Interactive Hashing can Simplify Zero-Knowledge Protocol Design Without Computational Assumptions

Extended Abstract

Ivan B. Damgård

Aarhus University

Abstract. We show that any 3-round protocol (in general, any bounded round protocol) in which the verifier sends only random bits, and which is zero-knowledge against an *honest* verifier can be transformed into a protocol that is zero-knowledge *in general*. The transformation is based on the interactive hashing technique of Naor, Ostrovsky, Venkatesan and Yung. No assumption is made on the computing power of prover or verifier, and the transformation therefore is valid in both the proof and argument model, and does not rely on any computational assumptions such as the existence of one-way permutations. The technique is also applicable to proofs of knowledge. The transformation preserves perfect and statistical zero-knowledge. As corollaries, we show first a generalization of a result by Damgård on construction of bit-commitments from zero-knowledge proofs. Other corollaries give results on non-interactive zero-knowledge, one-sided proof systems, and black-box simulation.

1 Introduction

In this paper, we consider protocols in which a prover tries to convince a verifier that some claim is true. Some protocols can be shown to not reveal anything to the verifier, other than the fact that the claim is indeed true, by demonstrating that the verifier could have simulated the protocol himself. Such a protocol is said to be zero-knowledge [8]. Such protocols can be considered in the proof-model, where the prover is unbounded while the verifier is polynomial time restricted; or in the argument model, where the prover is poly-time bounded, while the verifier may (in most cases) be unbounded [1].

It is well known that the design of zero-knowledge proof or arguments is a difficult task. A main complicating factor is the demand that the protocol must be secure against dishonest behavior by both the prover and the verifier. For example, by allowing the prover too much control over the conversation in an effort to protect her privacy, we risk also allowing a dishonest prover to cheat. It would be much easier if we could assume that the verifier was honest.

[1] This does not mean that an honest verfier needs infinite computing power to execute the protocol, only that the protocol is secure, even against a cheating verifier with unbounded resources

D.R. Stinson (Ed.): Advances in Cryptology - CRYPTO '93, LNCS 773, pp. 100-109, 1994.
© Springer-Verlag Berlin Heidelberg 1994

In this paper, we consider this problem for protocols in which the verifier sends only random bits, sometimes called public-coin protocols. This is quite a large class of protocols, for example the general zero-knowledge proof systems and arguments for any NP-problem in [7], [2] are of this type. where the private are

For any bounded round public-coin protocol secure (zero-knowledge) against an *honest* verifier, we present a generic method transforming it into a protocol that is zero-knowledge against *any* verifier. This is based on the interactive hashing technique of [10]. The transformation preserves the proving capabilities of the original protocol, i.e. if it was a proof system, resp. an argument, resp. a proof of knowledge, the resulting protocol will be a proof/argument/proof of knowledge for the same problem. Also, if the original protocol was perfect or statistical zero-knowledge, so is the transformed protocol. In this paper, we show explicitly only the case of 3-round protocols. The generalization to any bounded number of rounds is in principle simple, but the proof is technically cumbersome. It is not clear how to generalize to any polynomial number of rounds. We will return to this problem later.

As a corollary of the transformation, we obtain a generalization of the result from [5], on constructing bit commitments from a 3-move public coin zero-knowledge proof of knowledge. The result from [5] needed a restriction on the number of random bits sent by the verifier, or on the error probability of the protocol; we show that this restriction is unnecessary.

Another corollary is that any non-interactive zero-knowledge proof can be transformed into an ordinary interactive zero-knowledge proof for the same problem. Also this transformation preserves perfect and statistical zero-knowledge.

Finally, we show that for bounded round, public coin, statistical zero knowledge proofs, requiring one-sidedness (i.e. completeness with probability 1) or black-box simulation is not a restriction on the set of statements that can be proved.

2 Related Work

The idea of transforming honest-verifier zero-knowledge into zero-knowledge in general was first studied by Bellare, Micali and Ostrovsky [4]. 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 [11], who give a transformation which is based on interactive hashing and preserves statistical zero-knowlegde. This transformation works for any protocol, but only in the zero-knowledge proof model, in which the verifier is assumed to be polynomial time bounded: the transformation relies on the existence of a one-way permutation which the verifier cannot invert.

Thus compared to [11], the contribution of this paper is a new technique for using interactive hashing, showing that if we restrict to bounded round public-coin protocols, we can get a transformation that does not need any computa-

tional assumptions and therefore works even if the verifier (and/or the prover) is unbounded. Moreover, our transformation preserves both perfect and statistical zero-knowledge.

Finally some recent independent work should be mentioned, in which Ostrovsky and Wigderson [12] show that the existence of honest verifier zero-knowledge proofs for non-trivial (i.e. non-BPP) problems impliy the existence of certain kinds of one-way functions. It appears that this could be immediately combined with the result in [3] that everything provable is provable in computational zero-knowledge assuming that one-way functions exist. This would give a transformation without assumptions (but one that would not preserve statistical or perfect zero-knowledge). However, the notion of one-way functions used in [12] is technically different from the standard one needed in [3], and the results therefore cannot be combined without solving some technical problems.

3 Notation and Definitions

In this section, the technical definitions and notation for zero-knowledge and probabilistic algorithms are given.

As the results below are only presented informally in this extended abstract, we omit technicalities here and present a minimum of notation.

We will restrict ourselves to 3-move public coin protocols for simplicity.

Thus we can describe the protocol we start with as follows: common input to the prover and verifier is a word x of length n bits in a language L. The prover P sends a message m_1, receives a random bit string c from the verifier V. Finally P sends a message m_2 to V, who then outputs accept or reject. We let $t = t(n)$ be the length of c.

Any function of n converging to 0 faster than any polynomial fraction will be called negligible.

We assume the following about the protocol: if $x \in L$, and P, V are honest, the probability that V accepts is at least $1 - \epsilon(n)$, where ϵ is negligible. If $x \notin L$, then the probability that V accepts, when talking to an arbitrary prover P^* is negligible. For arguments, we need to replace "arbitrary prover" by "any polynomial time prover". We do not make this explicit here, however, because the results below do not depend on any such restriction.

Finally, we assume that the protocol is zero-knowledge against the honest verifier: there is an expected polynomial time probabilistic machine S, which on input x produces a simulated conversation which is indistinguishable from the real conversation between P and the honest V on input x. We do not make explicit here, which flavor of zero-knowledge we talk about (perfect, statistical or computational) because the construction to follow will work for all flavors. In accordance with the usual models, we do assume, however, that cheating verifiers are polynomial time bounded in the case of computational zero-knowledge, but may be unbounded in case of statistical or perfect zero-knowledge.

4 The Transformation

For the transformation of the protocol, we need the technique of interactive hashing [10], which we repeat here with some changes to match our context: we work with the vector space $GF(2)^t$, and we assume that P is capable of computing some function g on values in this space.

1. P selects a random t-bit vector c, which is kept secret from V.
2. V selects at random $t-1$ vectors in $GF(2)^t$, $h_1, ..., h_{t-1}$, such that the h_i's are linearly independent over $GF(2)$.
3. For $j = 1$ to $t-1$:
   - V sends h_j to P.
   - P sends $b_j := h_j \cdot c$ (the inner product) to V.
4. Both parties compute the two vectors c_1, c_2, with the property that $c_i \cdot h_j = b_j$ for $j = 1..t-1$, where of course c is one of c_1, c_2. We say that the hashing isolates the values c_1, c_2 and that P answers consistently with c.
5. V sends $v = 1$ or 2 to P.
6. P returns $g(c_v)$ to V.

In [10], the following is proved about this procedure (some of the wording has been changed):

Lemma 1 At the end of Step 4, an arbitrary cheating V^* playing the role of V has no Shannon information about which of c_1, c_2 P has answered consistently with.

Lemma 2 Let P^* be any machine which plays the role of P and is capable of returning both $g(c_1)$ and $g(c_2)$ with probability ϵ. Then there is a probabilistic polynomial time algorithm M using P^* as an oracle, which can, on input a random c, compute $g(c)$ with probability $T(t, \epsilon)$, where T is a function polynomial in $1/t$ and ϵ. This probability is over the choice of c and internal coin tosses of M.

While the proof of Lemma 1 is easy, the proof of Lemma 2 is very technical and complicated. We refer to [10] for details.

The reader may notice that the counterpart of P^* in [10] (called S' there) was assumed to be polynomial time bounded. But this was only necessary there because g was the inverse of a one-way permutation, and the purpose was to show in a proof by contradiction that S' could compute this inverse. However, the analysis in [10] essentially shows that *for any* strategy used by P^* to choose the b_j's, at least one of c_1, c_2 will be almost uniformly chosen. Their construction of M will therefore work in our case for any P^*.

We are now ready to show our transformed protocol for proving that $x \in L$. The prover and verifier from the original protocol will be called P_0, V_0.

Intuitively, the reason why the original protocol may not be secure against an arbitrary verifier V^*, is that even though we can, using S, generate valid looking transcripts m_1, c, m_2 of the protocol where c is uniform, this distribution of c may not be the one that V^* would generate. In particular, there could be some dependency between c and m_1 that a random c-value has only negligible probability of satisfying. Hence trying to use the honest-verifier simulator directly might take expected exponential time. We therefore apply in the transformation the interactive hashing to cut down the number of possible c's to a small number. This increases our chance of hitting the right c-value in the simulation enough to make the expected time polynomial. We cut down to 2 possibilities for simplicity because this allows us to use the results of [10] without modifications. In practice, one could gain efficiency by stopping the hashing earlier, leaving more possibilities for c. But note that if the transformation is to remain provably secure, the number of possible values must be kept polynomial.
The transformed protocol consists of repeating the following $n = |x|$ times:

1. P starts running P_0 on input x. She sends the m_1 generated by P_0 to V.
2. P and V go through the interactive hashing process described above. The value $g(c)$ is defined to be the m_2 P_0 would return given that the initial message was m_1 and the challenge from V_0 was c. Therefore, to compute $g(c_v)$, P passes c_v on to P_0, gets m_2 back and sends it to V.
3. V uses V_0 to decide if the conversation m_1, c_v, m_2 would lead to accept by V_0. If so, he outputs accept, otherwise he rejects.

Remark: to improve readability, we have simplified things a little by defining g as a simple function of c: in general there may be more than one valid answer from P given that the first part of the conversation was m_1, c, so that $g(c)$ would in fact be a random variable. This makes no difference in the following, however.

Lemma 3 If (P_0, V_0) is perfect resp. statistical resp. computational honest verifier zero-knowledge, then (P, V) is perfect resp. statistical resp. computational zero-knowledge.

Proof Sketch We show a simulator S^*, working with any verifier V^*. One iteration of steps 1-3 above will be called a round. It is sufficient to show how to simulate 1 round:

1. Run the honest-verifier simulator S on input x. This results in a conversation m_1, c, m_2. Send m_1 to V^*.
2. Go through the interactive hashing with V^* while answering consistently with c.
3. Receive v from V^*. If $c = c_v$, send m_2 to V^*, stop. Else rewind V^* to the start of Step 1. Go to 1.

It is clear that, by Lemma 1, the probability that $c = c_v$ would be exactly $1/2$ if the c we use in Step 2 had been uniformly chosen. The c we actually use is produced by S. In the perfect/statistical zero-knowledge case c cannot be distinguished from a uniform choice by any unbounded verifier. In the computational case c is only computationally indistinguishable from a uniform choice, but the difference cannot be told by a polynomially bounded verifier. Hence in any case the probability that $c = c_v$ is away from $1/2$ by at most a negligible amount, and therefore simulation of one round needs an expected number of rewinds that is $O(1)$. Thus the complete simulation takes expected linear time.

To show correctness of the output distribution, first note that it is sufficient for all flavors of indistinguishability to show that the simulation of 1 round is indistinguishable from a real execution, since we have used a back-box simulation, which is closed under serial composition.

Then observe the following: the messages sent by the verifier V^* in real conversations are computed from m_1 chosen by P_0 and answers from P consistent with a random c-value. Our simulator uses m_1, c as produced by S, but otherwise uses the same algorithm as P to compute the answers in the interactive hashing. The final message m_2 is a random sample of P_0's final message, given that the first part of the conversation was m_1 and c_v as chosen by V^*. In the simulation we have a random sample of what S would produce, given that the first part of the conversation was m_1, c_v.

This immediately implies that if the output of S is perfectly indistinguishable from real conversations between P_0 and V_0, then the output of S^* is perfectly indistinguishable from conversations between P and V^*. Statistical indistinguishability introduces at most a negligible deviation from real conversations.

For the computational case, we give a proof by contradiction: assume that we have a successful distinguisher D for S^*. Then we can turn D into a distinguisher for S: it is well known that indistinguishablity does not depend on whether the distinguisher gets 1 or any polynomial number of samples of the distributions in question. So assume we are given a list L containing a linear number of samples produced by either S or P_0, V_0. We then use the algorithm of S^* to make from this a conversation C, seemingly between P and V^*. This is done by changing Step 1 such that we take the next sample in the input list, in stead of running S. The only problem is if many rewinds are necessary, so that L is exhausted before we finish. But this only happens with negligible probability, as we have a linear number of samples. We give the conversation produced to D and output its result.

It is now clear that if L was produced by S, C will be statistically indistinguishable from the output of S^*, while if it was produced by P_0, V_0, C will be statistically indistinguishable from conversations between P and V. Therefore, if D is a successful distinguisher for S^*, we get a successful distinguisher for S.

Lemma 4 For any prover P^* that convinces the verifier in step 1-3 of the transformed protocol with probability $1/2 + \epsilon$, there is a prover P_0^* in the original protocol which convinces V_0 with probability $T(t(n), \epsilon)$, where T is a function

polynomial in $1/t(n)$ and ϵ. P_0^* is polynomial time, using P^* as an oracle.

Proof We show the algorithm of P_0^*:

1. Start running P^*. Get m_1, and send it to V_0.
2. Receive c from V_0.
3. Use the procedure of Lemma 2 to get an valid $m_2 = g(c)$ from P^*. Send it to V_0

To see that this works, observe that the success probability of $1/2 + \epsilon$ of P^* implies that for at least a fraction ϵ of the possible choices of the h_j's, we can, by rewinding P^* get correct replies to both $v = 1$ and 2. Therefore P^* satisfies the requirements of Lemma 2, and we get that we convince V_0 with probability $T(t(n), \epsilon)$.

Theorem 1 Assume that a language L has an honest verifier zero-knowlegde proof system, resp. argument that is a 3-round public coin protocol. Any such protocol can be transformed into an ordinary zero-knowledge proof, resp. argument for L.

Similarly, any honest verifier zero-knowledge proof of knowledge for a predicate Pre that is a 3-round public coin protocol can be transformed into an ordinary zero-knowledge proof of knowledge for Pre.

Both types of transformations preserve perfect and statistical zero-knowledge. The transformations are efficient in the sense that provers and verifiers in the transformed protocols are polynomial time machines that use the original provers and verifiers as oracles.

Proof In all cases, the transformation given above will work.

The statements on zero-knowledge are clear from Lemma 3.

If the original protocol is a proof system (an argument), it is clear from Lemma 4 that the transformed protocol is also a proof system (an argument).

Lemma 4 also shows that if the original protocol was a proof of knowledge, so is the transformed one: we can construct a knowledge extractor for P^*, by first using Lemma 4 to get a prover P_0^* in the original protocol and then use the knowledge extractor we know exists for the original protocol.

Remark on generalizations of Theorem 1 To generalize Theorem 1 to any bounded number of rounds, we make the transformation by simply doing one interactive hashing process for each random string sent by the verifier. Zero-knowledge is proved essentially as before. Showing soundness of the transformed proof can be done by proving a multi-round version of Lemma 4 by essentially applying the technique of Lemma 2 once for each interactive hashing done. It appears, however, that the overall success probability of the resulting reduction will tend to 0 exponentially in the number of rounds, and therefore it is not clear how to generalize Theorem 1 to any polynomial number of rounds.

5 Consequences of the main result

5.1 Bit Commitments from Zero-Knowledge Proofs

In [5], it was shown how to construct bit commitments from 3-move public-coin zero-knowledge proofs of knowledge. These commitments will hide the bits committed to statistically, resp. perfectly if the protocol used is statistical, resp. perfect zero-knowledge.

The technique used in [5], however, only works if either the number of bits sent by the verifier is constant, or the error probability decreases sufficiently fast as a function of the security parameter.

What we have shown in the previous section is how to transform any 3-move public coin protocol into one that is essentially equivalent to a protocol where the verifier sends only 1 bit. Hence the technique from [5] can be used on the transformed protocol, and we get directly from Theorem 1:

Corollary 1 Assume there exists any 3-move public coin zero-knowledge proof of knowledge for a problem of which hard instances can be sampled efficiently. Then bit commitments exist. The commitments will hide the bits committed to statistically, resp. perfectly if the protocol used is statistical, resp. perfect zero-knowledge. The commitments are computationally binding.

The prime interest of this result is its ability to produce perfectly or statistically hiding bit commitments. Such commitments otherwise seem to require one-way permutations or collision intractable hash functions, and neither assumption seems to follow from the assumption of Corollary 1.

5.2 Non-Interactive Zero-Knowledge

In the non-interactive zero-knowledge model [1], prover and verifier both have access to a uniformly chosen, random bit string. Based on this string, the prover can produce a proof consisting of just 1 message, that can be checked by the verifier. A simulator in this model produces both a simulated shared random string and a proof. Security from both the verifier's and prover's point of view is based on trust in the randomness of the shared string.

Thus the non-interactive model is less powerful than the ordinary one because interaction is not allowed, but more powerful because a shared random string is assumed as a part of the model. It is therefore not immediately clear, if a language that has a non-interactive (perfect) zero-knowledge proof also has an ordinary (perfect) zero-knowledge proof.

From Theorem 1, however, we can get the following:

Corollary 2 If a language L has a non-interactive perfect/statistical/computational zero-knowledge proof, then L has an ordinary perfect/statistical/computational zero-knowledge proof.

Proof Just observe that from the non-interactive proof we can trivially get an interactive honest-verifier zero-knowledge proof, by just letting the verifier choose the "shared" random string, and then let the prover respond with the proof. This is of course a 3-round public coin protocol (where the prover's first message is empty), so we can use Theorem 1.

5.3 One-sidedness and Black-box Simulation

In [11], Ostrovsky, Venkatesan and Yung show a transformation from honest verifier zero-knowledge to ordinary zero-knowledge. As mentioned in the introduction, this transformation works for both public and secret coin protocols, but needs to assume existence of one-way permutations. They show show two implications, which also rely on one-way permutations. These results were first shown in [4], under a stronger computational assumption. With our result, we get those two corollaries for bounded round public coin protocols without computational assumptions.

In [6], it is shown that any language which has an Arthur-Merlin proof (public coin protocol), also has a *one-sided* Arthur-Merlin proof, i.e. one in which the verfier always accepts if the common input is in L. Their proof is a transformation that builds a one-sided proof system in which the number of rounds in the original proof system is preserved. The transformed protocol is not necessarily zero-knowledge, even if the original protocol was. But it can (with minor modifications) be shown to be honest verifier statistical zero-knowledge, if the original protocol was honest verifier statistical zero-knowledge. This means that we get the following:

Corollary 3 If a language L has a bounded round public coin statistical zero-knowledge proof system, it also has a one-sided public coin statistical zero-knowledge proof system.

A final implication concerns black-box simulation, which is a special case of zero-knowledge, in which the simulator is only allowed to use the verifier as an oracle. It is unknown in general whether black-box simulation is more restrictive than the most liberal definition, where the simulator is allowed to depend on the verifier. But for public coin bounded round protocols, we can show that black box simulation is not a restriction for any flavor of zero-knowledge, simply from the fact that the simulation guaranteed by Theorem 1 and its generalization is black-box:

Corollary 4 If a language L has a bounded round public coin perfect, statistical or computational zero-knowledge proof system, it also has a public coin perfect, statistical or computational *black-box* zero-knowledge proof system.

6 Open Problem

The main open problem arising from the results in this paper is of course whether a *general* transformation from honest verifier zero-knowledge can be found that preserves perfect and statistical zero-knowledge. It is worth noting that this problem would be immediately solved if perfectly hiding bit commitments could be implemented based on any one-way function. But also this problem is open so far.

References

1. Blum, De Santis, Micali and Persiano: *Non-Interactive Zero-Knowledge*, SIAM Journal of Computing, vol.20, no.6, 1991.
2. G.Brassard, D.Chaum and C.Crépeau: *Minimum Disclosure Proofs of Knowledge*, JCSS.
3. Ben-Or, Goldreich, Goldwasser, Håstad, Killian, Micali and Rogaway: *Everything Provable is Provable in Zero-Kowledge*, Proc. of Crypto 88.
4. Bellare, Micali and Ostrovsky: *The (true) Complexity of Statsitical Zero-Knowledge*, STOC 90.
5. I.Damgård: *On the Existence of Bit Commitment Schemes and Zero-Knowledge Proofs*, Proc. of Crypto 89, Springer Verlag LNCS series.
6. Goldreich, Mansour and Sipser: *Poofs that Never Fail and Random Selection*, FOCS 87.
7. O.Goldreich, S.Micali and A.Wigderson: *Proof that Yield Nothing but their Validity and a Methodology of Cryptographic Protocol Design*. Proc. of FOCS 86.
8. S.Goldwasser, S.Micali and C.Rackoff: *The Knowledge Complexity of Interactive Proof Systems*, SIAM J.Computing, Vol.18, pp.186-208, 1989.
9. Impagliazzo and Yung: *Direct Minimum-Knowledge Computations*, Crypto 87.
10. Naor, Ostrovsky, Venkatesan and Yung: *Zero-Knowledge Arguments for NP can be based on General Complexity Assumptions*, Proc. of Crypto 92, Springer Verlag LNCS series.
11. Ostrovsky, Venkatesan and Yung: *Interactive Hashing Simplifies Zero-Knowledge Protocol Design*, to appear in Proc. of EuroCrypt 93, Springer Verlag LNCS Series.
12. Ostrovsky and Wigderson: *One-way functions are essntial for non-trivial zero-knowledge proofs*, preliminary manuscript.