modifications.

▶ **Lemma 26** ([8]). *There is an explicit, efficient transformation from any distributional one-way function to a standard one-way function.*

▶ **Lemma 27** ([13]). *There is an explicit, efficient transformation from any weak one-way function to a standard one-way function.*

▶ Remark 28. Both results cited above work as stated for converting (infinitely-often or auxiliary-input) weak or distributional one-way functions to (infinitely-often or auxiliary-input) one-way functions.

2.4 Zero-Knowledge Protocols

In this section, we formally define zero-knowledge protocols, specifically *non-interactive zero-knowledge* (NIZK) and *public-coin zero-knowledge* proofs and arguments.

2.4.1 Non-Interactive Zero-Knowledge

We describe the non-interactive zero-knowledge proofs or arguments in the *common reference string* model. In particular, there is no interaction between the prover and the verifier. Both parties refer to a common reference string r, which is randomly sampled, to run the protocol. For an NP language $\mathcal{L}$, on input x, the honest prover holds the NP witness w and generates a message $\pi \leftarrow \mathsf{P}(x, w; r)$; then the verifier computes $0/1 \leftarrow \mathsf{V}(x; r, \pi)$, where by convention, 1 denotes the acceptance of the proof.

▶ **Definition 29** (Non-Interactive Zero-Knowledge, NIZK). *For $\epsilon_c, \epsilon_s, \epsilon_z : \mathbb{N} \to [0, 1]$ and a language $\mathcal{L} \in \mathsf{NP}$, an $(\epsilon_c, \epsilon_s, \epsilon_z)$-*Non-Interactive Zero-Knowledge (NIZK) proof *for $\mathcal{L}$ consists of algorithms $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$, where Gen samples the* common reference string *in polynomial time, V is a deterministic polynomial-time verifier and P is a computationally unbounded prover. Let n be the length of the input x and $\mathcal{R}_\mathcal{L}$ denote the corresponding NP relation. The protocol should satisfy the following properties.*

1. Completeness. *For any $x \in \mathcal{L}_n$ and any witness w such that $(x, w) \in \mathcal{R}_\mathcal{L}$*

$$\Pr_{\substack{r \leftarrow \mathsf{Gen}(1^n) \\ \pi \leftarrow \mathsf{P}(x,w;r)}} [\mathsf{V}(x; r, \pi) = 1] \geq 1 - \epsilon_c(n).$$

2. Soundness. *For any $x \in \{0, 1\}^n \setminus \mathcal{L}_n$ and any prover algorithm P^*, the following holds*

$$\Pr_{\substack{r \leftarrow \mathsf{Gen}(1^n) \\ \pi^* \leftarrow \mathsf{P}^*(x;r)}} [\mathsf{V}(x; r, \pi^*) = 1] \leq \epsilon_s(n),$$

3. Computational Zero-Knowledge. *There exists a probabilistic polynomial-time simulator Sim such that for any $x \in \mathcal{L}_n$ and $(x, w) \in \mathcal{R}_\mathcal{L}$*

$$\Delta_c(\mathsf{Sim}(x); \mathsf{View}\langle \mathsf{P}, \mathsf{V}\rangle(x, w)) \leq \epsilon_z(n),$$

where the transcript $\mathsf{View}\langle \mathsf{P}, \mathsf{V}\rangle(x, w)$ represents the view of the verifier in the protocol with input x and witness w given to the prover, consisting of the common reference string r and the prover's message π.

If the honest prover P is constrained to be computationally efficient, and the soundness condition is required to hold only against computationally efficient provers P^*, the protocol is called an NIZK argument.

▶ Remark 30. The simulator $\mathsf{Sim}(x)$ is randomized on input x. When we need a deterministic description, we explicitly expose the random coins and include them as part of the input, which is written as $\mathsf{Sim}(x; \rho)$. Equivalently, $\mathsf{Sim}(x)$ represents the distribution of $\mathsf{Sim}(x; \rho)$ when ρ is drawn uniformly randomly. For convenience, we denote by $\mathsf{Sim}_i(x)$ the simulator $\mathsf{Sim}(x)$ restricted to outputting only the i-th message. For example, in the NIZK protocols, $\mathsf{Sim}_1(x)$ only samples the marginal distribution on the common reference string r while $\mathsf{Sim}_2(x)$ simulates the distribution of π.

2.4.2 Public-Coin Zero-Knowledge

Beyond non-interactive protocols, we study the broader class of public-coin zero-knowledge arguments or proofs.

▶ **Definition 31** (Public-Coin Zero-Knowledge). *For $\epsilon_c, \epsilon_s, \epsilon_z : \mathbb{N} \to [0, 1]$ and a language $\mathcal{L}$, an $(\epsilon_c, \epsilon_s, \epsilon_z)$-*public-coin Zero-Knowledge (ZK) proof *for $\mathcal{L}$ consists of algorithms (P, V), where V is a deterministic polynomial-time verifier and P is a computationally unbounded prover. In an execution of the protocol V is given as input the instance x and P the instance x and a witness w. The protocol should satisfy the following.*

1. Syntax and notation. *In each round, first a uniformly random string r_i of pre-specified length is sampled and sent to the prover, and the prover responds with a message π_i computed as $\pi_i \leftarrow \mathsf{P}_i(x, w; r_1, \pi_1, \ldots, r_i)$. At the end, the verifier decides whether to accept the transcript $(r_1, \pi_1, \ldots, r_k, \pi_k)$ by computing $0/1 \leftarrow \mathsf{V}(x; r_1, \pi_1, \ldots, r_k, \pi_k)$. Denote by $\langle \mathsf{P}, \mathsf{V} \rangle(x, w)$ the output of the protocol on a common input x and a witness w (held only by the prover), and $\mathsf{View}\langle \mathsf{P}, \mathsf{V} \rangle(x, w)$ represents the transcript of the execution, consisting of $(r_1, \pi_1, \ldots, r_k, \pi_k)$. Here, k is the number of rounds of the protocol.*
2. Completeness. *For any $x \in \mathcal{L}_n$, for any w that $(x, w) \in \mathcal{R}_{\mathcal{L}}$, $\Pr[\langle \mathsf{P}, \mathsf{V} \rangle(x, w) = 1] \geq 1 - \epsilon_c(n)$.*
3. Soundness. *For any $x \in \{0,1\}^n \setminus \mathcal{L}_n$, for any prover P^*, $\Pr[\langle \mathsf{P}^*, \mathsf{V} \rangle(x) = 1] \leq \epsilon_s(n)$.*
4. Computational Zero-knowledge. *There exists a probabilistic polynomial-time simulator Sim, such that for any $x \in \mathcal{L}_n$ and $(x, w) \in \mathcal{R}_{\mathcal{L}}$, $\Delta_c(\mathsf{Sim}(x); \mathsf{View}\langle \mathsf{P}, \mathsf{V} \rangle(x, w)) \leq \epsilon_z(n)$. If the honest prover P is constrained to be computationally efficient, and the soundness condition is required to hold only against computationally efficient provers P^*, the protocol is called a* ZK argument.

▶ Remark 32. We often think of the public coins r_i's as being sent by the verifier to the prover. We consider the process of the verifier sending a randomness and the prover replying with a message as one round in the protocol, where each round contains two messages. For convenience in notation, when the first message in the protocol is from the prover, we sometimes pretend that there is an empty message from the verifier before that.

3 Auxiliary-Input One-Way Functions

In this section, we show that if a language has a Zero-Knowledge proof or argument with errors satisfying certain conditions, then for any instance, we can define a function such that any inverter for that function can be used to decide the membership of that instance in the language. Worst-case hardness of the language then gives us an auxiliary-input one-way function, which will be used in later sections to construct one-way functions from hard languages that have such proof systems.

We do this for Non-Interactive ZK protocols in Section 3.1 and for general public-coin ZK protocols in Section 3.2. In Section 3.3, we use additional ideas to improve the range of errors that can be used, at the cost of the proof being non-black-box, and yielding only infinitely often secure OWFs.

▶ Remark 33. In this section, we state the main lemmas for both ZK proofs and ZK arguments. To avoid excessive repetition, in the proofs of these lemmas, we only deal with the case of arguments. It may be verified that these proofs do not rely on the efficiency of the honest prover, and work nearly as is for proof systems as well.

3.1 Reductions from NIZK

▶ **Lemma 34.** *For some $\epsilon_c, \epsilon_s, \epsilon_z : \mathbb{N} \to [0, 1]$, suppose a language $\mathcal{L} \in \mathsf{NP}$ has an $(\epsilon_c, \epsilon_s, \epsilon_z)$-NIZK proof or argument (with deterministic verification). Then there exists a reduction R, which is a polynomial-time oracle-aided algorithm, and a polynomial-time computable function family $\mathcal{F} = \{f_x\}_{x \in \{0,1\}^*}$ such that, for any probabilistic polynomial-time algorithm $\mathcal{A}$, any polynomial p, and all large enough $n \in \mathbb{N}$,*

1. *For any $x \in \mathcal{L}_n$, if $\mathcal{A}$ inverts f_x with probability at least $(1 - 1/p(n))$, then*

$$\Pr\left[R^{\mathcal{A}}(x) = 1\right] \geq 1 - \epsilon_c(n) - \epsilon_z(n) - \frac{1}{p(n)}.$$

2. *For any $x \in \{0,1\}^n \setminus \mathcal{L}_n$,*

$$\Pr\left[R^{\mathcal{A}}(x) = 1\right] \leq \epsilon_s(n).$$

▶ Remark 35. Owing to space constraints, we omit this proof and several subsequent ones, and refer the readers to the full version of our paper.

▶ **Corollary 36.** *For $\epsilon_c, \epsilon_s, \epsilon_z : \mathbb{N} \to [0,1]$, suppose there is an* NP *language $\mathcal{L} \notin$* ioP/poly *that has an $(\epsilon_c, \epsilon_s, \epsilon_z)$-NIZK proof or argument with $\epsilon_c + \epsilon_s + \epsilon_z <_n 1$. Then auxiliary-input one-way functions exist.*

Proof. Let R be the oracle-aided algorithm and $\mathcal{F}$ be the function family guaranteed by Lemma 34. There exists a polynomial q such that:

$$\epsilon_c(n) + \epsilon_s(n) + \epsilon_z(n) + \frac{1}{q(n)} < 1$$

holds for all sufficiently large $n \in \mathbb{N}$. Then, let $p = 2q$. Suppose that there is a (non-uniform) PPT algorithm $\mathcal{A}$ that, for infinitely many $n \in \mathbb{N}$, for every $x \in \mathcal{L}_n$, inverts f_x with probability at least $(1 - 1/p(n))$. By Lemma 34, for every $x \in \mathcal{L}_n$,

$$\Pr\left[R^{\mathcal{A}}(x) = 1\right] \geq 1 - \epsilon_c(n) - \epsilon_z(n) - \frac{1}{p(n)};$$

and for every $x \in \{0,1\}^n \setminus \mathcal{L}_n$,

$$\Pr\left[R^{\mathcal{A}}(x) = 1\right] \leq \epsilon_s(n).$$

Since both $\mathcal{A}$ and R are polynomial-time algorithms and

$$1 - \epsilon_c(n) - \epsilon_z(n) - \frac{1}{2q(n)} >_n \epsilon_s(n),$$

it contradicts $\mathcal{L} \notin$ ioP/poly. Hence, $\mathcal{F}$ is a weak auxiliary-input one-way function: for any efficient non-uniform adversary, for all sufficiently large n, there exists a function f_x, $x \in \{0,1\}^n$ on which the adversary cannot successfully invert f_x with probability at least $(1 - 1/p(n))$. By [13], the existence of weak ai-OWFs implies the existence of (standard) ai-OWFs, which concludes the proof. ◀

3.2 Reductions from Public-Coin ZK

▶ **Lemma 37.** *For some $\epsilon_c, \epsilon_s, \epsilon_z : \mathbb{N} \to [0,1]$, $t : \mathbb{N} \to \mathbb{N}$, suppose a language $\mathcal{L} \in$* NP *has an $(\epsilon_c, \epsilon_s, \epsilon_z)$-public-coin ZK proof or argument with t messages. Then there exists a reduction R, which is a polynomial-time oracle-aided algorithm, and a polynomial-time computable function family $\mathcal{F} = \{f_x\}_{x \in \{0,1\}^*}$ such that, for any probabilistic polynomial-time algorithm $\mathcal{A}$, any polynomial p, and all large enough $n \in \mathbb{N}$,*

1. *For any $x \in \mathcal{L}_n$, if $\mathcal{A}$ distributionally inverts f_x with deviation at most $1/p(n)$, then*

$$\Pr\left[R^{\mathcal{A}}(x) = 1\right] \geq 1 - \epsilon_c(n) - (t(n) - 1) \cdot \epsilon_z(n) - \frac{t(n)}{p(n)}.$$

2. *For any $x \in \{0,1\}^n \setminus \mathcal{L}_n$,*

$$\Pr\left[R^{\mathcal{A}}(x) = 1\right] \leq \epsilon_s(n).$$

▶ **Corollary 38.** *For $\epsilon_c, \epsilon_s, \epsilon_z : \mathbb{N} \to [0,1]$ and $t : \mathbb{N} \to \mathbb{N}$, suppose there is an* NP *language $\mathcal{L} \notin$* ioP/poly *that has a t-message public-coin $(\epsilon_c, \epsilon_s, \epsilon_z)$-ZK proof or argument with $\epsilon_c + \epsilon_s + (t-1)\epsilon_z <_n 1$. Then auxiliary-input one-way functions exist.*

3.3 Reductions from Constant-Round Public-Coin ZK

For the lemma below, as opposed to the other (similar) ones, we consider the number of rounds in the protocol rather than the number of messages. Recall that a round consists of a message from the verifier, followed by a message from the prover. If the protocol starts with the prover sending a message, we think of it as starting with an empty message from the verifier instead.

▶ **Lemma 39.** *For some $\epsilon_c, \epsilon_s, \epsilon_z : \mathbb{N} \to [0,1]$ and any constant $k \in \mathbb{N}$, suppose a language $\mathcal{L} \in \mathsf{NP}$ has a k-round public-coin $(\epsilon_c, \epsilon_s, \epsilon_z)$-ZK proof or argument. Then there exists a polynomial-time oracle-aided algorithm R and families of oracle-aided functions $\mathcal{F}_1, \ldots, \mathcal{F}_k$ satisfying the following.*

For $i \in [k]$, the family $\mathcal{F}_i = \{f_{i,x}\}_{x \in \{0,1\}^*}$ consists of functions that require access to $(k-i)$ oracles, and are polynomial-time computable given these oracles. For any sequence of probabilistic polynomial-time algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_k$, any polynomial p, and all large enough n, we have the following.

1. *For any $x \in \mathcal{L}_n$, if for all $i \in [k]$, $\mathcal{A}_i$ distributionally inverts $f_{i,x}^{\mathcal{A}_{i+1},\ldots,\mathcal{A}_k}$ with deviation at most $1/p(n)$, then*

$$\Pr\left[R^{\mathcal{A}_1 \ldots \mathcal{A}_k}(x) = 1\right] \geq 1 - \epsilon_c(n) - k \cdot \epsilon_z(n) - \frac{3k-2}{p(n)} - \mathsf{negl}(n).$$

2. *For any $x \in \{0,1\}^n \setminus \mathcal{L}_n$,*

$$\Pr\left[R^{\mathcal{A}_1 \ldots \mathcal{A}_k}(x) = 1\right] \leq \epsilon_s(n).$$

▶ Remark 40. In this approach, the functions are defined recursively and the number of function families is closely related to the round number k, so we can only achieve the implication result for any constant k. We do not know how to make it work beyond a finite number of rounds. This issue has also been observed in other work that employ similar recursive constructions [3, 10].

▶ **Corollary 41.** *For $\epsilon_c, \epsilon_s, \epsilon_z : \mathbb{N} \to [0,1]$ and a constant $k \in \mathbb{N}$, suppose there is an NP language $\mathcal{L} \notin \mathsf{P/poly}$ that has a k-round public-coin $(\epsilon_c, \epsilon_s, \epsilon_z)$-ZK proof or argument with $\epsilon_c + \epsilon_s + k \cdot \epsilon_z <_n 1$. Then infinitely-often auxiliary-input one-way functions exist.*

▶ Remark 42. Our approach to obtaining a contradiction relies on the overlap of input lengths on which all of the algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_k$ succeed in their respective inversions. The negation of this assumption only implies the existence of infinitely-often one-way functions. Given that we only get infinitely-often security anyway, we are able to weaken our assumption to $\mathcal{L} \notin \mathsf{P/poly}$ instead of needing $\mathcal{L} \notin \mathsf{ioP/poly}$ as in our other results.

4 One-Way Functions

We complete our transformation by extending the result of [7] that shows that assuming there are zero-knowledge arguments for NP, auxiliary-input one-way functions imply standard one-way functions. We follow the approach taken by [4] in the context of weak zero-knowledge arguments, showing that the auxiliary-input OWFs obtained in the previous sections also imply standard one-way functions. The resulting theorems are as follows.

▶ **Theorem 43.** *If $\mathsf{NP} \not\subseteq \mathsf{ioP/poly}$ and, for some $\epsilon_c, \epsilon_s, \epsilon_z : \mathbb{N} \to [0,1]$ such that $\epsilon_c + \epsilon_s + \epsilon_z <_n 1$, every language in NP has an $(\epsilon_c, \epsilon_s, \epsilon_z)$-NIZK proof, then one-way functions exist. The same holds for NIZK arguments.*

▶ **Theorem 44.** *If* NP $\not\subseteq$ ioP/poly *and, for some* $\epsilon_c, \epsilon_s, \epsilon_z : \mathbb{N} \to [0,1]$ *and* $t : \mathbb{N} \to \mathbb{N}$ *such that* $\epsilon_c + \epsilon_s + (t-1) \cdot \epsilon_z <_n 1$*, every language in* NP *has a* t*-message public-coin* $(\epsilon_c, \epsilon_s, \epsilon_z)$*-ZK proof, then one-way functions exist. The same holds for ZK arguments.*

▶ **Theorem 45.** *If* NP $\not\subseteq$ P/poly *and, for some* $\epsilon_c, \epsilon_s, \epsilon_z : \mathbb{N} \to [0,1]$ *and constant* $k \in \mathbb{N}$ *such that* $\epsilon_c + \epsilon_s + k \cdot \epsilon_z <_n 1$*, every language in* NP *has a* k*-round public-coin* $(\epsilon_c, \epsilon_s, \epsilon_z)$*-ZK proof, then infinitely often one-way functions exist. The same holds for ZK arguments.*

The proofs of these theorems follow from combining the respective lemmas in Section 3 with lemmas stated later in this section. Theorems 43 and 44 are proven in Section 4.2.1, and Theorem 45 in Section 4.2.2.

As observed in [4, Section 5] (which in turn draws on the approach in [9]), the overall implication from weak ZK to OWFs can be broken up into three parts. The first part consists of showing that having such a weak ZK protocol for worst-case hard language implies an auxiliary-input OWF, and this we have shown in Section 3 (for various flavors of ZK protocols). The two remaining steps are the following:

1. Show that auxiliary-input OWFs imply that there exist what are called in [4] as *one-sided average-case hard* languages.
2. Show that (the corresponding flavor of) weak ZK protocols for such one-sided average-case hard languages imply standard OWFs.

We outline these transformations below for each flavor of weak ZK that we have considered so far. Some of the proofs below are along the lines of those of similar lemmas from prior work, especially from [4].

4.1 One-Sided Average-Case Hardness from ai-OWF

This lemma appears in [4, Lemma 4]. This result is somewhat independent, and does not have anything to do with weak zero-knowledge. We can thus use it as is.

▶ **Definition 46** (One-Sided Average-Case BPP)**.** *The class* 1AvgBPP/poly *consists of average-case problems* $(\mathcal{L}, \mathcal{D})$ *for which there is a non-uniform probabilistic polynomial-time algorithm* $\mathcal{A}$ *and functions* $a, b : \mathbb{N} \to [0,1]$ *with* $a >_n b$*, such that for all sufficiently large* n

1. *For every* $x \in \mathcal{L}_n$*,* $\Pr[\mathcal{A}(x) = 1] \geq a(n)$*.*
2. $\Pr_{x \leftarrow \mathcal{D}_n}[\mathcal{A}(x) = 1 | x \notin \mathcal{L}_n] \leq b(n)$*.*

▶ **Lemma 47** ([4, Lemma 4])**.** *Suppose that there exists an auxiliary-input one-way function. Then there exists an* $\mathcal{L} \in$ NP *such that* $(\mathcal{L}, \mathcal{U}) \notin$ io1AvgBPP/poly *and* $\Pr_{x \leftarrow \mathcal{U}_n}[x \notin \mathcal{L}_n] \geq 1/2$ *for all* $n \in \mathbb{N}$*.*

The following variant also follows from the proof of the above lemma.

▶ **Lemma 48.** *Suppose that there exists an infinitely-often ai-OWF, then there exists a language* $\mathcal{L} \in$ NP *such that* $(\mathcal{L}, \mathcal{U}) \notin$ 1AvgBPP/poly *and* $\Pr_{x \leftarrow \mathcal{U}_n}[x \notin \mathcal{L}_n] \geq 1/2$ *for all* $n \in \mathbb{N}$*.*

4.2 OWF from One-Sided Average-Case Hardness

4.2.1 NIZK and Public-Coin ZK

Combining Lemma 34 and Lemma 37 with the amplification for converting weak and distributional one-way functions into standard form [13, 8], we obtain the following lemma.

▶ **Lemma 49.** *For some $\epsilon_c, \epsilon_s, \epsilon_z : \mathbb{N} \to [0,1]$ and $t : \mathbb{N} \to \mathbb{N}$, consider a language $\mathcal{L} \in \mathsf{NP}$ with an $(\epsilon_c, \epsilon_s, \epsilon_z)$-NIZK protocol (in which case $t = 2$) or a t-message $(\epsilon_c, \epsilon_s, \epsilon_z)$-public-coin ZK protocol. For any polynomials p_1, p_2, there is a reduction R, which is a polynomial-time oracle-aided algorithm, and a polynomial-time computable function family $\mathcal{F} = \{f_x\}_{x\in\{0,1\}^*}$ such that for any probabilistic polynomial-time algorithm $\mathcal{A}$ and all large enough n,*

1. *When $x \in \mathcal{L}_n$, if $\mathcal{A}$ inverts f_x with probability at least $1/p_1(n)$, then*

$$\Pr\left[R^{\mathcal{A}}(x) = 1\right] \geq 1 - \epsilon_c(n) - (t(n) - 1) \cdot \epsilon_z(n) - 1/p_2(n).$$

2. *When $x \in \{0,1\}^n \setminus \mathcal{L}_n$,*

$$\Pr\left[R^{\mathcal{A}}(x) = 1\right] \leq \epsilon_s(n).$$

Note that our constructions of R and $\mathcal{F}$ in the proofs of Lemma 34 and Lemma 37 do not satisfy the above conditions. However, reductions and functions satisfying the above lemma can be obtained from the previous construction by applying the techniques in [13, 8]. Next, we use the following analogue of [4, Lemma 5].

▶ **Lemma 50.** *For $\epsilon_c, \epsilon_s, \epsilon_z : \mathbb{N} \to [0,1]$ and $t : \mathbb{N} \to \mathbb{N}$, consider any language $\mathcal{L}$ such that $(\mathcal{L},\mathcal{U}) \notin \mathsf{io1AvgBPP/poly}$, and $\Pr_{x\leftarrow\mathcal{U}_n}[x \notin \mathcal{L}_n] \geq 1/2$ for all $n \in \mathbb{N}$. If there is an $(\epsilon_c, \epsilon_s, \epsilon_z)$-NIZK protocol (in which case $t = 2$) or a t-message $(\epsilon_c, \epsilon_s, \epsilon_z)$-public-coin ZK protocol for $\mathcal{L}$ with $\epsilon_c + \epsilon_s + (t-1) \cdot \epsilon_z <_n 1$, then one-way functions exist.*

Combining Corollary 36, Lemma 47 and Lemma 50 yields Theorem 43. Analogously, for any round public-coin ZK protocol, by putting Corollary 38, Lemma 47 and Lemma 50 together, we obtain Theorem 44.

4.2.2 Constant-Round Public-Coin ZK

Lemma 39 combined with [8] yields the following lemma.

▶ **Lemma 51.** *For some $\epsilon_c, \epsilon_s, \epsilon_z : \mathbb{N} \to [0,1]$ and any constant $k \in \mathbb{N}$, suppose a language $\mathcal{L} \in \mathsf{NP}$ has a k-round public-coin $(\epsilon_c, \epsilon_s, \epsilon_z)$-ZK proof or argument. For any polynomials p_1, p_2, there exists a polynomial-time oracle-aided algorithm R and families of oracle-aided functions $\mathcal{F}_1, \ldots, \mathcal{F}_k$ satisfying the following.*

For $i \in [k]$, the family $\mathcal{F}_i = \{f_{i,x}\}_{x\in\{0,1\}^*}$ consists of functions that require access to $(k-i)$ oracles, and are polynomial-time computable given these oracles. For any sequence of probabilistic polynomial-time algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_k$ and all large enough n, we have the following.

1. *For every $x \in \mathcal{L}_n$, if for all $i \in [k]$, $\mathcal{A}_i$ inverts $f_{i,x}$ with probability at least $1/p_1(n)$, then*

$$\Pr\left[R^{\mathcal{A}_1 \ldots \mathcal{A}_k}(x) = 1\right] \geq 1 - \epsilon_c(n) - k \cdot \epsilon_z(n) - 1/p_2(n)$$

2. *For every $x \in \{0,1\}^n \setminus \mathcal{L}_n$,*

$$\Pr\left[R^{\mathcal{A}_1 \ldots \mathcal{A}_k}(x) = 1\right] \leq \epsilon_s(n).$$

▶ **Lemma 52.** *For $\epsilon_c, \epsilon_s, \epsilon_z : \mathbb{N} \to [0,1]$ and a constant $k \in \mathbb{N}$, consider any language $\mathcal{L}$ such that $(\mathcal{L},\mathcal{U}) \notin \mathsf{1AvgBPP/poly}$, and $\Pr_{x\leftarrow\mathcal{U}_n}[x \notin \mathcal{L}_n] \geq 1/2$ for all $n \in \mathbb{N}$. If there is a k-round $(\epsilon_c, \epsilon_s, \epsilon_z)$-public-coin ZK protocol for $\mathcal{L}$ with $\epsilon_c + \epsilon_s + k \cdot \epsilon_z <_n 1$, then infinitely-often one-way functions exist.*

Theorem 45 follows directly from Corollary 41, Lemma 48, and Lemma 52.

All the omitted proofs can be found in the full version of our paper.

References

1 Benny Applebaum and Eliran Kachlon. NIZK amplification via leakage-resilient secure computation. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part VII*, volume 16006 of *Lecture Notes in Computer Science*, pages 462–479. Springer, 2025. doi:10.1007/978-3-032-01907-3_15.

2 Nir Bitansky and Nathan Geier. Amplification of non-interactive zero knowledge, revisited. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part IX*, volume 14928 of *Lecture Notes in Computer Science*, pages 361–390. Springer, 2024. doi:10.1007/978-3-031-68400-5_11.

3 Jan Buzek and Stefano Tessaro. Collision resistance from multi-collision resistance for all constant parameters. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part V*, volume 14924 of *Lecture Notes in Computer Science*, pages 429–458. Springer, 2024. doi:10.1007/978-3-031-68388-6_15.

4 Suvradip Chakraborty, James Hulett, and Dakshita Khurana. On weak nizks, one-way functions and amplification. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part VII*, volume 16006 of *Lecture Notes in Computer Science*, pages 580–610. Springer, 2025. doi:10.1007/978-3-032-01907-3_19.

5 Suvradip Chakraborty, James Hulett, Dakshita Khurana, and Kabir Tomer. Non-trivial zero-knowledge implies one-way functions. Cryptology ePrint Archive, Paper 2026/331, 2026. URL: https://eprint.iacr.org/2026/331.

6 Vipul Goyal, Aayush Jain, and Amit Sahai. Simultaneous amplification: The case of non-interactive zero-knowledge. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 608–637. Springer, 2019. doi:10.1007/978-3-030-26951-7_21.

7 Shuichi Hirahara and Mikito Nanashima. One-way functions and zero knowledge. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 1731–1738. ACM, 2024. doi:10.1145/3618260.3649701.

8 Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 230–235. IEEE Computer Society, 1989. doi:10.1109/SFCS.1989.63483.

9 Yanyi Liu, Noam Mazor, and Rafael Pass. A note on zero-knowledge for NP and one-way functions. *IACR Cryptol. ePrint Arch.*, page 800, 2024. URL: https://eprint.iacr.org/2024/800.

10 Changrui Mu and Prashant Nalini Vasudevan. Instance-hiding interactive proofs - (extended abstract). In Elette Boyle and Mohammad Mahmoody, editors, *Theory of Cryptography - 22nd International Conference, TCC 2024, Milan, Italy, December 2-6, 2024, Proceedings, Part I*, volume 15364 of *Lecture Notes in Computer Science*, pages 3–34. Springer, 2024. doi:10.1007/978-3-031-78011-0_1.

11 Rafail Ostrovsky. One-way functions, hard on average problems, and statistical zero-knowledge proofs. In *Proceedings of the Sixth Annual Structure in Complexity Theory Conference, Chicago, Illinois, USA, June 30 - July 3, 1991*, pages 133–138. IEEE Computer Society, 1991. doi:10.1109/SCT.1991.160253.

12 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. doi:10.1109/ISTCS.1993.253489.

13 Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91. IEEE Computer Society, 1982. doi:10.1109/SFCS.1982.45.