A Perfect Zero-Knowledge Proof for a Problem Equivalent to Discrete Logarithm

Oded Goldreich *Eyal Kushilevitz*

Dept. of Computer Sc.
Technion
Haifa, Israel

ABSTRACT

An interactive proof is called *perfect zero-knowledge* if the probability distribution generated by any probabilistic polynomial-time verifier interacting with the prover on input a theorem ϕ, can be generated by another probabilistic polynomial time machine which only gets ϕ as input (and interacts with nobody!).

In this paper we present a *perfect* zero-knowledge proof system for a decision problem which is computationally equivalent to the Discrete Logarithm Problem. Doing so we provide additional evidence to the belief that *perfect* zero-knowledge proofs exist in a non-trivial manner (i.e. for languages not in BPP). Our results extend to the logarithm problem in any finite Abelian group.

1. INTRODUCTION

One of the most basic questions in complexity theory is how much knowledge should be yield in order to convince a polynomial-time verifier of the validity of some theorem. This question was raised by Goldwasser, Micali and Rackoff [GMR], with special emphasis on the extreme case where nothing but the validity of the theorem is given away in the process of proving the theorem. Such proofs are known as *zero-knowledge* proofs and have been the focus of much attention in recent years. Loosely speaking, whatever can be efficiently computed after participating in a zero-knowledge proof can be efficiently computed when just assuming the validity of the assertion.

The definition of zero-knowledge considers two types of probability distributions:

1) A distribution generated by a probabilistic polynomial-time verifier after participating in an interaction with the prover.

2) A distribution generated by a probabilistic polynomial-time machine on input the theorem.

Zero-knowledge means that for each distribution of type (1) there exists a distribution of type (2) such that these two distributions are "essentially equal". The exact definition of zero-knowledge depends on the exact interpretation of "essentially equal" distributions. Two extreme cases are of particular interest:

Research was partially supported by the Fund for Basic Research Administered by the Israeli Academy of Sciences and Humanities.

- *Perfect zero-knowledge.* This notion is derived when interpreting "essentially equal" in the most conservative way; namely, exactly equal.
- *Computational zero-knowledge.* This notion is derived when interpreting "essentially equal" in a very liberal way; namely, requiring that the distribution ensembles are polynomially indistinguishable. Loosely speaking, two distribution ensembles are polynomially indistinguishable if they can not be told apart by any probabilistic polynomial time test. For definition see [Y].

1.1. Known Results

Assuming the existence of one-way permutations, Goldreich, Micali and Wigderson showed that any language in NP has a computational zero-knowledge proof [GMW]. Using this result one can also show that whatever can be proven through an efficient interactive proof, can be proven through such a computational zero-knowledge proof [BGGHMR,IY]. Thus, assuming the existence of one-way permutations, the question of which languages have computational zero-knowledge proofs is closed.

Much less is known about perfect zero-knowledge. Clearly any language in BPP has a trivial perfect zero-knowledge proof (in which the prover is inactive...). Several languages believed not to be in BPP were shown to have perfect zero-knowledge proofs. These includes Quadratic Residuosity and Quadratic non-Residuosity [GMR], Graph Isomorphism and Graph non-Isomorphism [GMW], and membership and non-membership in a subgroup generated by a given group element [TW]. (It should be noticed that Tompa and Woll's proof of "possession of the Discrete Logarithm" is in fact a proof of membership in a subgroup generated by a primitive element. So are the proofs given by [CEGP,CG]).

The complexity of languages which have a perfect zero-knowledge interactive proofs was studied by Fortnow [F] and then by Aiello and Hastad [AH]. They prove that if a language L has a perfect zero-knowledge interactive proof system then both L and $\bar{L}$ have two-step interactive proofs. This implies that languages having perfect zero-knowledge proofs fall quite low in the polynomial time hierarchy (i.e. as low as $\Pi_2^P \cap \Sigma_2^P$). Using a result of Boppana, Hastad and Zachos [BHZ], such languages can also not be NP-complete, unless the polynomial time hierarchy collapses to its second level.

Perfect zero-knowledge proofs should not be confused with the perfect zero-knowledge *pseudo-proofs* presented by Brassard and Crepeau [BC]. By a *pseudo-proof* we mean that the verifier is convinced only if he believes that the prover is a polynomial-time machine with some auxiliary input (which is fixed before the protocol starts), and if some intractability assumption does hold. For example, *if factoring is intractable* then every NP language has a perfect zero-knowledge *pseudo-proof* [BC]. Brickell et. al. presented a perfect zero-knowledge *pseudo-proof* for a problem equivalent to the discrete logarithm problem, assuming the existence of any one-way permutation [BCDG]. It should be noted however that the class of languages having perfect zero-knowledge *pseudo-proof* does not seem to have the same complexity as the class of languages having perfect zero-knowledge proofs. Furthermore, assuming the intractability of factoring every language having an interactive proof has a perfect zero-knowledge pseudo-proof, and thus the class of languages having such proofs collides with the class of languages having computational zero-knowledge proofs.

1.2. Our Results

In this paper we present a *perfect* zero-knowledge proof system for a decision problem which is computationally equivalent to the Discrete Logarithm Problem. Doing so, we present a perfect zero-knowledge proof for a problem which is widely believed to be intractable. Thus, we provide additional evidence to the belief that *perfect* zero-knowledge proofs exist in a non-trivial manner (i.e. for languages not in BPP).

Let p be a prime, and g be a primitive element in the multiplicative group modulo p. The *Discrete Logarithm Problem* (DLP) is to find, given integers p, g and y, an integer x such that $g^x \equiv y \mod p$. Solving DLP is considered intractable, in particular when $p-1$ has large prime factors. The best algorithms known for this problem run in subexponential time ($\exp\{O(\sqrt{\log p \log\log p})\}$), see Odlyzko's survey [O]. It has been shown that determining* whether $x \le \frac{p-1}{2}$ is computationally equivalent to finding x, on inputs p, g and $g^x \mod p$ [BM]. This is the case even if x is guaranteed to lie either in the interval $[1, \varepsilon p]$ or in the interval $[\frac{p-1}{2}+1, \frac{p-1}{2}+\varepsilon p]$, where $0<\varepsilon<1/6$ is a constant or a function bounded below by $(log_2 p)^{-O(1)}$. This promise problem is hereby referred to as $DLP\,1$.

In this paper, we present a perfect zero-knowledge proof for $DLP\,1$. Using the computational equivalence with DLP, we have a perfect zero-knowledge proof for a problem considered computationally hard. Both our protocol and the computational equivalence of DLP and $DLP\,1$ extend to any finite Abelian group, in which the group operation can be implemented in polynomial-time and the order of the group is known (or can be efficiently found). (In the case of acyclic groups, one needs first to define the problems.)

It should be noted that DLP is always at least as hard as testing membership in a subgroup generated by an element of the group. In some cases, for example when $p-1=2q$ and q is prime, determining membership in a subgroup is easy (see Appendix), while solving DLP in the multiplicative group mod p is considered hard.

2. PRELIMINARIES

2.1. Promise Problems and Interactive Proofs

Loosely speaking a *promise problem* is a partial decision problem. That is, a decision problem in which only a subset of all possible inputs is being considered.
Formally a *promise problem* is a pair of predicates (Q,R). A Turing machine M *solves* the promise problem (Q,R) if for every z which satisfying $Q(z)$ machine M halts and it answer "yes" iff $R(z)$. When $\neg Q(z)$ we do not care what M does. This definition is originates from [ESY].

*) In fact, Blum and Micali proved a much stronger statement. Namely, that guessing this bit with success probability greater than $\frac{1}{2}+\varepsilon$ is as hard as retrieving x [BM].

We are going to extend the definition of interactive proofs given in [GMR] to promise problems. Intuitively, an interactive proof system for a promise problem (Q,R) is a two-party protocol for a "powerful" *prover* P and a probabilistic polynomial-time *verifier* V satisfying the following two conditions with respect to the common input, denoted z. If $Q(z) \wedge R(z)$ then with a very high probability the verifier is "convinced" of $R(z)$, when interacting with the prover. If $Q(z) \wedge \neg R(z)$ then no matter what the prover does, he cannot fool the verifier (into believing that "$R(z)$ is true"), except for with very low probability. When $\neg Q(z)$ nothing is required.

Definition 1: An *interactive proof for a promise problem* (Q,R) is a pair of interacting Turing-machines $<P,V>$, satisfying the following three conditions:

0) V is a probabilistic polynomial-time machine which share its input with P and they can communicate to each other using special communication tapes.

1) *Completeness condition*: For every constant $c>0$, and all sufficiently long z if $Q(z) \wedge R(z)$ then
$$Prob(V \text{ will accept } z \text{ after interacting with } P) \geq 1-|z|^{-c}.$$

2) *Soundness condition*: For every Turing machine P^*, every constant $c>0$, and all sufficiently long z if $Q(z) \wedge \neg R(z)$ then
$$Prob(V \text{ will reject } z \text{ after interacting with } P^*) \geq 1-|z|^{-c}.$$

2.2. Perfect Zero-Knowledge Proofs for Promise Problems

Here, again, we are going to extend the definition given by [GMR] to promise problems.

Definition 2: Let $<P,V>$ be an interactive proof system for a promise problem (Q,R), and V^* be an arbitrary verifier. Denote by $<P,V^*>(z)$ the probability distribution on all the read-only tapes of V^* when interacting with P (the prover) on common input z. We say that the proof system $<P,V>$ is a *perfect-zero-knowledge* for (Q,R) if for all polynomial-time verifier V^*, there exists a probabilistic machine M_{V^*} running in expected polynomial-time such that for every z satisfying $Q(z) \wedge R(z)$ the distributions $M_{V^*}(z)$ and $<P,V^*>(z)$ are equal.

2.3. The Discrete Logarithm Problem and a Related Promise Problem

Let p be a prime. The set of integers $[1,p-1]$ forms a cyclic group of $p-1$ elements under multiplication *mod* p which is denoted Z_p^*. The *Discrete Logarithm* problem (DLP) with input p,g and y is to find $x \in [1,p-1]$ such that $y \equiv g^x \bmod p$. (We use the notation $x = Dlog_g y$).

Let y be an element of Z_p^* and let g be a primitive element (a generator). We define the Half predicate H as follows:

$$H(p,g,y) \Leftrightarrow Dlog_g y \in [\frac{p-1}{2}+1, p-1]$$

Let $n = \log_2 p$ and let $\varepsilon(n) < \frac{1}{2}$ be a fraction bounded below by $\frac{1}{n^{O(1)}}$. We define the following predicate:

$$Q_\varepsilon(p,g,y) \Leftrightarrow g \text{ is a generator of } Z_p^* \text{ and } Dlog_g y \text{ in } [1, \varepsilon(n)(p-1)] \text{ or } [\frac{p-1}{2}+1, \frac{p-1}{2}+\varepsilon(n)(p-1)]$$

When it will be clear from the context we will shorten $H(p,g,y)$ and $Q_\varepsilon(p,g,y)$ by $H(y)$ and $Q(y)$ respectively.

The promise problem defined by the pair of predicates (Q_ε, H) will be called in this work *DLP* 1. Blum and Micali have shown that the *DLP* 1 is polynomially-equivalent to the original *DLP* in the group Z_p^* [BM].

2.4. Notations

1) Let s and t be two integers such that $1 \le s, t \le p-1$. $[s,t]$ is denoted the set of integers $\{s, s+1, \cdots, t-1, t\}$ in case $s \le t$ or $\{s, s+1, \cdots, p-1, 1, 2, \ldots, t\}$ in case $s > t$.
2) Let S be a set. The notation $r \in_R S$ means that r is chosen at random with uniform probability distribution among the elements of S.

3. THE PROTOCOL FOR DLP1 IN Z_p^*

In this section we will introduce a perfect zero-knowledge protocol for the promise problem *DLP* 1. In order to make the protocol more clear we will first introduce a protocol which is perfect zero-knowledge with respect to the honest verifier.

3.1. Protocol 1 - Perfect Zero Knowledge Proof with respect to the Honest Verifier

Here is a protocol for the promise problem $(Q_c(p,g,y), H(p,g,y))$ where c is a constant such that $0 < c \le 1/6$:

common input: The integers p, g and y as previously defined.

The following 3 steps are executed $n = \log_2 p$ times (unless the verifier *rejects* previously), each time using independent random coin tosses.

V1) The verifier chooses at random a bit $b \in_R \{0,1\}$ and an integer $r \in_R [1, 2c(p-1)]$. The verifier computes $\alpha = y^b g^r$ and sends α to the prover.

P1) The prover computes $\beta = H(\alpha)$ and sends it to the verifier.

V2) If $\beta \ne b$, then the verifier *rejects*.

If all n rounds are completed without the verifier rejects then the verifier *accepts*.

Theorem 1: *Assuming $c < 1/6$ then protocol 1 constitutes an interactive proof system for DLP1.*

Proof: Recall that x denoted $Dlog_g y$ (i.e $y \equiv g^x \ mod\ p$).
Completeness: If $Q(y) \wedge H(y)$ then $x \in [\frac{p-1}{2}+1, \frac{p-1}{2}+c(p-1)]$ and then, according to the ranges in which b and r are chosen from, in each round $\beta = H(\alpha) = H(y^b \cdot g^r) = H(g^{bx+r}) = b$.
Soundness: If $Q(y) \wedge \neg H(y)$ then we have $x \in [1, c(p-1)]$. Therefore if the verifier chooses $b=0$ then $Dlog_g \alpha \in [1, 2c(p-1)]$ and if he chooses $b=1$ then $Dlog_g \alpha \in [x+1, x+2c(p-1)]$. In this case, for any prover P' we are looking for the probability that V does not reject in a single round:
Prob(V does not reject)= $Prob(P'(\alpha)=b)$

$$= Prob(b=0) \cdot Prob(P'(\alpha)=b \mid b=0) + Prob(b=1) \cdot Prob(P'(\alpha)=b \mid b=1)$$

$$=\frac{1}{2}\cdot Prob(P'(g^r)=0) + \frac{1}{2}\cdot Prob(P'(yg^r)=1)$$

$$=\frac{1}{2}\cdot\frac{1}{2c(p-1)}\ (\ \sum_{i=1}^{x} Prob(P'(g^i)=0) + \sum_{i=x+1}^{2c(p-1)} (Prob(P'(g^i)=0) + Prob(P'(g^i)=1))$$

$$+ \sum_{i=2c(p-1)+1}^{2c(p-1)+x} Prob(P'(g^i)=1)\ )$$

$$\le\frac{1}{2}\cdot\frac{1}{2c(p-1)}\cdot\left[x+(2c(p-1)-x)+x\right]$$

$$=\frac{1}{2}\cdot\left[\frac{x}{2c(p-1)}+1\right]$$

$$\le\frac{1}{2}\cdot\left[\frac{c(p-1)}{2c(p-1)}+1\right]=\frac{3}{4}$$

Therefore in n iterations the probability that the verifier will not reject this input is exponentially low. (i.e. $(3/4)^n$) □

Remark: It is clear that this protocol is perfect zero-knowledge with respect to the honest verifier V. The simulator M_V chooses the random tape for V, and therefore knows the b which V will choose and can compute $\beta=H(\alpha)=b$.

The interactive proof for DLP1 presented above is probably not zero-knowledge with respect to arbitrary verifiers: a cheating verifier interacting with the prover may send $\alpha's$ which he wants to know $H(\alpha)$, he could also choose $r \notin [1,2c(p-1)]$ and get in this way some additional information about x. The way to prevent this, is to let the verifier first "prove" to the prover that he "knows" $H(\alpha)$. This is done in the following protocol.

3.2. Protocol 2 - Perfect Zero Knowledge Proof with respect to Any Verifier

The previous protocol will be modified. The modification follows an idea of [GMR] used also in [GMW] and simplified by [Bh]. However in our case the modification is more complex.

In the following protocol we provide an interactive proof to the promise problem $(Q_{\frac{c}{n^2}}(p,g,y),H(p,g,y))$ where c is a constant such that $0<c\le\frac{1}{12}$.

common input: The integers p,g and y as previously defined.

The following 5 steps are repeated $n=\log_2 p$ times (unless the verifier *rejects* previously), each time using independent random coin tosses.

V1) The verifier chooses at random a bit $b \in_R \{0,1\}$ and an integer $r \in_R [\left\lfloor\frac{p-1}{4}\right\rfloor+1, \left\lfloor\frac{p-1}{4}\right\rfloor+c(p-1)]$. The verifier computes $\alpha=y^b g^r$ and sends α to the prover. In addition to α he computes n pairs of integers. The $i-th$ pair is denoted α_i and is constructed in the following way: The verifier chooses at random $\gamma_i \in_R \{0,1\}$ and $r_{i,0},r_{i,1} \in_R [1,c(p-1)]$. He computes $\alpha_{i,0}=y^{\gamma_i}\cdot g^{r_{i,\gamma_i}}$ and $\alpha_{i,1}=y^{\gamma_i+1 \bmod 2}\cdot g^{r_{i,\gamma_i+1 \bmod 2}}$ and at last sets $\alpha_i=(\alpha_{i,0},\alpha_{i,1})$. The verifier sends the list of pairs to the prover.

P1) The prover chooses at random, a subset $I \subseteq \{1,2,\ldots,n\}$ with uniform probability distribution among all 2^n subsets. The prover sends I to the verifier.

V2) If I is not a subset of $\{1,2,\ldots,n\}$ then the verifier halts and *rejects*. Otherwise, the verifier replies with $\{(\gamma_i, r_{i,0}, r_{i,1}) : i \in I\}$ and $\{(\gamma'_i = \gamma_i + b + 1 \; mod \; 2, \; r'_i = r + r_{i,b+1 \; mod \; 2}) : i \in \bar{I}\}$. (where $\bar{I} = \{1,2,\ldots,n\} - I$).

P2) For every $i \in I$ the prover checks that α_i is constructed according to the protocol. (i.e. $r_{i,0}, r_{i,1} \in [1, c(p-1)]$ and $\alpha_i = (y^{\gamma_i} \cdot g^{r_{i,\gamma}}, y^{\gamma_i+1 \; mod \; 2} \cdot g^{r_{i,\gamma+1 \; mod \; 2}})$). He also checks for every $i \in \bar{I}$ that $r'_i \in [\left\lfloor \frac{p-1}{4} \right\rfloor + 2, \left\lfloor \frac{p-1}{4} \right\rfloor + 2c(p-1)]$ and $y \cdot g^{r'_i} = \alpha \cdot \alpha_{i,\gamma'_i}$. If either conditions is violated the prover stops. Otherwise, the prover computes $\beta = H(\alpha)$ and sends it to the verifier.

V3) If $\beta \neq b$, then the verifier *rejects*. Otherwise he continues.

If all n rounds are completed without the verifier rejects then the verifier *accepts*.

Theorem 2: *Protocol 2 constitutes a perfect zero-knowledge interactive proof system for DLP1.*

Proof: We will first prove that Protocol 2 is an interactive proof for DLP1 and then we will show that it is perfect zero-knowledge. Recall again that $x = Dlog_g y$.

Completeness: Similar to the completeness in theorem 1.

Soundness: We are going to prove that although α and the list of pairs $S = \{\alpha_1 \cdots \alpha_n\}$ can give information to the prover, there is a big enough probability that α and S will not give him anything that will help him to convince V that $x \in [\frac{p-1}{2}+1, \frac{p-1}{2} + \frac{c(p-1)}{n^2}]$ when in fact $x \in [1, \frac{c(p-1)}{n^2}]$.

We call α *good* if it is constructed using $r \in [\left\lfloor \frac{p-1}{4} \right\rfloor + \frac{c(p-1)}{n^2} + 1, \left\lfloor \frac{p-1}{4} \right\rfloor + c(p-1) - \frac{c(p-1)}{n^2}]$. Otherwise α is *bad*. Intuitively, when α is good the prover can not learn anything about b from α, for any $x \in [1, \frac{c(p-1)}{n^2}]$ (since in this case $Prob(b=0 \mid y^b \cdot g^r = \alpha) = \frac{1}{2}$). The probability that α is bad is $\frac{2}{n^2}$.

Similarly we will call a pair α_i *good* if both $r_{i,0}$ and $r_{i,1}$ are in $[\frac{c(p-1)}{n^2}+1, c(p-1) - \frac{c(p-1)}{n^2}]$. Otherwise α_i is *bad*. The list of pairs S is *good* if every α_i is good, and is *bad* otherwise. The probability that a pair α_i is bad is less than $\frac{4}{n^2}$ and the probability that S is bad is therefore less than $\frac{4n}{n^2}$.

We remark here that since P' has infinite power we can assume without loss of generality that P' is deterministic. Therefore for any α and S the prover P' always chooses the same subset I, denoted $f(\alpha, S)$.

Our first claim is the following:

$\forall$ *good* α $\forall$ *good* S $\forall I \; \forall r'_i \; \forall \gamma'_i$

$Prob(b{=}0 \mid y^b g^r{=}\alpha \;\wedge\; f(\alpha,S){=}I \;\wedge\; \forall i \in \bar{I}\,(y \cdot g^{r'_i}{=}\alpha \cdot \alpha_{i,\gamma_i} \wedge r'_i{=}r{+}r_{i,\gamma_i})) \;=\; \frac{1}{2}$

The reason is that when α and S are good then assigning any value to b yields a unique values to all the other variables r , $r_{i,0}$ and $r_{i,1}$. Thus, assuming $I{=}f(\alpha,S)$, there are only two elements in the conditional probability space, one corresponds to $b{=}0$ and the other to $b{=}1$. Using this claim we will show now that the probability that P' will convince V in single round is low:

$Prob(P'(\,\alpha,S\;,\{\gamma_i, r_{i,0}, r_{i,1}{:}i \in f(\alpha,S)\}\,,\{\gamma'_i, r'_i{:}i \notin f(\alpha,S)\})=b\;)$

$$\le Prob(P'(\alpha\,,S\;,\{\gamma_i, r_{i,0}, r_{i,1}{:}i \in f(\alpha,S)\}\,,\{\gamma'_i, r'_i{:}i \notin f(\alpha,S)\})=b \mid \alpha \text{ and } S \text{ are good})$$

$$+Prob(\alpha \text{ is bad or } S \text{ is bad})$$

$$\le \frac{1}{2} + \frac{4n+2}{n^2}$$

Therefore the probability that P' will mislead V (i.e. provide correct β's) in all n rounds is exponentially low.

Zero-knowledge: For every interactive machine V^*, we will present a machine M_{V^*} so that for every input satisfying $Q(p,g,y) \wedge H(p,g,y)$ then $M_{V^*}(p,g,y) = \langle P, V^* \rangle (p,g,y)$. The machine M_{V^*} uses V^* as a subroutine.

The idea of the simulator M_{V^*} is to cause V^* to yield all the information needed for calculating $H(\alpha)$. This is done by executing V^* several times with the same random tape, so that V^* will send the same α and S. Machine M_{V^*} will try to get for one of the pairs α_i the information $\{\gamma_i, r_{i,0}, r_{i,1}\}$ in one round and $\{\gamma'_i, r'_i\}$ in another. If this information is constructed according to the protocol (M_{V^*} will check it) then this is enough for calculating $H(\alpha)$.

Following is a detailed description of M_{V^*}. Machine M_{V^*} starts by choosing a random tape $r \in_R \{0,1\}^q$ for V^*, where $q{=}poly(|p,g,y|)$ is a bound on the running time of V^* on the current input (Clearly, V^* reads at most q bits from its random tape). M_{V^*} places r on its record tape and proceeds in n rounds as follows.

Round j:

S1) M_{V^*} initiates V^* on the input (p,g and y) and random tape r, and reads from the communication tape of V^* the pairs α and $\alpha_1 \cdots \alpha_n$. M_{V^*} chooses a random subset I and places it on the communication tape of V^*. M_{V^*} also appends I to its record tape.

S2) M_{V^*} reads from the communication tape of V^* $\{(\gamma_i, r_{i,0}, r_{i,1}){:}i \in I\}$ and $\{(\gamma'_i, r'_i){:}i \in \bar{I}\}$. For every $i \in I$ machine M_{V^*} checks whether $\gamma_i \in \{0,1\}$, $r_{i,0}, r_{i,1} \in [1, c(p-1)]$ and whether $\alpha_i{=}(y^{\gamma_i} \cdot g^{r_{i,\gamma}},\; y^{\gamma_i+1 \bmod 2} \cdot g^{r_{i,\gamma+1 \bmod 2}})$. It also checks for every $i \in \bar{I}$ whether $r'_i \in [\left\lfloor \frac{p-1}{4} \right\rfloor + 2, \left\lfloor \frac{p-1}{4} \right\rfloor + 2c(p-1)]$ and $y \cdot g^{r'_i}{=}\alpha \cdot \alpha_{i,\gamma_i}$. If either conditions is violated M_{V^*} outputs its record tape and stops. Otherwise, M_{V^*} continues to step (S3).

S3) The purpose of this step is to find $H(\alpha)$. This is done by repeating the following procedure (until $H(\alpha)$ is found):

(S3.1) Machine M_{V^*} chooses at random a subset $K \subseteq \{1,2,\ldots,n\}$ not equal to I. Machine

M_{V^*} initiates V^* on the same input and the same (!) random tape r and places K as the first message on the read-only communication tape of V^*. Consequently, machine M_{V^*} reads from the communication tape of V^* $\{(\delta_i, s_{i,0}, s_{i,1}) : i \in K\}$ and $\{(\delta'_i, s'_i) : i \in \bar{K}\}$.

(S3.2) M_{V^*} checks whether the information he received is ok. (The same tests as he does for the answers to I). If it is not ok he returns back to step (S3.1). Otherwise M_{V^*} finds i such that $i \in I \cap \bar{K}$ or $i \in \bar{I} \cap K$. Such an i exists since $I \neq K$, without loss of generality we assume that $i \in I \cap \bar{K}$. Machine M_{V^*} sets $\beta = (\gamma_i + \delta'_i + 1) mod 2$.

(S3.3) In parallel to (S3.1) and (S3.2), try to find $H(\alpha)$ by exhaustive search. (Make one try per each invocation of V^*.)

S4) Once β is found, machine M_{V^*} appends β to its record tape, thus completing round j.

If all rounds are completed then M_{V^*} outputs its record tape and halts.

We now have to prove the validity of the construction. First, we will prove that the simulator M_{V^*} indeed terminates in expected polynomial-time. Next, we will prove that the output distribution produced by M_{V^*} does equal the distribution over V^*'s tapes (when interacting with P). Once these two claims are proven, the Theorem follows.

Claim 1: Machine M_{V^*} terminates in expected polynomial time.

Proof: We consider the expected running time on a single round with respect to a particular random tape r. We call a subset $I \subseteq \{1,2,...,n\}$ *good* if V^* answers properly on message I with random tape r. Denote by g_r the number of good subsets with respect to random tape r. Clearly, $0 \le g_r \le 2^n$. We will compute the expected number of times V^* is invoked in round j as a function of g_r. We need to consider three cases:

Case 1 ($g_r \ge 2$): In case the subset I chosen in step (S1) is good, we have to consider the probability that another subset K is also good. In case the set I chosen in step (S1) is bad, the round is completed immediately. Thus, the expected number of invocations is

$$\frac{g_r}{2^n} \cdot \left(\left[\frac{g_r - 1}{2^n - 1} \right]^{-1} + 1 \right) + \frac{2^n - g_r}{2^n} \cdot 1 < \frac{g_r}{g_r - 1} + 1 \le 3$$

Case 2 ($g_r = 1$): With exponentially small probability (i.e. 2^{-n}) the subset I chosen in step (S1) is good. In this case we find β by exhaustive search (in stage (S3.3)). Otherwise, the round is completed immediately. Thus, the expected complexity of M_{V^*} in case 2 is bounded by one invocation of V^* and an additional $(p-1) \cdot 2^{-n} \le 1$ step.

Case 3 ($g_r = 0$): The subset I chosen in step (S1) is always bad, and thus M_{V^*} invokes V^* exactly once and then halts.

The claim follows by additivity of expectation and the fact that V^* is polynomial-time. □

Claim 2: The probability distribution $M_{V^*}(p,g,y)$ is identical to the distribution $<P, V^*>(p,g,y)$.

Proof: Both distributions consists of a random r, and sequence of elements, each being either (I, β) (with good I) or a bad I, with random I. In $<P, V^*>(p,g,y)$ we have $\beta = H(\alpha)$ we need to show that this is the case also in $M_{V^*}(p,g,y)$. i.e. we will prove that when I is good then M_{V^*} succeeds in finding $H(\alpha)$. But this is true because either he finds $H(\alpha)$ by exhaustive search or find an i in which γ_i , $r_{i,0}$, $r_{i,1}$, δ'_i and s'_i are all correct. (i.e. $r_{i,0}, r_{i,1} \in [1, c(p-1)]$, $s'_i \in [\left\lfloor \frac{p-1}{4} \right\rfloor +2, \left\lfloor \frac{p-1}{4} \right\rfloor +2c(p-1)]$, $\alpha_{i,j} = y^{\gamma_i + j \; mod \; 2} \cdot g^{r_{i,\gamma_i + j \; mod \; 2}}$ and $y \cdot g^{s'_i} = \alpha \cdot \alpha_{i,\delta'_i}$). In this case we have:

$$
\begin{aligned}
H(\alpha) &= H(yg^{s'_i} \cdot (\alpha_{i,\delta'_i})^{-1}) \\
&= H(yg^{s'_i} \cdot (y^{\gamma_i + \delta'_i \; mod \; 2} \cdot g^{r_{i,\gamma_i+\delta'_i \; mod \; 2}})^{-1}) \\
&= H(y^{\gamma_i + \delta'_i + 1 \; mod \; 2} \cdot g^{s'_i - r_{i,\gamma_i+\delta'_i \; mod \; 2}}) \\
&= H(y^{\gamma_i + \delta'_i + 1 \; mod \; 2}) = \gamma_i + \delta'_i + 1 \; mod \; 2 \; \square
\end{aligned}
$$

The Theorem follows. $\square$

Remark 1: It is not hard to see that instead of executing the protocol sequentially, we can execute all the rounds in parallel.

Remark 2: Let s and t be two integers such that $1 \le s, t \le p-1$. We define $dist(s,t)$ to be the minimal distance between s and t over the circle of numbers $[1, p-1]$. Consider the following promise problem hereby referred to as $DLP2$: Promised that $x \in [s,t]$ or $x \in [(s+\frac{p-1}{2}) \; mod \; (p-1), (t+\frac{p-1}{2}) \; mod \; (p-1)]$ and $dist(s,t) < \frac{p-1}{12n^2}$ does $x \in [(s+\frac{p-1}{2}) \; mod \; (p-1), (t+\frac{p-1}{2}) \; mod \; (p-1)]$? An easy modification to protocol 2 yields a perfect zero-knowledge interactive proof system for $DLP2$:

Protocol 3

common input: p, g and y as before and also s.

1) P and V both perform $y' := y \cdot g^{-s+1}$

2) P and V perform protocol 2 on input p, g and y'.

Theorem 2': *Protocol 3 constitutes a perfect zero-knowledge interactive proof system for DLP2.*

Proof: Since it is promised that $Dlog_g y \in [s,t]$ or $Dlog_g y \in [(s+\frac{p-1}{2}) \; mod \; (p-1), (t+\frac{p-1}{2}) \; mod \; (p-1)]$ and that $dist(s,t) < \frac{p-1}{12n^2}$ then after executing step 1 we have $Dlog_g y' \in [1, \frac{p-1}{12n^2}]$ or $Dlog_g y' \in [\frac{p-1}{2}+1, \frac{p-1}{2}+\frac{p-1}{12n^2}]$ and now our theorem follows from theorem 2.

4. EXTENSIONS

4.1. Generalization of the Protocol to other Cyclic Groups

Let G be an arbitrary cyclic group such that the following conditions holds:

1) The group operation of G can be implemented in polynomial-time.

2) The order of G (to be denoted N) is either given or can be computed in polynomial-time.

We can extend the definitions of the *DLP* and the *DLP* 1 in the obvious way. The needed modifications are to replace any multiplication *mod p* by the group-operation of G and to replace $p-1$ by the group order (N).

With the same modifications our protocol will be a perfect zero-knowledge proof for the promise problem *DLP* 1 in G (since the protocol does not make any use of the special structure of Z_p^*, but merely its being cyclic). What we still have to show is that the *DLP* 1 is polynomially equivalent to the *DLP* itself in any cyclic group. The Blum-Micali proof (used in Z_p^*) extends easily only to groups in which N is even and both testing quadratic-residuosity and taking square-root can be performed in polynomial time. Unfortunately, this does not seem to be the case in all groups and a different argument is needed. We present a proof for the equivalence of *DLP* and *DLP* 1 based on ideas of Kaliski [Ka].

We define the oracle LOG_G as follows:

$LOG_G(g,y,s,d)=0$ if $Dlog_g y \in [s,(s+d) \bmod N]$

$LOG_G(g,y,s,d)=1$ if $Dlog_g y \in [(s+\left\lfloor \frac{N}{2} \right\rfloor) \bmod N, (s+\left\lfloor \frac{N}{2} \right\rfloor+d) \bmod N]$

In any other case the answer of the oracle LOG_G is unexpected.

It should be noticed that when $d<\frac{N}{12n^2}$ this oracle solves the promise problem *DLP* 2 for which protocol 3 is a perfect zero-knowledge proof.

Theorem 3: The following 2 problems are polynomially equivalent for any cyclic group G of N elements and a generator g:

1) . Given $g,y \in G$ such that g is a generator of G find x such that $x=Dlog_g y$. (*DLP*)

2) Given $y \in G$, a generator $g \in G$, $0<s<N$ and d such that $0<d<\frac{N}{12n^2}$ compute $LOG_G(g,y,s,d)$. (*DLP* 2)

Proof: It is obvious that if we know to solve the first problem we can solve the second one. We will prove the other direction by presenting an algorithm that solves the *DLP* using the oracle $LOG_G(g,y,s,d)$. The algorithm is based on the following elementary lemma:

Lemma 1: For any cyclic group G of order N and for every $y \in G$:

If $Dlog_g y^2 \in [s,t]$ then $Dlog_g y$ is in $[\left\lceil \frac{s}{2} \right\rceil, \left\lfloor \frac{t}{2} \right\rfloor]$ or in $[\left\lceil \frac{s+N}{2} \right\rceil, \left\lfloor \frac{t+N}{2} \right\rfloor]$

Proof (of the Lemma): Let $x \in [s,t] \subseteq \{0,1,\cdots,N-1\}$ and try to find a number w such that $x=2 \cdot w$. We deal with two cases:

Case 1: N is odd. Since N is odd there exists a unique number $2^{-1} \bmod N$. In this case one can easily verify that if x is even then $w=\frac{x}{2} \in [\left\lceil \frac{s}{2} \right\rceil, \left\lfloor \frac{t}{2} \right\rfloor]$ and if x is odd then $w=\frac{x}{2} \in [\left\lceil \frac{s+N}{2} \right\rceil, \left\lfloor \frac{t+N}{2} \right\rfloor]$.

Case 2: N is even. Since N is even $2^{-1} \bmod N$ not exists. In this case only for even x's we have such w. Actually we have two such numbers: $w_1=\frac{x}{2}$ and $w_2=\frac{x+N}{2}$. It is easy to verify that

$w_1=\frac{x}{2} \in [\left\lceil \frac{s}{2} \right\rceil, \left\lfloor \frac{t}{2} \right\rfloor]$ and $w_2=\frac{x+N}{2} \in [\left\lceil \frac{s+N}{2} \right\rceil, \left\lfloor \frac{t+N}{2} \right\rfloor]$.

Now taking $x=Dlog_g y^2$ the lemma follows for every N. □

Note that if the interval in which $Dlog_g y^2$ is found is of size d then the intervals in which $Dlog_g y$ can be found are of size $\lceil d/2 \rceil$. This is used in the following algorithm.

Algorithm 1: (The input is $y \in G$ and a generator g)

```
(1)  Let $n=\log_2 N$
(2)  Compute $y_1=y$ , $y_2=y_1^2$ , $y_3=y_2^2$ ... $y_n=y_{n-1}^2$ . /* $y_i=y^{2^{i-1}}$ */
(3)  Let $s=0$.
(4)  Let $d=\frac{N}{12n^2}$
(5)  For $k=n$ to 1 do
         If ($LOG_G(g, y_k, s, d)=0$) then $s'=s$
                                  else $s'=(s+\left\lfloor \frac{N}{2} \right\rfloor) \bmod N$
         $s=\lceil s'/2 \rceil$
         $d=\lceil d/2 \rceil$
     end
(6)  If ($g^s=y$) output $s$
     else if ($g^{s+1}=y$) output $s+1$
     else $s=s+\frac{N}{12n^2}$ ; goto (4)
```

The idea is that we are trying to find an s such that $Dlog_g y_n=Dlog_g y^{2^n}$ is in the range of size $\frac{N}{12n^2}$ starting from s. Assume that we are in the right interval then according to the lemma in each round in step (5) we reduce by a factor of 2 the size of interval in which we are looking for $Dlog_g y_k$. Therefore at the end after $n=log_2 N$ rounds we are looking for $Dlog_g y_1=Dlog_g y$ in an interval of size 2. Now, we check which of the two numbers in the interval is $Dlog_g y$. If both are not fitted then the current s is wrong and we increase it by $\frac{N}{12n^2}$ and try again. After at most $\frac{N}{d}=12n^2$ iterations we should find the right s. Therefore the number of times we will have to execute steps (4-6) is $O(n^2)$. Now, assuming that LOG_G is polynomial-time and recall the assumptions about G (i.e. N is known or can be computed in polynomial-time and the group operation can also be implemented in polynomial time) then this algorithm is also polynomial-time. □

4.2. Generalization of the Results to Acyclic Groups

In an acyclic group which is finite and Abelian we do not have a generator but a *generating-tuple* $\bar{g}=(g_1,g_2,\cdots,g_k)$. Any element $y \in G$ can be uniqely expressed as $y=g_1^{x_1}\cdots g_k^{x_k}$. The order of each g_i is denoted N_i and the number of elements in the group is

$N = N_1 \cdot N_2 \cdots N_k$. The *DLP* and the *DLP* 1 are defined with respect to g_1. (For example the *DLP* in such a group is: Given y - find x_1 such that $\exists x_2 \cdots x_k \mid y = g_1^{x_1} \cdots g_k^{x_k}$).

Our protocol with some modifications will work here too. We have to assume that we know (or can compute in polynomial-time) not only the group size N but also N_1. We should replace every occurance of N in the previous protocol by N_1 and also everything done with respect to g has to be done with respect to g_1. In addition we should randomize everything by elements chosen at random from the subgroup generated by $(g_2, g_3, \cdots, g_k)$. For example in step (V1) of the protocol the verifier should compute $\alpha = y^b \cdot g_1^{r_1} \cdot g_2^{r_2} \cdots g_k^{r_k}$, where $r_1 \in_R [\left\lfloor \frac{N_1}{4} \right\rfloor + 1, \left\lfloor \frac{N_1}{4} \right\rfloor + cN_1]$ and $r_2 \cdots r_k \in_R [1, N]$.

Using the same modifications described above we can also modify theorem 3 to show that the *DLP* and the *DLP* 1 are still equivalent in an acyclic group.

APPENDIX: Determining Membership in a Subgroup - Special Case

In this Appendix we consider the problem of determining membership in a subgroup generated by an element g in Z_p^*, when $p-1=2q$ and q is prime. We will show that in this special case, testing membership in a subgroup is easy. This should be contrasted with the believed intractability of *DLP* also for this case.

One can readily verify that if $p-1=2q$ with q prime then Z_p^* has $q-1$ primitive elements (i.e. elements of order $p-1$), $q-1$ elements of order q, one element of order 2, and one element of order 1 (i.e. the identity). Furthermore, all the elements of order q and the identity element form a subgroup which is generated by any of the elements of order q. Thus, the question of whether a is in the subgroup generated by g reduces (in this case!) to testing the order of both a and b (a is in the subgroup generated by b iff the order of a divides the order of b). Finally note that testing the order of an element is easy (in this case!).

REFERENCES

[AH] Aiello, W., and J. Hastad, "Perfect Zero-Knowledge Languages can be Recognized in Two Rounds", *Proc. 28th FOCS*, 1987, pp. 439-448.

[BGGHMR] Ben-or, M., O. Goldreich, S. Goldwasser, J. Hastad, S. Micali, and P. Rogaway, This proceedings.

[Bh] Benaloh, (Cohen), J.D., "Cryptographic Capsules: A Disjunctive Primitive for Interactive Protocols", *Crypto86*, Abstract #21, Santa Barbara, California, August 1986.

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

[BHZ] Boppana, R., J. Hastad, and S. Zachos, "Does Co-NP Have Short Interactive Proofs?", *IPL*, 25, May 1987, pp. 127-132.

[BC] Brassard, G., and C. Crepeau, "Non-Transitive Transfer of Confidence: A Perfect Zero-Knowledge Interactive Protocol for SAT and Beyond", *Proc. 27th FOCS*, pp. 188-195, 1986.

[BCDG] Brickell E.F., D. Chaum, I. Damgard, and J. van de Graaf, "Gradual and Verifiable Release of a Secret", preprint, 1987.

[CEGP] Chaum, D., J.H. Evertse, J. van de Graaf, and R. Peralta, "Demonstrating Possession of a Discrete Logarithm without Revealing It", *Crypto86*, Abstract #20, Santa Barbara, California, August 1986.

[CG] Chaum, D. and J. van de Graaf, "An Improved protocol for Demonstrating Possession of a Discrete Logarithm without Revealing It", *Eurocrypt87*, Amsterdam, The Netherlands, April 1987, IV-15 to IV-21.

[ESY] Even, S., A.L. Selman, and Y. Yacobi, "The Complexity of Promise Problems with Applications to Public-Key Cryptography" *Information and Control*, Vol. 61, 1984, pp. 159-173.

[F] Fortnow, L., "The Complexity of Perfect Zero-Knowledge", *Proc. of 19th STOC*, pp. 204-209, 1987.

[GMW] Goldreich, O., S. Micali, and A. Wigderson, "Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design", *Proc. 27th FOCS*, pp. 174-187, 1986. Submitted to *JACM*.

[GMR] Goldwasser, S., S. Micali, and C. Rackoff, "Knowledge Complexity of Interactive Proofs", *Proc. 17th STOC*, 1985, pp. 291-304. To appear in *SIAM Jour. on Computing.*

[IY] Impagliazo, R., and M. Yung, "How to show that IP has Zero-Knowledge Proofs", private communication.

[Ka] Kaliski, B., "A Pseudo-Random Bit Generator Bases on Elliptic Logarithms", Ph.D. thesis, MIT, in preparation.

[O] Odlyzko, A., "Discrete Logarithm in finite fields and their cryptographic significance", Preprint.

[TW] Tompa, M., and H. Woll, "Random Self-Reducibility and Zero-Knowledge Interactive Proofs of Possession of Information", *Proc. 28th FOCS*, 1987, pp. 472-482.

[Y] Yao, A.C., "Theory and Applications of Trapdoor Functions", *Proc. of the 23rd IEEE Symp. on Foundation of Computer Science*, 1982, pp. 80-91.