Honest Verifier vs Dishonest Verifier in Public Coin Zero-Knowledge Proofs

Ivan Damgård[*] Oded Goldreich[†] Tatsuaki Okamoto[‡]
Avi Wigderson[§]

September 12, 1995

Abstract

This paper presents two transformations of public-coin/Arthur-Merlin proof systems which are zero-knowledge *with respect to the honest verifier* into (public-coin/Arthur-Merlin) proof systems which are zero-knowledge *with respect to any verifier*.

The first transformation applies only to constant-round proof systems. It builds on Damgård's transformation (see *Crypto93*), using ordinary hashing functions instead of the interactive hashing protocol (of Naor, Ostrovsky, Venkatesan and Yung – see *Crypto92*) which was used by Damgård. Consequently, the protocols resulting from our transformation have much lower round-complexity than those derived by Damgård's transformation. As in Damgård's transformation, our transformation preserves statistical/perfect zero-knowledge and does not rely on any computational assumptions. However, unlike Damgård's transformation, the new transformation is not applicable to argument systems or to proofs of knowledge.

The second transformation can be applied to proof systems of arbitrary number of rounds, but it only preserves statistical zero-knowledge. It assumes the existence of secure commitment schemes and transforms any public-coin proof which is statistical zero-knowledge with respect to the honest into one which is statistical zero-knowledge (in general). It follows, by a result of Ostrovsky and Wigderson (1993), that any language which is "hard on the average" and has a public-coin proof system which is statistical zero-knowledge with respect to the honest verifier, has a proof system which is statistical zero-knowledge (with respect to any verifier).

[*]Dept. of Computer Science, Aarhus Univesity, Denmark and BRICS, Basic Research In Computer Science, center of the Danish National Research Foundation.

[†]Dept. of Computer Science and Applied Math., Weizmann Institute of Science, Rehovot, Israel. Work done while visiting BRICS, Basic Research In Computer Science, center of the Danish National Research Foundation. Supported in part by grant No. 92-00226 from the United States – Israel Binational Reseach Foundation (BSF), Jerusalem, Israel.

[‡]NTT Laboratories, Yokosuka-shi, 238-03 Japan. Work done while visiting AT&T Bell Laboratories, Murray Hill, NJ, USA

[§]Institute for Computer Science, Hebrew University, Jerusalem, Israel. Work done while visiting BRICS, Basic Research In Computer Science, center of the Danish National Research Foundation. This research was partially supported by a grant from the Wolfson Research Awards, administered by the Israeli Academy of Sciences and Humanities.

Part I

Hashing Functions can Simplify Zero-Knowledge Protocol Design (too) [1]

1 Introduction to Part I

Zero-knowledge proof systems, introduced by Goldwasser, Micali and Rackoff [16], are a key tool in the design of cryptographic protocols. The results of Goldreich, Micali and Wigderson [14] guarantee that such proof systems can be constructed for any NP-statement, provided that one-way functions exist. However, the general construction presented in [14] and subsequent works may yield quite inefficient proof systems for particular applications of interest. Thus, developing methodoligies for the design of zero-knowledge proofs is still of interest.

Designing proof systems which are merely zero-knowledge with respect to the honest verifier (i.e., the verifier specified for the system) is much easier than constructing proof systems which are zero-knowledge in general (i.e., with respect to any efficient strategy of trying to extract knowledge from the specified prover). For example, the simple 1-round interactive proof for Graph Non-Isomorphism [2] is zero-knowledge with respect to the honest verifier. Yet, cheating verifiers may extract knowledge from this system and a non-trivial modification, which utilizes proofs of knowledge and increases the number of rounds, is required to make it zero-knowledge in general. Likewise, assuming the existence of one-way function, there exist constant-round interactive proofs for any NP-language which are zero-knowledge with respect to the honest verifier. Yet, constant-round interactive proofs for NP which are zero-knowledge in general are known only under seemingly stronger assumptions and are also more complex (cf., [11]).

In view of the relative simplicity of designing protocols which are zero-knowledge with respect to the honest verifier, a transformation of such protocols into protocols which are zero-knowledge in general (i.e., w.r.t. any verifier) may be very valuable. Assuming various intractability assumptions, such transformations have been presented by Bellare et. al. [2], and Ostrovsky et. al. [23]. A transformation which does not rely on any intractability assumptions has been presented by Damgård in *Crypto93*. His transformation (of honest-verifier zero-knowledge into general zero-knowledge) has two shortcomings. Firstly, it can be applied only to constant-round protocols of the Arthur-Merlin type (i.e., in which the verifier's messages are uniformly distributed in the set of strings of specified length). Secondly, the transformation produces protocols of very high round complexity; specifically, the round complexity of the resulting protocol is linear in the randomness complexity of the original one.

In this part of paper, we improve the round complexity of Damgård's transformation, while preserving the class of interactive proofs to which it can be applied. Our transformation only increases the number of rounds by a factor of two. However, it also increases the error probability of the proof system by a non-negligible amount which can be made arbitrarily small. This increase is inevitible in view of a result of Goldreich and Krawcyzk [12], see discussion in subsection 3.4. Thus, to get proof systems with negligible error probability, one may repeat the protocols resulting from our transformation a non-constant number of times. Still, the resulting proof systems will have much lower round complexity than those resulting from Damgård's transformation.

We preserve some of the positive properties of Damgård's transformation. In particular, our transformation does not rely on any computational assumptions and preserves perfect and almost-perfect (statistical) zero-knowledge. However, unlike Damgård's transformation, the new transformation is not applicable to argument systems (i.e., the BCC model [4]) or to proofs of knowledge.

Our transformation builds on Damgård's work [6]. In his transformation, the random messages

[1] by Ivan Damgård, Oded Goldreich and Avi Wigderson.

[2] To be convinced that G_0 and G_1 are not isomorphic, the verifier randomly selects n random isomorphic copies of each graph, randomly shuffles all these copies together, and asks the prover to specify the origin of each copy.

sent by the verifier (in each round) are replaced by a multi-round interactive hashing protocol, which in turn originates in the work of Ostrovsky, Venkatesan and Yung [22]. Instead, in our transformation, the random messages sent by the verifier are replaced by a $\frac{3}{2}$-round protocol, called Random Selection. The Random Selection protocol uses a family of ordinary hashing functions; specifically, we use a family of t-wise indepedent functions, for some parameter t (which is polynomial in the input length).

We believe that the Random Selection protocol may be of independent interest. Thus, a few words are in place. The goal of this protocol is to allow two parties to select a "random" n-bit string. There is a parameter ε which governs the quality of this selection and the requirement is asymmetric with respect to the two parties. Firstly, it is required that if the first party follows the protocol then, no matter how the second player plays, the output of the protocol will be at most ε away (in norm-1) from uniform. Secondly, it is required that if the second party follows the protocol then, no matter how the first player plays, no string will appear as output of the protocol with probability greater than $\mathrm{poly}(n/\varepsilon) \cdot 2^{-n}$. Our Random Selection protocol has the additional property of being simulatable in the sense that, given a possible outcome, it is easy to generate a (random) transcript of the protocol which ends with this outcome.

Other Related Work

The idea of transforming honest verifier zero-knowledge into zero-knowledge in general was first studied by Bellare, Micali and Ostrovsky [2]. Their transformation needed a computational assumption of a specific algebraic type. Since then several constructions have reduced the computational assumptions needed. The latest in this line of work is by Ostrovsky, Venkatesan and Yung [23], who give a transformation which is based on interactive hashing and preserved statistical zero-knowledge. Their transformation relies on existence of a one-way permutation. The transformation works for any protocol, provided that the verifier is probabilistic polynomial-time.

In the other part of this paper, a secure commitment scheme[3] is used to transform honest-verifier zero-knowledge Arthur-Merlin proofs (with unbounded number of rounds) into (general) zero-knowledge Arthur-Merlin proofs. This transformation increases the round-complexity of the proof system by an additive term which is linear in the number of coin tosses used in the original proof system.

An indirect way of converting protocols which are zero-knowledge with respect to the honest verifier into ones which are zero-knowledge in general, is available through a recent result of Ostrovsky and Wigderson [24]. They have proved that the existence of honest verifier zero-knowledge proof system for a language which is "hard on the average" implies the existence of one-way functions. Combined with the results of [14] and [19, 3], this yields a (computational and general) zero-knowledge proof for the same language. Thus, computational honest-verifier zero-knowledge interactive proofs, for "hard on the average" languages, get transformed into computational zero-knowledge interactive proofs for these languages. However, perfect honest-verifier zero-knowledge proofs (for such languages) do not get transformed into *perfect* zero-knowledge proofs.

A two-party protocol for random selection, with unrelated properties, has been presented in [10]. This protocol guarantees that, as long as one party plays honestly, the outcome of the protocol hits any set $S \subset \{0,1\}^n$ with probability at most $\tilde{O}(\sqrt{|S|/2^n})$, where $\tilde{O}(\varepsilon) \stackrel{\rm def}{=} \varepsilon \cdot \mathrm{polylog}(1/\varepsilon)$.

Another two-party protocol for random selection, with other unrelated properties, has been presented in [13]. Loosely speaking, this protocol allows a computationally restricted party, interacting with a powerful and yet untrustful party, to uniformly select an element in an easily recognizable set $S \subset \{0,1\}^n$.

Remarks Concerning this Part of the Paper

We use the standard definitions of interactive proofs and zero-knowledge, except for the following minor modification. We require the simulator (in the definition of zero-knowledge) to to run in

[3] Secure commitment schemes exist provided that one-way functions exist [18, 20] and the latter exist if some languages which is hard on the average have proof systems which are zero-knowledge with respect to the honest verifier [24].

strictly polynomial-time (rather than in expected polynomial-time), but we allow it to produce output only with some non-negligible probability (rather than always). Clearly, this definition implies the standard one, but the converse is not known to hold – see [9]. This definition is more convenient for establishing zero-knowledge claims and in particular for our purposes, but our results do not depend on it (and can be derived under the standard definitions).

Due to space limitations the proofs of all propositions have been omitted. The complete proofs appear in our technical report [7].

2 Random Selection

We consider a randomized two-party protocol for selecting strings. The two parties to the protocol are called the *challenger* and the *responder*. These names are supposed to reflect the asymmetric requirements (presented below) as well as the usage of the protocol in our zero-knowledge transformation. Loosely speaking, we require that

- if the challenger follows the protocol then, no matter which strategy is used by the responder, the output of the protocol is almost uniformly distributed;
- if the responder follows the protocol then, no string may appear with probability much greater than its probability under the uniform distribution. Furthermore, for any string which may appear as output, when an arbitrary challenger strategy is used, one can efficiently generate a random transcript of that protocol ending with this output.

We postpone the formal specification of these properties to the analysis of the protocol presented below. Actually, we present two version of the protocol.

Construction 1 (Random Selection Protocol – two versions): *Let n and $m < n$ be integers*[4]*, and $H_{n,m}$ be a family of functions, each mapping the set of n-bit long strings onto*[5] *the set of m-bit long strings.*

C1: *the challenger uniformly selects $h \in H_{n,m}$ and sends it to the responder;*

R1:
- (version 1): *the responder uniformly selects $x \in \{0,1\}^n$, computes $\alpha = h(x)$ and sends α to the challenger;*
- (version 2): *the responder uniformly selects $\alpha \in \{0,1\}^m$ and sends it to the challenger;*

C2: *the challenger uniformly selects a preimage of α under h and outputs it.*

We remark that if version 1 is used and both parties follow the protocol then the output is uniformly distributed in $\{0,1\}^n$. However, the interesting case is when one of the parties deviates from the protocol. In this case, the protocol can be guaranteed to produce "good" output, provided that "good" families of hash functions are being used as $H_{n,m}$. These functions must have relatively succient representation as well as strong random properties. Furthermore, given a function h, it should be easy to evaluate h on a given image and to generate a random preimage (of a given range element) under h. Using the algorithmic properties of $H_{n,m}$ it follows that the instructions specified in the above protocol can be implemented in probabilistic $\mathrm{poly}(n/\varepsilon)$-time, which for $\varepsilon = 1/\mathrm{poly}(n)$ means $\mathrm{poly}(n)$-time.

Construction 2 (Preferred family $H^t_{n,m}$): *Let n, $m < n$ and $t = \mathrm{poly}(n)$ be integers. We associate $\{0,1\}^n$ with the finite field $GF(2^n)$ and consider the set of $(t-1)$-degree polynomials over this field. For each such polynomial f, we consider the function h so that, for every $x \in \{0,1\}^n$, $h(x)$ is the m most significant bits of $f(x)$. The family $H^t_{n,m}$ consists of all such functions h. The canonical description of a function $h \in H^t_{n,m}$ is merely the sequence of t smallest coefficients of the corresponding polynomial. Finaly, we modify the functions in $H^t_{n,m}$ so that for each $h \in H^t_{n,m}$ and every $x' \in \{0,1\}^m$ it holds $h(x'0^{n-m}) \overset{\mathrm{def}}{=} x'$.*

[4] In particular, we will use $m \overset{\mathrm{def}}{=} n - 4\log_2(n/\varepsilon)$, where ε is an error-bound parameter.

[5] We stress that each function in $H_{n,m}$ rages over all $\{0,1\}^m$. Thus, the challenger may always respond in step C2 even if the responder deviates from the protocol or version 2 is used.

In the sequel, we will use the family $H_{n,m} \overset{\mathrm{def}}{=} H^n_{n,m}$. We now list the following, easy to verify, properties of the above family.

P1 There is a $\mathrm{poly}(n)$-time algorithm that, on input a function $h \in H^t_{n,m}$ and a string $x \in \{0,1\}^n$, outputs $h(x)$.

P2 The number of preimages of an image y under $h \in H^t_{n,m}$ is bounded above by $2^{n-m} \cdot t$; furthermore, there exists a $\mathrm{poly}(2^{n-m}t)$-time algorithm that, on input y and h, outputs the set $h^{-1}(y) \overset{\mathrm{def}}{=} \{x : h(x) = y\}$. (The algorithm works by trying all possible extensions of y to an element of $GF(2^t)$; for each such extension it remains to find the roots of a degree $t-1$ polynomial over the field.)

P3 $H^t_{n,m}$ is a family of almost t-wise independent hashing functions in the following sense: for every t distinct images, $x_1, ..., x_t \in (\{0,1\}^n - \{0,1\}^m 0^{n-m})$, for a uniformly chosen $h \in H^t_{n,m}$, the random variables $h(x_1), ..., h(x_t)$ are indepedently and uniformly distributed in $\{0,1\}^m$.

2.1 The output distribution for honest challeger

We now turn to analyze the output distribution of the above protocol, assuming that the challenger plays according to the protocol. In the analysis we allow the responder to deviate arbitrarily from the protocol and thus as far as this analysis goes the two versions in Construction 1 are equivalent. The analysis is done using the "random" properties of the family $H^t_{n,m}$. Recall that the statistical difference between two random variable X and Y is

$$\frac{1}{2} \sum_{\alpha} |\mathrm{Prob}(X = \alpha) - \mathrm{Prob}(Y = \alpha)|$$

We say that X is ε-*away* from Y if the statistical difference between them is ε.

Proposition 1 *Let n be an integer, $\varepsilon \in [0,1]$ and $m \overset{\mathrm{def}}{=} n - 4\log_2(n/\varepsilon)$. Suppose that $H_{n,m}$ is a family of almost n-wise independent hashing functions. Then, no matter which strategy is used by the responder, provided that the challenger follows the protocol, the output of the protocol is at most $(2\varepsilon + 2^{-n})$-away from uniform distribution.*

2.2 The output distribution for honest responder

We now show that no matter what strategy is used by the challenger, if the responder follows the protocol then the set of possible outputs of the protocol must constitute a non-negligible fraction of the set of n-bit long strings. This claim holds for both versions of Construction 1. Furthermore, we show that no single string may appear with probability which is much more than 2^{-n} (i.e., its probability weight under the uniform distribution).

Proposition 2 *Suppose that $H_{n,m} = H^t_{n,m}$ is a family of hashing functions satisfying property (P2), for some $t = \mathrm{poly}(n)$. Let C^* be an arbitrary challenger strategy. Then, for every $x \in \{0,1\}^n$, the probability that an execution of* version 1 *of the protocol with challenger strategy C^* ends with output x is at most $(t \cdot 2^{n-m}) \cdot 2^{-n}$.*

Proposition 3 *Let C^* be an arbitrary challenger strategy. Then, for every $x \in \{0,1\}^n$, the probability that an execution of* version 2 *of the protocol with challenger strategy C^* ends with output x is at most 2^{-m}. Furthermore, for every deterministic challenger strategy $\mathbf{c}$, exactly 2^m strings may appear as output, each with probability exactly 2^{-m}.*

2.3 Simultability property of the protocol

We conclude the analysis of the above protocol by showing that, one can efficiently generate random transcripts of the protocol having a given outcome. Throughout this analysis, we assume that the responder follows the instruction specified by the protocol. As in the proof of the last two propositions, it suffices to consider an arbitrary deterministic challenger strategy, denoted $\mathbf{c}$.

Now, suppose that $H_{n,m} = H_{n,m}^t$ is a family of hashing functions satisfying property (P1), for some $t = \mathrm{poly}(n)$. Then, on input x and access to a function $\mathbf{c}\colon \{0,1\}^* \mapsto \{0,1\}^*$, we can easily test if $\mathbf{c}(h(x)) = x$, where $h \stackrel{\mathrm{def}}{=} \mathbf{c}(\lambda)$. In case the above condition holds, the triple $(h, h(x), x)$ is the only transcript of the execution of the protocol, with challenger strategy $\mathbf{c}$, which ends with output x. Otherwise, there is no execution of the protocol, with challenger strategy $\mathbf{c}$, which ends with output x. Thus,

Proposition 4 *Consider executions of the Random Selection protocol in which the challenger strategy, denoted* $\mathbf{c}$*, is an arbitrary function and the responder plays according to the protocol. There exists a polynomial-time oracle machine that, on input* $x \in \{0,1\}^n$ *and* $h \in H_{n,m}$ *and oracle access to a function* $\mathbf{c}$*, either generates the unique transcript of a* $\mathbf{c}$*-execution which outputs* x *or indicates that no such execution exists.*

2.4 Setting the Parameters

Proposition 1 motivates us to set ε (the parameter governing the approximation of the output in case of honest challenger) as small as possible. On the other hand, Propositions 2 and 3 motivates us to maintain the difference $n - m$ small and in paricular logarithmic (in n). Recalling that $n - m = 4\log_2(n/\varepsilon)$, this suggests setting $\varepsilon = 1/p(n)$ for some fixed positive polynomial p.

3 The Zero-Knowledge Transformation

Our transformation is restricted to interactive proofs in which the verifier sends the outcome of every coin it tosses. Such interactive proofs are called *Arthur-Merlin games* [1] or *public-coins interactive proofs* (cf., [17]). Note that in such interactive proofs the verifier moves, save the last, may consist merely of tossing coins and sending their outcome. (In its last move the verifier decides, based on the entire history of the communication, whether to accept the input or not.) Without loss of generality, we may assume that in every round of such an interactive proof the verifier tosses at least $4\log(|x|/\varepsilon)$ coins, where x is the common input to the interactive proof and ε specifies the desired bound on the statistical distance (between one round in the resulting interactive proof and the original one). Furthermore, assume for sake of simplicity that at each round the verifier tosses the same number of coins, denoted n.

3.1 The Transformation

In the following description, we use the second version of the Random Selection protocol presented in Construction 1. This simplifies the construction of the simulator for the transformed interactive proof. The first version can be used as well, at the expense of some modification in the simulator construction.

The protocol transformation consists of replacing each verifier move (except the last, decision move) by an execution of the Random Selection protocol, in which the verifier plays the role of the challenger and the prover plays the role of the responder.

Construction 3 (transformation of round i in (P,V) interaction): *Let* (P,V) *be an interactive proof system in which the verifier* V *only uses public coins, let* $\varepsilon(n) = 1/\mathrm{poly}(n)$ *be the desired error in the Random Selection protocol,* $m \stackrel{\mathrm{def}}{=} m(n) \stackrel{\mathrm{def}}{=} n - 4\log_2(n/\varepsilon(n))$ *and* $H_{n,m}$ *be as specified in Construction 2* (for $t = n$)*. The* i^{th} *round of the* (P,V) *interaction, on common input* x*, is replaced by the following two rounds of the resulting interactive proof* (P',V')*. Let*

$(h_1, \alpha_1, r_1, \beta_1, ..., h_{i-1}, a_{i-1}, r_{i-1}, \beta_{i-1})$ *be the history so far of the interaction between prover P' and verifier V'. Then, the next two rounds consist of an execution of the* (second version of the) *Random Selection protocol follows by P' mimicing the response of P. Namely, in the first round, the verifier V' uniformly selects $h_i \in H_{n,m}$ and sends it to the prover P' who replies with a_i uniformly selected in $\{0,1\}^m$. In the second round, the verifier V' uniformly selects $r_i \in h_i^{-1}(a_i)$ and sends it to the prover P' who replies with $\beta_i \stackrel{\rm def}{=} P(x, r_1, ..., r_i)$.*

The final decision of the new verifier V' mimics the one of the original verifier V; namely,

$$V'(h_1, \alpha_1, r_1, \beta_1, ..., h_t, a_t, r_t, \beta_t) = V(r_1, \beta_1, ..., r_t, \beta_t)$$

3.2 Preservation of Completeness and Soundness

In this subsection, we may assume that V' follows the interactive proof. Thus, if for some $x \in \{0,1\}^*$, prover P always convinces V on common input x then P' always convinces V' on this common input. We stress that both V' and P' run in polynomial-time when given oracle access to V and P, respectively. Thus, the new verifier is a legitimate one. Furthermore, if the original prover P, working in polynomial-time with help of a suitable auxiliary input, could convince the original verifier to accept some common input, then the resulting prover P' could do the same (i.e., can convince V' to accept this common input, while working in polynomial-time with help of the same auxiliary input).

We have just seen that the completeness properties of the original interactive proof is preserved, by the transformation, in a strong sense. Soundness properties are preserved as well, but with some slackness which results from the imperfectness of the Random Selection protocol. In particular,

Proposition 5 *Let $\mu : \{0,1\}^* \mapsto [0,1]$ be a function bounding the probability that verifier V accepts inputs when interacting with any* (possibly cheating) *prover. Namely, $\mu(x)$ is a bound on the probability that V accepts x. Suppose that on input x, the interactive proof (P, V) runs for $t(|x|)$ rounds. Then, $\mu'(x) \stackrel{\rm def}{=} \mu(x) + O(t(|x|) \cdot \varepsilon(|x|))$ is a function bounding the probability that verifier V' accepts inputs when interacting with any* (possibly cheating) *prover.*

proof: Recall that V' plays the role of the challenger in the Random Selection protocol. Thus, the proposition follows quite immediately from Proposition 1. ■

We stress that the above proposition remains valid no matter which of the two version of Random Selection is used. The same holds with respect to the comments regarding completeness (made above).

3.3 Zero-Knowledge

In this subsection, we may assume that P' follows the interactive proof. Assuming that P is zero-knowledge with respect to the verifier V, we prove that P' is zero-knowledge with respect to any probabilistic polynomial-time verifier strategy. This statement holds for the three versions of zero-knowledge; specifically, perfect, almost-perfect (statistical), and computational zero-knowledge.

Proposition 6 *Let (P, V) be a constant-round Arthur-Merlin interactive proof. Suppose that P is perfect* (resp. almost-perfect) [resp. computational] *zero-knowledge* with respect to the honest verifier *V over the set $L \subseteq \{0,1\}^*$. Then P' is perfect* (resp. almost-perfect) [resp. computational] *zero-knowledge* (with respect to any probabilistic polynomail-time verifier) *over the set $L \subseteq \{0,1\}^*$.*

A few comments regarding the proof: Let M be a simulator witnessing the hypothesis of the proposition. Then, for every $x \in L$, with non-negligible probability $M(x)$ halts with output, and given that this happens the output has distributed indistinguishable from that of $(P, V)(x)$. For every verifier strategy V^* interacting with P', we construct a simulator M^*, which uses M and V^* as black-boxes, as follows. By uniformly selecting and fixing coin tosses for V^*, *we may assume that V^* is deterministic.*

On input x, the simulator M^* invokes M and assuming $M(x)$ halts with output, sets $(r_1, \beta_1, ..., r_t, \beta_t) \stackrel{\text{def}}{=} M(x)$; otherwise M^* also halts with no output. The simulator M^* now tries to form transcripts of the Random Selection protocol which end with output r_1, r_2 through r_t, respectively. (Here we use the simulatability of the Random Selection protocol.) A transcript with output r_1 is formed as follows. M^* feeds V^* with input x and obtains h_1, which can be assumed as in Propositions 2 and 3 to be in $H_{n,m}$. Next, M^* computes $a_1 = h_1(r_1)$ and feeds V^* with (x, a_1). If V^* replies with r_1, we've succeeded in forming a transcript for the first invokation of Random Selection and we proceed to the next. (This happens with non-negligible probability.) Otherwise, M^* halts with no output. We note that for the next invokations of Random Selection, V^* is fed with the entire history so far; for example, to obtain h_2 we feed V^* with (x, a_1, β_1) and next we feed it with (x, a_1, β_1, a_2), where $a_2 = h_2(r_2)$. If all t rounds were completed successfully[6], M^* halts with output $(h_1, a_1, r_1, \beta_1, ..., h_t, a_t, r_t, \beta_t)$.

To complete the proof we prove six claims. Firstly, we show that in each of the three cases (perfect, almost-perfect, or computational zero-knowledge), the simulator M^* produces output with non-negligible probability. Secondly, for each of the three cases, we establish the required relationship between the transcript of the real interaction and the output of the simulator. As expected, the proofs become more involved as we move from perfect to computational zero-knowledge.

The above proposition remains valid even if one uses the first version of the Random Selection protocol. However, a slightly more complex simulator will have to be used. The reason being that in the first version (of the Random Selection protocol) the a_i's are not selected uniformly but are rather weighted by the number of their preimages under the corresponding h_i's. Thus, r_i's which are mapped to a_i's with small preimage may be less likely in the real interactions. To compensate for this phenomenon, one may modify the simulator so that it skews the probabilities in the same manner. Namely, when producing a transcript with less likely r_i's, the simulator will discard it with some probability. The required probability (with which to discard transcripts) can be easily computed.

3.4 Conclusions

Combining Propositions 5 and 6, we get

Theorem 1 *Let $\mu : \mathsf{N} \mapsto [0,1]$. Suppose L has a constant-round Arthur-Merlin proof system, with error bound μ, which is perfect* (resp. almost-perfect) [resp. computational] *zero-knowledge* with respect to the honest verifier. *Then, for every positive polynomial $p(\cdot)$, L has a constant-round Arthur-Merlin proof system, with error bound $\mu'(n) \stackrel{\text{def}}{=} \mu(n) + \frac{1}{p(n)}$, which is perfect* (resp. almost-perfect) [resp. computational] *zero-knowledge* (with respect to any probabilistic polynomial-time verifier). *Furthermore, the zero-knowledge property can be demonstrated using a black-box simulation. Also, if the original system had no error on inputs in L then the same holds for the new system.*

Theorem 1 does not preserve the error probability of the original system. This seems inevitible, in light of [12]. Recall that there are languages believed not to be in $\mathcal{BPP}$ which have constant-round Arthur-Merlin proof systems, with exponentially small error probability, which are zero-knowledge with respect to the honest verifier. For example, Graph Isomorphism has such a system (for perfect zero-knowledge), and assuming the existence of one-way functions, every language in $\mathcal{NP}$ has such a system (for computational zero-knowledge) [14]. Now, a stronger version of Theorem 1, say one in which $\mu'(n) - \mu(n)$ is a negligible function of n, would imply that these languages have constant-round Arthur-Merlin (balck-box) zero-knowledge proof systems (with negligible error probability). But, according to [12], languages having constant-round Arthur-Merlin (balck-box) zero-knowledge proof systems lie in $\mathcal{BPP}$. Needless to say that $\mathcal{NP}$ and even Graph Non-Isomorphism are believed not to lie in $\mathcal{BPP}$.

[6]This happens with probability $p(|x|)^t$, where $p(\cdot)$ is the non-negligible probability that we've completed successfuly a single round. This is the reason we can handle any constant number of rounds.

We now compare the round complexity of the protocols resulting from our transformation to those resulting from Damgård's transformation of [6]. Suppose we start with a c-round proof system which uses $r(n)$ random coins and has error $\mu(n)$. Clearly, $\mu(n) \geq 2^{-r(n)}$ and $r(n) > \log_2 n$ (otherwise the language is in BPP [15]). Now, the proof system resulting from Damgård's transformation will have $c + r(n)$ rounds and maintain the error bound of the original proof system. On the other hand, the protocol resulting from our transformation will have $2c$ rounds and error $\mu(n) + \frac{1}{\mathrm{poly}(n)}$. In case $\mu(n)$ is non-negligible, we have a clear advatage. Otherwise, to make the comparison fair, we use sequentail repetitions to reduce the error in the protocols resulting from our transformation to the bound $\mu(n)$. This requires $\log_{\mathrm{poly}(n)}(1/\mu(n))$ repetitions yielding round complexity bounded by $\frac{\log_2(1/\mu(n))}{\log_2 n} \leq \frac{r(n)}{\log_2 n}$. (Typically, $\mu(n)$ is much larger than $2^{-r(n)}$.)

Part II
Using Commitment Schemes to Simplify Zero-Knowledge Protocol Design [7]

4 Introduction to Part II

In this part, we will show another transformation, which can be applied to *arbitrary number of round* statistical zero-knowledge proofs, assuming the existence of secure commitment schemes (i.e., one-way functions [18, 20]). This assumption can be replaced by the restriction on the applicable languages, that they are "hard on the average" (not in $\mathcal{AVBPP}$) [24].

This result can be considered to improve the two previous results partially: one is the result by Ostrovsky, Venkatesan and Yung [23] and the other is by Damgård[6] (see Introduction of Part I). That is, our result generalizes the assumption of [23], from one-way permutations to *one-way functions*, although our transformation is only applicable to *public coin* proof systems. On the other hand, this result relaxes the round complexity restriction for applicable proof systems, from constant number of rounds to *arbitrary number of rounds*, although our transformation does not preserve perfect zero-knowledge, and the applicable languages should not be in $\mathcal{AVBPP}$.

The technique of using the bit-commitment for the transformation can be also applied to the argument model [4]. In this transformation, the roles of the committer and receiver are reversed (i.e., the verifier is the committer.)

5 The Zero-Knowledge Transformation

Theorem 2 *If language L has a statistical zero-knowledge public-coin proof against a "honest verifier", then L has a statistical zero-knowledge public-coin proof against "any verifier", assuming the existence of secure bit-commitment schemes (i.e., one-way functions).*

Proof

Let (M, A) be a statistical zero-knowledge public-coin proof against a "honest verifier", A, for language L. Then we will construct a statistical zero-knowledge public-coin proof, (M^*, A^*), against any verifier, A^*, for L.

We assume

1. If $x \in L$, then $\mathrm{Prob}[(M, A)(x)$ accepts $] \geq 1 - 1/2^n$
2. If $x \notin L$, then for any $\widetilde{M}$, $\mathrm{Prob}[(\widetilde{M}, A)(x)$ accepts $] \leq 1/2^n$,

[7] by Tatsuaki Okamoto.

where n is the size of x.

Suppose that the conversation of $(M, A)(x)$ is $(\alpha_1, \beta_1, \ldots, \alpha_k, \beta_k)$, where α_i $(i = 1, \ldots, k)$ is the i-th public coin message by A, and β_i is the i-th message by M. Let l_i be the (bit) size of α_i.

Let BC be Naor's bit-commitment function based on a pseudo-random generator, G, [20]. That is, Naor's bit-commitment protocol is as follows:

1. [Commit stage:]

   Receiver (R) sends a ($3n$ bits) random string, t, to Committer (C).

   C randomly selects a (n bits) seed, s, of a pseudo-random generator, G, and calculates $BC(s, t, b) = G^{(3n)}(s) \oplus bt$, where $b \in \{0, 1\}$ is the bit C is committed to, bt is t (if $b = 1$) or 0^{3n} (if $b = 0$), and $G^{(3n)}(s)$ is the first $3n$ bits output of $G(s)$. P sends $BC(s, t, b)$ to R.

2. [Reveal stage:]

   C sends s and b to R, and R checks the validity.

A pseudo-random generator exits if and only if a one-way function exists [18].

Next, we show the protocol of (M^*, A^*) using Naor's bit-commitment protocol.

Protocol (M^*, A^*)

Common input: x

What to prove: $x \in L$.

Repeat the following protocol for i from 1 to k sequentially. Here, when $i = j$, we suppose that (M^*, A^*) has already executed the protocol for i from 1 through $j - 1$. (When $i = 1$, suppose that no protocol has been executed before.)

1. Repeat the following protocol for I from 1 to l_i sequentially.

   (a) A^* sends a ($3n$ bits) random string, $t_I^{(i)}$, to M^*.

   (b) M^* randomly selects a (n bits) seed, $s_I^{(i)}$, of a pseudo-random generator, and a random bit, $b_I^{(i)} \in \{0, 1\}$. M^* calculates $BC(s_I^{(i)}, t_I^{(i)}, b_I^{(i)})$, and sends it to A^*.

   (c) A^* sends a random bit, $c_I^{(i)} \in \{0, 1\}$, to M^*.

   (d) M^* sends $s_I^{(i)}$ and $b_I^{(i)}$ to A^*.

   (e) A^* checks the validity of $s_I^{(i)}$ and $b_I^{(i)}$, and if it is invalid A^* halts. Otherwise, go to the next step.

2. M^* sets
$$\alpha_i \leftarrow (b_1^{(i)} \oplus c_1^{(i)}, b_2^{(i)} \oplus c_2^{(i)}, \ldots, b_{l_1}^{(i)} \oplus c_{l_i}^{(i)}).$$
   M^* runs M with α_i as the i-th message by A and gets the i-th message by B, β_i. Here, we suppose that M, given $(\alpha_1, \ldots, \alpha_{i-1})$, has already outputs $(\beta_1, \ldots, \beta_{i-1})$ sequentially. M^* sends β_i to A^*.

Finally, for $i = 1, \ldots, k$, A^* sets
$$\alpha_i \leftarrow (b_1^{(i)} \oplus c_1^{(i)}, b_2^{(i)} \oplus c_2^{(i)}, \ldots, b_{l_1}^{(i)} \oplus c_{l_i}^{(i)}).$$
Then, A^* runs A with $(\alpha_1, \ldots, \alpha_k)$ as A's random string, and $(\beta_1, \ldots, \beta_k)$ as messages from M. If A accepts, then A^* accepts.

[End of Protocol (M^*, A^*)]

[Completeness]

If $x \in L$ and M^* and A^* are honest, then, clearly, (M^*, A^*) accepts x with the same probability by (M, A), where M and A are also honest.

[Soundness]

If $x \notin L$, we will show that for any prover, $\widetilde{M^*}$, $(\widetilde{M^*}, A^*)(x)$ accepts with probability less than $\epsilon(n)$.

First, we assume that there exist $\widetilde{M^*}$ and a constant a such that $(\widetilde{M^*}, A^*)(x)$ accepts with probability greater than $1/n^a$. Here, we suppose that $\widetilde{M^*}$ is deterministic, by selecting the optimum coin flips of $\widetilde{M^*}$ which maximize the accept probability of $(\widetilde{M^*}, A^*)(x)$.

Then we will show that $\widetilde{M^*}$ must break the condition of Naor's bit-commitment.

For any $\widetilde{M}$, $\mathrm{Prob}[(\widetilde{M}, A)(x)$ accepts $] \leq 1/2^n$, and (M^*, A^*) is the same as (M, A) except the procedure of determining $\{\alpha_i\}$. Hence, if $(\widetilde{M^*}, A^*)(x)$ accepts with probability greater than $1/n^a$ for a constant a, then $(\alpha_1, \ldots, \alpha_k)$, which is input to A by A^* to decide the acceptance, must be in a negligible ($< 1/2^n$) fraction, Γ, of $\{(\alpha_1, \ldots, \alpha_k)\}$ with probability greater than $1/n^a$ for a constant a. Here, Γ is fixed when $\widetilde{M^*}$ is fixed.

On the other hand, from the condition of Naor's bit-commitment, the committer $(\widetilde{M^*})$ can change the committed value with probability at most $1/2^n$. Since A^* sends a true random bits c_I (for $I = 1, \ldots, l_i$; $i = 1, \ldots, k$), $e_I^{(i)}$ is uniformly distributed with probability greater than $1 - 1/2^n$. Hence, $(\alpha_1, \ldots, \alpha_k) = (e_1^{(1)}, \ldots, e_{l_k}^{(k)})$ is uniformly distributed with probability greater than $(1 - 1/2^n)^{\sum_{i=1}^k l_i} > 1 - \epsilon(n)$. Therefore, the probability that $(\alpha_1, \ldots, \alpha_k) \in \Gamma$ is at most $(1/2^n)(1 - \epsilon(n)) + \epsilon(n) < \epsilon(n)$.

Thus, if $(\widetilde{M^*}, A^*)(x)$ accepts with probability greater than $1/n^a$ for a constant a, then $\widetilde{M^*}$ must break the condition of Naor's bit-commitment.

[Zero-knowledgeness (Black-box simulation zero-knowledgeness]

When $x \in L$, for any verifier A^*, simulator $\widetilde{S}$ for (M^*, A^*), which utilizes A^* as a black-box, can be constructed as follows:

[Simulator $\widetilde{S}$]

1. For $x \in L$, $\widetilde{S}$ runs Simulator S for (M, A), then gets the simulated conversation, $(\alpha_1, \beta_1, \ldots, \alpha_k)$ of $(M, A)(x)$. Let
$$(e_1^{(i)}, e_2^{(i)}, \ldots, e_{l_i}^{(i)}) = \alpha_i,$$
for $i = 1, \ldots, k$.

2. Repeat the following procedure for i from 1 to k, and for I from 1 to l_i, sequentially. (So, totally, $(\sum_{i=1}^k l_1)$ procedures are repeated sequentially.) We denote each procedure by $[i, I]$. Here, when $i = j$ and $I = J$, we suppose that $\widetilde{S}$ has already executed the procedures for i from 1 through $j - 1$ and the procedures for I from 1 through $J - 1$ in the procedure for $i = j$. (i.e., $[1, 1], \ldots, [1, l_1], \ldots, [j-1, 1], \ldots, [j-1, l_{j-1}], [j, 1], \ldots, [j, J-1]$.)
So, the initial status of A^* in the following procedure is the final status of A^* just before the procedure. Let $Init_{[i,I]}$ be the initial status of A^* in procedure $[i, I]$,

   During the following procedure $[i, I]$, $\widetilde{S}$ can make A^* to $Init_{[i,I]}$ from the first initial status of A^* (i.e., $Init_{[1,1]}$). Since a simulated conversation from [1,1] through $[i, I-1]$ has been fixed, $\widetilde{S}$ can make A^* to $Init_{[i,I]}$ just by simulating the fixed simulated conversation from [1,1] through $[i, I-1]$ again. (Then the execution is straightforward and no trial and error.) (Note: $[i, 0]$ means $[i-1, l_{i-1}]$.) When $i = 1$, suppose that no procedure has been executed before.

   (a) $\widetilde{S}$ runs A^* and gets a ($3n$ bits) string, $t_I^{(i)}$ from A^*.

   (b) $\widetilde{S}$ randomly selects a (n bits) seed, $s_I^{(i)}$, of a pseudo-random generator, and a random bit, $b_I^{(i)} \in \{0, 1\}$. $\widetilde{S}$ calculates $BC(s_I^{(i)}, t_I^{(i)}, b_I^{(i)})$, and gives it to A^*.

   (c) $\widetilde{S}$ runs A^* and gets a bit, $c_I^{(i)} \in \{0, 1\}$, from A^*.

(d) $\widetilde{S}$ checks whether the following equation holds or not:

$$b_I^{(i)} \oplus c_I^{(i)} = e_I^{(i)}.$$

If it holds, then $\widetilde{S}$ goes to the next procedure, $[i, I+1]$. (Note: $[i, l_i + 1]$ means $[i+1, 1]$.) Otherwise, $\widetilde{S}$ makes A^* to $Init_{[i,I]}$ and returns to the first step of this procedure, $[i, I]$.

3. Finally $\widetilde{S}$ arranges these values in the order of (M^*, A^*) protocol, and outputs them.

Next, we will show that $\widetilde{S}$ terminates in expected polynomial-time.

Since A^* is a polynomial time bounded Turing machine, from the property of the bit-commitment protocol,

$$|\mathrm{Prob}[c_I^{(i)} \mid b_I^{(i)} = 0] - \mathrm{Prob}[c_I^{(i)} \mid b_I^{(i)} = 1]| < \epsilon(n).$$

Therefore, if $b_I^{(i)}$ is randomly selected,

$$\mathrm{Prob}[b_I^{(i)} \oplus c_I^{(i)} = e_I^{(i)}] > 1/2 - \epsilon(n).$$

Thus, in each procedure, the expected repetition number is less than $1/(1/2 - \epsilon(n)) < 2 + 4\epsilon(n)$. Clearly, after procedure $[i, I]$ is completed, the simulated conversation from $[1, 1]$ to $[i, I]$ is not affected by the following procedures. (i.e., there is no back-track.) Hence, totally, $\widetilde{S}$ terminates in expected time of polynomial (i.e., $O(2(\sum_{i=1}^{k} l_1) \times T)$; where T is the running time of each procedure described above).

Next, we will show that the simulated conversation is statistically close to the real conversation.

Since this is a black-box simulation, if the simulated messages of M^* is statistically close to the real messages, then the total simulation is also statistically close to the real conversation.

To prove this, it is sufficient to show that the simulated α_i is statistically close to the real one. Since (M, A) is a statistical zero-knowledge proof, the distribution of the simulated $\alpha_i = (e_1^{(i)}, \ldots, e_{l_i}^{(i)})$ (output of simulator S for (M, A)) is statistically close to the uniform distribution. On the other hand, the real α_i is also statistically close to the uniform distribution. This is because: (same as the related part of the proof that $\widetilde{S}$ terminates in expected polynomial-time)

$$|\mathrm{Prob}[c_I^{(i)} \mid b_I^{(i)} = 0] - \mathrm{Prob}[c_I^{(i)} \mid b_I^{(i)} = 1]| < \epsilon(n),$$

and $b_I^{(i)}$ is truly random in the real conversation. Hence,

$$\mathrm{Prob}[e_I^{(i)} = b_I^{(i)} \oplus c_I^{(i)} = 0] > 1/2 - \epsilon(n).$$

Thus, the simulated α_i is statistically close to the real one.

□

We can immediately obtain the following corollary from Theorem 2 and [24].

Corollary 1 *If language L has a statistical zero-knowledge public-coin proof and L is not in $\mathcal{AVBPP}$, then L has a statistical zero-knowledge public-coin proof against "any verifier".*

By using the commitment scheme reversely, we can obtain the following:

Corollary 2 *If language L has a statistical zero-knowledge public-coin argument against a "honest verifier", then L has a statistical zero-knowledge public-coin argument against "any verifier", assuming the existence of secure bit-commitment schemes (i.e., one-way functions).*

References

[1] L. Babai. *Trading Group Theory for Randomness*, Proc. of 17th STOC, pages 421–420, 1985.

[2] M. Bellare, S. Micali and R. Ostrovsky: *The (true) Complexity of Statistical Zero-Knowledge*, Proc. of STOC 90.

[3] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, J. Killian, S. Micali and P. Rogaway: *Everything Provable is Provable in Zero-Knowledge*, Proc. of Crypto 88.

[4] G. Brassard, D. Chaum and C. Crépeau: *Minimum Disclosure Proofs of Knowledge*, JCSS.

[5] G. Brassard, C. Crépeau and M. Yung: *Everything in NP can be Argued in Perfect Zero-Knowledge in a Constant Number of Rounds*, 16th ICALP, pp. 123–136, 1989.

[6] I. Damgård: *Interactive Hashing can Simplify Zero-Knowledge Protocol Design Without Computational Assumptions*, Proc. of Crypto 93.

[7] I. Damgård, O. Goldreich, and A. Wigderson: *Hashing Functions can Simplify Zero-Knowledge Protocol Design (too)*, BRICS Technical Rerport RS-94-39, Nov. 1994.

[8] U. Feige and A. Shamir: *Zero-Knowledge Proofs of Knowledge in Two Rounds*, Advances in Cryptology – Crypto89 (proceedings), pp. 526–544, 1990.

[9] O. Goldreich: *Foundation of Cryptography – Fragments of a Book*, February 1995. Available from the Electronic Colloquium on Computational Complexity (ECCC), `http://www.eccc.uni-trier.de/eccc/`.

[10] O. Goldreich, S. Goldwasser and N. Linial: *Fault-Tolerant Computation without Assumptions: the Two-Party Case*, 32nd FOCS, pp. 447–457, 1991.

[11] O. Goldreich and A. Kahan: *How to Construct Constant-Round Zero-Knowledge Proof Systems for NP*, to appear in Journal of Crypology,

[12] O. Goldreich and H. Krawcyzk: *On the Composition of Zero-Knowledge Proof Systems*, 17th ICALP, pp. 268–282, 1990.

[13] O. Goldreich, Y. Mansour and M. Sipser: *Proofs that Never Fail and Random Selection*, Proc. of FOCS 87.

[14] O. Goldreich, S. Micali and A. Wigderson: *Proofs that yield Nothing but their Validity and a Methodology of Cryptographic Protocol Design*, Proc. of FOCS 86.

[15] O. Goldreich and Y. Oren: *Definitions and Properties of Zero-Knowledge Proof Systems*. Jour. of Crypto., Vol. 7, pp. 1-32, 1994.

[16] S. Goldwasser, S. Micali and C. Rackoff: *The Knowledge Complexity of Interactive Proof Systems*, SIAM J. Computing, Vol. 18, pp. 186–208, 1989.

[17] S. Goldwasser and M. Sipser. *Private Coins versus Public Coins in Interactive Proof Systems*, Proc. of 18th STOC, pages 59–68, 1986.

[18] J. Hastad, R. Impagliazzo, L.A. Levin and M. Luby: *Construction of Pseudorandom Generator from any One-Way Function*, manuscript, 1993. See preliminary versions by Impagliazzo et. al. in 21st STOC and Hastad in 22nd STOC.

[19] R. Impagliazzo and M. Yung, *Direct Minimum-Knowledge Computations*, Advances in Cryptology - Crypto87 (proceedings), 1987, pp. 40–51.

[20] M. Naor: *Bit Commitments from Pseudorandomness*, Proc. of Crypto 89.

[21] M. Naor, R. Ostrovsky, R. Venkatesan and M. Yung: *Zero-Knowledge Arguments for NP can be Based on General Complexity Assumptions*, Proc. of Crypto 92.

[22] R. Ostrovsky, R. Venkatesan and M. Yung: *Fair Games Against an All-Powerful Adversary*, presented at DIMACS Complexity and Cryptography Workshop, October 1990, Princeton.

[23] R. Ostrovsky, R. Venkatesan and M. Yung: *Interactive Hashing Simplifies Zero-Knowledge Protocol Design*, Proc. of EuroCrypt 93.

[24] R. Ostrovsky and A. Wigderson: *One-Way Functions are Essential for Non-Trivial Zero-Knowledge*, Proc. 2nd Israel Symp. on Theory of Computing and Systems, 1993.