On the Existence of 3-Round Zero-Knowledge Protocols

Satoshi Hada Toshiaki Tanaka

KDD R&D Laboratories,
2-1-15 Ohara, Kamifukuoka, Saitama 356-8502, Japan.
E-mail: {sa-hada, tl-tanaka}@kdd.co.jp

Abstract. In this paper, we construct a 3-round zero-knowledge protocol for any NP language. Goldreich and Krawczyk proved that a 3-round black-box simulation zero-knowledge protocol exists only for BPP languages. However, there is no contradiction here. That is, our proposed protocol achieves a weaker notion of zero-knowledge: auxiliary-input non-uniform zero-knowledge. Since this notion has not been investigated in the literature, we classify several zero-knowledge notions including it and discuss the relationships among them. Our main contribution is to provide a non-black-box simulation technique. It is based on a novel computational assumption related to the Diffie-Hellman problem. Although this assumption is strong and non-standard, its non-standard nature seems essential for our simulation technique.

Keywords: Zero-knowledge, interactive proof, Diffie-Hellman problem.

1 Introduction

The fundamental notion of zero-knowledge (ZK) introduced by Goldwasser, Micali and Rackoff plays a central role in modern cryptography [GMR85]. In this paper, we investigate the methodology underlying ZK in order to construct a 3-round ZK protocol for NP.

1.1 Background on Zero-Knowledge Protocol

Consider an interactive protocol in which a prover convinces a verifier that some common input x belongs to some underlying language L (In this paper, L is in NP). The length of x is denoted by n and one measures complexity in terms of n. The verifier is always a probabilistic polynomial-time machine. We focus on two properties: "soundness" and "zero-knowledge." Each can be formalized in two ways depending on whether or not we restrict the adversary (the cheating prover and the cheating verifier) to a resource bound machine.

The soundness asks that if $x \notin L$, any cheating prover can not convince the verifier to accept, except with negligible error probability. This notion is formalized in two ways: "proofs" and "arguments." These provide the statistical soundness and the computational soundness, respectively. The former requires

that even a computationally unrestricted cheating prover should be unable to make the verifier accept $x \notin L$, except with negligible probability [GMR85]. On the other hand, the latter requires that any cheating prover restricted to probabilistic polynomial-time should be unable to make the verifier accept $x \notin L$, except with negligible probability [BrCr86][BCC88]. Although the notion of arguments is weaker than the notion of proofs, it is good enough for cryptographic applications. The soundness of arguments will typically depend on the complexity assumptions such as the discrete logarithm assumption. Whenever we talk of proofs or arguments, we always mean ones with negligible error probability.

Zero-knowledge asks that when $x \in L$, an interaction with the prover yields no information (other than the fact $x \in L$) to any cheating verifier. Again, this notion is formalized in two ways: "statistical ZK" (SZK) and "computational ZK" (CZK). The former requires that even a computationally unrestricted cheating verifier will not gain useful information, except with negligible probability. On the other hand, the latter requires that any resource bound cheating verifier (probabilistic polynomial-time machine or polynomial-size circuit family) will not gain useful information, except with negligible probability. Clearly, SZK is a special case of CZK.

In this paper, unless stated explicitly, ZK protocols mean CZK arguments. Our proposed protocol is a CZK argument.

1.2 Classification of Zero-Knowledge

Our proposed protocol achieves the notion of auxiliary-input non-uniform ZK. Since this notion has not been investigated in detail so far, we classify several relevant ZK notions and discuss the relationships among them.

ZK was originally formalized in [GMR85] as follows: for any probabilistic polynomial-time machine $\hat{V}$ (the cheating verifier), there exists a probabilistic polynomial-time machine $S_{\hat{V}}$ (the simulator) which produces a probability distribution which is computationally indistinguishable from the distribution of conversations of $\hat{V}$ with the prover P. This original definition (GMRZK) is not suitable for cryptographic applications since it is not closed under sequential composition [GoKr96]. In cryptographic applications, the verifier can have some additional a-priori information.

In order to overcome the above problem, auxiliary-input zero-knowledge (AIZK) was introduced in [GoOr94]. AIZK is defined by augmenting GMRZK with auxiliary-input, that is, the simulation requirement is extended to deal with non-uniform verifiers with an auxiliary-input, where the simulator takes the same auxiliary-input used by the verifier. It was shown that AIZK is closed under sequential composition [GoOr94].

Black-box simulation zero-knowledge (BSZK) requires the existence of a universal simulator that, using any non-uniform verifier $\hat{V}$ as a black-box, succeeds in simulating the interaction of $\hat{V}$ with the prover P. It was shown that BSZK implies AIZK [GoOr94]. Although BSZK is the most restrictive among the above definitions, almost all known ZK protocols are BSZK.

All the above definitions are "semi-uniform" in the sense that it uses uniform machines but quantifies over all common inputs $x \in L$. The non-uniform formalization of ZK appeared in [Go93], where all machines are modeled by a family of polynomial-size circuits. We consider two non-uniform formalizations here: non-uniform zero-knowledge and auxiliary-input non-uniform zero-knowledge.

Non-uniform zero-knowledge (NUZK) is a non-uniform variant of GMRZK. That is, it requires that for any family of polynomial-size circuits $\hat{V}$, there exists a family of (probabilistic) polynomial-size circuits $S_{\hat{V}}$ which produces a probability distribution which is computationally indistinguishable from the distribution of conversations of $\hat{V}$ with the prover P. It is important to note that NUZK does not imply GMRZK [Go98-2]. In fact, one can devise artificial protocols for sparse languages such that it achieves the notion of NUZK but not GMRZK. For example, consider the following interactive proof for a sparse language $L_{sp} = \{1^n\}_{n \in \mathbf{N}}$. The prover sends the verifier a hard function $K(\cdot)$ of the common input $x \in L_{sp}$ (e.g., K is a non-recursive function indicating whether the nth Turing machine accepts every input). The verifier accepts iff x is of the form 1^n. Certainly, this is an interactive proof for L_{sp}. It is not GMRZK since there is no way to simulate in probabilistic polynomial-time the interaction in which the prover sends the value of $K(x)$. On the other hand, it is still NUZK since the simulator may just incorporate the hard bit (i.e., the nth circuit will incorporate the bit indicating if the nth Turing machine accepts every input). This shows that NUZK is a very weak notion of ZK and does not satisfy the intuitive requirement of ZK. Also, the result of [GoKr96] can be extended to show that NUZK is not closed under sequential composition.

Auxiliary-input non-uniform zero-knowledge (AINUZK) is defined by augmenting the notion of NUZK with auxiliary-input. The above interactive proof for a sparse language achieves not only NUZK but also AINUZK. That is, AINUZK also does not satisfy the intuitive requirement of ZK. However, AINUZK has an advantage over NUZK since it is closed under sequential composition [GoOr94][Go93]. Our proposed protocol achieves this notion.

Let $Cl(def)$ denote the class of all interactive proofs and arguments satisfying the requirements of definition def. In the light of the above, it holds that

$$Cl(BSZK) \subseteq Cl(AIZK) \subset Cl(GMRZK) \subset Cl(NUZK)$$

and

$$Cl(AIZK) \subset Cl(AINUZK) \subset Cl(NUZK).$$

It is an open problem whether $Cl(BSZK)$ equals $Cl(AIZK)$ [GoOr94].

1.3 Motivation and Contribution

The round complexity, the number of messages exchanged, is a standard complexity measure for the efficiency of ZK protocols. Several researchers constructed constant round ZK protocols for NP [BCY89][FeSh89] [GoKa96] [1] [BJY97]. The

[1] ZK protocols constructed in [GoKa96] are proofs rather than arguments.

lower bounds on the round complexity have been investigated from the practical and theoretical viewpoint. Goldreich and Oren proved that only languages in BPP have 2-round AIZK protocols [GoOr94]. Their result can be extended to prove that only languages in P/poly have 2-round AINUZK protocols. Furthermore, Goldreich and Krawczyk proved that only languages in BPP have 3-round BSZK protocols [GoKr96] [2]. Since the argument in [GoKr96] uses the notion of black-box simulation in an essential way, their result does not apply to the weaker notions such as AIZK and AINUZK. Therefore, with respect to AIZK and AINUZK, it is an interesting open problem whether there exists a 3-round ZK protocols for a non-trivial language, i.e., a language not known to be in BPP and P/poly.

As mentioned above, almost all known ZK protocols are BSZK, that is, the zero-knowledge property has been demonstrated using the black-box simulation of the verifier. In fact, it seems hard to conceive an alternative way to demonstrate ZK property. Therefore, it seems hard to construct a 3-round ZK protocol for a non-trivial language. In other words, in order to construct such protocols, a new simulation technique is needed.

In this paper, we construct a 3-round AINUZK protocol for any NP language. Our result does not contradict the result of [GoKr96] since our proposed protocol does not achieve the notion of BSZK. Our main contribution is to provide a non-black-box simulation technique. It is based on a novel computational assumption related to the Diffie-Hellman problem. We call it the strong Diffie-Hellman assumption (SDHA). Although this assumption is strong and non-standard, its non-standard nature seems essential for our simulation technique.

Organization. In Section 2, we give the definitions of AINUZK arguments and some standard complexity assumptions. Section 3 describes our proposed protocol. In Section 4, we formalize SDHA. In Section 5, we prove the correctness of our proposed protocol. In Section 6, we conclude with some remarks.

2 Preliminaries

In this section, we give the definitions of AINUZK arguments and some standard complexity assumptions. Most of this section follows [BJY97] and [Go98-1].

2.1 Auxiliary-Input Non-Uniform Zero-Knowledge Arguments

We deal with NP languages and let $W_L(x)$ denote the witness set of x which belongs to an NP language L. We say that a function $\nu(\cdot) : \mathsf{N} \to \mathsf{R}$ is negligible if for every polynomial $poly(\cdot)$ and all sufficiently large n's, it holds that $\nu(n) < 1/poly(n)$. Also, we say that a function $g(\cdot) : \mathsf{N} \to \mathsf{R}$ is overwhelming if $g(\cdot) = 1 - \nu(\cdot)$ for some negligible function $\nu(\cdot)$.

[2] The proofs in [GoKr96] are for CZK proofs. However, their result extends to CZK arguments. See Remarks 6.3 and 6.5 in that paper.

We consider two probabilistic polynomial-time interactive machines called the prover and the verifier. Initially both machines have access to a common input tape which includes x of length n. The prover and the verifier send messages to one another through two communication tapes. After exchanging a polynomial number of messages, the verifier stops in an accept state or in a reject state. Each machine, denoted by A, only sees its own tapes, namely, the common input tape, the random tape, the auxiliary-input tape and the communications tapes. In particular, the prover's auxiliary input tape includes a witness $w \in W_L(x)$. Let $A(x, m, r)$ denote A's next message, where x is the common input, r the random coins and m the messages so far. We let $A_x(\cdot, \cdot) = A(x, \cdot, \cdot)$ and $A_{x,r}(\cdot) = A(x, \cdot, r)$. When A takes an auxiliary input y, we write A_x^y and $A_{x,r}^y$ for A_x and $A_{x,r}$, respectively. Let $\mathsf{Acc}(P_x, V_x)$ denote the probability that V accepts when interacting with P on the common input x. The probability is taken over the random tapes of both machines.

Definition 1. Let P, V be two probabilistic polynomial-time interactive machines. We say that (P, V) is an argument for L if the following two conditions are satisfied:

- Completeness: For every $x \in L$, every $w \in W_L(x)$, $\mathsf{Acc}(P_x^w, V_x) = 1$.
- Soundness: For every probabilistic polynomial-time machine $\hat{P}$ (the cheating prover), every polynomial $poly(\cdot)$, all sufficiently long $x \notin L$ and all y's,

$$\mathsf{Acc}(\hat{P}_x^y, V_x) < \frac{1}{poly(|x|)}.$$

We recall the notion of computational indistinguishability of probability distributions used in the definition of zero-knowledge.

Definition 2. Let L be an NP language. An *ensemble* indexed by $L \times \{0,1\}^*$ is a sequence $\{E_{x,y}\}_{x \in L, y \in \{0,1\}^*}$ of probability distributions, one for each $(x, y) \in L \times \{0,1\}^*$. Let $E = \{E_{x,y}\}_{x \in L, y \in \{0,1\}^*}$ and $E' = \{E'_{x,y}\}_{x \in L, y \in \{0,1\}^*}$ be two ensembles over a common index set $L \times \{0,1\}^*$. A *distinguisher* is a polynomial-size circuit family $D = \{D_{x,y}\}_{x \in L, y \in \{0,1\}^*}$. We say that E and E' are computationally indistinguishable if for every distinguisher D, every polynomial $poly(\cdot)$, all sufficiently long $x \in L$ and all $y \in \{0,1\}^*$,

$$|\Pr[D_{x,y}(v) = 1] - \Pr[D_{x,y}(v') = 1]| < \frac{1}{poly(|x|)},$$

where v and v' are chosen according to the distribution $E_{x,y}$ and $E'_{x,y}$, respectively.

A *view* of the verifier is an ensemble which consists of the common input, the auxiliary input, the random coins and the sequence of messages by the prover and the verifier during the interaction. Let $\mathsf{View}(P_x, V_x^y) = [x, y, r, m]$ denote V's view after interacting with P, where x is the common input, y the auxiliary input to V, r the random coins of V and m the sequence of messages sent by P and V. Note that r is empty when V is a family of polynomial-size circuits.

Definition 3. Let P, V be two probabilistic polynomial-time interactive machines. We say that (P, V) is an auxiliary-input non-uniform zero-knowledge for L if for every family of polynomial-size circuits $\hat{V}$ (the cheating verifier), there exists a family of (probabilistic) polynomial-size circuits $S_{\hat{V}}$ (the simulator) such that the following two ensembles are computationally indistinguishable:

$$\{S_{\hat{V}}(x, y)\}_{x \in L, y \in \{0,1\}^*} \text{ and } \{\mathsf{View}(P_x, \hat{V}_x^y)\}_{x \in L, y \in \{0,1\}^*}.$$

We remark that it is not required that simulator $S_{\hat{V}}$ can be effectively constructed given a verifier $\hat{V}$, but rather that it exists.

2.2 DLA and DHA

We give two standard complexity assumptions related to the discrete logarithm problems. All exponentiations in this paper are in Z_p^* (the definition of the prime p will be clear by the context). To simplify the notations, we omit the expression "mod p".

We recall the discrete logarithm assumption (DLA). In this paper, we need a stronger definition of DLA in which we assume nothing on the distribution of a prime p and a base g.

Definition 4. Let L_{PQG} denote the set $\{(p, q, g)\}$ of primes and generators, where p and q are primes such that $p = 2q + 1$ and g is an element of order q in Z_p^* (a generator of a subgroup of Z_p^* of order q).

L_{PQG} can be recognized in probabilistic polynomial-time with negligible error probability by testing primality for p and q in probabilistic polynomial-time [SS77][Ra80], and verifying that g is not the identity and that $g^q = 1$. Furthermore, there exists a probabilistic polynomial-time algorithm which, on input 1^n, outputs $(p, q, g) \in L_{PQG}$ such that p is of length n.

Assumption 5 (DLA). For every family of polynomial-size circuits $I = \{I_n\}_{n \geq 1}$, every polynomial $poly(\cdot)$ and all sufficiently large n's,

$$\mathsf{Inv}_{DL}^I(p, q, g) = Prob[\ I_n(p, q, g, g^a) = a\ ] < \frac{1}{poly(n)},$$

where (p, q, g) is any instance in L_{PQG} such that p is of length n. The probability is taken over the choice of a uniformly at random in Z_q.

The Diffie-Hellman assumption (DHA) says that the Diffie-Hellman problem [DH76] is intractable in the same setting as DLA.

Assumption 6 (DHA). For every family of polynomial-size circuits $I = \{I_n\}_{n \geq 1}$, every polynomial $poly(\cdot)$ and all sufficiently large n's,

$$\mathsf{Inv}_{DH}^I(p, q, g) = Prob[\ I_n(p, q, g, g^a, g^b) = g^{ab}\ ] < \frac{1}{poly(n)},$$

where (p, q, g) is any instance in L_{PQG} such that p is of length n. The probability is taken over the choice of a, b uniformly at random in Z_q.

3 Protocol Description

In this section, we construct a 3-round ZK protocol for any NP language (called 3R-ZK). Our starting point is a 3-round public-coin honest-verifier ZK protocol for an NP language L_{NP}. We transform it into a 3-round secret-coin any-verifier ZK protocol for the same language L_{NP}.

3.1 The Starting Protocol

We require that the starting protocol (M, A) satisfies the following properties. Let M_1, Y, M_2 denote the messages exchanged in the starting protocol. M_1 and M_2 are the first and the second messages that the prover sends to the verifier, respectively. Y is the verifier's challenge.

B0. It is a public-coin protocol in which the challenge Y (the verifier's public coins) is chosen uniformly at random in any polynomially samplable subdomain of $\{0,1\}^*$.

B1. The prover can be implemented in probabilistic polynomial-time when it is given as its auxiliary-input an NP witness.

B2. It satisfies a *strong soundness* property which requires that for every common input $x \notin L_{NP}$ and every possible first message M_1, there exists at most one verifier's challenge Y such that the prover may answer properly in its second message M_2.

B3. It is zero-knowledge with respect to a prescribed verifier for the protocol, i.e., honest-verifier zero-knowledge (HVZK). Formally, for the prescribed verifier B, there exists a probabilistic polynomial-time simulator S_{HV} (the honest-verifier simulator) such that the following two ensembles are computationally indistinguishable:

$$\{S_{HV}(x,y)\}_{x\in L, y\in\{0,1\}^*} \text{ and } \{\mathsf{View}(M_x, A_x^y)\}_{x\in L, y\in\{0,1\}^*}.$$

For example, the parallel composition of Blum's ZK protocol for the Hamiltonian circuit problem satisfies all the above properties [Bl86] (See also Chapter 4, Exercise 16 in [Go98-1]).

Theorem 7 [Bl86][Go98-1]. *Assuming the existence of non-uniformly secure commitment schemes, there exists a starting protocol satisfying all the above properties for any NP language.*

3.2 Our Proposed Protocol

Before describing our proposed protocol, we review the general approach to constructing constant round ZK protocols for NP. Only BPP languages have constant round *public-coin* BSZK protocols [GoKr96]. Therefore, only feasible way of constructing constant round BSZK protocols for NP is to let the verifier use "secret coins". That is, the coins inducing the message sent by the verifier are kept secret from the prover. In fact, all the previous constant round ZK protocols

are secret-coin protocols [BCY89][FeSh89][BMO90] [GoKa96][BJY97]. We note that in all the previous protocols, the verifier demonstrates that it knows the secret coins. As a result, the simulator can get the verifier's secret coins in the simulation. For example, in [FeSh89] the verifier executes a witness-hiding protocol. In [BCY89][BMO90][GoKa96] the verifier executes a commitment protocol [3]. In [BJY97] the verifier executes a cut-and-choose type protocol. Furthermore, the previous constant round ZK protocols are designed so that once the simulator gets the verifier's secret coins, it can complete the simulation without any NP witness, whereas as long as the cheating prover does not know the verifier's secret coins, the soundness condition is satisfied.

We construct a 3-round ZK protocol by transforming the starting protocol into a 3-round secret-coin protocol in which the challenge Y is generated by an interaction of the prover and the verifier, rather than by the verifier itself. In the light of the above, the resulting protocol should satisfy the following properties:

R1. The verifier demonstrates that it knows the coins inducing the message sent by it, while keeping these coins secret from the prover.
R2. The knowledge of the verifier's secret coins enables the simulator to answer properly in the second message given a fixed challenge Y.
R3. As long as the verifier's secret coins are kept secret, it is computationally difficult for a cheating prover to answer properly in the second message given a fixed challenge Y.

The property R1 has been implemented by a notion of "proof of knowledge" [TW87][FFS88][BeGo92], specifically by bit-commitment protocols, witness-hiding protocols and cut-and-choose type protocols. However, these protocols require an interaction which we can not afford here (Note that in 3-round protocols, the verifier sends the message once for all). Therefore, it seems impossible to satisfy the property R1. Nevertheless, we resolve this difficulty using a new type of computational assumption so that the protocol resulting from our transformation can satisfy all the properties.

Protocol: 3R-ZK for an NP language L_{NP}.
Common Input: a common input x of length n.
Prover's Witness: an NP witness in $W_{L_{NP}}(x)$.
P1: The prover P computes the first message as follows:
  P1-1 P generates an instance $(p,q,g) \in L_{PQG}$ such that p is of length n. P also generates uniformly a random number $a \in Z_q$. Then P computes $A = g^a$.
  P1-2 P computes M_1 according to the starting protocol.
  P sends (M_1, p, q, g, A) to the verifier V.
V1: V checks whether (p,q,g) is in L_{PQG} and p is of length n. If this is true, V generates a random number $b \in Z_q$ (the secret coins), computes $(B,X) = (g^b, A^b)$ and sends (B,X) to P. Otherwise V rejects.

[3] In a commitment protocol, the secret coins are revealed finally.

P2: P checks whether $X = B^a$. If this is false P stops, otherwise P computes the second message as follows:

P2-1 P generates a random number $c \in Z_q$ and computes $(C, Y) = (g^c, B^c)$ (These may also be computed as $(C, Y) = (A^c, X^c)$).

P2-2 P computes M_2 using Y as the challenge according to the starting protocol.

P sends (M_2, C, Y) to V.

V2: V checks whether the following two conditions are satisfied:

V2-1 V checks whether $Y = C^b$.

V2-2 V checks whether M_2 is valid using Y as the challenge according to the starting protocol.

If either condition is violated V rejects, otherwise V accepts.

We explain that 3R-ZK satisfies all the properties. With respect to R1, we can make the following observation. The verifier's secret coins b are kept secret if we assume that it is computationally intractable to compute b from (g, A, B, X). Furthermore, it seems that the verifier must raise (g, A) to bth power in order to pass the prover's check $X = B^a$ in P2. That is, we assume that *the verifier knows the secret coins b whenever it holds that $X = B^a$*. This is a new type of computational assumption which we call SDHA-1. SDHA-1 can be formalized as follows: for any verifier $\hat{V}$, there exists another verifier $\hat{V}'$ such that $\hat{V}'$ outputs not only (B, X) but also the secret coins b whenever $\hat{V}$ outputs (B, X) satisfying $X = B^a$. We may say that R1 is implemented using a computational assumption rather than a notion of "proof of knowledge".

R2 is satisfied since given Y and b, it is easy to compute the second message C such that $Y = C^b$. When the simulator uses $\hat{V}'$ (but not $\hat{V}$) as a black-box, it can get not only (B, X) but also b. As a result, it is possible to complete the simulation.

As in R1, it seems that the prover must raise (g, B) or (A, X) to cth power in order to pass the verifier's check $Y = C^b$ in V2-1 (Note that there are two ways of computing (C, Y) in P2-1). We assume that *the prover knows the value of c such that $Y = B^c$ or X^c whenever it holds $Y = C^b$*. We call this assumption SDHA-2. Assume that R3 is not satisfied. Then, given a fixed challenge Y, the cheating prover can compute the second message C satisfying $Y = C^b$ for randomly chosen (B, X). Under SDHA-2, this means that, given Y, it is easy to compute the discrete logarithm c such that $Y = B^c$ or X^c for randomly chosen (B, X). This contradicts DLA.

The above discussion roughly shows that 3R-ZK is a ZK protocol for L_{NP}. The formal proof is given in Section 5.

4 The Strong Diffie-Hellman Assumption

In this section, we formalize SDHA. There are two versions of SDHA: SDHA-1 and SDHA-2. They are required for the ZK property and the soundness, respectively.

Assumption 8 (SDHA-1). Firstly, DLA is assumed to hold in this assumption. Let I be a family of polynomial-size circuits which takes as input (p, q, g, g^a) and tries to output (B, X) such that $X = B^a$. For every family of polynomial-size circuits $I = \{I_n\}_{n \geq 1}$, there exists another family of polynomial-size circuits $I' = \{I'_n\}_{n \geq 1}$ which, on input (p, q, g, g^a), outputs (B', X', b) satisfying the following two conditions, where the probability is taken over the choice of a uniformly at random in Z_q, (p, q, g) is any instance in L_{PQG} and p is of length n.

1. $I'_n(p, q, g, g^a)$ is statistically close to $I_n(p, q, g, g^a)$ on the first two outputs (B, X).
2. For every polynomial $poly(\cdot)$ and all sufficiently large n's,

$$Prob[\ X' = B'^a \ \wedge \ B' \neq g^b\ ] < \frac{1}{poly(n)}.$$

Roughly, the above conditions say that whenever I outputs (B, X) such that $X = B^a$, I' outputs not only (B, X) but also b such that $B = g^b$ (i.e., $X = (g^a)^b$) with overwhelming probability.

The proposition below shows that SDHA-1 implies DHA. However, it is unlikely that DHA implies SDHA-1.

Proposition 9 . *Under SDHA-1, DHA holds.*

Proof(sketch). Assume that DHA does not hold. Then there exists a family of polynomial-size circuits I which, on input $g, A = g^a, B = g^b$, outputs $B = g^b$ and $X = g^{ab}$ with non-negligible probability. Under SDHA-1, there exists another family of polynomial-size circuits I' that computes not only (B, X) but also the discrete logarithm b such that $B = g^b$. This contradicts DLA. □

Assumption 10 (SDHA-2). Firstly, DLA is assumed to hold in this assumption. Let I be a family of polynomial-time circuits which takes as input $(p, q, g, g^a, g^b, g^{ab})$ and tries to output (C, Y) such that $Y = C^b$. For every family of polynomial-size circuits $I = \{I_n\}_{n \geq 1}$, there exists another family of polynomial-size circuits $I' = \{I'_n\}_{n \geq 1}$ which takes as input $(p, q, g, g^a, g^b, g^{ab})$ and outputs (C', Y', c) satisfying the following two conditions, where the probability is taken over the choice of b uniformly at random in Z_q, (p, q, g) is any instance in L_{PQG}, p is of length n and a is any element in Z_q.

1. $I'_n(p, q, g, g^a, g^b, g^{ab})$ is statistically close to $I_n(p, q, g, g^a, g^b, g^{ab})$ on the first two outputs (C, Y).
2. For every polynomial $poly(\cdot)$ and all sufficiently large n's,

$$Prob[\ Y' = C'^b \ \wedge \ Y' \neq (g^b)^c \ \wedge \ Y' \neq (g^{ab})^c\ ] < \frac{1}{poly(n)}.$$

Roughly, the above conditions say that whenever I outputs (C, Y) such that $Y = C^b$, I' outputs not only (C, Y) but also c such that $Y = (g^b)^c$ or $Y = (g^{ab})^c$ (i.e., $C = g^c$ or $(g^a)^c$) with overwhelming probability.

SDHA-2 implies SDHA-1 and so we call SDHA-2 SDHA. We remark that in both assumptions, it is not required that I' can be effectively constructed given I, but rather that it exists. This is similar to the definition of AINUZK.

5 Main Theorem

We prove that 3R-ZK is an AINUZK argument.

Theorem 11. *There exists a 3-round auxiliary-input non-uniform computational zero-knowledge argument for any NP language under SDHA.*

Proof. There exists a non-uniformly secure commitment scheme under DLA [BM84]. By Theorem 7, the starting protocol exists under SDHA. Therefore, Theorem 11 follows combining Lemma 12 and 13. □

Our result does not apply to the notion of AIZK. However, if we strengthen SDHA with auxiliary-input, we can prove that 3R-ZK is AIZK. This issue is taken up in the last section.

5.1 3R-ZK is AINUZK

Lemma 12. *3R-ZK is AINUZK for L_{NP} under SDHA-1.*

Proof. Firstly, we focus on the computation of the cheating verifier $\hat{V}_x^y$ in V1, where the inputs are (p, q, g, A, M_1) and the outputs are (B, X) such that $X = B^a$. We consider it as the computation of I in SDHA-1. The inputs (p, q, g, A) to $\hat{V}_x^y$ play the role of the inputs (p, q, g, g^a) to I in SDHA-1. Furthermore, we can consider that the other quantities such as the common input x, the auxiliary-input y and the message M_1 are incorporated into I in SDHA-1. Therefore, we can apply SDHA-1 to the cheating verifier $\hat{V}_x^y$: for any cheating verifier $\hat{V}_x^y$, there exists a family of polynomial-size circuits $\hat{V}_x^{'y}$ which outputs not only (B, X) but also b such that $B = g^b$.

As mentioned in the observation of the property R2, when the simulator uses $\hat{V}_x^{'y}$ as a black-box, it can get the secret coins b and complete the simulation with b. For simplicity, we describe the simulator $S_{\hat{V}}$ as a probabilistic polynomial-time machine.

Machine: Simulator $S_{\hat{V}}$.
Input: x, y.
Output: a view $[x, y, (M_1, p, q, g, A)(B, X)(M_2, C, Y)]$.
S1. $S_{\hat{V}}$ performs P1-1 to get (p, q, g, A).
S2. $S_{\hat{V}}$ runs the honest-verifier simulator S_{HV} in the property B3 of the starting protocol to get (M_1, Y, M_2) such that Y is an element of order q in Z_p^*.
S3. $S_{\hat{V}}$ runs $\hat{V}_x^{'y}$ on (M_1, p, q, g, A) to get (B, X, b). $S_{\hat{V}}$ checks whether $X = B^a$. If this is false, $S_{\hat{V}}$ stops and outputs $[x, y, (M_1, p, q, g, A)(B, X)]$.
S4. $S_{\hat{V}}$ checks whether $B = g^b$. If this is false, $S_{\hat{V}}$ aborts.

S5. $S_{\hat{V}}$ computes C such that $Y = C^b$.
S6. $S_{\hat{V}}$ outputs $[x, y, (M_1, p, q, g, A)(B, X)(M_2, C, Y)]$.

Clearly $S_{\hat{V}}$ runs in probabilistic polynomial-time. The probability that $S_{\hat{V}}$ aborts in S4 is negligible under SDHA-1.

Now we show that the output distribution ensemble $\{S_{\hat{V}}(x,y)\}_{x\in L, y\in\{0,1\}^*}$ is computationally indistinguishable from $\{\mathsf{View}(P_x, \hat{V}_x^y)\}_{x\in L, y\in\{0,1\}^*}$. The proof is by contradiction. Assume there exists a distinguisher D that can distinguish $\{S_{\hat{V}}(x,y)\}_{x\in L, y\in\{0,1\}^*}$ from $\{\mathsf{View}(P_x, \hat{V}_x^y)\}_{x\in L, y\in\{0,1\}^*}$. Then we can construct a distinguisher D' that distinguishes the output distribution ensemble $\{S_{HV}(x,y)\}_{x\in L, y\in\{0,1\}^*}$ from $\{\mathsf{View}(M_x, A_x^y)\}_{x\in L, y\in\{0,1\}^*}$, in contradiction to the property B3 of the starting protocol (M, A). Given a view (M_1, Y, M_2) of the starting protocol (from either distributions), D' extends it to a view of 3R-ZK and invokes D on the extended view as follows:

Distinguisher: $D' = \{D'_{x,y}\}_{x\in L_{NP}, y\in\{0,1\}^*}$.
Input: (M_1, Y, M_2), where we assume that $(p, q, g) \in L_{PQG}$ is fixed in advance and that Y is an element of order q in Z_p^*.
Output: 0 or 1.
D1: $D'_{x,y}$ runs $S_{\hat{V}}$ to extend the input (M_1, Y, M_2) to a view $[x, y, (M_1, p, q, g, A)\ (B, X)\ (M_2, C, Y)]$, where the input (M_1, Y, M_2) is used as a result of S2, respectively.
D2: $D'_{x,y}$ invokes $D_{x,y}$ on the extended view.

If the input (M_1, Y, M_2) is from $\mathsf{View}(M_x, A_x^y)$, the extended view is distributed statistically close to $\mathsf{View}(P_x, \hat{V}_x^y)$. This is because the probability that $S_{\hat{V}}$ aborts is negligible and the output distributions of $\hat{V}_x^y$ and $\hat{V}_x'^y$ are statistically close on (B, X) under SDHA-1. On the other hand, if the input (M_1, Y, M_2) is from $S_{HV}(x,y)$, the extended view is distributed exactly alike the distribution $S_{\hat{V}}(x,y)$. Therefore, D' distinguishes the two ensembles in the property B3 of the starting protocol. □

Since $S_{\hat{V}}$ does not use $\hat{V}$ as a black-box, the above argument does not show that 3R-ZK is BSZK. SDHA-1 does not say that the circuit for $\hat{V}'$ can be effectively constructed given the circuit for $\hat{V}$, but rather that it exists. However, it is sufficient for our purpose since the definition of AINUZK also does not require that the circuit for $S_{\hat{V}}$ can be effectively constructed given the circuit $\hat{V}$. That is, under SDHA-1, for any verifier $\hat{V}$, there exist another one $\hat{V}'$ and the simulator $S_{\hat{V}}$. Therefore, the above argument shows that 3R-ZK is AINUZK.

5.2 3R-ZK is an Argument

Lemma 13. *3R-ZK is an argument for L_{NP} under SDHA-2.*

Proof. The completeness is trivially satisfied because of the property B1 of the starting protocol. We focus on the soundness. Assume that 3R-ZK does not have

negligible error probability. Then there exist a cheating prover $\hat{P}$, a polynomial $p_0(\cdot)$ and an infinite set $G = \{(x,y)\}$ of common inputs not in L_{NP} and auxiliary inputs such that $\mathrm{Acc}(\hat{P}^y_x, V_x) > 1/p_0(|x|)$ for all $(x,y) \in G$. We will show that this contradicts DLA. Let K be the set of all integers n for which G contains a common input x of length n. We will show that there exists an inverter $I = \{I_n\}_{n\in K}$ such that for all $n \in K$, there exist $(p,q,g) \in L_{PQG}$ (p is of length n) such that $\mathrm{Inv}^I_{DL}(p,q,g)$ is overwhelming.

Firstly, we construct a (probabilistic) polynomial-size circuit family $PG = \{PG_n\}_{n\in K}$ which outputs a (bad) instance $(p,q,g) \in L_{PQG}$ such that the discrete logarithm modulo p is easy to compute. For each $n \in K$ we fix some common input and some auxiliary input $(x,y) \in G$ and they are incorporated into PG_n. We allow PG_n to use $\hat{P}^y_x$ as a black-box. We allow PG_n to feed the random coins for $\hat{P}^y_x$. For simplicity, we describe PG_n as an expected polynomial-time machine and its expected running time is clearly $O(p_0(n)poly(n))$.

Machine: Prime Generator $\{PG_n\}_{n\in K}$.
Input: 1^n.
Output: (r, M_1, p, q, g, A).
Step 0: PG_n initiates $\hat{P}^y_x$ on the random coins r.
Step 1: PG_n runs both $\hat{P}^y_{x,r}$ and V_x to get a transcript $[(M_1,p,q,g,A)\ (B,X)\ (M_2,C,Y)]$. If this is rejecting, PG_n goes back to Step 0.
Step 2: PG_n outputs (r, M_1, p, q, g, A).

Let $\mathrm{Acc}(\hat{P}^y_{x,r}, V_x, (M_1,p,q,g,A))$ denote the conditional probability that V_x accepts $x \notin L_{NP}$ in interacting with $\hat{P}^y_{x,r}$ when the conversation so far is (M_1,p,q,g,A). We say that (p,q,g) is *bad* if $\mathrm{Acc}(\hat{P}^y_{x,r}, V_x, (M_1,p,q,g,A)) > 1/2p_0(n)$. Since $\mathrm{Acc}(\hat{P}^y_x, V_x) > 1/p_0(n)$, it must be that the probability (over r and the random coins of V leading to (M_1,p,q,g,A)) that $\mathrm{Acc}(\hat{P}^y_{x,r}, V_x, (M_1, p, q, g, A)) > 1/2p_0(n)$ is at least $1/2$. Therefore there exists a bad instance $(p,q,g) \in L_{PQG}$ for all $n \in K$ and the probability that the output $(p,q,g) \in L_{PQG}$ of PG_n is bad is at least $1/2$. We ignore the possibility that $(p,q,g) \notin L_{PQG}$ since it happens with negligible probability.

Before we describe an inverter $I = \{I_n\}_{n\in K}$, we apply SDHA-2 to $\hat{P}^y_{x,r}$. We consider the computation of the prover $\hat{P}^y_{x,r}$ in P2, where the inputs are $(p,q,g,A,M_1)(B,X)$ and the outputs are (M_2,C,Y) such that $Y = C^b$. We consider it as the computation of I in SDHA-2. The inputs (p,q,g,A,B,X) to $\hat{V}^y_{x,r}$ play the role of the inputs (p,q,g,g^a,g^b,g^{ab}) to I in SDHA-2. Furthermore, we can consider that the other quantities such as the random coins r, the message M_1, the common input x and the auxiliary-input y are incorporated into I in SDHA-2. Therefore, we can apply SDHA-2 to the cheating prover $\hat{P}^y_{x,r}$: for any cheating prover $\hat{P}^y_{x,r}$, there exists a family of polynomial-size circuits $\hat{P}'^y_{x,r}$ which outputs in P2 not only (M_2,C,Y) but also c such that $Y = B^c$ or X^c.

From now on, we assume that PG_n has output a bad instance $(p,q,g) \in L_{PQG}$ along with (r, M_1, A). We construct an inverter $I = \{I_n\}_{n\in K}$, where a common input x, an auxiliary input y, the output (r, M_1, A) of PG_n and a

such that $A = g^a$ are incorporated into each machine I_n. Furthermore, I_n is allowed to use $\hat{P}'^{y}_{x,r}$ as a black-box. For simplicity, we describe I_n as an expected polynomial-time machine.

Machine: Inverter $\{I_n\}_{n \in K}$.
Input: A bad (p, q, g) output by PG_n, a random number β of order q in Z_p^*.
Output: The discrete logarithm α such that $\beta = g^\alpha$.
Step 1: I_n generates a random number $b \in Z_q$ and computes $B = \beta g^b$ and $X = B^a$.
Step 2: I_n runs $\hat{P}'^{y}_{x,r}$ on $((M_1, p, q, g, A)(B, X))$ to get its response (M_2, C, Y, c). I_n checks whether (M_1, Y, M_2) is accepting according to the starting protocol. If this is false, I_n goes back to Step 1, otherwise I_n goes to Step 3.
Step 3: I_n checks whether $Y = B^c$ or $Y = X^c$. If this is false, I_n goes back to Step 1, otherwise I_n goes to Step 4.
Step 4: I_n generates a random number $b' \in Z_q$ and computes $B' = g^{b'}$ and $X' = B'^a$. If $B = B'$ then I_n outputs $\alpha = b' - b \bmod q$ (Note that $\beta g^b = g^{b'}$). Otherwise, I_n goes to Step 5.
Step 5: As in Step 2, I_n runs $\hat{P}'^{y}_{x,r}$ on $((M_1, p, q, g, A)(B', X'))$ to get its response (M_2', C', Y', c'). I_n checks whether (M_1, Y', M_2') is accepting according to the starting protocol. If this is false, I_n goes back to Step 4, otherwise I_n goes to Step 6.
Step 6: I_n checks whether $Y' = B'^{c'}$ or $Y' = X'^{c'}$. If this is false, I_n goes back to Step 4, otherwise I_n goes to Step 7.
Step 7: I_n outputs α such that $\beta = g^\alpha$.

I_n tries to get two different accepting transcripts. Since we assumed that (p, q, g) is bad, the expected running time of I_n is $O((p_0(n))^2 poly(n))$.

In Step 7, it holds that $Y = \beta^{ac} g^{abc}$ or $\beta^c g^{bc}$ and it also holds that $Y' = g^{ab'c'}$ or $g^{b'c'}$. Since $x \notin L_{NP}$, Y must equal Y' by the property B2 of the starting protocol. So it is easy to compute α such that $\beta = g^\alpha$ from the values (a, b, b', c, c'). Therefore we conclude that for all $n \in K$, there exists $(p, q, g) \in L_{PQG}$ such that $\mathrm{Inv}^I_{DL}(p, q, g)$ is overwhelming. This contradicts DLA. □

6 Concluding Remarks

We introduced a novel computational assumption SDHA so that we could provide a non-black-box simulation technique and construct a 3-round AINUZK protocol for NP. However, SDHA is strong and non-standard. It is fundamentally different from the standard complexity assumptions such as DLA and DHA in the sense that SDHA has double quantification (i.e., for every adversary, there exists another one such that something holds) whereas the standard assumptions have one quantifier (i.e., for every adversary, something holds). It is unlikely that SDHA holds under some standard complexity assumptions. Therefore, it is interesting and necessary to study the validity of SDHA.

Our result can not apply to AIZK. When we apply SDHA-1 to a probabilistic polynomial-time verifier $\hat{V}$, the resultant $\hat{V}'$ is a circuit family and the simulator $S_{\hat{V}}$ is also a circuit family. This does not satisfy the requirement of the definition of AIZK. One may think that we can prove that 3R-ZK achieves GMRZK or AIZK if SDHA-1 is formalized in the uniform model, where both I and I' are modeled by probabilistic polynomial-time machines. It is not the case since such uniform assumption (called uniform SDHA-1) can not deal with the common input x and the auxiliary-input y.

We can prove that 3R-ZK achieves the notion of AIZK if we further strengthen uniform SDHA-1 with auxiliary-input, that is, if we assume that uniform SDHA-1 holds even when both I and I' are allowed to take arbitrary (same) auxiliary-inputs (We call this strengthened assumption auxiliary-input uniform SDHA-1). This result leads us to an interesting corollary: under auxiliary-input uniform SDHA-1, it holds that $Cl(BSZK) \subset Cl(AIZK)$ unless NP $\subseteq$ BPP. However, auxiliary-input uniform SDHA-1 seems unreasonable [Go98-2]. Consider I as a universal machine which takes as its auxiliary-input the description of any circuit C which, given (g, g^a), outputs (B, X) such that $X = B^a$. Then, auxiliary-input uniform SDHA-1 says that I' can *reverse-engineer* the circuit C and output b such that $B = g^b$. Therefore, auxiliary-input uniform SDHA seems very strong and problematic.

Acknowledgments

This paper owes much to the valuable comments and suggestions of Oded Goldreich. He kindly helped us to revise our earlier version of this paper. We would like to thank the anonymous referees of Crypto'98 for valuable comments. We also thank Masahiro Wada and Koji Nakao for their encouragement.

References

[BeGo92] M. Bellare and O. Goldreich , "On Defining Proofs of Knowledge, " Proceedings of Crypto'92, 1992.

[BJY97] M. Bellare, M. Jakobsson and M. Yung, "Round-Optimal Zero-Knowledge Arguments Based on any One-Way Function, " Proceedings of Eurocrypt' 97, 1997.

[BMO90] M. Bellare, S. Micali and R. Ostrovsky, " Perfect Zero-Knowledge in Constant Rounds," Proceedings of 22nd STOC, 1990.

[Bl86] M. Blum, "How to Prove a Theorem So No One Else Can Claim It," Proceedings of the International Congress of Mathematicians, pp.1444-1451, 1986.

[BM84] M. Blum and S. Micali, "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits," SIAM Journal on Computing, Vol.13, No.4, pp.850-863, 1984.

[BCC88] G. Brassard, D. Chaum and C. Crépeau, "Minimum Disclosure Proofs of Knowledge, " Journal of Computer and System Sciences, Vol. 37, No. 2, pp. 156-189, 1988.

[BCY89] G. Brassard, C. Crépeau and M. Yung, " Everything in NP Can Be Argued in Perfect Zero-Knowledge in a Bounded Number of Rounds, " Proceedings of 16th ICALP, pp.123-136, 1989.

[BrCr86] G. Brassard and C. Crépeau, "Non-Transitive Transfer of Confidence : A Perfect Zero-Knowledge Interactive Protocol for SAT and Beyond, " Proceedings of 27th FOCS, 1986.

[DH76] W. Diffie and M. Hellman, "New Directions in Cryptography," IEEE Trans. Inform. Theory, Vol.22, No.6, pp.644-654, 1976.

[FFS88] U. Feige, A. Fiat, and A. Shamir, "Zero Knowledge Proofs of Identity," Journal of Cryptology, Vol.1, pp.77-94, 1988.

[FeSh89] U. Feige and A. Shamir, "Zero Knowledge Proofs of Knowledge in Two Rounds," Proceedings of Crypto'89, pp.526-544, 1989.

[Go93] O. Goldreich, "A Uniform-Complexity Treatment of Encryption and Zero-Knowledge," Journal of Cryptology, Vol.6, No. 1, pp.21-53, 1993.

[Go98-1] O. Goldreich, "Foundations of Cryptography (Fragments of a Book - Version 2.03)," February 27, 1998.

[Go98-2] O. Goldreich, private communication, May 1998.

[GoKa96] O. Goldreich and A. Kahan, "How to Construct Constant-Round Zero-Knowlege Proof Systems for NP," Journal of Cryptology, Vol.9, No. 3, pp.167-190, 1996.

[GoKr96] O. Goldreich and H. Krawczyk, "On the Composition of Zero-Knowledge Proof Systems," SIAM Journal on Computing, Vol.25, No.1, pp.169-192, 1996.

[GMW91] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems," Journal of the ACM, Vol.38, No.1, pp.691-729, 1991.

[GoOr94] O. Goldreich and Y. Oren, "Definitions and Properties of Zero-Knowlege Proof Systems," Journal of Cryptology, Vol.7, No. 1, pp.1-32, 1994.

[GMR85] S. Goldwasser, S. Micali, and C. Rackoff, "The Knowledge Complexity of Interactive Proofs," Proceedings of 17th STOC, pp.291-304, 1985.

[Ra80] M. O. Rabin, "Probabilistic Algorithm for Testing Primality, " Jounal of Number Theory, Vol 12, pp.128-138, 1980.

[SS77] R. Solovay and V. Strassen, "A Fast Monte-Carlo Test for Primality," SIAM Journal on Computing, Vol.6, No.1, pp.84-86, 1977.

[TW87] M. Tompa and H. Woll, "Random Self-Reducibility and Zero Knowledge Interactive Proofs of Possession of Information," Proceedings of 28th FOCS, pp.472-482, 1987.