Strong Proofs of Knowledge
(preliminary version)

Oded Goldreich
Department of Computer Science and Applied Mathematics
Weizmann Institute of Science, Rehovot, Israel.

May 31, 1998

Abstract

The concept of proofs-of-knowledge, introduced in the seminal paper of Goldwasser, Micali and Rackoff, plays a central role in various cryptographic applications. An adequate formulation, which enables modular applications of proofs of knowledge inside other protocols, was presented by Bellare and Goldreich. However, this formulation depends in an essential way on the notion of expected (rather than worst-case) running-time. Here we present a seemingly more restricted notion which maintains the main feature of the prior definition while referring only to machines running in strict probabilistic polynomial-time (rather than to expected polynomial-time).

1 Introduction

The reader is referred to [1] for a discussion of the intuitive notion of a proof-of-knowledge (cf., [6]), and the previous attempts to define it [2, 7], cumlinating in the definition presented in [1]. We also assume that the reader is familiar with the definition given in [1].

The definition given in [1] relies in a fundamental way on the notion of *expected* running-time. Throughout the years we remained bothered by this feature, and recently – while working on [4] – we have decided to look for an alternative. Specifically, we present a more stringent definition in which the knowledge extractor is required to run in *strict* polynomial-time (rather than in *expected* polynomial-time). We call proof systems for which this more stringent definition holds, *strong proofs of knowledge* (in contrast to ordinary *proofs of knowledge* as defined in [1]).

There are two reasons to prefer strong proofs of knowledge over ordinary ones. Firstly, we feel more comfortable with the notion of strict polynomial-time than with the notion of expected polynomial-time. For example, it is intuitively unclear why a machine which runs for time 2^n on an 2^{-n} fraction of its coin-tosses (and in linear time otherwise) should be considered fundamentally different than a machine which runs for time 2^{n^2} on the same fraction. Secondly, it seems much more convinient to work (i.e., to compose) strict polynomial-time computations rather than expected polynomial-time ones.

However, there seems to be a loss in going from ordinary proofs of knowledge to strong ones: Not all proofs of knowledge are known to be *strong proofs of knowledge*. Furthermore, we conjecture that their are proofs of knowledge which are not *strong proofs of knowledge*. Still, zero-knowledge strong-proofs-of-knowledge do exist for all NP-relations, provided that one-way functions exist.

2 The Definition

Recall, we assume that the reader is familiar with the definition given in [1], as well as its underlying motivation.

Definition 1 (System of strong proofs of knowledge): *Let R be a binary relation. We say that an efficient strategy V is a* strong knowledge verifier for the relation R *if the following two conditions hold.*

- Non-triviality: *There exists an interactive machine P so that for every $(x, y) \in R$ all possible interactions of V with P on common-input x and auxiliary-input y are accepting.*
- Strong Validity: *There exists a negligible function $\mu : \mathbb{N} \mapsto [0,1]$ and a probabilistic* (strict) polynomial-time *oracle machine K such that for every strategy P and every $x, y, r \in \{0,1\}^*$, machine K satisfies the following condition:*

  Let $P_{x,y,r}$ be a prover strategy, in which the common input x, auxiliary input y and random-coin sequence r have been fixed, and denote by $p(x)$ the probability that the interactive machine V accepts, on input x, when interacting with the prover specified by $P_{x,y,r}$. Now, if $p(x) > \mu(|x|)$ then, on input x and access to oracle $P_{x,y,r}$, with probability at least $1 - \mu(|x|)$, machine K outputs a solution s for x. That is,

  $$\text{If } p(x) > \mu(|x|) \text{ then } \mathsf{Pr}[(x, K^{P_{x,y,r}}(x)) \in R] > 1 - \mu(|x|) \quad (1)$$

  The oracle machine K is called a strong knowledge extractor.

An interactive pair (P, V) so that V is a strong knowledge verifier for a relation R and P is a machine satisfying the non-triviality condition (with respect to V and R) is called a system for strong proofs of knowledge *for the relation R.*

Thus, it is required that whenever $p(x) > \mu(|x|)$ (i.e., unless the prover convinces the verifier with negligible probability), the extractor fails with negligible probability. Our choice to bound the failure probability of the extractor by the specific negligible function μ (which serves mainly as bound on $p(x)$) is rather arbitrary. What is important is to have this failure probability be a negligible function of $|x|$. Actually, in case membership in the relation R can be determined in polynomial-time, one may reduce the failure probability from $1 - \frac{1}{\text{poly}(n)}$ to $2^{-\text{poly}(n)}$, while maintaining the polynomial running-time of the extractor. Finally, we note that the extractor presented below has failure probability 0.

3 On the existence of strong proofs of knowledge

Some zero-knowledge proof (of knowledge) systems for NP are in fact strong proofs of knowledge. In particular, consider n sequential repetitions of the following basic proof system for the *Hamiltonian Cycle* (HC) problem (which is NP-complete). We consider directed graphs (and the existence of directed Hamiltonian cycles), and employ a commitment scheme $\{C_n\}$ as above.

Construction 2 (Basic proof system for HC):

- Common Input: *a directed graph $G = (V, E)$ with $n \stackrel{\text{def}}{=} |V|$.*

- Auxiliary Input to Prover: *a directed Hamiltonian Cycle, $C \subset E$, in G.*
- Prover's first step (P1): *The prover selects a random permutation, π, of the vertices of G, and commits to the entries of the adjacency matrix of the resulting permuted graph. That is, it sends an n-by-n matrix of commitments so that the $(\pi(i),\pi(j))^{\text{th}}$ entry is $C_n(1)$ if $(i,j) \in E$, and $C_n(0)$ otherwise.*
- Verifier's first step (V1): *The verifier uniformly selects $\sigma \in \{0,1\}$ and sends it to the prover.*
- Prover's second step (P2): *If $\sigma = 0$ then the prover sends π to the verifier along with the revealing (i.e., preimages) of all n^2 commitments. Otherwise, the prover reveals to the verifier only the commitments to n entries $(\pi(i),\pi(j))$ with $(i,j) \in C$.* (By revealing a commitment c, we mean supply a preimage of c under C_n; that is, a pair (σ, r) so that $c = C_n(\sigma, r)$.)
- Verifier's second step (V2): *If $\sigma = 0$ then the verifier checks that the revealed graph is indeed isomorphic, via π, to G. Otherwise, the verifier just checks that all revealed values are 1 and that the corresponding entries form a simple n-cycle. (Of course in both cases, the verifier checks that the revealed values do fit the commitments.) The verifier accepts if and only if the corresponding condition holds.*

The reader may easily verify that sequentially repeating the above for n times yields a zero-knowledge proof system for HC, with soundness error 2^{-n}. We argue that the resulting system is also a strong proof of knowledge of the Hamiltonian cycle. Intuitively, the key observation is that each application of the basic proof system results in one of two possible situations depending on the verifier choice, σ. In case the prover answers correctly in both cases, we can retrieve an Hamiltonian cycle in the input graph. On the other hand, in case the prover fails in both cases, the verifier will reject regardless of what the prover does from this point on. This observation suggests the following construction of a strong knowledge extractor (where we refer to repeating the basic proof systems n times and set $\mu(n) = 2^{-n}$).

Strong knowledge extractor for Hamiltonian cycle: On input G and access to the prover-strategy oracle P^*, we proceed in n iterations, starting with $i = 1$. Initially, T (the transcript so far), is empty.

1. Obtain the matrix of commitments, M, from the prover strategy (i.e., $M = P^*(T)$).
2. Extract the prover's answer to both possible verifier moves. Each of these answers may be correct (i.e., passing the corresponding verifier check) or not.
3. If both answers are correct then we recover a Hamiltonian cycle. In this case the extractor outputs the cycle and halts.
4. In case a single answer, say the one for value σ, is correct and $i < n$, we let $T \leftarrow (T, \sigma)$, and proceed to the next iteration (i.e., $i \leftarrow i + 1$). Otherwise, we halt with no output.

It can be easily verified that if the extractor halts with no output in iteration $i < n$ then the verifier (in the real interaction) accepts with probability zero. Similarly, if the extractor halts with no output in iteration n then the verifier (in the real interaction) accepts with probability 2^{-n}. Thus, whenever $p(G) > 2^{-n}$, the extractor succeeds in recovering a Hamiltonian cycle (with probability 1).

References

[1] M. Bellare and O. Goldreich. On Defining Proofs of Knowledge. In *Crypto92*, Springer-Verlag LNCS (Vol. 740), pages 390–420.

[2] U. Feige, A. Fiat and A. Shamir. Zero-Knowledge Proofs of Identity. *J. of Crypto.*, Vol. 1, 1988, pages 77–94.

[3] O. Goldreich. *Foundation of Cryptography – Fragments of a Book*. February 1995. Available from `http://theory.lcs.mit.edu/~oded/frag.html`.

[4] O. Goldreich. *Secure Multi-Party Computation*. In preparation, 1998.

[5] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *J. of the ACM*, Vol. 38, No. 1, pages 691–729, 1991. Preliminary version in *27th FOCS*, 1986.

[6] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM J. on Comput.*, Vol. 18, pages 186–208, 1989. Preliminary version in *17th STOC*, 1985.

[7] M. Tompa and H. Woll. Random Self-Reducibility and Zero-Knowledge Interactive Proofs of Possession of Information. University of California (San Diego), Computer Science and Engineering Department, Technical Report Number CS92-244, June 1992. Preliminary version in *28th FOCS*, pages 472–482, 1987.