Honest-Verifier Statistical Zero-Knowledge Equals General Statistical Zero-Knowledge

Oded Goldreich* Amit Sahai† Salil Vadhan‡

Abstract

We show how to transform any interactive proof system which is statistical zero-knowledge with respect to the honest-verifier, into a proof system which is statistical zero-knowledge with respect to any verifier. This is done by limiting the behavior of potentially cheating verifiers, *without using computational assumptions or even referring to the complexity of such verifier strategies*. (Previous transformations have either relied on computational assumptions or were applicable only to constant-round public-coin proof systems.)

Our transformation also applies to public-coin (aka Arthur-Merlin) computational zero-knowledge proofs: We transform any Arthur-Merlin proof system which is computational zero-knowledge with respect to the honest-verifier, into an Arthur-Merlin proof system which is computational zero-knowledge with respect to any probabilistic polynomial-time verifier.

A crucial ingredient in our analysis is a new lemma regarding 2-universal hashing functions.

1 Introduction

Zero-Knowledge proofs, introduced by Goldwasser, Micali and Rackoff [GMR89], are fascinating and extremely useful constructs. Their fascinating nature is due to their seemingly contradictory nature; they are both convincing and yet yield nothing beyond the validity of the assertion being proven. Their applicability in the domain of cryptography is vast; they are typically used to force malicious parties to behave according to a predetermined protocol (which requires parties to provide proofs of the correctness of their secret-based actions without revealing these secrets).

Zero-knowledge proofs come in many flavors. Arguably, the most important parameters refer to the strength of the zero-knowledge (or simulability) condition. These are captured by two parameters: The first parameter is the *type* of adversary which is supposed to learn nothing while verifying an assertion. The simplest type is a *honest-verifier*; that is, one which follows the protocol (and ends up with the transcript of the interaction). Zero-knowledge with respect to an honest-verifier is already a fascinating notion from a conceptual as well as a complexity-theoretic point of view. However, cryptographic applications typically require robustness against arbitrary (or arbitrary feasible) behavior which typically deviates from the protocol. This is the general (or standard) notion of zero-knowledge. A major open problem in the area is *whether honest-verifier zero-knowledge equals general zero-knowledge.* A positive answer to this question may also lead the way to a useful methodology: First construct a honest-verifier zero-knowledge proof to the problem at hand, and next transform it to a general zero-knowledge proof. To describe our contribution to the above open problem, we need first to discuss a second major parameter of the zero-knowledge framework – the *notion of learning nothing*.

The requirement that the *verifier learns nothing* from the proof is formulated by saying that the transcript of its interaction with the prover can be *simulated* by the verifier itself. That is, there exists an efficient procedure than on input a valid assertion produces a distribution which is "similar" to the distribution of transcripts of the executions of the proof system on that assertion. The key parameter is the interpretation of "similarity". Three notions have been commonly considered in the literature (cf., [GMR89, For89]). Perfect Zero-Knowledge (PZK) requires that the two distributions be identical. *Statistical Zero-Knowledge* (SZK) requires that these distributions be statistically close (i.e., the variation distance between them is negligible). Finally, Computational Zero-Knowledge (CZK) refers to the case that these distributions are computationally indistinguishable (cf., [GM84, Yao82]).

Assuming the existence of one-way functions, any language which has an interactive proof, has also a *Computational* Zero-Knowledge one (cf., [GMW91, IY87, BGG+88]). Thus, assuming the existence of one-way functions, the above problem (i.e., of honest-verifier ZK versus general ZK) is long resolved for the case of Computational Zero-Knowledge. Still, it is open whether one can prove that honest-verifier CZK equals general CZK, *without assuming* the existence of one-way functions. We resolve this problem for the special case of public-coin (aka Arthur-Merlin) proof systems –

Theorem 1 *Every language having an Honest-Verifier Computational Zero-Knowledge public-coin proof system, also has a general Computational Zero-Knowledge* (public-coin) *proof system.*

We note that it is known that the existence of honest-verifier CZK for languages outside BPP yields a *weak* form of one-way functions [OW93]. However, this weak form of one-way functions does NOT seem to suffice for constructing general CZK proofs for the same language (in general).

*Department of Computer Science, Weizmann Institute of Science, Rehovot, ISRAEL. E-mail: oded@wisdom.weizmann.ac.il. Work done while visiting LCS, MIT. Supported by DARPA grant DABT63-96-C-0018.

†Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139. E-mail: amits@theory.lcs.mit.edu. Supported by an NDSEG/DOD Graduate Fellowship and partially by DARPA grant DABT63-96-C-0018.

‡Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139. E-mail: salil@math.mit.edu. Supported by an NDSEG/DOD Graduate Fellowship and partially by DARPA grant DABT63-96-C-0018.

The main focus of this paper is the honest-verifier versus general verifier problem for *Statistical* Zero-Knowledge. We fully resolve the problem in this case –

Theorem 2 *Every language having an Honest-Verifier Statistical Zero-Knowledge proof system, also has a general* (public-coin) *Statistical Zero-Knowledge proof.*

Results of similar nature were previously achieved under intractability assumptions (cf., [BMO90, OVY93, Oka96]). A weaker unconditional result was claimed in [DOY97]. All these are discussed in detail below. But first we need to be somewhat more precise about the notions and issues discussed above.

1.1 Formal Setting

The basic notions of interactive proofs [GMR89] are recalled in Appendix A. Throughout this subsection we fix a language L, and an interactive proof system, (P, V), for it.[1] The basic paradigm of zero-knowledge is that for every verifier of a certain class, there should be an efficient non-interactive machine, called the simulator, which is able to "simulate well" the view of the verifier in real interactions with the prescribed prover (i.e., P). The two main issues we consider are (1) which verifiers should be simulated, and (2) the quality of simulation.

Which verifiers should be simulated (or honest-verifier versus general zero-knowledge): The two standard classes are the class consisting merely of the prescribed verifier V (aka the honest-verifier), and the class consisting of all probabilistic polynomial-time interactive machines (i.e., feasible cheating strategies for the verifier).

For the case of statistical zero-knowledge, we will consider even a wider (in fact the widest possible) class – the class of all possible verifier strategies (including non-computable ones). This will make our result even stronger. But how can an efficient machine (i.e., the simulator) simulate the behavior (let alone interaction) of a non-computable verifier strategy? The clue is the familiar notion of a reduction, captured in this context by the notion of a *black-box simulator*. The latter is a probabilistic polynomial-time oracle machine which is given oracle access to the verifier strategy.[2] We comment that the notion of black-box simulation was considered before for other reasons (cf., [GO94, GK96]).

The quality of simulation (or SZK versus CZK): When defining *statistical zero-knowledge* (w.r.t. a class of verifiers), one requires that for every verifier, V^*, in the class there exists an efficient simulator, S^*, such that the following two distribution ensembles are statistically close (i.e., the variation distance is eventually smaller than $1/p(|x|)$ for every positive polynomial p):

1. $\{(P, V^*)(x) : x \in L\}$, where $(P, V^*)(x)$ denotes the view of V^* when interacting with P on common input x. Recall that this view consists of x, all internal coin tosses of V^*, and all messages received from P.
2. $\{S^*(x) : x \in L\}$.

The variation distance between the two distribution ensembles is called the *simulator deviation* . In case there exists a black-box simulator, denoted S, the second distribution ensemble is $\{S^{V^*}(x) : x \in L\}$, where $S^{V^*}(x)$ denotes the output distribution of S on input x and oracle access to V^*.

When defining *computational zero-knowledge* (with respect to a class of verifiers), one instead requires that the two distributions above are computationally indistinguishable (cf., [GM84, Yao82]). That is, for every probabilistic polynomial-time algorithm, D, the following quantity is negligible (i.e., is eventually smaller than $1/p(|x|)$ for every polynomial p):

$$|\Pr(D((P, V^*)(x)) = 1) \; - \; \Pr(D(S^{V^*}(x)) = 1)|$$

In our definitions of zero-knowledge, we require that the simulators run in strict polynomial-time, as in [Gol95].

Notations: Let $\mathcal{HVSZK}$ (resp., $\mathcal{SZK}$) denote the class of languages having interactive proofs which are statistical zero-knowledge with respect to the honest-verifier (resp., with respect to any probabilistic polynomial-time verifier). The classes $\mathcal{HVCZK}$ and $\mathcal{CZK}$ are defined analogously for computational zero-knowledge.

Public-coin (or Arthur-Merlin) proof systems. As we refer to this notion, let us recall that *public-coin proof systems* are interactive proof systems in which the prescribed verifier's strategy amounts to the following: In each round, the verifier tosses a predetermined number of coins and sends the outcome to the prover, and at the end it decides whether to accept by applying a predicate to the (full) sequence of messages it has sent and received. For each of the classes $\mathcal{C}$ above, we denote by $\mathcal{C}|_{\mathrm{AM}}$ the subclass of *public coin* (or Arthur-Merlin) proof systems having the corresponding zero-knowledge property.

1.2 Previous work

Clearly, $\mathcal{SZK} \subseteq \mathcal{HVSZK}$ (resp., $\mathcal{CZK} \subseteq \mathcal{HVCZK}$), $\mathcal{BPP} \subseteq \mathcal{SZK} \subseteq \mathcal{CZK}$ (resp., $\mathcal{HVSZK} \subseteq \mathcal{HVCZK}$), and $\mathcal{C}|_{\mathrm{AM}} \subseteq \mathcal{C} \subseteq \mathcal{IP}$ for each of these four ZK classes.

1.2.1 On the complexity of various ZK classes.

The situation with respect to computational ZK is as follows.

Positive for CZK: *Assuming the existence of one-way functions*, $\mathcal{CZK}|_{\mathrm{AM}} = \mathcal{IP}$ (cf., [GMW91, IY87, BGG+88, HILL, Nao91]), and so under this assumption the status of all computational zero-knowledge classes is resolved.

"Negative" for CZK: If one-way functions do not exist then only "easy on the average languages" have honest-verifier (computational) zero-knowledge proofs [OW93]. This result *almost* complements the positive result above.

Open for CZK: Does $\mathcal{HVCZK} = \mathcal{CZK}$ hold unconditionally? (Or put otherwise, can it be proven without assuming the existence of one-way functions?)

Recall, this paper resolves this open problem for the case of public-coin proof systems; that is, we show that $\mathcal{HVCZK}|_{\mathrm{AM}} = \mathcal{CZK}|_{\mathrm{AM}}$. As for statistical ZK we have

Positive for SZK: Several computational problems, believed to be hard, are known to have statistical zero-knowledge proof systems; for example, Quadratic Residuosity [GMR89], Graph Isomorphism [GMW91], a problem equivalent to the Discrete Logarithm Problem [GK93], Statistical Difference [SV97], and a gap promise problem for lattices [GG98].

Negative for SZK: $\mathcal{HVSZK} \subseteq \mathcal{AM} \cap \mathrm{co}\mathcal{AM}$ [For89, AH87].

[1] All our results extend also to promise problems.

[2] That is, assuming deterministic strategies, each query is parsed as a sequence of prover messages representing a prefix of the interaction, and the answer is the response of this verifier strategy to such a prefix. Probabilistic verifier strategies are considered by first randomly selecting and fixing a deterministic strategy, and then proceeding as above.

Inside HVSZK: A key result regarding SZK is that any honest-verifier statistical zero-knowledge proof can be transformed into one using only public-coins [Oka96]. That is, $\mathcal{HVSZK} = \mathcal{HVSZK}|_{\rm AM}$. It is also known that $\mathcal{HVSZK}$ is closed under complement [Oka96, SV97].

Open for SZK: Does $\mathcal{HVSZK} = \mathcal{SZK}$ hold?

Recall, this paper resolves this open problem, showing that $\mathcal{HVSZK} = \mathcal{SZK}$ (and in fact $\mathcal{HVSZK} = \mathcal{SZK}|_{\rm AM}$).

1.2.2 Previous transformation of honest-verifier to general ZK

Conditional results for SZK: The problem of relating $\mathcal{HVSZK}$ to $\mathcal{SZK}$ was first studied in [BMO90]. They showed that the two classes coincide, provided that the Discrete Logarithm Problem is hard. At the time, it seemed puzzling that computational assumptions can be used in the supposedly "information theoretic" context of statistical zero-knowledge. However, a careful examination reveals that the standard class $\mathcal{SZK}$ does refer to computational limitations: It is required to simulate only all probabilistic polynomial-time verifiers. The computational assumption is thus used to restrict the behavior of cheating verifiers. This approach was carried to its climax in [Oka96] (cf., [DGOW95, Part 2]): Using any bit commitment scheme (and thus any one-way function [HILL, Nao91]) it was shown that $\mathcal{HVSZK}|_{\rm AM} = \mathcal{SZK}|_{\rm AM}$. Combined with the $\mathcal{HVSZK} = \mathcal{HVSZK}|_{\rm AM}$ result cited above, one gets that the existence of one-way functions implies $\mathcal{HVSZK} = \mathcal{SZK}$ (and in fact $\mathcal{HVSZK} = \mathcal{SZK}|_{\rm AM}$).

Unconditional results for constant-round ZK: The only unconditional transformations of honest-verifier SZK (resp., CZK) known before, referred to the class of *constant-round public-coin* proof systems (cf., [Dam94, DGW94]). It was shown that if L has a HVSZK (resp., HVCZK) public-coin proof system of a constant number of rounds then $L \in \mathcal{SZK}|_{\rm AM}$ (resp., $L \in \mathcal{CZK}|_{\rm AM}$).

Weak SZK: In [DOY97] it is claimed that any language in $\mathcal{HVSZK}$ has an interactive proof, (P, V), with the following non-standard statistical zero-knowledge property: For every positive polynomial p, and every probabilistic polynomial-time verifier V^*, there exists a probabilistic polynomial-time simulator S_p^* (with running-time depending on p) so that the variation distance between the probability ensembles, $\{(P, V^*)(x) : x \in L\}$ and $\{S_p^*(x) : x \in L\}$, is at most $1/p(|x|)$.[3]

1.3 Restating our results

We obtain the first unconditional general transformation of honest-verifier zero-knowledge to general zero-knowledge.

Theorem 3 (main result): *There exists an efficient transformation of Honest-Verifier Statistical* (resp., Computational) *Zero-Knowledge* public-coin *proof systems, into general Statistical* (resp., Computational) *Zero-Knowledge* public-coin *proof systems. Furthermore,*

1. *The resulting proof systems has twice as many rounds as the original one.*
2. *The resulting prover strategy can be implemented in probabilistic polynomail-time given oracle access to the original prover strategy.*
3. *The completeness error of the resulting proof system is exponentially vanishing. In case the original proof system has perfect completeness, so does the resulting one.*
4. *The soundness error of the resulting proof system is bounded above by $1/p(|x|)$, where p is an arbitrary polynomial determined by the transformation.*
5. *The resulting proof system has a black-box zero-knowledge simulator.*
6. *In case of Statistical Zero-Knowledge, the resulting simulator is strong (i.e., it can handle arbitrary verifier strategies), and its simulation error is at most $\mathrm{poly}(|x|) \cdot \epsilon(x) + 2^{-\Omega(|x|)}$, where $\epsilon(x)$ is the simulation error of the original system.*

Theorems 1 and 2 follow, where in case of Statistical Zero-Knowledge we use Okamoto's result by which $\mathcal{HVSZK} = \mathcal{HVSZK}|_{\rm AM}$ [Oka96, Thm. 1].

We stress that, in contrast to the previously mentioned conditional results, our result for (unbounded) *statistical* zero-knowledge is unconditional and guarantees (black-box) simulation of all possible verifier strategies (not only polynomial-time ones). Theorem 3 also provides a transformation for a wide class of *computational* zero-knowledge proof systems – that is, the class of public-coin proof systems. We view our result as a significant step towards showing that $\mathcal{HVCZK} = \mathcal{CZK}$ without relying on any intractability assumptions.

Soundness error and number of rounds: The transformation of Theorem 3 increases the number of rounds of the original proof system only by a factor of 2. However, the resulting protocol has noticeable soundness error. That is, for any positive polynomial p, we can achieve a soundness error of $1/p(|x|)$. The soundness error may be further decreased, while preserving the zero-knowledge property, by *sequential* repetition of the proof system. In particular, to achieve negligible soundness error it suffices to use $\omega(1)$ sequential repetitions. This is unavoidable, unless $\mathcal{NP} \subseteq \mathcal{BPP}$, since only $\mathcal{BPP}$ languages may have black-box simulation zero-knowledge public-coin proofs with constant number of rounds and negligible error probability [GK96].[4]

Completeness error: By first applying the transformation of [FGM$^+$89], we may eliminate completeness error altogether (at the cost of at most one additional round and not preserving the complexity of the prover). (Recall that the transformation of [FGM$^+$89] increases the simulation error by at most an exponentially vanishing amount.)

Corollaries: Many known results regarding the class $\mathcal{HVSZK}$ translate to the class $\mathcal{SZK}$ (and respectively results for $\mathcal{HVCZK}|_{\rm AM}$ translate to $\mathcal{CZK}|_{\rm AM}$). For example, using known results regarding $\mathcal{HVSZK}$, one obtains that $\mathcal{SZK}$ is closed under complement, equals $\mathcal{SZK}|_{\rm AM}$, has a complete promise problem, etc. A somewhat less straightforward corollary is the following.

Corollary 4 *Every language in $\mathcal{SZK}$ has a SZK proof system with perfect completeness in which the soundness error and the simulation deviation are exponentially vanishing.*

Given Theorem 3 (and the discussion above), the only non-obvious part in Corollary 4 is the claim about the simulation error. Here we rely on the result of [SV97] by which every language in $\mathcal{HVSZK}$ has a 1-round interactive proof system for which the honest-verifier can be simulated with exponentially vanishing simulation error. We

[3] The first author was unable to verify the claims and arguments given in [DOY97].

[4] Recall that if one-way functions exist then $\mathcal{NP}$ has constant-round public-coin proofs with negligible soundness error which are honest-verifier computational zero-knowledge [GMW91]. So, if Theorem 3 were to preserve all its features while resulting in a proof system with negligible soundness error then $\mathcal{NP} \subseteq \mathcal{BPP}$ would follow (assuming that one-way functions exist).

also use a careful analysis of the $\mathcal{HVSZK}$ to $\mathcal{HVSZK}|_{\rm AM}$ transformation of [Oka96] by which this transformation increases the simulation error by at most an exponentially vanishing amount. And lastly, applying Theorem 3, we use its item 6.

1.4 Techniques

Theorem 3 is proven by modifying the transformation presented in [DGW94]. Whereas the proof systems resulting from that transformation could be simulated only for a constant number of rounds, our modified transformation can be simulated for any (polynomial) number of rounds. Both transformations apply to honest-verifier Arthur-Merlin zero-knowledge proofs (both statistical and computational).

In the transformation of [DGW94], each ℓ-bit long (random) message sent by Arthur is replaced by an invocation of a 2-round *Random Selection* protocol, for generating strings in $\{0,1\}^\ell$. For any fixed positive polynomial p, a Random Selection protocol with the following two properties was presented [DGW94]:

1. As long as Arthur plays according to the protocol, Merlin may cause the outcome to deviate from uniform distribution over $\{0,1\}^\ell$ by at most $1/p(\ell)$. (That is, the variation distance is at most $1/p(\ell)$.)
2. As long as Merlin plays according to the protocol, Arthur may not cause any ℓ-bit string to appear as the outcome with probability greater than $p(\ell)^4 \cdot 2^{-\ell}$. In particular, when Arthur applies a deterministic cheating strategy, the outcome of the protocol is uniformly distributed over some set of $\frac{2^\ell}{p(\ell)^4}$ strings.

The proof system resulting from the above transformation is simulated in [DGW94] by running the honest-verifier simulator, and hoping that all Arthur-messages included in the transcript fall in the sets mentioned in Item (2) above. If the proof system uses only a constant number of invocations of the Random Selection protocol, then the above suffices for producing a black-box simulation with respect to any cheating Arthur-strategy. This approach fails when we have a non-constant number of rounds (Random Selection invocations).

In this paper we modify the above transformation as follows. Rather than selecting a message, we use the Random Selection protocol to specify (in a succinct manner) a set of 2^n messages. Merlin is then supposed to select a message for Arthur, uniformly from this set. An immediate concern is that this allows Merlin to select a string which is advantageous for cheating. However, this only increases Merlin's cheating probability by a factor of 2^n per each round. (We can first make the original proof system have an even smaller soundness error, so this should not scare us.) So the question is what we gained by doing so. Intuitively, we gained not having to simulate the Random Selection protocol for *any* possible outcome. Rather than having to simulate an execution which results in any specific ℓ-bit output, α, we only need to simulate an execution which results in a random set of strings containing α. The distinction is important since executions of the former type may exist only for a $1/\mathrm{poly}(\ell)$ fraction of the possible α's, whereas – as we show – executions of the latter type exists and can be efficiently generated for all but a $2^{-\Omega(n)}$ fraction of the α's. Proving the last statement is a major technical undertaking of the paper. It is reduced to proving the following lemma which may be of independent interest:

Lemma 5 (Hashing Lemma): *There exists a universal constant, $c > 0$, so that the following holds, for every $\epsilon, \delta > 0$. Let D and T be finite sets, H be a 2-universal family of hash functions from D to T, and $e \in T$. Let $S \subseteq H$ such that $|S| \geq \delta|H|$, and X be a random variable ranging over a finite set D having collision probability at most $\frac{\epsilon}{|T|}$* (i.e., $\sum_{x\in D} \Pr[X=x]^2 \leq \frac{\epsilon}{|T|}$). *Then the statistical difference between the following two random processes is at most $c \cdot \epsilon^{1/c}\delta^{-c}$.*

(A) *Select h uniformly in S, and let x be selected from X conditioned on $h(X) = e$. Output (h, x).*

(B) *Let $x \leftarrow X$, and h be selected uniformly among all $h \in H$ satisfying $h(x) = e$. Output (h, x).*

Actually, a special case of this lemma, where X is uniform over D (and $|T| = \epsilon \cdot |D|$) suffices for the current proof of Theorem 3. Thus, only a proof of this special case is given in this version. The stronger version was developed for an alternative proof, discovered first, which is totally superseded by the current proof.

2 Notation

Whenever we consider an interactive proof system, x will denote the common input and n will be the length of x. For notational convenience, we will often hide dependence on x or n when it is clear. For example, we write r instead of $r(n)$.

If X and Y are random variables, we write $\|X - Y\|$ for their *statistical difference* (or *variation distance*), defined as $\|X-Y\| = \frac{1}{2}(\sum_x |\Pr[X=x] - \Pr[Y=x]|)$. By $x \leftarrow X$, we mean taking a sample x from random variable X. If S is a set $x \in_R S$ indicates that x is chosen uniformly from S.

3 The starting proof system

Theorem 3 is proven by combining two transformations. The first transformation is obtained by parallel repetition, and is stated without proof below.[5] The protocols resulting from this transformation are the starting point for our main transformation, stated in the next section.

Lemma 3.1 *Let L be a language having a* honest-verifier *statistical (resp., computational) zero-knowledge* public-coin *proof systems of r rounds. Then L has such a* (r-round honest-verifier) *zero-knowledge* (public-coin) *proof system in which*

1. *The prover strategy can be implemented in probabilistic polynomial-time given oracle access to the original prover stategy.*
2. *The completeness error is exponentailly vanishing, and in case the original proof system has perfect completeness so does the resulting one.*
3. *Soundness error is less than $2^{-n\cdot(r+1)}$.*
4. *For $L \in \mathcal{HVSZK}$: The simulator deviation is at most a polynomial factor greater than the original one.*

4 The transformation

Fix a language L in $\mathcal{HVSZK}$ or $\mathcal{HVCZK}|_{\rm AM}$ and let (M, A) be the proof system guaranteed by Lemma 3.1. Let $r = r(n)$ be the number of rounds of (M, A) and let $\ell = \ell(n)$ be the length of A's messages. We may describe this proof system as follows:

[5] Recall that *honest-verifier* zero-knowledge properties are preserved under parallel repetition.

Original Proof System (M, A)**, on input** x**:**

1. In round $i \quad (i = 1, 2, \ldots, r)$,
   (a) A chooses a message $\alpha_i \in_R \{0,1\}^\ell$ and sends it to M.
   (b) M sends a response $\beta_i \leftarrow M(\alpha_1, \beta_1, \alpha_2, \beta_2, \ldots, \alpha_i)$ to A.
2. After round r, machine A deterministically decides whether to accept or reject.

The reason such a protocol could be zero-knowledge against the honest verifier but not against dishonest verifiers is that nothing prevents A from choosing the α_i's maliciously rather than uniformly. The idea of our transformation is to replace A's random choices with a Random Selection protocol (to be described in Section 5) which guarantees that the α_i's are statistically close to uniform, regardless of how A behaves. The new protocol, denoted $(\mathbf{M}, \mathbf{A})$, proceeds as follows.

Transformed Proof System $(\mathbf{M}, \mathbf{A})$**, on input** x**:**

1. In stage $i \quad (i = 1, 2, \ldots, r)$,
   (a) $\mathbf{M}$ and $\mathbf{A}$ use the Random Selection protocol, $RS_{2nr(n),\ell(n)}(n)$, to select $\alpha_i \in \{0,1\}^\ell$.
   (b) $\mathbf{M}$ sends the response $\beta_i \leftarrow M(\alpha_1, \beta_1, \alpha_2, \beta_2, \ldots, \alpha_i)$ to $\mathbf{A}$.
2. After stage r, machine $\mathbf{A}$ accepts or rejects as A would on transcript $(\alpha_1, \beta_1, \ldots, \alpha_r, \beta_r)$.

We will prove the following about the Transformed Proof System:

Lemma 4.1 *The Transformed Proof System* $(\mathbf{M}, \mathbf{A})$ *has the following properties:*

1. *The number of rounds is twice the number of rounds in* (M, A).
2. $\mathbf{M}$ *can be implemented in probabilistic polynomial time given oracle access to* M.
3. *The completeness error is exponentially vanishing. In case* (M, A) *has perfect completeness, so does* $(\mathbf{M}, \mathbf{A})$.
4. *Soundness error* $1/n$.
5. *When* (M, A) *is Honest-Verifier Statistical (resp., Computational) Zero-Knowledge,* $(\mathbf{M}, \mathbf{A})$ *is Statistical (resp., Computational) Zero-Knowledge, and this zero-knowledge property is exhibited by a black-box simulator.*
6. *In the case of Statistical Zero-Knowledge, the simulator deviation is at most* $2^{-\Omega(n)}$ *greater than that of* (M, A).

Theorem 3, follows immediately from Lemmas 3.1 and 4.1.[6] We now informally explain why Lemma 4.1 holds. All of these properties depend on facts about our Random Selection protocol which will be proven in subsequent sections. Property 1 follows from the fact that our Random Selection Protocol consists of 2 rounds with Merlin sending the last message. Property 2 is clear, given that the Merlin's strategy in the Random Selection protocol can be implemented in probabilistic polynomial time.

Property 3, the completeness error, follows from the fact that (M, A) has exponentially vanishing completeness error and the fact that when $\mathbf{M}$ behaves honestly in the Random Selection protocol, the α's will have only have a statistical difference of $2^{-\Omega(n)}$ from uniform. It is obvious that perfect completeness is preserved by our transformation. For soundness (Property 4, we will show that in our Random Selection protocol, a cheating $\mathbf{M}$ cannot make the output lie in any set $S \subset \{0,1\}^\ell$ with probability greater that $2^n \cdot \frac{|S|}{2^\ell} + \frac{1}{2nr}$. This gives $\mathbf{M}$ essentially an extra 2^n factor of freedom (compared to what M has) at each stage. Over r stages, we expect $\mathbf{M}$ to succeed with probability 2^{rn} times greater than M can. But since the original (M, A) protocol has soundness error $2^{-(r+1)n}$, $\mathbf{M}$ still has only an exponentially small chance of succeeding. The additive error term of $1/2nr$ also accumulates to give an additional additive factor of $1/2n$ to the soundness error over r rounds, yielding a total soundness error less than $1/n$. A more detailed proof of soundness will be given in the full version of the paper [GSV98].

The proof of zero-knowledgeness (Properties 5 and 6) is the major technical undertaking of the paper, and it too reduces to properties of our Random Selection protocol. We will demonstrate that no matter what strategy the verifier follows, the α_i's will be distributed statistically close to uniform. Moreover, we will show that the Random Selection protocol satisfies a *strong simulability property*: Using the verifier algorithm as a black-box subroutine and given a random $\alpha \in \{0,1\}^\ell$, one can efficiently simulate the distribution of Random Selection transcripts which yield α. Thus, a simulator for the Transformed Proof System could operate as follows: Run the honest verifier simulator for the original proof system to produce a transcript of α_i's and β_i's; then use the strong simulator for the Random Selection protocol to "fill in" how the α_i's are chosen. These intuitive arguments will be made precise in the next few sections.

5 Random Selection

Let q and ℓ be any polynomials. In this section, we describe an Arthur-Merlin protocol $RS_{q,\ell}(n) = (M_{RS}, A_{RS})(n)$ for randomly selecting a string in $\{0,1\}^{\ell(n)}$. The protocol employs the Random selection protocol $DGW_{q,\ell}(n) = (M_D, A_D)$ of [DGW94] as a subprotocol, and the following presentation is adapted from that paper.

For notational convenience, we will write q to mean $q(n)$ and ℓ to mean $\ell(n)$. Let $\mathcal{H}$ be the space of affine linear functions from $\{0,1\}^\ell$ to $\{0,1\}^{\ell-n}$, i.e. $h \in \mathcal{H}$ is of the form $h(x) = Ax + b$ for some appropriately sized matrix A and vector b.[7] For $\alpha \in \{0,1\}^\ell$, we write $\mathcal{H}_\alpha$ for $\{h \in \mathcal{H}\colon h(\alpha) = 0\}$. Let $s = \ell \cdot (\ell - n) + (\ell - n)$ and $t = s - 4\log_2(3qs)$. Note that elements of $\{0,1\}^s$ can be viewed as elements from $\mathcal{H}$. The protocol $DGW_{q,\ell}$ utilizes a space of functions $\mathcal{F}$ from $\{0,1\}^s$ to $\{0,1\}^t$ satisfying the following properties:

1. Each $f \in \mathcal{F}$ has a description of size $\mathrm{poly}(n)$.
2. There is a $\mathrm{poly}(n)$-time algorithm that, on input $f \in \mathcal{F}$ and $h \in \{0,1\}^s$, outputs $f(h)$.
3. There is a $\mathrm{poly}(n)$-time algorithm that, on input $f \in \mathcal{F}$, $y \in \{0,1\}^t$, lists all the elements of $f^{-1}(y)$. In particular, $|f^{-1}(y)| \leq p(n)$ for some polynomial p.
4. For every $y \in \{0,1\}^s$ and $f \in \mathcal{F}$, $f^{-1}(y)$ is nonempty.
5. $\mathcal{F}$ is a family of almost s-wise independent hashing functions in the following sense: For every s distinct points $h_1, \ldots, h_s \in (\{0,1\}^s \setminus \{0,1\}^t 0^{s-t})$, for a uniformly chosen $f \in \mathcal{F}$, the random variables $f(h_1), \ldots, f(h_s)$ are independently and uniformly distributed in $\{0,1\}^t$. (This property is used only for the proof of the soundness condition of the protocol, found in [DGW94].)

[6] For ease of presentation, we only show how to obtain a soundness error of $1/n$, but this can be replaced with any inverse polynomial.

[7] Any 2-universal family for which the required computations are feasible can be used; we use this particular family for simplicity and ease of presentation.

An explicit construction of such a family is given in [DGW94]. We can view each $f \in \mathcal{F}$ as defining a partition of $\{0,1\}^s$ into 2^t cells of the form $f^{-1}(y)$, each of size $\mathrm{poly}(n)$. For notational convenience, we will sometimes write *cell* y to refer to the cell $f^{-1}(y)$.

We now describe the protocol of [DGW94]:

The DGW Random Selection Protocol $DGW_{q,\ell} = (M_D, A_D)(n)$:

1. A_D selects $f \in_R \mathcal{F}$, and sends it to M_D (i.e., A_D selects a random partition).
2. M_D selects $y \in_R \{0,1\}^t$, and sends it to A_D (i.e., M_D uniformly selects a cell).
3. A_D selects $h \in_R f^{-1}(y)$ (i.e. A_D uniformly selects an element of the cell).
4. Output h.

If, at any step, A_D or M_D do not select an object from the appropriate set, whatever message they send is interpreted as a canonical element of that set. In [DGW94], it was shown that the above protocol has the following properties (roughly speaking):

1. (Soundness) For any Merlin strategy M_D^*, the output distribution on $\mathcal{H} = \{0,1\}^s$ of (M_D^*, A_D) deviates from uniform by at most $1/q$ (in statistical difference).
2. (Simulability) Let A_D^* be any strategy for Arthur. At least a $1/\mathrm{poly}(n)$ fraction of the h's in $\{0,1\}^s$ occur as possible outputs of the interaction (M_D, A_D^*) and given such an h, one can simulate in $\mathrm{poly}(n)$-time A_D^*'s view of an interaction resulting in h.

The main hindrance in applying the protocol as used by [DGW94] is that the simulator is only guaranteed to work for a $1/\mathrm{poly}(n)$ fraction of the h's. The new technique of this paper is to interpret the output $h \in \mathcal{H}$ of the DGW protocol as a set of strings (namely $h^{-1}(0)$), from which a single string α is randomly selected by Merlin. It is this α, rather than h, that is the output of the Random Selection protocol. Thus, we only need to simulate the Random Selection protocol for a random α rather than a random h. For a given α, there are exponentially many hash functions h such that $h(\alpha) = 0$. Because this space of h's is so large and covers the α's near-uniformly, we are able to perform the simulation for a $1 - 2^{-\Omega(n)}$ fraction of the α's.

A full description of our Random Selection protocol follows.

Our Random Selection Protocol $RS_{q,\ell} = (M_{RS}, A_{RS})(n)$:

1–3. As in $DGW_{q,\ell}(n)$.

4. M_{RS} selects $\alpha \in_R h^{-1}(0)$. (If $h^{-1}(0) = \emptyset$ then α is defined to be 0^ℓ.)
5. Output α.

As with the DGW protocol, if A_{RS} or M_{RS} do not select an object from the appropriate set at any step, whatever message they send is interpreted as a canonical element of that set. The properties of this protocol are described in the following Proposition.

Proposition 1 *For any polynomials q and ℓ, the Random Selection protocol $RS_{q,\ell}$ is a 2-round protocol with the following properties:*

1. *(Efficiency) Both M_{RS} and A_{RS} can be implemented in time* $\mathrm{poly}(n)$ *and the protocol is public-coin for both parties.*
2. *(Soundness) For all Merlin strategies M_{RS}^* and all sets $S \subset \{0,1\}^\ell$, the probability that the output of $(M_{RS}^*, A_{RS})(n)$ lies in S is at most*
$$2^n \cdot \frac{|S|}{2^\ell} + \frac{1}{q}$$
3. *(Strong Simulability) There exists a black-box simulator S_{RS} running in time* $\mathrm{poly}(n)$*, such that for all deterministic[8] Arthur strategies A_{RS}^*, the statistical difference between the following distributions is $2^{-\Omega(n)}$:*
   - **(I)** *Execute $(A_{RS}^*, M_{RS})(n)$, let $\alpha \in \{0,1\}^\ell$ be the output of the protocol, and let v be A_{RS}^*'s view of the interaction (i.e., v is a transcript (f, y, h, α)).[9]*
   - **(II)** *Choose α uniformly from $\{0,1\}^\ell$. Output $(S_{RS}^{A_{RS}^*}(\alpha), \alpha)$.*

Remark. The α's are included in the outputs of Distributions (I) and (II) above to force the simulator to produce a transcript for an *externally specified* α (rather than an α which it generates on its own while producing the transcript.)

Proof: Efficiency is immediate from the description of the protocol and the properties of the families $\mathcal{F}$ and $\mathcal{H}$. For Soundness, let M_{RS}^* be any cheating Merlin strategy and consider an execution of the protocol (M_{RS}^*, A_{RS}). Notice that that the probability that the output α lies in some set S is bounded above by the probability that $h^{-1}(0)$ contains an element of S. Now, for h chosen *uniformly* from $\mathcal{H}$ (instead of by the protocol), the probability that $h^{-1}(0)$ contains an element of S is at most
$$\sum_{\alpha \in S} \Pr_{h \in_R \mathcal{H}}[h(\alpha) = 0] = \frac{|S|}{2^{\ell - n}}.$$
In our protocol, h is chosen using the DGW protocol. It shown in [DGW94, Prop. 1] that a cheating Merlin can cause at most a $1/q$ statistical difference from the uniform distribution on $\mathcal{H}$, and so the Soundness property follows.

We now describe the simulator which will be used to establish Strong Simulability. Recall that p is polynomial bound on the size of $f^{-1}(y)$ for any $f \in \mathcal{F}$, s is the description length for elements of $\mathcal{H}$, and functions in $\mathcal{F}$ map $\{0,1\}^s$ to $\{0,1\}^t$, where $t = s - 4\log_2(3qs)$.

The simulator $S_{RS}^{A_{RS}^*}$, on input $\alpha \in \{0,1\}^\ell$, proceeds as follows:

S1. Let $f \in \mathcal{F}$ be the first message sent by A_{RS}^*.

S2. Repeat the following up to $n \cdot 2(3sq)^4 \cdot p$ times:
   (a) Choose h' uniformly from $\mathcal{H}_\alpha$ (Recall that $\mathcal{H}_\alpha = \{h \colon h(\alpha) = 0\}$.
   (b) Let $y = f(h')$ (i.e., y is the cell containing h'). Compute $k \stackrel{\mathrm{def}}{=} |f^{-1}(y) \cap \mathcal{H}_\alpha|$. With probability $1 - \frac{1}{k}$, proceed to next iteration of Step S2. (Otherwise continue.)
   (c) Let $h = A_{RS}^*(y)$, that is, the element (hereafter called the *cell representative*) of cell y that A_{RS}^* gives in Step 3 after being sent y in Step 2.
   (d) If $h(\alpha) = 0$, output $((f, y, h, \alpha), \alpha)$ and terminate the simulation. Otherwise, proceed to next iteration of Step S2.

S3. If the simulator failed to produce output so far, output `fail`.

[8] The restriction to deterministic Arthur strategies is only for ease of presentation, as a simulator for randomized Arthur strategies can uniformly select and fix Arthur's coins and then use the simulator for deterministic strategies. When we use the Random Selection simulator as a subroutine in the simulator for the Transformed Protocol in Section 6, the coins of Arthur will have already been fixed by the outer simulator.

[9] In Section 1.1, we defined the Verifier's view to consist of his random coins and the Prover's messages. Here, we do not include random coins, as they are irrelevant for deterministic strategies. We also include Arthur's messages — this is unnecessary as they are functions of Merlin's messages, but it will be convenient for our presentation.

From the various properties of the families $\mathcal{F}$ and $\mathcal{H}$, such as the fact that $f^{-1}(y)$ can be enumerated in time $\mathrm{poly}(n)$, and the fact that s, q, and p are all $\mathrm{poly}(n)$, we see immediately that the running time of $S_{RS}^{A^*_{RS}}$ is $\mathrm{poly}(n)$.

Let us now show that Distributions (I) and (II) in Proposition 1 have statistical difference $2^{-\Omega(n)}$. Each produces output of the form $((f, y, h, \alpha), \alpha)$. In both cases, f is the (deterministically chosen) first message of A^*_{RS} and $y = f(h)$, so it suffices to show that the distributions restricted to their (h, α) components are statistically close. We therefore define the Distributions (I') and (II') to be the Distributions (I) and (II) restricted to their (h, α) components. To analyze these distributions, we make use of the following Lemma, the proof of which is in Section 7. (As stated in the introduction, we can also prove a much more general form of this lemma. The proof is omitted in this abstract.)

Lemma 5.1 *There exists a universal constant $c > 0$, so that the following holds: Let $\mathcal{H}$ be the family of affine-linear maps from $D = \{0,1\}^\ell$ to $T = \{0,1\}^{\ell'}$, i.e. $h \in \mathcal{H}$ is of the form $h(x) = Ax + b$ for some matrix A and vector b. Let $S \subset \mathcal{H}$ be such that $|S| \geq \delta|\mathcal{H}|$. Let $\varepsilon = \frac{|T|}{|D|}$. Then*

Part 1: *The statistical difference between the following two distributions is at most $(c \cdot \varepsilon^{1/c}\delta^{-c})$:*

(A) *Choose $h \in_R S$. Let $x \in_R h^{-1}(0)$. Output (h, x).*
(B) *Choose $x \in_R D$. Let $h \in_R S \cap \mathcal{H}_x$. Output (h, x).*

Part 2: *For at least a $1 - (c \cdot \varepsilon^{1/c}\delta^{-c})$ fraction of $x \in D$,*

$$\frac{|S \cap \mathcal{H}_x|}{|\mathcal{H}_x|} \geq \delta/2.$$

When we apply the lemma, we take $\ell' = \ell - n$, $\varepsilon = 2^{-n}$, and $S = \{A^*_{RS}(y) \colon y \in \{0,1\}^t\}$. In other words, S is the set all possible *cell representatives* that A^*_{RS} can send in Step 3 of the protocol (M_{RS}, A^*_{RS}). Notice that

$$\delta \stackrel{\mathrm{def}}{=} \frac{|S|}{|\mathcal{H}|} = \frac{2^t}{2^s} = 2^{-4\log_2(3sq)} = \frac{1}{(3sq)^4}.$$

and so, $c \cdot \varepsilon^{1/c}\delta^{-c} = 2^{-\Omega(n)}$. Now, observe that the protocol (M_{RS}, A^*_{RS}) selects h uniformly from S. (Recall that A^*_{RS} is deterministic.) Thus, Distribution (I') is exactly Distribution (A) of Lemma 5.1. Now we will show that the Distribution (II') is statistically close to Distribution (B).

Let us consider a single iteration of Step S2 in $S_{RS}^{A^*_{RS}}$. In such an iteration, h' is chosen uniformly from $\mathcal{H}_\alpha$, and $y = f(h')$. We write $f(\mathcal{H}_\alpha)$ to denote the set of images of elements of $\mathcal{H}_\alpha$ under f (i.e., $f(\mathcal{H}_\alpha) = \{f(h) \colon h \in \mathcal{H}_\alpha\}$). In other words, $f(\mathcal{H}_\alpha)$ is the set of cells intersecting $\mathcal{H}_\alpha$. We want to establish that the distribution of h's produced by the simulator will be uniform in $S \cap \mathcal{H}_\alpha$. In order for this to happen, y must be uniformly selected from $f(\mathcal{H}_\alpha)$. If f was chosen honestly by A^*_{RS}, we would expect it to be one-to-one on the set $\mathcal{H}_\alpha$, since $\mathcal{H}_\alpha$ is a vanishingly small fraction of the domain. However, f is chosen adversarially, so we must do some work to ensure uniformity:

Notice that for any $y_0 \in f(\mathcal{H}_\alpha)$, the probability that $f(h') = y_0$ when uniformly selecting $h' \in \mathcal{H}_\alpha$ is exactly $|\mathcal{H}_\alpha \cap f^{-1}(y_0)|/|\mathcal{H}_\alpha|$. In Step S2b, any such choice is maintained with probability $1/|\mathcal{H}_\alpha \cap f^{-1}(y_0)|$. Thus the probability that $y = y_0$ after Steps S2a and S2b in S_{RS} is exactly $1/|\mathcal{H}_\alpha|$. This is independent of y_0, and therefore y is a uniformly chosen element of $f(\mathcal{H}_\alpha)$ — that is, a uniformly chosen cell intersecting $\mathcal{H}_\alpha$. (These probabilities sum up to $|f(\mathcal{H}_\alpha)|/|\mathcal{H}_\alpha|$, which may be less than 1; this is due to the possibility that the iteration ends prematurely in Step S2b.)

Now, since, in Step S2c, $h = A^*_{RS}(y)$ is taken to be the representative of cell y, the function h is uniformly distributed over the representatives of cells which intersect $\mathcal{H}_\alpha$. In Step S2d, we abandon any h not in $\mathcal{H}_\alpha$, so the resulting distribution on h is uniform over cell representatives in $\mathcal{H}_\alpha$, that is, uniform over $S \cap \mathcal{H}_\alpha$. Thus a single iteration of the loop produces an h uniformly chosen from $S \cap \mathcal{H}_\alpha$, if it manages to produce output at all. This is identical to how h is chosen in Distribution (B) of Lemma 5.1. So, to show that the Distribution (II') is statistically close to Distribution (B), we need only to show that the probability that the repeat loop fails to produce output in all its $\mathrm{poly}(n)$ iterations is $2^{-\Omega(n)}$ for at least a $1 - 2^{-\Omega(n)}$ fraction of the α's in $\{0,1\}^\ell$. We do this by showing that each iteration produces output with probability at least n times the reciprocal of the number of iterations.

There are two places in which an iteration can be exited, causing it to fail to produce output — Steps S2b and S2d. Observe that the simulator never exits in Step S2d if h' chosen in Step S2a lies in S, because then h will equal h'. This occurs with probability $|S \cap \mathcal{H}_\alpha|/|\mathcal{H}_\alpha|$. By Lemma 5.1, for at least a $1 - 2^{-\Omega(n)}$ fraction of $\alpha \in \{0,1\}^\ell$, this quantity is at least $\delta/2 = 1/2(3sq)^4$.

Now suppose that h' has been chosen in S. The probability of not exiting in Step S2b is at least $1/|f^{-1}(y)|$, which is at least $1/p$ by the properties of the family $\mathcal{F}$. Thus, for a $1 - 2^{-\Omega(n)}$ fraction of the α's, a single iteration produces output with probability at least $1/(2(3sq)^4 \cdot p)$. Since there are $(2(3sq)^4 \cdot p) \cdot n$ iterations, output is produced with probability $1 - 2^{-\Omega(n)}$.

We have shown that Distribution (I') is identical to Distribution (A) in Lemma 5.1 and Distribution (II') has a statistical difference of $2^{-\Omega(n)}$ from Distribution (B). So, by Lemma 5.1, we conclude that Distributions (I) and (II) have statistical difference $2^{-\Omega(n)}$ and Strong Simulability is established. ■

6 Simulating the Transformed Protocol

In this section, we describe the simulator for the protocol $(\mathbf{M}, \mathbf{A})$ of Section 4. Let S be the simulator for the honest verifier in the original protocol (M, A). We will give a universal simulator $\mathbf{S}$ for $(\mathbf{M}, \mathbf{A})$ which uses any verifier strategy $\mathbf{A}^*$ as a black-box.

The simulator $\mathbf{S}^{\mathbf{A}^*}$, on input x:

1. Uniformly choose and fix random coins c for $\mathbf{A}^*$ to obtain a deterministic strategy $A^{(1)}$.
2. Run the original honest-verifier simulator to obtain a transcript $(\alpha_1, \beta_1, \ldots, \alpha_r, \beta_r) \leftarrow S(x)$.
3. For $i = 1$ to r, do the following:
   (a) Run the strong simulator for the Random Selection protocol, on input α_i with Arthur strategy $A^{(i)}$, to obtain a simulated transcript t_i of the Random Selection protocol (i.e., $t_i \leftarrow S_{RS}^{A^{(i)}}(\alpha_i)$).
   (b) Let $A^{(i+1)}$ be the state of $A^{(i)}$ after additional history t_i, α_i, β_i.
4. Output $(t_1, \alpha_1, \beta_1, \ldots, t_r, \alpha_r, \beta_r; c)$.

To prove that the above simulator has the desired properties, we first consider its output distribution in the case that the original honest-verifier simulator S is perfect: Let $\overline{\mathbf{S}}^{\mathbf{A}^*}$ be the output distribution of $\mathbf{S}^{\mathbf{A}^*}$ if the output of S in Step 2 is replaced with a true sample $(\alpha_1, \beta_1, \ldots, \alpha_r, \beta_r)$ of the protocol (M, A). By an induction argument using the strong simulability property of the Random Selection protocol, it is easy to show the following:

Claim 6.1 *$\overline{\mathbf{S}}^{\mathbf{A}^*}(x)$ and $(\mathrm{M}, \mathbf{A}^*)(x)$ have statistical difference at most $2^{-\Omega(n)}$.*

The proof of Claim 6.1 can be found in the full version of the paper [GSV98]. Now we deduce Lemma 4.1, Parts 5 and 6, from Claim 6.1.

Statistical Zero-Knowledge. Using the output of S instead of a true sample from (M, A) can increase the simulator deviation by at most $\|S(x)-(M,A)(x)\|$, which is exactly the simulator deviation for the protocol (M, A).

Computational Zero-Knowledge. We claim that the probability ensembles $X_1 \stackrel{\text{def}}{=} \{(\mathbf{M}, \mathbf{A}^*)(x)\}_{x\in L}$ and $X_2 \stackrel{\text{def}}{=} \{\mathbf{S}^{\mathbf{A}^*}(x)\}_{x\in L}$ are computationally indistinguishable for any probabilistic polynomial-time $\mathbf{A}^*$. Consider the ensemble $X_3 \stackrel{\text{def}}{=} \{\overline{\mathbf{S}}^{\mathbf{A}^*}(x)\}_{x\in L}$. By Claim 6.1, X_1 and X_3 are statistically close and therefore computationally indistinguishable. We claim that X_2 and X_3 are computationally indistinguishable, for any probabilistic polynomial-time $\mathbf{A}^*$. This holds because any distinguisher D between X_2 and X_3 can be transformed into a distinguisher D' between $\{(M, A)(x)\}_{x\in L}$ and $\{S(x)\}_{x\in L}$, which are computationally indistinguishable by hypothesis. The new distinguisher D' operates as follows: Given a transcript T of either of the latter two ensembles, perform the procedure specified by $\mathbf{S}^{\mathbf{A}^*}$, replacing the execution in Step 2 with T, and feed the output of $\mathbf{S}^{\mathbf{A}^*}$ to D. When T is selected according to $\{(M, A)(x)\}_{x\in L}$, D is fed with ensemble X_3, whereas when T is selected according to $\{S(x)\}_{x\in L}$, D is fed with ensemble X_2.

Remark. The above proof actually shows that, for any (not just probabilistic polynomial-time) verifier $\mathbf{A}^*$, if $(\mathbf{M}, \mathbf{A}^*)$ and $\mathbf{S}^{\mathbf{A}^*}$ can be distinguished by algorithm D, then there is an algorithm no more powerful than $\mathbf{A}^*$ and D (i.e., a probabilistic polynomial time machine with oracle access to $\mathbf{A}^*$ and D) that can distinguish the original honest-verifier proof system (M, A) from its simulator S. So, if the honest-verifier simulator produces transcripts indistinguishable from (M, A) by any machine running in, say, quasi-polynomial time, then the new protocol $(\mathbf{M}, \mathbf{A})$ is zero-knowledge against all quasi-polynomial time verifiers.

7 Proof of Hashing Lemma

Here we provide a proof of the Hashing Lemma used to establish the main result of this paper. We restate the lemma here:

Lemma 7.1 (Hashing Lemma) *There exists a universal constant $c > 0$, so that the following holds: Let $\mathcal{H}$ be the family of affine-linear maps from $D = \{0,1\}^{\ell}$ to $T = \{0,1\}^{\ell'}$, i.e. $h \in \mathcal{H}$ is of the form $h(x) = Ax + b$ for some matrix A and vector b. Let $S \subset \mathcal{H}$ be such that $|S| \geq \delta|\mathcal{H}|$. Let $\varepsilon = \frac{|T|}{|D|}$. Then*

Part 1: *The statistical difference between the following two distributions is at most $c \cdot \varepsilon^{1/c}\delta^{-c}$:*

$A = (A_{\mathcal{H}}, A_X)$: Let $h \in_R S$. Let $x \in_R h^{-1}(0)$. Output (h, x).

$B = (B_{\mathcal{H}}, B_X)$: Let $x \in_R D$. Let $h \in_R S \cap \mathcal{H}_x$. Output (h, x).

Part 2: *For at least a $1 - (c \cdot \varepsilon^{1/c}\delta^{-c})$ fraction of $x \in D$,*

$$\frac{|S \cap \mathcal{H}_x|}{|\mathcal{H}_x|} \geq \frac{1}{2} \cdot \frac{|S|}{|H|} \geq \frac{\delta}{2}.$$

Proof: We define a *perfect* hash function $h \in \mathcal{H}$ to be one of the form $h(x) = Ax + b$, where the matrix A is full rank (and hence h is surjective). Note that a straightforward calculation shows that at most an ε fraction of the functions in $\mathcal{H}$ are *not* perfect.

We first establish Part 1 of the Hashing Lemma for the special case of perfect hash functions.

Sublemma 7.2 *Part 1 of the Hashing Lemma holds when S contains only perfect hash functions.*

Proof: First, we consider the relationship between distributions A_X and B_X.

Claim 7.3 $\|A_X - B_X\| \leq \frac{3\varepsilon^{1/3}}{\delta}$.

Proof: Note B_X is uniform over D. To establish the claim, it suffices to show that for all $C \subseteq D$,

$$\left|\Pr\left[A_X \in C\right] - \frac{|C|}{|D|}\right| \leq \frac{3\varepsilon^{1/3}}{\delta}.$$

Note $\left|\Pr\left[A_X \in C\right] - \frac{|C|}{|D|}\right| = \left|\Pr\left[A_X \in (D \setminus C)\right] - \frac{|D\setminus C|}{|D|}\right|$, so it suffices to consider sets C such that $\frac{|C|}{|D|} \geq \frac{1}{2}$. From the definition of A, we observe:

$$\Pr\left[A_X \in C\right] = \frac{1}{|S|}\sum_{h\in S}\frac{|h^{-1}(0)\cap C|}{|h^{-1}(0)|} = \frac{1}{|S|}\sum_{h\in S}\varepsilon\cdot|h^{-1}(0)\cap C|$$

where the last equality is due to our assumption that every $h \in S$ is perfect, and hence $|h^{-1}(0)| = 1/\varepsilon$.

To analyze the expression above, which refers to a sum over $h \in S$, we first consider the behaviour of the sum over all $h \in \mathcal{H}$. Here, we can use Chebyshev's inequality. Consider the probability space uniform over $\mathcal{H}$, and define, for every $x \in C$, an indicator random variable:

$$\chi_x(h) = \begin{cases} 1 \text{ if } h(x) = 0 \\ 0 \text{ otherwise} \end{cases}$$

Let $W_C(h) = \varepsilon \cdot |h^{-1}(0) \cap C| = \varepsilon \cdot \sum_{x\in C} \chi_x(h)$. Since $\mathcal{H}$ is a 2-universal family of hash functions, the χ_x's are pairwise independent with $\Pr_{h\in\mathcal{H}}[\chi_x(h) = 1] = \frac{1}{|T|} = \frac{1}{\varepsilon\cdot|D|}$. Thus, we have that:

$$\mathrm{E}_{h\in\mathcal{H}}[W_C(h)] = \varepsilon \cdot \sum_{x\in C}\mathrm{E}_{h\in\mathcal{H}}[\chi_x(h)] = \frac{|C|}{|D|}.$$

$$\mathrm{Var}_{h\in\mathcal{H}}[W_C(h)] = \varepsilon^2 \cdot \sum_{x\in C}\mathrm{Var}_{h\in\mathcal{H}}[\chi_x(h)] < \varepsilon \cdot \frac{|C|}{|D|}.$$

By Chebyshev's inequality,

$$\Pr_{h\in\mathcal{H}}\left[\left|W_C(h) - \frac{|C|}{|D|}\right| > \varepsilon^{1/3}\cdot\frac{|C|}{|D|}\right] < \frac{\mathrm{Var}[W_C]}{\left(\varepsilon^{1/3}\cdot\frac{|C|}{|D|}\right)^2} < \frac{\varepsilon^{1/3}|D|}{|C|} \leq 2\varepsilon^{1/3}$$

where the last inequality is because $|C| \geq |D|/2$. Since $\frac{|S|}{|\mathcal{H}|} \geq \delta$, we can apply the above to the probability space uniform over S and conclude,

$$\Pr_{h\in S}\left[\left|W_C(h) - \frac{|C|}{|D|}\right| > \varepsilon^{1/3}\frac{|C|}{|D|}\right] < \frac{2\varepsilon^{1/3}}{\delta}.$$

Recall,

$$\Pr[A_X \in C] = \frac{1}{|S|}\sum_{h\in S} W_C(h).$$

Hence, for all but at most $\frac{2\varepsilon^{1/3}}{\delta}\cdot|S|$ terms in the sum, we have that $\left|W_C(h) - \frac{|C|}{|D|}\right| \le \varepsilon^{1/3}\frac{|C|}{|D|}$. Since for every h it is true that $0 \le W_C(h) \le 1$, we have,

$$\left|\Pr[A_X \in C] - \frac{|C|}{|D|}\right| \le \varepsilon^{1/3}\frac{|C|}{|D|} + \frac{2\varepsilon^{1/3}}{\delta} \le \frac{3\varepsilon^{1/3}}{\delta}.$$

And the claim is proved. ■

We are now ready to complete the proof of this sublemma. For all $x \in D$ and all $h \in S$ such that $h(x) = 0$, we have, by Bayes' Law:

$$\begin{aligned}\Pr[A_{\mathcal{H}} = h | A_X = x] &= \frac{\Pr[A_X = x | A_{\mathcal{H}} = h]\cdot\Pr[A_{\mathcal{H}} = h]}{\Pr[A_X = x]} \\ &= \frac{|h^{-1}(0)|^{-1}\cdot|S|^{-1}}{\Pr[A_X = x]} = \frac{\varepsilon\cdot|S|^{-1}}{\Pr[A_X = x]}\end{aligned}$$

where the last step is because for all perfect h, $|h^{-1}(0)| = 1/\varepsilon$. Note that this value has no dependence on h. Hence, for every x, given $A_X = x$, the distribution $A_{\mathcal{H}}$ is uniform over $\{h \in S : h(x) = 0\}$. Note that for all x, given $B_X = x$, $B_{\mathcal{H}}$ is also uniform over the same set. Thus, conditioned on the value of x, the distributions $A_{\mathcal{H}}$ and $B_{\mathcal{H}}$ are identical.

Hence $\|A - B\| = \|A_X - B_X\| \le \epsilon_1$, and the sublemma is established. ■

Before we argue Part 1 of the Hashing Lemma in general, we will show how Part 2 follows from Sublemma 7.2. In the sequel, it will be convenient to introduce the following notation: For any subset $I \subseteq \mathcal{H}$, we will write I_x to denote the set $\{h \in I : h(x) = 0\}$.

In order to apply Sublemma 7.2, we will consider the subset $S' \subset S$ of all perfect hash functions in S. Since less than an ε fraction of all hash functions are not perfect, $|S'| \ge (1-\frac{\varepsilon}{\delta})|S| \ge (\delta-\varepsilon)\cdot|\mathcal{H}|$. Similarly, we define the following two modifications of the distributions A and B, using S' instead of S:

$A' = (A'_{\mathcal{H}}, A'_X)$: Let $h \in_R S'$. Let $x \in_R h^{-1}(0)$. Output (h, x).

$B' = (B'_{\mathcal{H}}, B'_X)$: Let $x \in_R D$. Let $h \in_R S' \cap \mathcal{H}_x$. Output (h, x).

The following claim establishes Part 2 of the Hashing Lemma:

Claim 7.4 *Let* $\epsilon_1 \stackrel{\text{def}}{=} \frac{3\varepsilon^{1/3}}{\delta-\varepsilon}$. *For at least a* $(1-\sqrt{\epsilon_1})$ *fraction of* $x \in D$, $\frac{|S_x|}{|\mathcal{H}_x|} \ge \delta/2$.

Proof: By the definition of A'_X,

$$\Pr[A'_X = x] = \frac{1}{|S'|}\sum_{h\in S'_x}\frac{1}{|h^{-1}(0)|} = \varepsilon\frac{|S'_x|}{|S'|}$$

where the last equality follows because $|h^{-1}(0)| = 1/\varepsilon$ for all $h \in S'$. However, by the Sublemma, $\|A'_X - B'_X\| \le \epsilon_1$. Note that B'_X is uniform over D, so for a $(1-\sqrt{\epsilon_1})$ fraction of $x \in D$, it must be that

$$\varepsilon\frac{|S'_x|}{|S'|} = \Pr[A'_X = x] \ge (1-\sqrt{\epsilon_1})\cdot\frac{1}{|D|}.$$

Thus,

$$\frac{|S_x|}{|\mathcal{H}_x|} \ge \frac{|S'_x|}{|\mathcal{H}_x|} \ge (1-\sqrt{\epsilon_1})\cdot\frac{|S'|}{\varepsilon|D|\cdot|\mathcal{H}_x|} = (1-\sqrt{\epsilon_1})\cdot\frac{|S'|}{|\mathcal{H}|}$$

where the last equality follows from $\varepsilon\cdot|D| = |T|$ and $|T|\cdot|\mathcal{H}_x| = |\mathcal{H}|$. Using the fact that $\frac{|S'|}{|\mathcal{H}|} \ge (1-\frac{\varepsilon}{\delta})\cdot\frac{|S|}{|\mathcal{H}|}$, we have, for a $(1-\sqrt{\epsilon_1})$ fraction of $x \in D$,

$$\frac{|S_x|}{|\mathcal{H}_x|} \ \ge\ (1-\sqrt{\epsilon_1})\cdot(1-\frac{\varepsilon}{\delta})\cdot\delta \ge \frac{\delta}{2}.$$

Note that the final inequality follows because we can safely assume that $\sqrt{\epsilon_1} + \frac{\varepsilon}{\delta} < \frac{1}{2}$. This is because we can freely assume that $c\cdot\varepsilon^{1/c}\delta^{-c} < 1$, since otherwise the statement of the Hashing Lemma becomes trivially satisfied. Since $\sqrt{\epsilon_1} + \frac{\varepsilon}{\delta}$ is upper bounded by $k\cdot\varepsilon^{1/k}\delta^{-k}$ for some constant k, our assumption can be made to imply that $\sqrt{\epsilon_1} + \frac{\varepsilon}{\delta} < \frac{1}{2}$ by choosing $c > 2k$. ■

Finally, we establish Part 1 of the Hashing Lemma in general by showing that the presence of imperfect hash functions will not disturb our computations. First, we see immediately that since $|S'| \ge (1-\frac{\varepsilon}{\delta})|S|$, the statistical difference between A and A' can be at most $\frac{\varepsilon}{\delta}$. To see that the statistical difference between B' and B is sufficiently small, it suffices to show that for almost all x, the probability that $B_{\mathcal{H}}$ outputs an imperfect hash function, given that $B_X = x$, is small. First we argue:

Claim 7.5 *For every* $x \in D$, $\Pr_{h\in\mathcal{H}_x}[h \textit{ is imperfect}] \le \varepsilon$.

Proof: Observe that for any $x \in D$, $\mathcal{H}_x$ consists exactly of those functions $h(y) = Ay + b$ where $b = -Ax$. Thus, there is exactly one function in $\mathcal{H}_x$ for every matrix A. Hence, the fraction of imperfect functions in $\mathcal{H}_x$ is precisely the fraction of matrices A that do not have full rank, which is at most ε. ■

For any $x \in D$, the probability that $B_{\mathcal{H}}$ outputs an imperfect hash function given that $B_X = x$ is

$$\Pr_{h\in S_x}[h \text{ is imperfect}] \le \Pr_{h\in\mathcal{H}_x}[h \text{ is imperfect}]\cdot\frac{|\mathcal{H}_x|}{|S_x|}.$$

Using Claim 7.4 and Claim 7.5 above, we have that for at least a $(1-\sqrt{\epsilon_1})$ fraction of $x \in D$, this probability is at most $\epsilon_2 \stackrel{\text{def}}{=} \varepsilon\cdot(2/\delta)$. Thus, $\|B - B'\| \le (1-\sqrt{\epsilon_1})\cdot\epsilon_2 + \sqrt{\epsilon_1} \le \epsilon_2 + \sqrt{\epsilon_1}$. We have already observed that $\|A' - A\| \le \frac{\varepsilon}{\delta}$, and Sublemma 7.2 showed that $\|B' - A'\| \le \epsilon_1$. Hence $\|A - B\| \le \epsilon_1 + \frac{\varepsilon}{\delta} + \epsilon_2 + \sqrt{\epsilon_1}$, and the Hashing Lemma is established. ■

Acknowledgments

We are grateful to Madhu Sudan for collaboration at early stages of this research, and to Shafi Goldwasser for helpful discussions and encouragement.

References

[AH87] William Aiello and Johan Håstad. Perfect zero-knowledge languages can be recognized in two rounds. In *Proceedings of the Twenty Eighth Annual Symposium on Foundations of Computer Science*, pages 439–448, 1987.

[BGG+88] Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In S. Goldwasser, editor, *Advances in Cryptology—CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 37–56. Springer-Verlag, 1990, 21–25 August 1988.

[BMO90] Mihir Bellare, Silvio Micali, and Rafail Ostrovsky. The (true) complexity of statistical zero-knowledge. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 494–502, 1990.

[Dam94] Ivan Damgård. Interactive hashing can simplify zero-knowledge protocol design. In *Proceedings of Crypto '95, Lecture Notes in Computer Science*, volume 403, pages 100–109. Springer-Verlag, 1994.

[DGOW95] Ivan Damgård, Oded Goldreich, Tatsuaki Okamoto, and Avi Wigderson. Honest verifier vs. dishonest verifier in public coin zero-knowledge proofs. In *Proceedings of Crypto '95, Lecture Notes in Computer Science*, volume 403. Springer-Verlag, 1995.

[DGW94] Ivan Damgård, Oded Goldreich, and Avi Wigderson. Hashing functions can simplify zero-knowledge protocol design (too). Technical Report RS-94–39, BRICS, November 1994. See Part 1 of [DGOW95].

[DOY97] Giovanni Di Crescenzo, Tatsuaki Okamoto, and Moti Yung. Keeping the SZK-verifier honest unconditionally. In Burton S. Kaliski Jr., editor, *Advances in Cryptology—CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 31–45. Springer-Verlag, 17–21 August 1997.

[FGM+89] Martin Fürer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stathis Zachos. On completeness and soundness in interactive proof systems. In Silvio Micali, editor, *Advances in Computing Research*, volume 5, pages 429–442. JAC Press, Inc., 1989.

[For89] Lance Fortnow. The complexity of perfect zero-knowledge. In Silvio Micali, editor, *Advances in Computing Research*, volume 5, pages 327–343. JAC Press, Inc., 1989.

[GG98] Oded Goldreich and Shafi Goldwasser. On the limits of non-approximability of lattice problems. These proceedings, 1998.

[GK93] Oded Goldreich and Eyal Kushilevitz. A perfect zero-knowledge proof system for a problem equivalent to the discrete logarithm. *Journal of Cryptology*, 6:97–116, 1993.

[GK96] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, 1996.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, February 1989.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the Association for Computing Machinery*, 38(1):691–729, 1991.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, Winter 1994.

[Gol95] Oded Goldreich. *Foundations of Cryptography (Fragments of a Book)*. Weizmann Institute of Science, February 1995. Available from `http://www.eccc.uni-trier.de/eccc/`.

[GSV98] Oded Goldreich, Amit Sahai, and Salil Vadhan. Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. *Electronic Colloquium on Computational Complexity*, 1998. `http://www.eccc.uni-trier/eccc/`.

[HILL] Johan Håstad, Russell Impagliazzo, Leonid Levin, and Michael Luby. Construction of pseudorandom generator from any one-way function. To appear in *SICOMP*. Preliminary versions by Impagliazzo et. al. in *21st STOC* (1989) and Håstad in *22nd STOC* (1990).

[IY87] Russell Impagliazzo and Moti Yung. Direct minimum-knowledge computations (extended abstract). In Carl Pomerance, editor, *Advances in Cryptology—CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 40–51. Springer-Verlag, 1988, 16–20 August 1987.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[Oka96] Tatsuaki Okamoto. On relationships between statistical zero-knowledge proofs. In *Proceedings of the Twenty Eighth Annual ACM Symposium on the Theory of Computing*, 1996. See also preprint of full version, Aug. 1997.

[OVY93] Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Interactive hashing simplifies zero-knowledge protocol design. In *Proceedings of Eurocrypt '93, Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[OW93] Rafail Ostrovsky and Avi Wigderson. One-way functions are essential for non-trivial zero-knowledge. In *Proceedings of the Second Israel Symposium on Theory of Computing and Systems*, 1993.

[SV97] Amit Sahai and Salil Vadhan. A complete promise problem for statistical zero-knowledge. In *Proceedings of the Thirty Eighth Annual Symposium on Foundations of Computer Science*, pages 448–457, 1997.

[Yao82] Andrew C. Yao. Theory and application of trapdoor functions. In *Proceedings of the Twenty Third Annual Symposium on Foundations of Computer Science*, pages 80–91, 1982.

A Definitions

Definition 6 (Interactive Proofs – IP) [GMR89]: *An* interactive proof system *with* completeness error $c : \mathbb{N} \mapsto \mathbb{N}$ *and* soundness error $s : \mathbb{N} \mapsto \mathbb{N}$ *for a language* L *is a two-party game, between a* verifier *executing a probabilistic polynomial-time strategy* (denoted V) *and a* prover *which executes a computationally unbounded strategy* (denoted P)*, satisfying*

- Completeness*: For every* $x \in L$*, the verifier* V *rejects with probability at most* $c(|x|)$*, after interacting with the prover* P *on common input* x*.*
- Soundness*: For every* $x \notin L$ *and every potential strategy* P^**, the verifier* V *accepts with probability at most* $s(|x|)$*, after interacting with* P^* *on common input* x*.*

In case $c \equiv 0$ *we say that the interactive proof has perfect completeness.*

Unless specified differently, an interactive proof system means one in which both the completeness and soundness errors are negligible (i.e., eventually smaller than $1/p(\cdot)$, for any polynomial p). Recall that completeness and soundness errors can be decreased by parallel repetitions of the proof system. Thus, a proof system with soundness and completeness errors which sum-up to a function bounded away from 1 (i.e., $c(n) + s(n) < 1 - 1/\mathrm{poly}(n)$), can be transformed into a proof system of the same number of rounds having exponentially decreasing completeness and soundness errors. This transformation preserves *honest-verifier* statistical (resp., computational) zero-knowledge. (Recall that zero-knowledge with respect to any verifier is not preserved, in general, under parallel repetition [GK96].)