Interactive Hashing Simplifies Zero-Knowledge Protocol Design

Rafail Ostrovsky* Ramarathnam Venkatesan† Moti Yung‡

(Extended abstract)

Abstract

Often the core difficulty in designing zero-knowledge protocols arises from having to consider every possible *cheating* verifier trying to extract additional information. We here consider a compiler which transforms protocols proven secure only with respect to the *honest* verifier into protocols which are secure against any (even cheating) verifier. Such a compiler, which preserves the zero-knowledge property of a statistically or computationally secure protocol was first proposed in [BMO] based on Discrte Logarithm problem. In this paper, we show how such a compiler could be constructed based on any one-way permutation using the recent method of *interactive hashing* [OVY-91]. This applies to both statistically and computationally secure protocols, preserving their respective security. Our result allows us to utilize DES-like permutations for such a compiler.

* University of California at Berkeley Computer Science Division, and International Computer Science Institute at Berkeley. E-mail: rafail@melody.berkeley.edu. Supported by NSF postdoctoral fellowship and ICSI. Part of this work was done at Bellcore and part at IBM T.J. Watson Research Center.

† Bellcore, Room 2M-344, 445 South St, Morristown, NJ 07960. E-mail: venkie@bellcore.com.

‡ IBM Research, T.J. Watson Research Center, Yorktown Heights, NY 10598. E-mail: moti@watson.ibm.com.

1 Introduction

An interactive proof involves two communicating parties, a prover and a verifier. The prover is computationally unbounded; alternatively, in applications, it is a polynomial-time machine possessing additional private knowledge. It tries to convince the probabilistic polynomial time verifier that a given theorem is true.

A zero-knowledge (ZK) proof is an interactive proof with an additional privacy constraint: the verifier does not learn why the theorem is true [GMR]. That is, whatever the polynomial-time verifier sees in a ZK-proof with the unbounded prover of a true theorem x, can be approximated by a probabilistic polynomial-time machine working solely on input x. A statistical zero-knowledge proof (SZK proof) is one for which this true view and approximate view are (information-theoretically) indistinguishable.

A methodology suggested in [BMO] is to design statistical or computational zero-knowledge protocols by assuming a canonical behavior of the verifier, and then translate such protocols to those where cheating is allowed. The mechanism proposed there, as well as the one in [GKa, NY] (for computational zero-knowledge proofs only) uses specific algebraic assumptions to achieve it.

The task of finding the necessary and sufficient complexity conditions needed for various primitives has attracted a lot of work, showing that many primitives, originally based on specific algebraic functions, need only one-way functions or permutations. For example, pseudo-random generators [BM-84], secure signature schemes [GoMiRi], computational ZK-proofs [GMR] were shown to be equivalent to the existence of general one-way functions [ILL, Ha-90, NY, Ro, OW]. Such efforts, not only develop the theoretical foundations of cryptography, but also enable the primitive implementations to be based on a larger possible concrete choices of underlying functions, thus making them more plausible.

The method of *interactive hashing* has been recently developed in [OVY-91] and applied [OVY-91, NOVY] to zero-knowledge arguments and information theoretically secure Oblivious Transfer protocols (also to commitments by/to powerful non-polynomial parties [OVY-92]). Here we show an extended use of this method with zero-knowledge protocols to provide a ZK-protocol design tool along the line of [BMO], but based on the existence of any *one-way permutation*. In particular, assuming that one-way permutations exist, we show that if a language L has a honest-verifier statistical zero-knowledge proof, then L has a (general) statistical zero-knowledge proof. We remark that our method applies to computational zero-knowledge as well. Previously, specific algebraic assumptions were needed in order to implement such tools [BMO, GKa, NY].

1.1 Organization of the paper

In section 2, we give the model and definitions. In Section 3, we present the main result on compiling protocols zero-knowledge against a honest verifier to general zero-knowledge protocols, and we show some implications. Section 4 outlines the compiler and its proof.

2 Definitions

We use standard notions of Turing machines (TM) and probabilistic polynomial time TM's (PPT), and interactive Turing machines [GMR]. We adopt the standard definition of computational and statistical indistinguishability (see, for example, [ILL, GMR]). Let us recall definitions of interactive proofs and zero-knowledge proofs, introduced and formalized in [GMR].

We assume that prover P is a probabilistic, infinite power, interactive TM and verifier V is a probabilistic, poly-time interactive TM [GMR]. We consider interactions between P and V, where they share the same input and can communicate. We say P convinces V to accept on x if P and V have common input x, and after the interaction V accepts. Let *view* of V be the transcript of the conversation between P and V which consists of all the messages between P and V and the portion of the random tape used by V (i.e. random coin tosses of V).

P and V form an interactive protocol for language L with security parameter k (k is the length of the input string), if the following two conditions are satisfied:

- **Completeness**: For all $x \in L$, P convinces V to accept with probability greater than $1 - \frac{1}{2^k}$, where probability is taken over coin tosses of P and V.
- **Soundness**: For all P' and for all $x \notin L$ probability that P' convinces V to accept on x is less than $\frac{1}{2^k}$.

IP $(= PSPACE)$ is the class of languages which can be accepted satisfying completeness and soundness conditions.

The zero-knowledge property:
For every PPT verifier V' let $M_{V'}$ be the probabilistic poly-time TM. The goal of $M_{V'}$ is to simulate the *view* of V', i.e. the conversation between P and V' on x. As such, it must produce a pair: $<$random tape used by V', conversation between P and $V' >$. We restrict our simulators to be average-PPT TM. An interactive protocol is *Statistical Zero-Knowledge* if for all V' there exists $M_{V'} \in PPT$ such that for all $x \in L$, the distributions of the conversation between P and V' on x and $M_{V'}(x)$ is statistically close. If the two distributions are computationally indistinguishable, this corresponds to *Computational Zero-Knowledge*.

Zero-knowledge with respect to honest verifier:
Finally, we are ready to specify what does it mean to have a protocol which works for *honest verifier* only. An interactive protocol is *Statistical Zero-Knowledge for Honest Verifier* if for the honest V (i.e. the one specified in the description of P, V) there exists $M_V \in PPT$ such that for all $x \in L$, the distributions of the conversation between P and V on x and $M_V(x)$ are statistically close. Similar definition holds for *Computational Zero-Knowledge Protocols for Honest Verifier*.

Let f be a length preserving function $f: \{0,1\}^* \rightarrow \{0,1\}^*$ computable in polynomial time.

Definition 2.1 [One-way function.] *f is one-way if for every probabilistic polynomial time algorithm A, for all polynomials p and all sufficiently large n,*

$$Pr[f(x) = f(A(f(x))) \mid x \in_R \{0,1\}^n] < 1/p(n).$$

If addition, if f is a permutation on $\{0,1\}^n, n > 0$, then we say that f is a *one-way permutation*. The above definition is of a *strong one-way function*. Its existence is equivalent to the existence of the weak one-way function [Y82]; a stronger equivalence is possible in the case of permutations (see [GILVZ]). A weak one-way function has the same definition as above, except the probability of successful inversion above is $1 - 1/n^c, c > 0$.

3 Main Result

We show that if there is any one-way permutation, then "honest verifier zero knowledge" is in fact just as strong as zero-knowledge.

Theorem 3.1 Suppose a one-way permutation exists. If a language L has an honest verifier statistical (respectively computational) zero knowledge protocol, then L has a statistical (respectively computational) zero knowledge protocol.

We remark that our transformation is constructive and that error probabilities are preserved, as in [BMO], it also works for zero-knowledge proof of knowledge.

3.1 Implications

The theorem has a few implications on languages and their proof systems (beyond giving a design tool). We discuss those briefly.

- *Black-box simulation:*
  Oren [Or] formalized the black box notion by saying that the simulator is a PPT oracle machine M which when asked to simulate a particular verifier $\widehat{V}$ is given that verifier as an oracle. Thus the same simulator works for all verifiers. Using our method we show that assuming any one-way permutation, black box simulation is not a restriction on zero-knowledge, i.e.: Suppose L has a (honest verifier) SZK (ZK) protocol and one-way permutation exists. Then, L has a black box simulation SZK (ZK) protocol.

- *Error probability one-sidedness* :
  Goldreich, Mansour and Sipser [GMS] define a one-sided proof system to be one in which completeness holds with probability 1 (that is the prover can always convince the verifier). An implication of our protocol tool is: If L has a (honest verifier) SZK proof system and one-way permutation exists. Then, L has a SZK one-sided proof system.

4 The Protocol Compiler and its Proof

Given a zero-knowledge for honest verifier proof system $(\overline{P}, \overline{V})$, we have to construct another prover/verifier pair (P, V) such that (P, V) is still an interactive proof system for L and for *any* (possibly cheating) verifier $\widehat{V}$ there exists a simulator $S_{\widehat{V}}$. We specify the protocol below. For completeness sake, first we recall what is interactive hashing [OVY-91], and show the interactive hashing-based bit commitment protocol [NOVY].

Remark: The bit commitment protocol parties are *efficient*, i.e. they need only perform polynomial time computations to execute the protocol.

Commit to a bit a

1. The verifier V selects $x \in_R \{0,1\}^n$ at random and computes $y \leftarrow f(x)$. V keeps both x and y secret from P.
2. The prover P selects $h_1, h_2, \ldots h_{n-1} \in \{0,1\}^n$ such that each h_i is a random vector over $GF[2]$

   such that $h_1, h_2, \ldots h_{n-1}$ are linearly independent over $GF[2]$
3. For j from 1 to $n-1$
   - P sends h_j to V.
   - V sends $r_j \leftarrow B(h_j, y)$ to P (where $B(u,v)$ is the bit resulting as the inner product of u and v).
4. At this point there are exactly two vectors $y_0, y_1 \in \{0,1\}^n$ such that for $i \in \{0,1\}$, $r_j = B(y_i, h_j)$ for all $1 \le j \le n-1$. y_0 is defined to be the lexicographically smaller of the two vectors. Both P and V compute y_0 and y_1. Let

$$d = \begin{cases} 0 & \text{if } y = y_a \\ 1 & \text{if } y = y_{1-a} \end{cases}$$

5. V computes d and sends it to R (d is "encrypting" the commitment bit a and given the inversion of one of y_0, y_1 and d, a is uniquely determined).

This committal reveals to P nothing about the committed bit (in the information-theoretic sense). On the other hand, V cannot later decommit to a value other than the one it committed without inverting a one-way permutation on a random challenge.

Next we present the compiler.

Compiler Protocol

1. V picks a sequence $a_i, 1 \leq i \leq 2t$ of random bits, and commits to them using *Interactive Hashing*. The commitment can be done in parallel for all bits.

2. P chooses at random t-subset of $\{1, \ldots, 2t\}$ and asks V to decommit bits a_j for j in the subset. Let $a'_i, i \leq t$ be the subsequence of unopened bits.

3. P picks t bits $b_1, \ldots, b_t$ at random and sends them to V.

4. V lets $c_i = b_i \oplus a'_i$ and $C = c_1 c_2 c_3 \ldots c_t$ be its secret random (tape) string.

5. P, V execute an old $(\overline{P}, \overline{V})$ protocol with V, running an $\overline{V}$, but using C as its secret coinflips. Moreover, for every message sent from V to P is accompanied by a *zero-knowledge argument* that $\overline{V}$ would really have sent this message if its coinflips were C. (Remark: Such a proof is possible and users are engaged in *Interactive Hashing* based on one-way permutation [NOVY] as a subroutine).

   More specifically, V begins by sending the message α_1 that would have been the first message $\overline{V}$ sent on coins C, and proves that indeed it has done this. The prover checks this proof, and if it is incorrect it aborts. Otherwise it sends whatever response β_1 the old prover $\overline{P}$ would have sent. This continues till the proof ends. (The available strongly committed bits, and the specification of the original protocols are the witness to the proofs communicated).

4.1 Proof of correctness

We have to prove completeness, soundness and the zero-knowledge property.

Completeness: For all x in L, the prover can still convince the verifier, since the success probability of the new P is essentially equivalent to the old one (by the simple fact that it is following the protocol).

Soundness: Interactive hashing hides committed bits in the information-theoretic sense, and thus the prover does not get any information about the random tape of the verifier (other then what follows from the original protocol during the initialization stage). Since all the subsequent rounds use zero-knowledge arguments of [NOVY] in addition to the messages of the old protocol, the soundness follows.

Zero-knowledge property: The simulator below proves this. We concentrate on statistical zero-knowledge here. The computational case is similar.

First, our new simulator runs the old simulator for honest verifier in order to obtain a pair $(\overline{C}, \alpha_1\beta_1 \dots \alpha_m\beta_m)$ consisting of coin tosses of the honest verifier $\overline{C} = \bar{c}_1\bar{c}_2 \dots \bar{c}_t$ and the transcript $\alpha_1\beta_1 \dots \alpha_m\beta_m$ of the conversation between the prover and the honest verifier. The new simulator, will now transform (with very high probability) this old transcript for honest verifier into one which is statistically close to the conversation between new prover/verifier pair as follows:

(1) It runs $\widehat{V}$ for step 1 to get its commitment of $a_1, \dots, a_{2t}$, using interactive hashing.
(2) At this point, the simulator uses the backtracking capability to run the protocol twice in order to learn what are the "unopened" bits. That is, it asks to reveal a random subset of t bits. Then it puts the verifier into the state it was in before the subset of t bits was requested to be revealed (but after the commitments) and now requests to open the complementary set of bits.
(3) Having the a_i, the simulator now picks $b_i = a_i \oplus \bar{c}_i$ for all $i = 1, \dots, t$ as being the prover's response (modifying bits) of step 3, and has thus makes $\overline{C}$ be the secret random string for the new V.

Recall that the simulator has in its possession the old conversation with coins fixed to $\overline{C}$. The zero-knowledge arguments executed at each round force cheating verifier to generate a conversation which is statistically close to the one we produced by using the honest verifier (with additional ZK arguments). The new simulator runs $\widehat{V}$ and gets what is supposed to be $\overline{V}$'s first message if it had $\overline{C}$, together with a proof (i.e. a zero-knowledge argument based on interactive hashing and assuming one-way permutations exist) that this is indeed the case. It examines the proof and if it is found incorrect the simulator aborts as the prover would have. But if not, then with very high probability, the message $\widehat{V}$ sent is *really* the message α_1 that the simulator expected at this stage. And to this message it can respond: it just has to send β_1. Continuing in this way the simulator soon has a transcript of the entire conversation, which (retracing through the argument) is statistically close to the real conversation. That is, the simulator generates exactly the correct conversation except if:

- $\widehat{V}$ manages to break the commitment scheme (i.e. invert a one-way permutation), or
- if it is able to cheat the prover in a zero-knowledge argument (which as well implies it can invert a one-way permutation, given the underlying construction).

Thus, we are done.

Conclusions: To summarize, we have presented a uniform way to compile honest-verifier zero-knowledge protocols into general zero-knowledge ones. This gives a design method which seems to be easier than considering all possible verifiers as a starting design point. The proof has some implications to properties of languages and their proofs, and it further demonstrates a wider applicability of the recent notion of interactive hashing.

References

[BM-84] M. Blum, and S. Micali "How to Generate Cryptographically Strong Sequences Of Pseudo-Random Bits" *SIAM J. on Computing,* Vol 13, 1984, pp. 850-864.

[BMO] Bellare, M., S. Micali and R. Ostrovsky, "The (True) Complexity of Statistical Zero Knowledge" STOC 90.

[BCC] G. Brassard, D. Chaum and C. Crépeau, *Minimum Disclosure Proofs of Knowledge*, JCSS, v. 37, pp 156-189.

[BCY] Brassard, G., C. Crépeau, and M. Yung, "Everything in NP can be Argued in Perfect Zero Knowledge in a Bounded Number of Rounds," ICALP 89. (also in Theoretical Computer Science, special issue of ICALP 89).

[Dam] I. B. Damgaard, *Collision Free Hash Functions and Public Key Signature Schemes* , Eurocrypt, 1987.

[GKa] Goldreich, O. and A. Kahn, personal communication.

[GILVZ] O. Goldreich, R. Impagliazzo, L. Levin, R. Venkatesan, and D. Zuckerman, *Security Preserving Amplification of Hardness*, FOCS 90.

[GMS] Goldreich, O., Y. Mansour, and M. Sipser, "Interactive Proof Systems: Provers that never Fail and Random Selection," FOCS 87.

[GMW1] Goldreich, O., S. Micali, and A. Wigderson, "Proofs that Yield Nothing but their Validity", FOCS 86.

[GMR] Goldwasser, S., S. Micali, and C. Rackoff, "The Knowledge Complexity of Interactive Proofs," *SIAM J. Comput.,* **18**(1), 186-208 (February 1989).

[GoMiRi] Goldwasser, S., S. Micali, and R. Rivest, "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks," *SIAM J. Comput.,* **17**(2), 281-308 (April 1988).

[Ha-90] J. Hastad, "Pseudo-Random Generators under Uniform Assumptions" *STOC 90*

[ILL] I. Impagliazzo, L. Levin and M. Luby, *Pseudo-random generation from one-way functions*, Proc. 21st Symposium on Theory of Computing, 1989, pp. 12-24.

[NOVY] M. Naor, R. Ostrovsky, R. Venkatesan, and M. Yung. "Perfect Zero-Knowledge Arguments for NP Can Be Based on General Complexity Assumptions", *Advances in Cryptology – Crypto '92*, Lecture Notes in Computer Science, Springer, to appear.

[NY] Naor, M. and M. Yung, "Universal One-Way Hash Functions and their Cryptographic Applications," STOC 89.

[Or] Oren Y., "On The Cunning Power of Cheating Verifiers: Some Observations About Zero Knowledge Proofs", FOCS 87.

[OVY-91] R. Ostrovsky, R. Venkatesan, and M. Yung. "Fair Games Against an All-Powerful Adversary", *SEQUENCES '91*, Positano, June, 1991 (Proc. Springer Verlag). Full version in: Computational Complexiity, DIMACS Series on Discrete Mathematics and Theoretical Computer Science (to appear), Ed. J. Cai, 1993.

[OVY-92] R. Ostrovsky, R. Venkatesan, M. Yung, *Secure Commitment Against A Powerful Adversary, STACS 92*, Springer Verlag LNCS Vol. 577, p. 439-448, 1992.

[OW] R. Ostrovsky, A. Wigderson *One-Way Functions are Essential for Non-Trivial Zero-Knowledge*, The second Israel Symposium on Theory of Computing and Systems (ISTCS93) 1993.

[Ro] J. Rompel "One-way Functions are Necessary and Sufficient for Secure Signatures" STOC 90.

[Y82] A. C. Yao, *Theory and Applications of Trapdoor functions*, Proceedings of the 23th Symposium on the Foundation of Computer Science, 1982, pp 80-91.