Super-Perfect Zero-Knowledge Proofs

Oded Goldreich and Liav Teichner

Abstract. We initiate a study of super-perfect zero-knowledge proof systems. Loosely speaking, these are proof systems for which the interaction can be perfectly simulated in strict probabilistic polynomial-time. In contrast, the standard definition of perfect zero-knowledge only requires that the interaction can be perfectly simulated by a strict probabilistic polynomial-time that is allowed to fail with probability at most one half.

We show that two types of perfect zero-knowledge proof systems can be transformed into super-perfect ones. The first type includes the perfect zero-knowledge interactive proof system for Graph Isomorphism and other systems of the same form, including perfect zero-knowledge arguments for NP. The second type refers to perfect non-interactive zero-knowledge proof systems. We also present a super-perfect non-interactive zero-knowledge proof system for the set of Blum integers.

An early version of this work appeared as TR14-097 of *ECCC*. The current revision is quite minimal.

1 Introduction

A standard exposition of the notion of zero-knowledge proofs may start by presenting the following oversimplified definition:

> An interactive proof system (P, V) for a set S is called zero-knowledge if for every probabilistic polynomial-time strategy V^* there exists a (strict) probabilistic polynomial-time algorithm (called a simulator) A^* such that $A^*(x)$ is distributed identically to the output of V^* after interacting with P on common input x.

(See, e.g., Definition 9.7 in [12, Sec. 9.2.1] and top page 201 in [10, Sec. 4.3.1].)

However (as stated at the bottom of page 201 in [10, Sec. 4.3.1]), the problem with this oversimplified definition is that it is not known to be materializable (for sets outside $\mathcal{BPP}$). Indeed, [12, Def. 9.7] is labeled "oversimplified" and [10, Sec. 4.3.1] avoids presenting it formally. Instead, the standard definition of *perfect* zero-knowledge (cf. [10, Def. 4.3.1]) relaxes the above requirement by allowing the simulator to output a special failure symbol (i.e., $\perp$) with probability at most one half, and requires a perfect simulation conditioned on not failing. We stress that in both cases, the simulator is required to run in *strict* polynomial-time.[1]

[1] Note that this definition of perfect zero-knowledge implies that a perfect simulation can be generated in *expected* (probabilistic) polynomial-time, but the latter does

In this work, we take the "bold" step of turning the oversimplified definition to an actual definition, which we call *super-perfect* zero-knowledge (ZK). We obtain a few positive results regarding this notion, indicating that it is not a vacuous notion; that is, that it can be materializable non-trivially (i.e., for sets outside $\mathcal{BPP}$). Actually, super-perfect zero-knowledge was implicitly considered by Malka [18, Sec. 4.1] (see further discussion below).

The following overview assumes familiarity with the basic definitions and notations, which are reviewed in Section 2.

1.1 Our results

We present several indications that super-perfect ZK exists beyond $\mathcal{BPP}$. Each of these results comes with some limitations (e.g., losing perfect completeness, being applicable only to argument systems or only to perfect NIZK, or holding only for a specific set).

The case of verifier-oblivious simulation failure. Our first result presents a sufficient condition for the existence of super-perfect ZK proof systems. It asserts that any perfect ZK proof system in which *all the relevant simulators output* $\perp$ *with probability that may depend on the input but not on the verifier* (whose interaction with the prover is simulated) *can be converted into super-perfect ZK proof system*. This transformation preserves the soundness error but not the completeness error; in particular, it does not preserve perfect completeness. Specifically, in Section 3.1, we prove

Theorem 1 (from perfect ZK to super-perfect ZK): *Suppose that* (P,V) *is an interactive proof system for* S *and that there exists a function* $p : S \to [0, 0.5]$ *such that for every probabilistic polynomial-time strategy* V^* *there exists a probabilistic polynomial-time algorithm* A^* *such that for every* $x \in S$ *it holds that* $\mathbf{Pr}[A^*(x) = \perp] = p(x)$ *and* $\mathbf{Pr}[A^*(x) = \gamma \mid A^*(x) \neq \perp] = \mathbf{Pr}[\langle P, V^* \rangle(x) = \gamma]$, *for every* $\gamma \in \{0,1\}^*$. *Then,* S *has a super-perfect zero-knowledge proof system. Furthermore:*

- *The soundness error is preserved and the increase in the completeness error is exponentially vanishing;*
- *black-box simulation is preserved;*
- *the communication complexities* (i.e., number of rounds and length of messages) *are preserved; and*
- *the new prover strategy can be implemented by a probabilistic polynomial-time oracle machine that is given oracle access to the original prover strategy.*

The same holds for computationally-sound proof systems (a.k.a argument systems).

not imply the former. Also recall that the issue does not arise for statistical zero-knowledge, since the failure probability can be made exponentially vanishing (by repeated trials), and then absorbed in the statistical deviation of the simulation. Ditto for computational zero-knowledge.

Theorem 1 is proved by observing that the transformation proposed by Malka [18, Sec. 4.1] applies whenever all simulators fail with the same probability (for each fixed input), and not merely when this probability equals one half. We stress that it is not even required that this probability (i.e., the function p) be efficiently computable. (Essentially, the new prover first invokes a simulator (say for V itself) and proceeds with the original proof system if and only if the output is not $\perp$; otherwise, the interaction is suspended and the verifier rejects.) As noted by Malka, one notable example of an interactive proof system that satisfies the foregoing condition (with $p = 1/2$) is the perfect zero-knowledge proof system for Graph Isomorphism of Goldreich, Micali, and Wigderson [13]. The condition holds also for numerous other interactive proofs that have the same form, including the perfect zero-knowledge arguments for $\mathcal{NP}$ of Naor *et al.* [19] (see also [10, Sec. 4.8.3]). Hence, *assuming the existence of one-way permutations, every set in $\mathcal{NP}$ has a super-perfect ZK argument system.*

The case of perfect ZK arguments. In contrast to the previous transformation, the following one does preserve perfect completeness. It refers to a *certain class* of perfect ZK *arguments*, and yields super-perfect ZK *arguments* with perfect completeness, assuming the existence of perfectly binding commitment schemes (which can be constructed based on any one-way permutation). The class (see Definition 3.4) includes the aforementioned proof system for Graph Isomorphism and the perfect zero-knowledge arguments for $\mathcal{NP}$ of Naor *et al.* [19].[2] For details see Section 3.2. (We mention that super-perfect ZK arguments (of perfect completeness) for $\mathcal{NP}$ are implicit in the work of Pass and Rosen [20, 21], were they are based on the existence of claw-free pairs of permutations and established using non-black-box simulators.)

The case of perfect NIZKs. Another case in which perfect completeness can be preserved is the case of *non-interactive zero-knowledge (NIZK)* proof systems. Specifically, we refer to perfect NIZK system in which *the probability that the simulator outputs $\perp$ is efficiently computable.* (Recall that in setting of NIZK there is only one simulator, and it simulates the distribution $(\omega, P(x,\omega))$ where ω is a uniformly distributed "common reference string" (and $P(x,\omega)$ is the proof provided by the prover for input x under the reference strong ω).)

Theorem 2 (from perfect NIZK to super-perfect NIZK): *Suppose that (P,V) is a non-interactive proof system for S and that there exist a polynomial-time computable function $p : S \to [0, 0.5]$ and a probabilistic polynomial-time algorithm A such that for every $x \in S$ it holds that $\mathbf{Pr}[A(x) = \perp] = p(x)$ and $\mathbf{Pr}[A(x) = \gamma \mid A(x) \neq \perp] = \mathbf{Pr}_\omega[(\omega, P(x,\omega)) = \gamma]$, for every $\gamma \in \{0,1\}^*$. Then, S has a super-perfect non-interactive zero-knowledge proof system. Furthermore:*

- *The completeness error is preserved and the increase in the soundness error is exponentially vanishing;*

[2] This holds only in the non-standard model of PPTs, discussed in Section 1.2. Ditto for the result of of Pass and Rosen [20, 21] (mentioned next).

- *the proof length is preserved; and*
- *the new prover algorithm can be implemented by a probabilistic polynomial-time oracle machine that is given oracle access to the original prover algorithm.*

This presumes a (non-standard) *model of probabilistic polynomial-time machines that are equipped with a special device that when fed with an integer n, returns an element uniformly distributed in $[n] \stackrel{\text{def}}{=} \{1, ..., n\}$.* (See discussion in Section 1.2.)

(Note that in the standard model, where such a device is not provided, a strict probabilistic polynomial-time can select a uniformly distributed element in $[n]$ if and only if n is a power of 2.) Unfortunately, we are not aware of any perfect NIZK systems to which Theorem 2 can be applied.[3] (We note that Theorem 2 is proved by a transformation akin to the one used in the proof of Theorem 1, except that the simulation error is moved to the soundness error rather than to the completeness error; this can be done because the common reference string can be trusted by both parties.) Using different ideas, we present a super-perfect NIZK system for a set that is widely believed to be outside of $\mathcal{BPP}$.

Theorem 3 (a super-perfect NIZK for Blum Integers): *Let B denote the set of all natural numbers that are of the form $p^e q^d$ such that $p \equiv q \equiv 3 \pmod 4$ are different odd primes and $e \equiv d \equiv 1 \pmod 2$. Then, B has a super-perfect NIZK.*

We also use the idea underlying the proof of Theorem 3 for presenting a promise problem that is complete for the class of promise problems having super-perfect NIZK of perfect completeness. The yes-instances of this promise problem are circuits that generate uniform distributions and the no-instances are circuits that generate distributions that cover at most half of the relevant range. For details, see Section 5.3.

1.2 Models of PPT

As noted above, the standard model of (strict) PPT refers to machines that can only toss fair coins, and such machines cannot generate a uniform distribution over $\{1, 2, 3\}$. In contrast, one may consider non-standard models (of PPT). One such model allows the machine to sample the uniform distribution over $\{1, ..., c\}$ for some fixed (constant) integer $c \geq 2$; indeed, this is a generalization of standard model where $c = 2$. A more powerful model is one in which a PPT machine is equipped with a special device that when fed with an integer n, returns an element uniformly distributed in $[n] \stackrel{\text{def}}{=} \{1, ..., n\}$. Actually, two such models are possible:

[3] We are only aware of the perfect NIZK arguments of Groth *et al.* [16], but these are in a more liberal model that allows the common reference string to be distributed according to any efficiently sampleable distribution.

1. A model in which the PPT machine provides n in binary, which allows the machine to obtain a uniform distribution over $[n]$ also when n is exponential in the machine's input length. This is the model used in Theorem 2.
2. A model in which the PPT machine provides n in unary (i.e., as 1^n), which allows the machine to obtain a uniform distribution over $[n]$ only when n is polynomial in its input length. This is the model used in our reference to [20, 21], where this ability is used to generate a random permutation over $[n]$.

Note that the issue does not arise in case the PPT machine is allowed to fail with bounded probability (as is the case with the PPT simulators underlying the definition of perfect ZK). We note that standard expositions of perfect ZK simulators seem to refer to the non-standard model of PPT, but they can be easily converted to the standard model by implementing the said device by a machine that is allowed to fail with bounded probability.[4]

1.3 Organization

We start (Section 2) by recalling the standard definitions underlying this work. Our results regarding interactive proofs and arguments are proved in Sections 3.1 and 3.2, respectively. In particular, the proof of Theorem 1 appears in Section 3.1. Our results regarding non-interactive ZK systems appear in Sections 4 and 5. In particular, Theorem 2 is proved in Section 4 and Theorem 3 is proved in Section 5. We conclude with some open problems that arise naturally from this work (Section 6).

2 Preliminaries

In this section, we recall the standard definitions underlying this work. For more details, see [10, Chap. 4].

2.1 Interactive systems

For (randomized) interactive strategies A and B, we denote by $\langle A, B\rangle(x)$ the output of B after interacting with A on common input x. Since A and B are randomized, $\langle A, B\rangle(x)$ is a random variables. We denote by U_ℓ a random variable uniformly distributed in $\{0,1\}^\ell$.

We say that a strategy is probabilistic polynomial-time (PPT) if the total time it spends when it interacts with any other strategy on common input x is $\mathrm{poly}(|x|)$, where the total time accounts for all computations performed at all stages of the interaction (including the final generation of output). We stress that, throughout this work, PPT mean "strict PPT"; that is, there exists a

[4] One can generate the uniform distribution over $[n]$ by selecting at random a uniformly distributed $r \in [2^{\log_2 \lceil n \rceil}]$, outputting r if $r \in [n]$, and announcing failure otherwise.

polynomial p such that the running time on any ℓ-bit input is always at most $p(\ell)$, regardless of the outcome of the coin tosses.[5]

Definition 2.1 (interactive proof systems, following Goldwasser, Micali and Rackoff [15]): *Let $\epsilon_c, \epsilon_s : \mathbb{N} \to [0,1)$ such that $\epsilon_c(\ell)$ and $\epsilon_s(\ell)$ are computable in $\mathrm{poly}(\ell)$-time and $\epsilon_c(\ell) + \epsilon_s(\ell) < 1 - 1/\mathrm{poly}(\ell)$. Let P and V be interactive strategies such that V is PPT. We say that (P,V) is an* interactive proof system for a set S with completeness error ϵ_c and soundness error ϵ_s *if the following two conditions hold:*

Completeness: *For every $x \in S$, it holds that $\mathbf{Pr}[\langle P,V\rangle(x)=1] \geq 1 - \epsilon_c(|x|)$.*
Soundness: *For every $x \notin S$ and every strategy P^*, it holds that $\mathbf{Pr}[\langle P^*,V\rangle(x)=1] \leq \epsilon_s(|x|)$.*

If $\epsilon_c \equiv 0$, then the system has perfect completeness.

When we talk of interactive proof systems without specifying their errors, the reader may think of any choice (e.g., $\epsilon_c = \epsilon_s = 1/3$ or $\epsilon_c(\ell) = \epsilon_s(\ell) = \exp(-\ell)$). Recall that interactive proof systems with "average error" that is bounded away from one half (i.e., $(\epsilon_c(\ell) + \epsilon_s(\ell))/2 < 0.5 - 1/\mathrm{poly}(\ell)$) can be converted to ones with negligible error by parallel or sequential composition. Lastly, recall that in computationally-sound systems (a.k.a argument systems) the soundness condition is required to hold only with respect to cheating strategies that can be implemented by polynomial-size circuits [8].[6]

Definition 2.2 (perfect and super-perfect zero-knowledge, following Goldwasser, Micali and Rackoff [15]): *Let (P,V) be an interactive proof system for S.*

Super-Perfect ZK: *The system (P,V) is* super-perfect zero-knowledge *if for every probabilistic polynomial-time strategy V^* there exists a* (strict) *probabilistic polynomial-time algorithm A^* such that for every $x \in S$ it holds that $A^*(x)$ is distributed identically to $\langle P,V^*\rangle(x)$.*

Perfect ZK: *The system (P,V) is* perfect zero-knowledge *if for every probabilistic polynomial-time strategy V^* there exists a* (strict) *probabilistic polynomial-time algorithm A^* such that for every $x \in S$ it holds that $\mathbf{Pr}[A^*(x) = \perp] \leq 1/2$ and*

$$\mathbf{Pr}[A^*(x)=\gamma \mid A^*(x)\neq\perp] = \mathbf{Pr}[\langle P,V^*\rangle(x)=\gamma],$$

for every $\gamma \in \{0,1\}^*$.

The same definition applies to argument systems. The honest-verifier *version of these definitions make a requirement only with respect to a strategy V_{hon} that behaves like V except that it outputs its entire view of the interaction* (i.e., its internal coin tosses as well as the sequence of all messages received from P).

[5] We denote the input length by ℓ, rather than by n, in order to avoid confusion with Section 5 where n denotes a large integer (which is part of the input).

[6] Specifically, for any polynomial p, all sufficiently long $x \notin S$, and any strategy P^* that can be implemented by a circuit of size at most $p(|x|)$, it holds that $\mathbf{Pr}[\langle P^*,V\rangle(x) = 1] \leq \epsilon_s(|x|)$.

Note that the failure probability in the case of perfect ZK (i.e., $\mathbf{Pr}[A^*(x) = \perp]$) can be reduced to $\exp(-\mathrm{poly}(|x|))$ by repeated applications of the original simulator.

While Graph Isomorphism (GI) has a perfect ZK proof system [13], it is not *a priori* clear whether GI has an *honest-verifier* super-perfect ZK proof system, let alone a full-fledged super-perfect ZK system. The problem is that the simulator (even just for the honest-verifier case) needs to generate a uniformly distributed permutation of the vertices of a graph, and it is not clear whether a (strict) PPT can do such a thing. This depends on whether a PPT is only allowed to toss fair coins or is also allowed to generate uniform distributions over arbitrary domains of feasible size – see discussion in Section 1.2.

Recall that the foregoing issue (i.e., the difficulty of perfectly generating uniform distributions over arbitrary domains) does not arise in case of perfect ZK, since a machine that is allowed to fail with bounded probability can easily generate such distributions. The latter comment refers both to the various simulators (establishing the ZK feature) and to the prescribed verifier itself.

2.2 Non-interactive systems

In the non-interactive setting both parties, modeled by standard algorithms, have access to a common reference string, which may be thought of as being generated by some trusted third party.

Definition 2.3 (non-interactive zero-knowledge, following Blum, Feldman and Micali [7]): *Let $\epsilon_c, \epsilon_s : \mathbb{N} \to [0,1)$ be as in Definition 2.1, and P and V be algorithms such that V is PPT. Let ρ be a positive polynomial. We say that (P,V) is an* non-interactive proof system for a set S with completeness error ϵ_c and soundness error ϵ_s *if the following two conditions hold.*

Completeness: *For every $x \in S$, it holds that*

$$\mathbf{Pr}_{\omega \leftarrow U_{\rho(|x|)}}[V(x,\omega,P(x,\omega))=1] \geq 1 - \epsilon_c(|x|).$$

Soundness: *For every $x \notin S$ and every function P^*, it holds that*

$$\mathbf{Pr}_{\omega \leftarrow U_{\rho(|x|)}}[V(x,\omega,P^*(x,\omega))=1] \leq \epsilon_s(|x|).$$

If $\epsilon_c \equiv 0$, then the system has perfect completeness.

Super-Perfect ZK: *The system (P,V) is* super-perfect zero-knowledge *if there exists a* (strict) *probabilistic polynomial-time algorithm A such that for every $x \in S$ it holds that $A(x)$ is distributed identically to $(\omega, P(x,\omega))$, where $\omega \leftarrow U_{\rho(|x|)}$.*

Perfect ZK: *The system (P,V) is* perfect zero-knowledge *if there exists a* (strict) *probabilistic polynomial-time algorithm A such that for every $x \in S$ it holds that $\mathbf{Pr}[A(x) = \perp] \leq 1/2$ and*

$$\mathbf{Pr}[A(x)=\gamma \mid A(x) \neq \perp] = \mathbf{Pr}_{\omega \leftarrow U_{\rho(|x|)}}[(\omega, P(x,\omega))=\gamma]$$

for every $\gamma \in \{0,1\}^*$.

Note that in Definition 2.3 the common reference string is uniformly distributed in $\{0,1\}^{\rho(|x|)}$. A popular relaxation, not used here, allows the common reference string to be taken from any efficiently sampleable distribution.

3 From perfect ZK to super-perfect ZK

In Section 3.1 we prove Theorem 1, which yields super-perfect zero-knowledge *proofs* with exponentially vanishing completeness error. In Section 3.2 we obtain super-perfect zero-knowledge *arguments with perfect completeness*, while assuming the existence of perfectly binding commitment schemes.

3.1 On super-perfect ZK interactive proofs

In the following transformation we assume, without loss of generality, that $p(x) < 2^{-|x|}$ for any $x \in S$. The transformation amounts to letting the prover perform the original protocol with probability $1 - p(x)$, and abort otherwise. Of course, the verifier will reject in case the prover aborts, and so perfect completeness is lost, but this will allow a super-perfect simulation. Note that this transformation relies on the hypothesis that, on any $x \in S$, all simulators output $\perp$ with the same probability. As stated in the introduction, the transformation is due to Malka [18, Sec. 4.1], although he states it only for the case of $p \equiv 1/2$. (For sake of simplicity, we also assume, w.l.o.g., that the original prover never sends the empty string, denoted λ.)

Construction 3.1 (the transformation used for establishing Theorem 1): *Let (P,V), S and p be as in the hypothesis of Theorem 1, and suppose that A' is a simulator for any fixed PPT strategy V'* (e.g., V' may equal V or V_{hon}) *such that for every $x \in S$ it holds that $\mathbf{Pr}[A'(x) = \perp] = p(x)$. Then, on common input x, the two parties proceed as follows.*

1. *The prover invokes $A'(x)$ and sends the empty message λ if and only if $A'(x) = \perp$. In such a case, the verifier will reject.*
2. *Otherwise, the parties execute (P,V) on the common input x.*

For every PPT strategy V^*, consider the simulator A^* guaranteed by the hypothesis of Theorem 1. On input x, the corresponding new simulator (for V^*) *computes $\gamma \leftarrow A^*(x)$, outputs γ if $\gamma \neq \perp$ and $V^*(x,\lambda)$ otherwise.*

Note that the foregoing protocol preserves the soundness error of V, whereas the completeness error on input $x \in S$ increases by at most $p(x) \leq 2^{-|x|}$ (i.e., from $\epsilon_{\mathrm{c}}(|x|)$ to $p(x) + (1 - p(x)) \cdot \epsilon_{\mathrm{c}}(|x|)$). Indeed, the verifier rejects if the prover got unlucky (i.e., $A'(x)$ yields $\perp$), and so a cheating prover gains nothing by claiming that it got $\perp$. The new simulators establishes the super-perfect ZK feature, and *Theorem 1 follows*. Noting that in the case of honest-verifier ZK the condition made in Theorem 1 hold vacuously, we immediate get the following corollary.

Corollary 3.2 (honest-verifier super-perfect ZK): *Every set S that has a honest-verifier perfect ZK proof system has a honest-verifier super-perfect ZK proof system. All additional features asserted in Theorem 1 hold as well.*[7]

More importantly, applying Theorem 1 (or rather Construction 3.1) to the perfect zero-knowledge arguments of Naor *et al.* [19] (see also [10, Sec. 4.8.3]), we obtain:

Corollary 3.3 (super-perfect ZK for $\mathcal{NP}$): *Assuming the existence of* (non-uniformly strong) *one-way permutations, every set in $\mathcal{NP}$ has a* (black-box) *super-perfect ZK argument system.*[8]

As stated in the introduction, super-perfect ZK arguments (with perfect completeness) for $\mathcal{NP}$ are implicit in [20] (see [21, Prop. 4.2]), where they are only claimed to be perfect ZK. Their claim, which refers to the non-standard model of PPT (in which a machine can sample $[n]$ uniformly at cost n), is conditioned on a seemingly stronger assumption (i.e., the existence of claw-free pairs of permutations), and is established using non-black-box simulators.[9] Hence, Corollary 3.3 is incomparable to the corresponding results that can be derived from [20, 21]. On the one hand, it is stronger, since it uses the standard model of PPT, and provides black-box simulators, while relying on a seemingly weaker assumption. On the other hand, it is weaker, since it does not provide perfect completeness. Obtaining perfect completeness is the focus of Section 3.2.

3.2 On super-perfect ZK arguments with perfect completeness

Assuming the existence of perfectly binding commitment schemes, we show that certain perfect ZK proof (or argument) systems can be transformed into super-perfect ZK *arguments* with perfect completeness. The transformation refers to perfect ZK proofs (or arguments) that have simulators that can always output a perfectly random prefix of the interaction that misses only the last message (from the prover). See Condition 3 below (whereas Conditions 1 and 2 are (a strong form of) the standard requirement from perfect ZK).[10]

[7] But, again, perfect completeness is lost.

[8] Again, the derived systems have exponentially vanishing completeness error.

[9] Specifically, the perfect ZK feature of their argument system is demonstrated using Barak's (non-black-box) simulation technique [3, 4], whereas such a demonstration actually yields a super-perfect simulator. This is the case because the simulation (constructed according to Barak's technique) amounts to executing the same protocol as the honest prover, while using the verifier's program as a NP-witness to a composed statement that the honest prover proves by using an NP-witness to the actual input. The need to use the non-standard model of PPT arises because in the known proof systems (e.g., [19]) the honest prover samples uniformly sets that have size that is not a power of 2.

[10] Specifically, Condition 2 requires perfect simulation of the interaction with P in case of non-failure, which is the standard requirement of perfect ZK, whereas Condition 1 requires that failure occurs with probability exactly 1/2 (rather than at most 1/2).

Definition 3.4 (an admissible class of perfect ZK protocols): *Let (P,V) be an argument system for S, and let P_0 denote the strategy derived from P by having it abort just before sending the last message. We say that P is* admissible *if for every probabilistic polynomial-time strategy V^* there exists a* (strict) *probabilistic polynomial-time algorithm A^* such that for every $x \in S$ the following three conditions hold.*

1. $\mathbf{Pr}[A^*(x)=(1,\cdot)] = 1/2$;
2. $\mathbf{Pr}[A^*(x)=(1,\gamma) \mid A^*(x)=(1,\cdot)] = \mathbf{Pr}[\langle P,V^*\rangle(x)=\gamma]$, *for every* $\gamma \in \{0,1\}^*$.
3. $\mathbf{Pr}[A^*(x) = (0,\gamma) \mid A^*(x) = (0,\cdot)] = \mathbf{Pr}[\langle P_0,V^*\rangle(x) = \gamma]$, *for every* $\gamma \in \{0,1\}^*$.

(Indeed, we parse the output of A^* as a pair of the form $(\sigma,\gamma) \in \{0,1\} \times \{0,1\}^*$, where $\sigma = 0$ indicates a failure to simulate interaction with P.)

Note that, in addition to requiring A^* to output $\langle P_0,V^*\rangle(x)$ whenever it fails to output $\langle P,V^*\rangle(x)$ (i.e., Condition 3), we also required the failure probability to be *exactly one half* (rather than at most 1/2). The latter condition can be assumed, without loss of generality, whenever p is efficiently computable, provided that we adopt the non-standard model of PPT machines in which a machine can sample $[n]$ uniformly at cost $\mathrm{poly}(\log n)$ (as discussed in Section 1.2). Furthermore, under a weaker non-standard PPT convention, in which a machine can sample $[n]$ uniformly at cost $\mathrm{poly}(n)$, both the perfect zero-knowledge proof system for Graph Isomorphism (of [13]) and the perfect ZK argument for any set in $\mathcal{NP}$ of Naor *et al.* [19] are admissible by Definition 3.4. (In both cases, the convention is required in order to allow a PPT machine to uniformly select a permutation over a set of a size larger than 2.) Actually, the perfect ZK argument for any set in $\mathcal{NP}$ of Naor *et al.* [19] are admissible by Definition 3.4 even when using the weaker non-standard model of PPT where the machine can sample (only) the uniform distribution over $[c]$ for $c = 6$ (equiv., for $c = 3$).[11]

In the following transformation, we shall use a perfectly binding commitment scheme, denoted C. That is, we shall assume that the distributions $C(0)$ and $C(1)$ are computationally indistinguishable (by polynomial-size circuits) although they have disjoint supports.[12] Such commitment schemes can be constructed, assuming the existence of one-way permutations (see [10, Sec. 4.4.1]). We denote the commitment to value v using coins s by $C_s(v)$.

Construction 3.5 (the transformation of admissible protocols): *Let (P,V) be an argument system for S such that P is admissible by Definition 3.4. On common input x, the two parties proceeds as follows.*

[11] Indeed, the perfect ZK argument system for $\mathcal{NP}$ based on 3-Colorability requires that the prover and simulator sample a random permutation of 3 elements. Furthermore, the simulator fails with probability exactly 1/3, for every input and every probabilistic polynomial-time strategy V^*.

[12] Note that in some sources (e.g. [10, Sec. 4.4.1]) the perfect binding property of commitment schemes only requires that the supports of $C(1)$ and $C(0)$ intersect on a set of negligible size, while we require that the supports of $C(0)$ and $C(1)$ are totally disjoint.

1. *The parties execute* (P_0, V) *on common input* x*; that is, they invoke the original protocol, except that the prover does not send its last message, denoted* β*.*
2. *The parties performs a standard coin tossing protocol* (see [11, Sec. 7.4.3.1])*. Specifically, the verifier sends a commitment* $c \leftarrow C(v)$ *to a random bit* v*, the prover responds* (in the clear) *with a random bit* u*, and the verifier de-commits to the commitment* (i.e., provides (v, s) such that $c = C_s(v)$).
3. *If the verifier has de-committed improperly* (i.e., $c \neq C_s(v)$)*, then the prover sends the empty message, denoted* λ*. Otherwise, if* $u = v$ *then the prover sends* 0*, and otherwise it sends* β (where we assume, w.l.o.g, that $\beta \notin \{0, \lambda\}$).
4. *If* $u = v$ *then the verifier accepts, otherwise* (i.e., $u \neq v$) *it acts as* $V(\alpha, \beta)$*, where* α *denotes the view of* V *in the interaction with* P_0 (as conducted in Step 1).

This transformation preserves the completeness error of (P, V), but the error in the computational-soundness grows from $\epsilon_s(\ell)$ to $(1+\epsilon_s(\ell)+\mu(\ell))/2$, where μ is a negligible function. Indeed, computational-soundness is established by observing that the prover can cause the verifier to accept only if either it guessed correctly the value committed by the verifier (i.e., if $u = v$) or it could have cheated anyhow in the corresponding (P, V) interaction. To reduce the computational-soundness error, one can always use sequential repetitions, whereas using parallel repetitions does not always work (because of issues with both computational-soundness (cf., e.g., [5]) and ZK (cf., e.g., [10, Sec. 4.5.4.1])).

Turning to the super-perfect ZK feature of the resulting protocol (and assuming that V^* always de-commits properly), we rely on the simulator's ability to set $u = v$ whenever it fails in its attempt to produce a full transcript. Hence, we establish the following claim.

Claim 3.6 (super-perfect simulations): *The prover strategy described in Construction 3.5 is super-perfect zero-knowledge.*

Proof: For every potential PPT strategy V^*, let A^* denote the corresponding simulator as guaranteed by Definition 3.4. The new simulator will act as follows.

1. It invokes A^* on input x, obtaining either a full transcript or a partial transcript.
   Recall that each event happens with probability 1/2, and that a full transcript has the form (α, β), where $\beta \notin \{0, \lambda\}$ is the prover's last message. For sake of convenience, set $\beta = 0$ in the case that A^* produced a partial transcript (denoted α).[13]
2. The simulator obtains a commitment c from V^*.
   Note that c determines a unique value, denoted v, such that a proper de-commitment of c yields v. (Here we rely on the perfect binding feature of C; that is, for every c there exist at most one v such that for some s it holds that $C_s(v) = c$.)

[13] Indeed, by Definition 3.4, the output of $A^{**}(x)$ has the form $(0, \alpha)$, with probability 1/2, and $(1, \alpha \circ \beta)$ otherwise.

3. The simulator obtains the reaction of V^* to both possible $u \in \{0,1\}$ (that the verifier expects as the prover's response in Step 2); that is, it obtains $d_u \leftarrow V^*(\alpha, u)$ for both $u \in \{0,1\}$.
4. If in both cases V^* acted improperly (i.e., did not provide a valid de-commitment to c), then the simulator selects u at random in $\{0,1\}$, and outputs $V^*(\alpha, u, \lambda)$.
   (Obviously, (α, u, λ) is what V^* sees in the real interaction with the prover.)
5. If in both cases V^* de-committed properly to the value v, then the simulator outputs $V^*(\alpha, v, 0)$ if $\beta = 0$ and $V^*(\alpha, 1-v, \beta)$ otherwise.
   (Here we rely on the fact that $\mathbf{Pr}[\beta{=}0] = 1/2$, whereas in the real interaction the prover responds with $u = v$ with probability $1/2$. Hence, in the real interaction the prover will respond with $\beta = 0$ if and only if $u = v$ (and otherwise, when $u = 1 - v$, it response will be the actual message $\beta \neq 0$.)
6. If V^* de-committed properly to the value v only when fed with a single value, denoted u^*, then we distinguish two cases.
   Case of $\beta = 0$: Output $V^*(\alpha, 1-u^*, \lambda)$.
   Case of $\beta \neq 0$: Output $V^*(\alpha, u^*, \beta)$ if $u^* \neq v$ and $V^*(\alpha, u^*, 0)$ otherwise.

It may be more intuitive to restructure the cases in Step 6 as follows:

Case of $u^* = v$ (i.e., proper de-commitment in response to v only): In this case, we output $V^*(\alpha, u^*, 0)$ if $\beta \neq 0$ and $V^*(\alpha, 1-u^*, \lambda)$ otherwise (i.e., $\beta = 0$).
Equivalently, output $V^*(\alpha, u^*, 0)$ with probability $1/2$ and $V^*(\alpha, 1-u^*, \lambda)$ otherwise.
In this case, in the real interaction, the prover selects $u = u^*$ with probability $1/2$ and seeing a proper de-commitment to v, responds with 0. Otherwise (i.e., $u \neq u^*$), the prover sees an improper de-commitment and responds with λ. Hence, the foregoing output distribution matches the transcript of the real interaction.

Case of $u^* \neq v$ (i.e., proper de-commitment in response to $1-v$ only): In this case, we output $V^*(\alpha, u^*, \beta)$ if $\beta \neq 0$ and $V^*(\alpha, 1-u^*, \lambda)$ otherwise.
Similarly, in this case the prover sees a proper de-commitment with probability $1/2$ and responds with $\beta \neq 0$ in that case (and otherwise it responds with λ).

Hence, in both cases considered in Step 6, the simulator produces the same distribution as in the real interaction. The same holds also in the situations considered in Steps 4 and 5. ∎

4 From perfect NIZK to super-perfect NIZK

While Construction 3.1 is applicable also in the context of NIZK, where the condition regarding p holds vacuously (cf. Corollary 3.2), this construction does not preserve perfect completeness. Our aim here is to preserve perfect completeness, and this can be done by "transferring" the simulation attempt from the

prover (who cannot be trusted to perform it at random) to the common reference string (which is uniformly distributed by definition). Specifically, we will establish Theorem 2, which presupposed that the failure probability function $p : S \to [0, 0.5]$ is efficiently computable. Actually, we assume, without loss of generality, that $p(x) < 2^{-|x|}$ for every $x \in \{0,1\}^*$ (and not merely for $x \in S$). (Again, we assume, w.l.o.g., that the original prover never outputs the empty string λ).

Construction 4.1 (the transformation): *Let (P, V), S, A and p be as in the hypothesis of Theorem 2; and let ρ denote the length of the common reference string and ρ' denote the number of coins used by the simulator A. The new NIZK for inputs of length ℓ is as follows.*

Common random string: *An $(\rho(\ell) + \rho'(\ell))$-bit string, denoted (ω, r), where r is interpreted as an integer in $\{0, ..., 2^{\rho'(\ell)} - 1\}$.*

Prover (on input $x \in \{0,1\}^\ell$): *If $r < p(x) \cdot 2^{\rho'(\ell)}$, then the prover outputs the empty message λ. Otherwise* (i.e., $r \geq p(x) \cdot 2^{\rho'(\ell)}$)*, the prover outputs $P(x, \omega)$.*

Verifier (on input $x \in \{0,1\}^\ell$ and alleged proof y): *If $r < p(x) \cdot 2^{\rho'(\ell)}$, then the verifier accepts. Otherwise* (i.e., $r \geq p(x) \cdot 2^{\rho'(\ell)}$)*, the verifier decides according to $V(x, \omega, y)$.*

The new simulator invokes $A(x)$ obtaining the value v. If $v = \perp$, then the simulator selects uniformly $\omega \in \{0,1\}^{\rho(\ell)}$ and $r \in \{0, ..., p(x) \cdot 2^{\rho'(\ell)} - 1\}$, and outputs $((\omega, r), \lambda)$. Otherwise (i.e., $v = (\omega, y)$)*, the simulator selects uniformly $r \in \{p(x) \cdot 2^{\rho'(\ell)}, ..., 2^{\rho'(\ell)} - 1\}$, and outputs $((\omega, r), y)$.*

The completeness error of the new system on input x is upper bounded by $(1 - p(x)) \cdot \epsilon_c(|x|) \leq \epsilon_c(|x|)$, whereas the soundness error is upper bounded by $p(|x|) + (1 - p(x)) \cdot \epsilon_s(|x|) \leq \epsilon_s(|x|) + 2^{-|x|}$, where ϵ_c and ϵ_s denote the error bounds of (P, V). Note that the distribution of the verifier's view both in the actual system and in its simulation equals $((U_{\rho(\ell)}, U_{\rho'(\ell)}), Y)$, where $Y = P(x, \omega)$ if $r \geq p(x) \cdot 2^{\rho'(\ell)}$ and $Y = \lambda$ otherwise.

Recall that the construction of the new simulator relies on the ability to generate uniform distributions on the sets $[p(x) \cdot 2^{\rho'(\ell)}]$ and $[2^{\rho'(\ell)} - p(x) \cdot 2^{\rho'(\ell)}]$, which is possible in the standard PPT model only if $p(x) = 1/2$. This was not the case above, since we started by reducing the simulation error to $p(x) \leq 2^{-|x|}$, which is the reason that Theorem 2 holds only in the non-standard PPT model (as stated in it). However, if we start with $p \equiv 1/2$, then Construction 4.1 yields the following.[14]

Corollary 4.2 (super-perfect NIZK in the standard PPT model): *Let (P, V), S, A and p be as in Construction 4.1, and suppose that $p \equiv 1/2$. Further suppose*

[14] Indeed, in this case the construction can be simplified. We may use a common reference string of the form $(\omega, \sigma) \in \{0,1\}^{\rho(\ell)+1}$, have the prover output $P(x, \omega)$ if and only if $\sigma = 1$, and have the verifier accept if either $\sigma = 0$ or $V(x, \omega, y)$, where y denotes the alleged proof.

that the sum of completeness and soundness error of (P,V) *are noticeably smaller than* $1/2$*. Then,* S *has a super-perfect non-interactive zero-knowledge proof system, where the simulation is in the standard PPT model, and the completeness and soundness errors are exponentially vanishing. Furthermore, if* (P,V) *has perfect completeness, then so does the resulting system.*

Note that applying Construction 4.1 to (P,V) yields a system with completeness error $\epsilon_{\mathrm{c}}' \stackrel{\mathrm{def}}{=} \epsilon_{\mathrm{c}}/2$ and soundness error at most $\epsilon_{\mathrm{s}}' \stackrel{\mathrm{def}}{=} (1+\epsilon_{\mathrm{s}})/2$, where ϵ_{c} and ϵ_{s} denote the error bounds of (P,V). Using the assumption $\epsilon_{\mathrm{c}}(\ell)+\epsilon_{\mathrm{s}}(\ell) < 0.5 - 1/\mathrm{poly}(\ell)$, we have $\epsilon_{\mathrm{c}}'(\ell)+\epsilon_{\mathrm{s}}'(\ell) < 1-1/\mathrm{poly}(\ell)$, which allows for error reduction that yields the stated (exponentially vanishing) error bounds. Interestingly, in this case, we apply error-reduction on the resulting NIZK system (rather than on the simulator A provided for the original NIZK system).

5 A super-perfect NIZK for Blum Integers

We first recall the definition of (generalized) Blum integers.

Definition 5.1 (Blum Integers): *A natural number is called a* (generalize) Blum Integer *if it is of the form* $p^e q^d$ *such that* $p \equiv q \equiv 3 \pmod 4$ *are different odd primes and* $e \equiv d \equiv 1 \pmod 2$*. The set of Blum integers is denoted* B*.*

The following standard notations will be used extensively. For any natural number n, we let $\mathbb{Z}_n$ denote the additive group modulo n, and $\mathbb{Z}_n^*$ denote the corresponding multiplicative group.

We let $Q_n \subseteq \mathbb{Z}_n^*$ denote the set of quadratic residues modulo n, and recall the definition of the Jacobi symbol modulo n, viewed as a function $\mathtt{JS}_n : \mathbb{Z} \to \{-1,0,1\}$, and a basic fact regarding it: For a prime p, it holds that $\mathtt{JS}_p(r) = 0$ if $r \equiv 0 \pmod p$, whereas $\mathtt{JS}_p(r) = 1$ if $(r \bmod p) \in Q_p$ and $\mathtt{JS}_p(r) = -1$ otherwise (i.e., $(r \bmod p) \in \mathbb{Z}_n^* \setminus Q_p$). For composite $n = n_1 n_2$, it holds that $\mathtt{JS}_n(r) = \mathtt{JS}_{n_1}(r) \cdot \mathtt{JS}_{n_1}(r)$, yet the Jacobi symbol modulo n can be computed efficiently also when not given the factorization of n. Note that $\mathtt{JS}_n(r) = 1$ for every $r \in Q_n$.

Another important set, first utilized in [2], is $S_n \stackrel{\mathrm{def}}{=} \{r \in \{1,...,\lfloor n/2 \rfloor\} : \mathtt{JS}_n(r) = 1\} \subset \mathbb{Z}_n^*$. For $n \in B$ it holds that $|S_n| = |\mathbb{Z}_n^*|/4$ (see Claims 5.3 and 5.5). We consider the following three functions:

1. The modular squaring function $g_n : \mathbb{Z} \to Q_n$ defined as $g_n(r) = r^2 \bmod n$.
2. The "first half" function $h_n : \mathbb{Z}_n \to \mathbb{Z}_{\lfloor n/2 \rfloor}$ defined as $h_n(r) = r$ if $r < n/2$ and $h_n(r) = n - r$ otherwise. Indeed, if $n \in B$, then h_n maps Q_n to S_n.
3. Their composition $f_n = h_n \circ g_n$; that is, $f_n(r) = h_n(g_n(r))$.

Abusing notation, we extend these functions to sets in the obvious manner.

5.1 Well known facts

The following well-known facts will be used in our construction and its analysis. The reader may consider skipping this subsection. We start by recalling two computational facts.

1. The set of prime powers is in $\mathcal{P}$.
   (Justification: Try all possible powers $e \in [\lceil \log_2 n \rceil]$, and use the primality tester of [1].)
2. The set $\{(n,r) : r \in S_n\}$ is in $\mathcal{P}$.
   (Justification: Recall that the Jacobi symbol is efficiently computable.)

We next recall a few elementary facts regarding the foregoing sets and functions.

Claim 5.2 (on the size of Q_n and $f_n(S_n)$):

1. *Suppose that $n = \prod_{i\in[k]} p_i^{e_i}$ such that the p_i's are different odd primes. Then, $|Q_n| = 2^{-k} \cdot |Z_n^*|$.*
2. *For every $n \in \mathbb{N}$, it holds that $|f_n(S_n)| \leq |Q_n|$.*

Proof: Part 1 holds since $r \in Z_n^*$ is in Q_n if and only if for every $i \in [k]$ it holds that $r \bmod p_i^{e_i}$ is in $Q_{p_i^{e_i}}$, whereas each $s \in Q_{p_i^{e_i}}$ has exactly two modular square root (which sum-up to $p_i^{e_i}$). Part 2 holds since $f_n(S_n) \subseteq f_n(Z_n^*) = h_n(Q_n)$. ■

Claim 5.3 (on $\mathtt{JS}_n(-1)$ and the form of n): *Suppose that $n = \prod_{i\in[k]} p_i^{e_i}$ such that the p_i's are different odd primes, and let $I = \{i \in [k] : p_i \equiv 3 \bmod 4\}$. Then, the following three conditions are equivalent: (1) $n \equiv 1 \pmod 4$; (2) $\mathtt{JS}_n(-1) = 1$; and (3) $\sum_{i\in I} e_i$ is even.*

In particular if n is a Blum integer, then $\mathtt{JS}_n(-1) = 1$.

Proof: Note that $n \equiv \prod_{i\in I} 3^{e_i} \equiv 3^{\sum_{i\in I} e_i} \pmod 4$, which implies that $n \equiv 1 \pmod 4$ if and only if $\sum_{i\in I} e_i$ is even. On the other hand, note that $\mathtt{JS}_n(-1) = \prod_{i\in[k]} \mathtt{JS}_{p_i}(-1)^{e_i} = \prod_{i\in I}(-1)^{e_i} = (-1)^{\sum_{i\in I} e_i}$, where the second equality holds since for every odd prime p it holds that $\mathtt{JS}_p(-1) = 1$ if and only if $p \equiv 1 \pmod 4$. ■

Claim 5.4 (on f_n when n is a Blum integer): *For $n \in B$, the function f_n is a permutation over S_n.*

Proof: Recall that $n \in B$ has the form $p^e q^d$ such that $p \equiv q \equiv 3 \pmod 4$ are distinct odd primes and $e \equiv d \equiv 1 \pmod 2$. First note that g_n is a permutation over Q_n, because $x^2 \equiv y^2 \pmod n$ implies that $x \equiv \pm y \pmod{p^e}$ whereas $|Q_{p^e} \cap \{r, p^e - r\}| = 1$ for every $r \in \mathbb{Z}_n^*$ (since $\mathtt{JS}_{p^e}(-1) = -1$). Ditto for the situation mod q^d. Next note that h_n is a bijection from Q_n to S_n, because $|Q_n \cap \{r, n-r\}| \leq 1$ for any $r \in \mathbb{Z}_n^*$. The claim (restated as $f_n(S_n) = S_n$) follows since $f_n(Q_n) = h_n(g_n(Q_n)) = h_n(Q_n) = S_n$ and $f_n(S_n) = f_n(h_n(S_n)) = f_n(h_n(Q_n)) = f_n(Q_n)$, where the last equality holds since $g_n(-1) = 1$ (and so for every $x \in Z_n^*$ it holds that $g_n(h_n(x)) = g_n(x)$). ■

Claim 5.5 (on the size of S_n): *If* $\mathtt{JS}_n(-1) = 1$, *then* $|S_n| \geq |Z_n^*|/4$, *where equality holds if* n *is not of the form* $2^e s^2$ *for some* $e, s \in \mathbb{N}$.

Proof: Using $\mathtt{JS}_n(-1) = 1$ it follows that the elements of $\mathbb{Z}_n^*$ that have Jacobi symbol 1 are paired such that $\mathtt{JS}_n(s) = 1$ if and only if $\mathtt{JS}_n(n-s) = 1$. This implies that $|S_n| = |\{s \in \mathbb{Z}_n^* : \mathtt{JS}_n(s)=1\}|/2$. The latter set contains half of $|Z_n^*|$ if there exists $r \in \mathbb{Z}_n^*$ such that $\mathtt{JS}_n(r) = -1$, since in that case $x \mapsto rx$ is a bijection of $\{s \in \mathbb{Z}_n^* : \mathtt{JS}_n(s) = 1\}$ to $\{s \in \mathbb{Z}_n^* : \mathtt{JS}_n(s) = -1\}$. Lastly, note that such r exists if and only if n is not of the form $2^e s^2$ for some $e, s \in \mathbb{N}$, whereas $\mathtt{JS}_{2^e s^2}(r) = \mathtt{JS}_s(r)^2 = 1$ for every $r \in \mathbb{Z}_{2^e s^2}^*$ (and in this case $|S_{2^e s^2}| = |\mathbb{Z}_{2^e s^2}|/2$). ■

5.2 The proof system

Recall that there exist (deterministic) polynomial-time algorithms for (1) deciding if a number is a prime (ditto for a prime power), and (2) deciding whether $r \in S_n$ when given n and r. The main observation underlying the proof system is that when $n \in B$ the function f_n is a permutation over S_n, whereas for $n \notin B$ it holds that $|f_n(S_n)| \leq |S_n|/2$ (provided that $n \equiv 1 \pmod 4$ and n is not a prime power). (Establishing the claim regarding $n \notin B$ is the core of the proof of Proposition 5.7 (below).) Hence, the proof system amounts to distinguishing the case $f_n(S_n) = S_n$ from the case $|f_n(S_n)| = |S_n|/2$ by *asking the prover to provide a pre-image under* f_n *of a uniformly distributed* $\omega \in S_n$ (where the case $\omega \notin S_n$ is treated separately).[15] The super-perfect simulator can provide such transcripts by uniformly selecting $r \in S_n$, and outputting $(f_n(r), r)$, where $f_n(r)$ represents the common reference string (and r be the prover's message/output).

Construction 5.6 (a non-interactive proof system for B):

Input: *A natural number* n. *Let* $\ell = \lceil \log_2 n \rceil$.

Common reference string: *An* ℓ*-bit string, denoted* ω, *interpreted as an integer in* $\mathbb{Z}_{2^\ell}$.

Prover: *If* $\omega \in S_n$ *and there exists* $r \in S_n$ *such that* $f_n(r) = \omega$, *then the prover outputs* r (otherwise it outputs 0).
(Note that for $n \in B$ and $\omega \in S_n$, there exists a unique $r \in S_n$ such that $f_n(r) = \omega$.)

Verifier: *When receiving an alleged proof* r, *the verifier proceeds as follows.*

1. Discarding obvious no-instances: *If* n *is a prime power or* $n \not\equiv 1 \pmod 4$, *then the verifier halts outputting 0* (indicating rejection).
2. Handling inputs with a small prime factor: *The verifier checks if there exists a prime* $p \in \{3, ..., \ell\}$ *that divides* n *and finds the largest* e *such that* p^e *divides* n. *If* n/p^e *is not a prime power, then the verifier rejects. Otherwise, letting* q^d *be this prime power* (i.e., $n = p^e q^d$), *the verifier accepts if* $p \equiv q \equiv 3 \pmod 4$ *and* $e \equiv d \equiv 1 \pmod 2$, *and rejects otherwise.*

[15] See Step 3. In addition, Steps 1 and 2 take care of other pathological cases. The main action takes place in Step 4.

3. *If $\omega \notin S_n$, then the verifier halts outputting 1* (indicating acceptance).
4. *If $r \in S_n$ and $f_n(r) = \omega$, then the verifier outputs 1. Otherwise, it outputs 0.*

Note that the foregoing system has soundness error *at least* 1/2, due to Step 3. However, as shown next, its soundness error is upper-bounded by a constant smaller than 1, and so we can apply straightforward error reduction (since the system has perfect completeness).

Proposition 5.7 (analysis of Construction 5.6): *Construction 5.6 constitutes a super-perfect NIZK for B with perfect completeness and soundness error smaller than* 16/17.

Proof: Suppose that $n = p^e q^d$ such that $p \equiv q \equiv 3 \pmod 4$ are different odd primes and $e \equiv d \equiv 1 \pmod 2$. Then, f_n is a permutation over S_n (see Claim 5.4), and perfect completeness holds (since no step of Construction 5.6 may cause rejection). In such a case, the super-perfect simulation proceeds as follows.

1. Select uniformly $r \in \mathbb{Z}_{2^\ell}$.
2. If $r \in S_n$ then output $(f_n(r), r)$, else output $(r, 0)$.

Note that the simulator's output is distributed identically to the distribution produced by the prover. In both distributions of pairs, denoted (ω, y), it holds that ω is distributed uniformly in $\mathbb{Z}_{2^\ell}$, whereas y is a function of ω (and n) determined as follows: If $\omega \notin S_n$, then $y = 0$, and otherwise $y \in S_n$ is the unique pre-image of ω under f_n. Hence, it remains to establish the soundness of the system.

Turning to the soundness condition, suppose that $n \notin B$. We may assume that n is not a prime power and that $n \equiv 1 \pmod 4$ (or else Step 1 would have rejected). We may also assume that n has no prime factor smaller than ℓ (or else Step 2 would have rejected).[16] Now, with probability $n/2^\ell > 1/2$, the random string ω is in $\mathbb{Z}_n$. Conditioned on this event, we consider the prime factorization of $n = \prod_{i \in [k]} p_i^{e_i}$ (where the p_i's are different odd primes), and show that $\omega \notin \mathbb{Z}_n^*$ is unlikely, whereas if $\omega \in \mathbb{Z}_n^*$ then the verifier rejects with probability at least 1/8.

First, recall that $|\mathbb{Z}_n^*| = \prod_{i \in [k]}((p_i - 1) \cdot p_i^{e_i - 1})$. Hence

$$\begin{aligned} \frac{|\mathbb{Z}_n \setminus \mathbb{Z}_n^*|}{|\mathbb{Z}_n|} &= 1 - \prod_{i \in [k]} \frac{p_i - 1}{p_i} \\ &\leq 1 - \left(1 - \frac{1}{\ell}\right)^k \end{aligned}$$

[16] This is the case since if $n = p^e n' \notin B$ for $e \geq 1$ and an odd prime $p \in [\ell]$ that does not divide n', then either n' is not a prime power or the prime factorization of n is found in Step 2 leading the verifier to reject.

where the inequality is due to $p_i \geq \ell$ for every $i \in [k]$. Hence, $\mathbf{Pr}_\omega[\omega \notin \mathbb{Z}_n^* \,|\, \omega \in \mathbb{Z}_n] \leq 1 - (1-\ell^{-1})^k < k/\ell = o(1)$, since $k \leq \log_\ell n = o(\ell)$. Considering the case that $\omega \in \mathbb{Z}_n^*$, and using the fact that $|S_n| \geq |\mathbb{Z}_n^*|/4$ (see Claims 5.3 and 5.5), we infer that $\omega \in S_n$ with probability

$$\begin{aligned}\frac{|S_n|}{2^\ell} &= \frac{n}{2^\ell} \cdot \frac{|Z_n^*|}{|Z_n|} \cdot \frac{|S_n|}{|Z_n|} \\ &> \frac{1}{2} \cdot (1 - o(1)) \cdot \frac{1}{4}\end{aligned}$$

which is larger than 2/17. Hence, the verifier executes Step 4 with probability greater than 2/17.

Recalling that $|f_n(S_n)| \leq 2^{-k} \cdot |\mathbb{Z}_n^*|$ (see Claim 5.2) while $|S_n| = |\mathbb{Z}_n^*|/4$, we infer that if $k \geq 3$, then Step 4 rejects with probability at least half. We are left with the case of $k = 2$, which means that $n = p^e q^d \notin B$ such that p and q are different odd primes (and $e, d \geq 1$). Hence, w.l.o.g., either $p \equiv 1 \pmod 4$ or $e \equiv 0 \pmod 2$, and $p^e \equiv 1 \pmod 4$ follows in both cases. We shall show that in this case $|f_n(S_n)| \leq |S_n|/2$, by showing that for each $r \in S_n$ there exists $r' \in S_n$ such that $r' \neq r$ and $f_n(r') = f_n(r)$.

For any $r \in S_n$, let $r_1 = r \bmod p^e$ and $r_2 = r \bmod q^d$. Consider the unique $s \in \mathbb{Z}_n^*$ such that $s \equiv -r_1 \pmod{p^e}$ and $s \equiv r_2 \pmod{q^d}$. Then, $s \neq r$ and $s \neq n - r$, whereas $s^2 \equiv r^2 \pmod n$, which implies $f_n(s) = f_n(r)$. On the other hand, $\mathtt{JS}_n(n-s) = \mathtt{JS}_n(s) = \mathtt{JS}_{p^e}(-r_1) \cdot \mathtt{JS}_{q^d}(r_2) = \mathtt{JS}_{p^e}(r_1) \cdot \mathtt{JS}_{q^d}(r_2) = \mathtt{JS}_n(r) = 1$, where the first equality uses $\mathtt{JS}_n(-1) = 1$ (which holds by $n \equiv 1 \pmod 4$ and Claim 5.3), the third equality uses $\mathtt{JS}_{p^e}(-1) = 1$ (which holds by $p^e \equiv 1 \pmod 4$ and Claim 5.3), and the last equality uses $r \in S_n$. Hence, either s or $n - s$ is in S_n (since $\mathtt{JS}_n(n-s) = \mathtt{JS}_n(s) = 1$), and it follows that $|\{r, s, n-s\} \cap S_n| \geq 2$. Having shown for each $r \in S_n$ there exists $r' \in S_n$ such that $r' \neq r$ and $f_n(r') = f_n(r)$, we conclude that $|f_n(S_n)| \leq |S_n|/2$.

Let us recap. If $n \notin B$, then the verifier reject with probability at least

$$\mathbf{Pr}_\omega[\omega \in S_n] \cdot \mathbf{Pr}_\omega[\omega \notin f(S_n) | \omega \in S_n] > \frac{2}{17} \cdot \frac{1}{2} = \frac{1}{17}$$

and the proposition follows. ■

5.3 A complete promise problem

Following Sahai and Vadhan [22], who identified promise problems that are complete for the class of promise problems that has statistical zero-knowledge proof systems, analogous results were obtained for statistical NIZK proof systems (see [14]) and perfect NIZK proof systems (see [18]). Following Malka [18, Sec. 2], we identify a very natural promise problem that is complete for super-perfect NIZK proof systems with perfect completeness. The promise problem is defined next.

Definition 5.8 (the promise problem $(\mathtt{U}_{\mathrm{yes}}, \mathtt{U}_{\mathrm{no}})$):

- *The set* $\mathtt{U}_{\mathrm{yes}}$ *consists of all circuits* $C : \{0,1\}^\ell \to \{0,1\}^m$ *such that* $C(U_\ell)$ *is distributed identically to* U_m.
- *The set* $\mathtt{U}_{\mathrm{no}}$ *consists of all circuits* $C : \{0,1\}^\ell \to \{0,1\}^m$ *such that the support of* $C(U_\ell)$ *has size at most* 2^{m-1}.

We assume that the circuits are given in a format in which it is easy to determine the number of bits in their inputs and in their outputs. We comment that the promise problem considered by Malka [18, Def. 2.2] is related but different (i.e., it required that for a yes-instance C it holds that $C(U_\ell)_{[m-1]} \equiv U_{m-1}$ and $\mathbf{Pr}[C(U_\ell)_m = 1] \geq 2/3$, whereas for a no-instance $\mathbf{Pr}[C(U_\ell)_m = 1] \leq 1/3$).

The definition of super-perfect NIZK proof systems extend naturally to promise problem (cf. [23]). Loosely speaking, a promise problem $(\Pi_{\mathrm{yes}}, \Pi_{\mathrm{no}})$ is a pair of non-intersecting sets, and the soundness condition refers only to inputs in Π_{no} (rather than to inputs in $\{0,1\}^* \setminus \Pi_{\mathrm{yes}}$). (The completeness and zero-knowledge conditions refer to all inputs in Π_{yes}.)

Theorem 5.9 ($(\mathtt{U}_{\mathrm{yes}}, \mathtt{U}_{\mathrm{no}})$ is complete for $\mathcal{SPNIZK}_1$): *Let* $\mathcal{SPNIZK}_1$ *denote the class of promise problems having a super-perfect NIZK proof system of perfect completeness. Then,* $(\mathtt{U}_{\mathrm{yes}}, \mathtt{U}_{\mathrm{no}})$ *is in* $\mathcal{SPNIZK}_1$ *and every problem in* $\mathcal{SPNIZK}_1$ *is Karp-reducible to* $(\mathtt{U}_{\mathrm{yes}}, \mathtt{U}_{\mathrm{no}})$.

Proof: The idea underlying the proof of Theorem 3 (presented in Section 5.2) can be used to present a super-perfect NIZK proof system of perfect completeness for $(\mathtt{U}_{\mathrm{yes}}, \mathtt{U}_{\mathrm{no}})$. Specifically, on input a circuit $C : \{0,1\}^\ell \to \{0,1\}^m$ and common reference string $\omega \in \{0,1\}^m$, the prover outputs a uniformly distributed string $r \in C^{-1}(\omega)$, and the verifier accepts if and only if $C(r) = \omega$. Perfect completeness and soundness error of $1/2$ are immediate by the definition of $(\mathtt{U}_{\mathrm{yes}}, \mathtt{U}_{\mathrm{no}})$, whereas super-perfect ZK is demonstrated by a simulator that uniformly selects $r \in \{0,1\}^\ell$ and outputs $(C(r), r)$, where $C(r)$ represents the simulated common reference string and r represents the simulated proof output by the prover on input C (and common reference string $C(r)$).

Assuming that $(\Pi_{\mathrm{yes}}, \Pi_{\mathrm{no}}) \in \mathcal{SPNIZK}_1$, we show a Karp-reduction of $(\Pi_{\mathrm{yes}}, \Pi_{\mathrm{no}})$ to $(\mathtt{U}_{\mathrm{yes}}, \mathtt{U}_{\mathrm{no}})$. Let (P, V) be the non-interactive proof systems of $(\Pi_{\mathrm{yes}}, \Pi_{\mathrm{no}})$, and A be the corresponding simulation. Let $\ell = \ell(|x|)$ denote the number of coin tosses used by A on input x, and m denote the length of the common reference string. (For sake of simplicity, we assume, without loss of generality, that V is deterministic and that the soundness error of (P, V) is at most $0.5 - 2^{-m}$, where the probability is taken over all possible choices of the common reference string.) Now, on input x, the reduction produces the following circuit $C_x : \{0,1\}^\ell \to \{0,1\}^m$.

1. On input $r \in \{0,1\}^\ell$, the circuit C_x invokes A on input x and coins r, and obtains the outcome (ω, y), where ω represents the simulated common reference string and y represents the simulated proof output by P on input x (and common reference string ω).
2. The circuit C_x outputs ω if $V(x, \omega, y) = 1$ and 0^m otherwise.

Observe that if $x \in \Pi_{\text{yes}}$ then $C_x(U_\ell) \equiv U_m$, whereas if $x \in \Pi_{\text{no}}$ then the support of $C_x(U_\ell)$ contains at most $(0.5 - 2^{-m}) \cdot 2^m + 1 = 2^{m-1}$ strings, where the extra unit is due to the possible case that $V(x, 0^m, P(x, 0^m)) \neq 1$. The claim follows. ■

6 Open Problems

All the following problems refer to ZK proof systems (rather than to ZK argument systems). For a wider perspective, we start with a well-known open problem regarding perfect ZK (see, e.g., [23, Chap. 8]).

Open Problem 6.1 (perfect ZK versus statistical ZK): *Let $\mathcal{SZK}$ be the class of sets having statistical* (a.k.a almost-perfect) *zero-knowledge interactive proof system. Prove or disprove, under reasonable assumptions, the conjecture by which not all sets in $\mathcal{SZK}$ have perfect zero-knowledge interactive proof systems. Ditto for having a perfect zero-knowledge proof system with perfect completeness.*

Recall that any set in $\mathcal{SZK}$ has a statistical zero-knowledge proof system with perfect completeness (via the transformation to public-coin systems and the use of Lautemann's technique [17]; see [23, Chap. 5] and [9], resp.). This is not known to be the case for perfect zero-knowledge. In particular, it is even not known whether all sets in $\mathcal{BPP}$ have perfect zero-knowledge proof systems with perfect completeness. (Indeed, all sets in co$\mathcal{RP}$ do have perfect zero-knowledge proof systems with perfect completeness, in which the prover remains silent.) Turning to the subject-matter of this work (i.e., super-perfect ZK), we ask:

Open Problem 6.2 (super-perfect ZK versus perfect ZK): *Let $\mathcal{PZK}$ be the class of sets having perfect zero-knowledge interactive proof system. Prove or disprove, under reasonable assumptions, the conjecture by which not all sets in $\mathcal{PZK}$ have super-perfect zero-knowledge interactive proof systems. Ditto for zero-knowledge proof systems with perfect completeness, where the question may refer both to the standard and non-standard models of PPT machines.*

Recall that the difference between the standard and non-standard models of PPT machines arises only with respect to super-perfect ZK. Indeed, one may ask the following

Open Problem 6.3 (super-perfect ZK: models of PPT): *Let S be a set having a super-perfect zero-knowledge interactive proof systems with perfect completeness under one of the two non-standard models of PPT machines. Does S necessarily has a super-perfect zero-knowledge interactive proof systems with perfect completeness under the standard model of PPT machines. Ditto for zero-knowledge proof system with non-perfect completeness.*

It is tempting to think that the question regarding non-perfect completeness can be resolved by applying Theorem 1, but this presumes that all (super-perfect ZK)

simulators use their distribution generating device in the same manner (i.e., with the same n's and for the same number of times).[17] The same questions arise with respect to NIZK.

Open Problem 6.4 (super-perfect NIZK versus perfect and statistical NIZK): *Address the non-interactive zero-knowledge analogues of Problems 6.1 and 6.2. Ditto for the perfect completeness version of Problem 6.3.*

Acknowledgments

We are grateful to Alon Rosen and Amit Sahai for useful discussions. This research was partially supported by the Minerva Foundation with funds from the Federal German Ministry for Education and Research.

References

1. M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. *Annals of Mathematics*, Vol. 160 (2), pages 781–793, 2004.
2. W. Alexi, B. Chor, O. Goldreich and C.P. Schnorr. RSA/Rabin Functions: Certain Parts are As Hard As the Whole. *SIAM Journal on Computing*, Vol. 17, April 1988, pages 194–209.
3. B. Barak. How to Go Beyond the Black-Box Simulation Barrier. In *42nd IEEE Symposium on Foundations of Computer Science*, pages 106–115, 2001.
4. B. Barak. Non-Black-Box Techniques in Crypptography. PhD Thesis, Weizmann Institute of Science, 2004.
5. M. Bellare, R. Impagliazzo and M. Naor. Does Parallel Repetition Lower the Error in Computationally Sound Protocols? In *38th IEEE Symposium on Foundations of Computer Science*, pages 374–383, 1997.
6. M. Blum, A. De Santis, S. Micali, and G. Persiano. Non-Interactive Zero-Knowledge Proof Systems. *SIAM Journal on Computing*, Vol. 20, No. 6, pages 1084–1118, 1991. (Considered the journal version of [7].)
7. M. Blum, P. Feldman and S. Micali. Non-Interactive Zero-Knowledge and its Applications. In *20th ACM Symposium on the Theory of Computing*, pages 103–112, 1988. See [6].
8. G. Brassard, D. Chaum and C. Crépeau. Minimum Disclosure Proofs of Knowledge. *Journal of Computer and System Science*, Vol. 37, No. 2, pages 156–189, 1988. Preliminary version by Brassard and Crépeau in *27th FOCS*, 1986.
9. M. Fürer, O. Goldreich, Y. Mansour, M. Sipser, and S. Zachos. On Completeness and Soundness in Interactive Proof Systems. *Advances in Computing Research: a research annual*, Vol. 5 (Randomness and Computation, S. Micali, ed.), pages 429–442, 1989.
10. O. Goldreich. *Foundation of Cryptography: Basic Tools*. Cambridge University Press, 2001.

[17] This presumption holds trivially when referring either to the honest-verifier version or to the NIZK version.

11. O. Goldreich. *Foundation of Cryptography: Basic Applications*. Cambridge University Press, 2004.
12. O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.
13. O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *Journal of the ACM*, Vol. 38, No. 3, pages 691–729, 1991. Preliminary version in *27th FOCS*, 1986.
14. O. Goldreich, A. Sahai, and S.P. Vadhan. Can Statistical Zero Knowledge Be Made Non-interactive? or On the Relationship of SZK and NISZK. In *Crypto99*, pages 467–484, 1999.
15. S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing*, Vol. 18, pages 186–208, 1989. Preliminary version in *17th STOC*, 1985. Earlier versions date to 1982.
16. J. Groth, R. Ostrovsky, and A. Sahai. Perfect Non-interactive Zero Knowledge for NP. In *25th Eurocrypt*, Springer Lecture Notes in Computer Science (Vol. 4004), pages 339–358, 2006.
17. C. Lautemann. BPP and the Polynomial Hierarchy. *Information Processing Letters*, Vol. 17, pages 215–217, 1983.
18. L. Malka. How to Achieve Perfect Simulation and A Complete Problem for Non-interactive Perfect Zero-Knowledge. In *5th TCC*, Springer Lecture Notes in Computer Science (Vol. 4948), pages 89–106, 2008.
19. M. Naor, R. Ostrovsky, R. Venkatesan and M. Yung. Zero-Knowledge Arguments for NP can be Based on General Assumptions. *Journal of Cryptology*, Vol. 11, pages 87–108, 1998. Preliminary version in *Crypto92*.
20. R. Pass and A. Rosen. New and improved constructions of non-malleable cryptographic protocols. *SIAM Journal on Computing*, Vol. 38 (2), pages 702–752, 2008.
21. R. Pass and A. Rosen. Concurrent Non-Malleable Commitments. *SIAM Journal on Computing*, Vol. 37 (6), pages 1891–1925, 2008.
22. A. Sahai and S. Vadhan. A Complete Promise Problem for Statistical Zero-Knowledge. *Journal of the ACM*, Vol. 50 (2), pages 196–249, 2003. Preliminary version in *38th FOCS*, 1997.
23. S. Vadhan. A Study of Statistical Zero-Knowledge Proofs. PhD Thesis, Department of Mathematics, MIT, 1999.
See `http://people.seas.harvard.edu/~salil/research/phdthesis.pdf`