ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.2, February 2008, pp.468−478
DOI: 10.3724/SP.J.1001.2008.00468

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

纯公钥模型下对NP语言的高效并发零知识证明系统*

邓 燚+, 林东岱

(中国科学院 软件研究所 信息安全国家重点实验室,北京 100080)

Efficient Concurrent Zero Knowledge Arguments for NP in the Bare Public-Key Model

DENG Yi+, LIN Dong-Dai

(State Key Laboratory of Information Security, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

+ Corresponding author: Phn: +86-10-62568254, E-mail: ydeng@is.iscas.ac.cn

Deng Y, Lin DD. Efficient concurrent zero knowledge arguments for NP in the bare public-key model. *Journal of Software*, 2008,19(2):468−478. http://www.jos.org.cn/1000-9825/19/468.htm

Abstract: This paper shows how to efficiently transform any 3-round public-coin honest verifier zero knowledge argument system for any language in NP into a 4 round (round-optimal) concurrent zero knowledge argument for the same language in the bare public-key model. The transformation has the following properties: 1) incurs only $O(1)$ (small constant, about 20) additional modular exponentiations. Compared to the concurrent zero knowledge protocol proposed by Di Crescenzo and Visconti in ICALP 2005, in which their transformation requires an overhead of $\Theta(n)$, the protocol is significantly more efficient under the same intractability assumptions; 2) yields a perfect zero knowledge argument under DL assumption. Note that the Di Crescenzo, *et al*.'s argument system enjoys only computational zero knowledge property. The transformation relies on a specific 3-round honest verifier zero knowledge proof of knowledge for committed discrete log. Such protocols that require only $O(1)$ modular exponentiations based on different kinds of commitment scheme are developed and they may be of independent interest.

Key words: concurrent zero knowledge; bare public-key model; proof of knowledge

摘 要: 提出了一种从3轮公开掷币的对任何NP语言的诚实验证者零知识证明系统到纯公钥模型下4轮(轮最优)对同一语言的具有并发合理性的并发零知识证明系统.该转化方法有如下优点:1) 它只引起 $O(1)$(常数个)额外的模指数运算,相比 Di Crescenzo 等人在 ICALP 05 上提出的需要$\Theta(n)$个额外的模指数运算的转化方法,该系统在效率上有着本质上的提高,而所需的困难性假设不变;2) 在离散对数假设下,该转化方法产生一个完美零知识证明系统.注意到 Di Crescenzo 等人提出的系统只具有计算零知识性质.该转化方法依赖于一个特殊的对承诺中的离散对数的 3 轮诚实验证者零知识的证明系统.构造了两个基于不同承诺方案的只需要常数个模指数运算的系统,这种系统可能有着独立价值.

关键词: 并发零知识;纯公钥模型;知识的证明

中图法分类号: TP309 **文献标识码**: A

* Supported by the National Natural Science Foundation of China under Grant No.60673069 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2003AA144030 (国家高技术研究发展计划(863))

Received 2006-05-18; Accepted 2006-10-10

1 Introduction

Zero knowledge (ZK for short) proof, a proof that reveals nothing but the validity of the assertion, was put forward in the seminal paper of Goldwasser, Micali and Rackoff[1]. Since its introduction, especially after the useful results demonstrated in Ref.[2], ZK proofs have become a fundamental tool in the design of cryptographic protocols. In recent years, the research is moving towards extending the security to cope with today's malicious communication environment. In particular, Dwork, *et al*.[3] introduced the stronger notion of concurrent zero knowledge and studied the effect of executing several instances of the same protocol concurrently. Though the concurrent zero knowledge protocols have wide applications in networks like Internet, unfortunately, they require logarithmic rounds for languages outside BPP in the plain model for the black-box case[4] and therefore are of round inefficiency. In the common reference string model, Damgard[5] showed that 3-round concurrent zero knowledge can be achieved efficiently. Surprisingly, using non-black-box technique, Barak[6] constructed a constant round non-black-box bounded concurrent zero knowledge protocol, which however is very inefficient.

We study the concurrent zero knowledge in a weak model with a very relaxed set-up assumption, the bare public-key model, which was introduced in Ref.[7] with the aim of getting constant round resettable zero knowledge arguments. Compared with some previous model such as common reference string model and the preprocessing model[8], this model seems to have minimal set-up assumption: It just assumes that each verifier stores a public key in a public file before any interaction with the prover begins. Note that we don't need any trust party to check something in the set-up stage. Despite its simplicity, the notion of soundness in this model, as Micali and Reyzin[9] pointed out, is subtler. There are four distinct notions of soundness: one time, sequential, concurrent and resettable soundness, each of which implies the previous one, moreover, there is NO black-box rZK satisfying resettable soundness for some non-trivial language.

Though there are arguments satisfying stronger notion of security than concurrent zero knowledge in the bare public-key model, such as resettable zero knowledge arguments with concurrent soundness[10], the study of concurrent zero knowledge in this model is still meaningful: almost known resettable zero knowledge arguments with concurrent soundness rely heavily on subexponential hardness assumption, while concurrent zero knowledge argument with concurrent soundness assumes only polynomial hardness assumption, as shown by Di Crescenzo and Visconti in Ref.[11] (a previous one with little flaw appeared in Ref.[12]).

Our results. We show an efficient transformation from any 3-round public-coin honest verifier zero knowledge argument system for any language in NP into a 4 round (round-optimal) concurrent *perfect* zero knowledge argument for the same language in the bare public-key model. Under *DL* assumptions, our transformation incurs only $O(1)$ (small *constant*, about 20) additional modular exponentiations. Compared to the concurrent zero knowledge protocol proposed by Di Crescenzo and Visconti in ICALP 2005[11], in which their transformation requires an overhead of $\Theta(n)$ (more than $7n$, where n is the security parameter) exponentiations, our protocol is significantly more efficient and enjoys stronger notion of security, i.e., perfect zero knowledge (note that the protocol in Ref.[11] enjoys only computational zero knowledge). To implement this efficient transformation, we also develop a 3-round public-coin proof of knowledge of committed discrete log that requires only several (*constant*) exponentiations, which may be of independent interest.

An independent work[13]. Very recently (after submission of this work), independent of this work, Visconti improved the results in Ref.[11] and also obtained an efficient concurrent *computational* zero knowledge argument for NP in BPK model[13] using a little more complicated technique. However, in this paper, beyond contribution to an efficient construction, we also achieve higher security, i.e., perfect zero knowledge, under assumption *weaker* than that assumed in Ref.[13] (our concurrent perfect zero knowledge argument relies on DL assumption, which is

weaker than DDH assumption used to achieve concurrent *computational* zero knowledge argument in Ref.[13]).

2 Preliminaries

In this section, we present some definitions and tools that will be used later.

We say a function $\mu(\cdot)$ is negligible if for every positive polynomial $f(\cdot)$ and all sufficiently large n, it holds that $\mu(n)<1/f(n)$. We denote by $\lambda\leftarrow_R\Lambda$ the process of picking a random element λ from Λ.

The BPK model. The bare public-key model (BPK model) makes the following assumptions: 1) a public file F that is a collection of records, and each containing a verifier's public key is available to the prover; 2) an honest prover P is an interactive deterministic polynomial-time algorithm that gives as inputs a security parameter n, a n-bit string $x\in L$, where L is an NP language, an auxiliary input w (the witness for $x\in L$, a public file F and a random tape r; 3) an honest verifier V is an interactive deterministic polynomial-time algorithm that works in two stages. In stage one, on inputting a security parameter n and a random tape, V generates a key pair (pk,sk) and stores pk in the file F. In stage two, on inputting sk, an n-bit string x and a random string y, V performs the interactive protocol with a prover, and outputs "accept x" or "reject x".

Definition 1. We say that the protocol (P,V) is complete for a language L in NP if for all n-bit string $x\in L$ and any witness w such that $(x,w)\in R_L$, here R_L is the relation induced by L, the probability that V interacts with P on the input w and outputs "reject x" is negligible in n.

Malicious prover and its attacks in the BPK model. Let s be a positive polynomial and P^* be a s-concurrent malicious prover. On inputting a public key of V, P^* performs at most s sessions as follows: 1) if P^* is already running $i-1$ sessions, $1<i-1<s$, it can output a special message "Starting x_i" to start a new protocol with V on the new statement; 2) at any point, it can output a message for any of its sessions, then immediately receives the verifier's response and continues.

A concurrent attack of a s-concurrent malicious prover P^* is executed in this way: 1) V runs on input n and a random string, and then obtains the key pair (pk,sk); 2) P^* runs on input n and pk. Whenever P^* starts a new session on a new statement chosen by itself, V is run on the new statement, a new random string and sk.

Definition 2. we say (P,V) satisfies concurrent soundness for a language L if for all positive polynomial s, for all s-concurrent malicious prover P^*, the probability that is in an execution of concurrent attack, V outputing "accept x" for a false statement $x\notin L$ is negligible in n.

Definition 3. An interactive argument system (P,V) in the BPK model is concurrent zero-knowledge if there exists a probabilistic polynomial-time algorithm M such that for any probabilistic polynomial-time algorithm V^*, for any polynomial s, for any $x_i\in L$, the length of x_i is n, $i=1,\ldots,s$, the following two distributions are indistinguishable:

1. the output of V^* that firstly generates F with s entries and interacts concurrently with s^2 instances of the honest prover: $P(x_i,w_i,pk_i,F)$, $1\le i,j\le s$, and each instance of the honest prover uses independent random strings, where w_i is a witness for $x_i\in L$, and pk is the j-th entry registered by V^* in F.

2. the output of M on inputs $x_1,\ldots,x_s$.

***Σ*-protocol**. A protocol (P,V) is said to be a Σ-protocol for relation R_L if it is of 3-move form (assume (a,e,z) is the three messages exchanged by prover P and verifier V in a session) and satisfies the following conditions:

1) *Completeness*: For all $(x,w)\in R_L$, if P has the witness w and follows the protocol, the verifier always accepts;
2) *Special soundness*: From any x and any pair of accepting transcripts (a,e,z) and (a,e',z'), where $e\neq e'$, one can efficiently compute w such that $(x,w)\in R_L$;
3) *Special honest-verifier ZK*: There exists a polynomial-time simulator M, which on input x and a random

challenge e outputs an accepting transcript of the form (a,e,z) which is (perfect/computational) indistinguishable from real transcript.

Many known efficient protocols such as Refs.[14,15] are Σ-protocols, furthermore, if one-way functions exist, there is a well-known standard Σ-protocol for Hamiltonian Cycle, and we can obtain Σ-protocols with special honest verifier perfect zero knowledge for NP by using perfect-hiding commitment scheme in the first message of this protocol.

Σ_{OR}-protocol is a special construction with Σ-protocol designed for "or-proof": given two statements $x_1 \in L_1$ and $x_2 \in L_2$, it allows a prover to show that he knows a witness for one of the above statements. As demonstrated in Ref.[3], given two protocols Σ_1 and Σ_2 for two relationships R_1 and R_2 respectively, we can construct Σ_{OR}-protocol for the following relationship efficiently: $R_{OR}=\{((x_1,x_2),w)|(x_1,w)\in R_1\}$ or $(x_2,w)\in R_2$. The new protocol is also a Σ-protocol and turns out to be *witness indistinguishable*.

In our construction, the verifier also executes a Σ_{OR}-protocol to prove the knowledge of one of the secret keys corresponding to his public key. Furthermore, as required in Ref.[11], we need a *partial-witness-independent* from this protocol: the message sent at its first round should have distribution independent from any witness for the statement to be proved. We can obtain this property using[15,16].

Commitment scheme. A commitment scheme is a two-phase two-party (the sender S and the receiver R) interactive protocol which has the following properties: 1) hiding: two commitments to different values (here we view a commitment as a variable indexed by the value that the sender committed to) are computationally distinguishable for every probabilistic polynomial-time (PPT, for short) R^*; 2) Binding: after sending the commitment to a value m, any PPT sender S^* cannot open this commitment to another value m' ($m\neq m'$) except with an negligible probability.

When the adversary which plays the role of the sender or the receiver is not restricted to be a PPT, we obtain two different types of commitment scheme with stronger security property. One is the perfect-hiding commitment scheme, in which the hiding property is required to hold against any (computational power unbounded) receiver, and the other is perfect-binding commitment scheme, in which the binding property is required to hold against any (computational power unbounded) sender. In this paper, we mainly use the perfect-hiding commitment scheme.

3 A New Building Block: Efficient Honest Verifier Zero Knowledge Proof of Knowledge of Committed Discrete Log

In this section we develop two Σ-protocols to prove the knowledge of committed discrete log based on Pedersen's commitment scheme and ElGamal commitment scheme, and these protocols are useful building blocks to implement the concurrent zero knowledge protocol in the BPK model efficiently, which will be described in next section.

Let security parameter be n, p and q be two primes such that $p=2q+1$, $|q|=n$, and let G_q denote the subgroup of Z_p^* with order q and g is a generator of the subgroup. Let $h_0=g^{x_0}$ and $h_1=g^{x_1}$, $C=Com(x_b,r)$ (b=0 or 1), where Com is a commitment scheme and r is the random string required in the commitment scheme. Our goal is to construct a Σ-protocol (with common inputs p, q, h_0, h_1, C and the description of the commitment scheme Com) in which the prover prove the following "or" statement:

Statement 1: $\exists x_b, r$, s.t. $h_0=g^{x_b}\wedge C=Com(x_b,r)$ or $h_1=g^{x_b}\wedge C=Com(x_b,r)$.

3.1 A Σ-protocol with special honest verifier *perfect* zero knowledge for statement 1 based on Pedersen's commitment scheme

Here we adopt the Pedersen's commitment scheme[17], which enjoys perfect-hiding property. We choose p,q,g,G_q described above as the description of the Pedersen's commitment scheme. To commit a value, the receiver picks a random numbers $h\in G_q$, and sends h to the sender. Then the sender commits to a value y as follows: it randomly chooses $r\in Z_q$, computes $C=g^y h^r$, sends C to the receiver. To decommit a commitment C, the sender delivers y and r.

In order to construct a Σ-protocol to prove the statement 1, we first construct a protocol to prove the following statement:

Statement 2: $\exists x_1, r$, s.t. $h_1 = g^{x_1} \wedge C = g^{x_1} h^r$.

Then using the technique from Refs.[15,16], it is easy to construct a Σ-protocol for statement 1 (based on the Pedersen's commitment scheme). Now we give the protocol for statement 2.

Protocol 1. A Σ-protocol for statement 2.

The common inputs: C, g, h, p, q, h_1.

The Prover's private input: x_1, r (such that $C = g^{x_1} h^r$, $h_1 = g^{x_1}$)

P step 1. The prover *P* picks $s,t\leftarrow_R Z_q$ randomly, computes $A=g^s h^t \bmod p$ and $B=h^t$. *P* sends A and B to the verifier.

V step 1. The verifier *V* picks $e\leftarrow_R Z_q$ randomly and sends e to the prover.

P step 2. The prover *P* computes $y=s+ex_1 \bmod q$ and $z=t+er \bmod q$, and sends y,z to the verifier.

V decision. The verifier accepts if only if $C^e A=g^y h^z \bmod p$ and $(C/h_1)^e B=h^z \bmod p$.

Proposition 1. Protocol 1 described above is a Σ-protocol with special honest verifier perfect zero knowledge for statement 2.

Proof: The completeness is straightforward. The property of the *Special soundness* follows from the fact: given two transcripts (A,B,e,y,z) and (A,B,e',y',z'), $e\neq e'$, we can compute x_1 and r from the equations $z=t+er \bmod q$, $z'=t'+e'r \bmod q$, $y=s+ex_1 \bmod q$ and $y'=s'+e'x_1 \bmod q$.

Furthermore, we construct a simulator *M* and show the property of special honest verifier *perfect* zero knowledge. *M* runs as follows. On inputting the common inputs C, g, h, p, q, h_1 and a challenge e, *M* picks randomly $y\leftarrow_R Z_q$ and $z\leftarrow_R Z_q$, then computes $A=g^y h^z C^{-e} \bmod p$, $B = h^z h_1^e C^{-e} \bmod p$, and outputs (A,B,e,y,z). It is easy to check that the output of *M* has the *identical* probability distribution as conversations between the honest *P*, *V* on the common input. □

Now we present a Σ-protocol for statement 1 using the technique in Refs.[15,16].

Protocol 2. A Σ-protocol for statement 1.

The common inputs: C, g, h, p, q, h_0, h_1.

The Prover's private inputs: x_b, r, b (such that $C = g^{x_b} h^r$, $h_b = g^{x_b}$)

P step 1. The prover *P* computes A_b and B_b according to Protocol 1 (with common inputs C, g, h, p, q, h_b), chooses $e_{1-b}\leftarrow_R Z_q$ randomly and runs *M* on input e_{1-b}, gets the output $(A_{1-b},B_{1-b},e_{1-b},y_{1-b},z_{1-b})$. *P* sends (A_b,B_b) and (A_{1-b},B_{1-b}) to the verifier.

V step 1. The verifier *V* picks $e\leftarrow_R Z_q$ randomly and sends e to the prover.

P step 2. *P* sets $e_b=e\oplus e_{1-b}$ and computes the last message (y_b,z_b) to the challenge e_b using (x_b,r) as witness according to Protocol 1. *P* sends e_b, e_{1-b}, y_b, z_b and (y_{1-b},z_{1-b}) to the verifier.

V decision. The verifier accepts if only if the two transcripts (A_b,B_b,e_b,y_b,z_b) and $(A_{1-b},B_{1-b},e_{1-b},y_{1-b},z_{1-b})$ are accepting.

It is clear that the above protocol is Σ-protocol . Furthermore, since the commitment scheme satisfies perfect-hiding property (therefore for every b, there exists x_b, r such that $C = g^{x_b} h^r$, $h_b = g^{x_b}$), the protocol enjoys special honest verifier perfect zero knowledge, and turns out to be perfectly witness distinguishable.

3.2 A Σ-protocol for statement 1 based on ElGamal's commitment scheme

Now we give another construction of the Σ-protocol for statement 1 based on Elgamal commitment scheme.

ElGamal commitment scheme is a basic application of ElGamal encryption scheme. p, q, g, G_q are described in the above subsection, but in this scheme we assumes that the DDH problem in G_q is hard. To commit a value y, the sender chooses a random numbers $x \leftarrow_R Z_q$ and computes $h=g^x$ (note that the sender chooses x itself and the committing stage does not require interaction), then it commits to a value y as follows: it randomly chooses $r \in Z_q$, computes $C=(C_1,C_2)=(g^r, g^y h^r)$, sends C to the receiver. To decommit a commitment C, the sender delivers y and r. The hiding property of this commitment scheme lies in DDH assumption on the subgroup G_q, and this scheme enjoys perfect-binding property: it is impossible to open a commitment C in different way for all powerful receivers.

As shown in the above subsection, to construct a Σ-protocol for statement 1, it is sufficient to construct a Σ-protocol for the following statement:

Statement 2′: $\exists x_1, r$, s.t. $h_1 = g^{x_1} \wedge C_1 = g^r, C_1 = g^r \wedge C_2 = g^{x_1} h^r$.

The common input for the Σ-protocol consists of $C=(C_1,C_2)$, (g,h,p,q), h_1 and the prover's private input is (x_1,r). The protocol runs as follows. In the prover's first step, the prover chooses $s,t \leftarrow_R Z_q$ randomly, computes $A_1=g^s \bmod p$, $A_2=g^t h^s \bmod p$ and $B=h^s \bmod p$, and sends A_1, A_2 and B to the verifier. Upon receiving the challenge e from the verifier, the prover computes $y=s+er \bmod q$ and $z=t+ex_1 \bmod q$, and sends y, z to the verifier. At last, the verifier accepts if only if the equations $C_1^e A_1 = g^y \bmod p$, $C_2^e A_2 = g^z h^y \bmod p$ and $(C_2/h_1)^e B = h^y \bmod p$ hold.

Proposition 2. The protocol described above is a Σ-protocol for statement 2′.

It is easy to prove this proposition in a similar way. However, this protocol enjoys special honest-verifier *computational ZK* only, that is, the transcript generated by the simulator is *computational* indistinguishable from the real transcript.

Analogously, We can construct a Σ-protocol for statement 1 using the or-proof technique shown in the above subsection. However, the resulting protocol enjoys only *computational* (not perfect) witness indistinguishability due to the computational-hiding property of the ElGamal commitment Scheme.

4 Efficient Concurrent Zero Knowledge Arguments with Concurrent soundness for NP in the BPK Model

In this section, we present a 4-round efficient concurrent zero knowledge argument with concurrent soundness in the BPK model for NP under the standard *DL* assumption.

For the sake of readability, we give some intuition before describing the protocol formally.

Considering a prover wanting to prove that a given string x is a member in a language L. Before the interaction with provers, the verifier generates a key pair and publishes the public key as follows. On inputting security parameter n, the verifier chooses two primes p and q such that $p=2q+1$, $|q|=n$, and let G_q denote the subgroup of Z_p^* with order q, and g is a generator of the subgroup, picks two random numbers x_0, $x_1 \in Z_q$, computes $h_0 = g^{x_0}$ and $h_1 = g^{x_1}$, then publishes h_0, h_1 as the public key, and keeps x_0, x_1 secret. Our argument consists of two-phase: in phase one of the argument, the verifier proves to the prover that he knows one of x_0, x_1 using 3-round *partial-witness-independent* Σ-protocol. In phase two, the prover uses a commitment scheme *Com* to compute a

commitment to a random strings y, and then proves the following statement using a Σ_{OR}-protocol: $x\in L$ or it commits to one of x_0, x_1 (i.e., y equals either x_0 or x_1).

We make two remarks here.

On differences between the approach in Ref.[11] and ours. We stress that there are two important differences between the approach to construct concurrent zero knowledge in BPK model in Ref.[11] and ours. The first is that in phase 2, Di Crescenzo, *et al*. have the prover committed to a "challenge" e' *bit by bit* for the Σ-protocol in which it proves that the statement to be proven is true using a dedicate atomic (concurrently sound) commitment scheme and sends this commitment along with the first message of this Σ-protocol. Upon receiving the challenge e and the prover answers the challenge $e''=e'\oplus e$. We remark that to commit to the "challeng" e' *bit by bit* is unavoidable in Di Crescenzo *et al*.'s approach. Our approach described above is to use the "or proof" technique to give a proof that the statement to be proven is true (which is also used in Ref.[12], and can be dated back to Ref.[18]. Note that Ref.[11] has shown that there is a flaw in the construction of Ref.[12], and we show that we can fix it using the commitment scheme *Com*), and this approach allows us to achieve a significantly more efficient concurrent zero knowledge protocol. The second one is, though our commitment scheme *Com* performs the same function as the statistically-binding commitment scheme in the atomic commitment scheme in Ref.[11], we show that computational-binding (with a perfect-hiding) commitment scheme suffices to achieve concurrent soundness of our concurrent zero knowledge argument. This allows us to achieve perfect zero knowledge.

On efficiency. We denote by Π_v the proof of knowledge given by the verifier in phase one, similarly, denote by Π_p the proof of knowledge given by the prover in phase two. We note that Σ-protocol for the discrete log requires only 3 exponentiations[15], so we can implement Π_v (i.e., "OR-proof" of knowledge of one of two discrete logs) using the technique shown in last section (also see Refs.[15,16]) with 6 exponentiations. For the protocol Π_p, with assuming a Σ-protocol for a language L in NP, we can construct Π_p by using the same "OR-proof" technique to combine the Σ-protocol for the language L and the Σ-protocol for statement 1. Note that this combination incurs only additional overhead of several constant exponentiations (i.e., the exponentiations required in the Σ-protocol for statement 1). Thus, we conclude that our whole transformation (from the Σ-protocol to the concurrent zero knowledge argument for a language L) requires only $O(1)$ additional exponentiations. Note that our transformation is significantly more efficient than the transformation in Ref.[1], which requires an overhead of $\Theta(n)$ exponentiations.

Now we describe the protocol fomally.

Protocol 3. A concurrent zero knowledge argument with concurrent soundness for L.

The common input: The public file F, n-bit string $x\in L$, an index i that specifies the i-th entry $pk_i=(p,q,g,h_0,h_1)$ of F.

The Prover's private input: A witness w for $x\in L$.

The Verifier's private input: A secret key x_b (such that $h_b=g^{x_b}$), here b is a random bit chosen by the verifier.

V step 1. Invokes the protocol Π_v in which V proves knowledge of x_b, and computes the first message of protocol Π_v and sends it to P.

P step 1

(1) Chooses a random number $y\in Z_q$, picks a random element $r\in Z_q$ and computes $C=Com(y,r)$ (note that if the Pedersen's commitment scheme is in use, then the verifier needs to send the first message), i.e., a random number $h\in Z_q$, to the prover in V in the committing phase step 1 because the Pedersen's scheme needs interaction in the committing phase.

(2) invokes the protocol Π_p, in which P proves to V that it knows a witness for $x \in L$ or there exist y, r, and b such that $C=Com(y,r)$ and $y=x_b$, and computes the first message a of protocol Π_p;

(3) Picks a random string (i.e., the challenge) as the second message of Π_v.

(4) Sends C, the first message a of protocol Π_p and the challenge of Π_v to V.

V step 2.

(1) Computes the last message of protocol Π_v according to the challenge send by P in P Step 1, and sends it to P.

(2) Sends a random challenge e of protocol Π_p to P.

P step 2. P checks whether the transcript of protocol Π_v is accepting. If so, P computes the last message z of protocol Π_p;

V decision. V accepts if only if (a,e,z) is an accepting transcript of Π_p.

Theorem 1. If *Com* is a secure (computational hiding and computational binding) commitment scheme, the protocol 3 described above is a concurrent zero knowledge argument with concurrent soundness for any language L in NP. Furthermore, if the commitment scheme *Com* enjoys *perfect-hiding* property, and the Σ-protocol for L (underlying the protocol Π_p) satisfies special honest verifier *perfect* zero knowledge property, protocol 3 is a *perfect* zero knowledge argument.

Let's convey some main ideas in the proof first. To see the above argument is a concurrent zero knowledge protocol, consider a simulator that extracts the secret key used in the proof of knowledge given by the Verifier, and then uses the secret key as the witness to simulate the verifier's view straight line. It is no hard to verify that this simulator works. For the concurrent soundness, we first note that the witness indistinguishability is preserved under concurrent composition. If a prover can convince the verifier on a false statement in a session, it must know one of the two discrete log (i.e., x_0, x_1) and therefore we can extract the discrete log. Importantly, the discrete log we extract from the prover will be independent of the discrete log used in the proof of knowledge given by the verifier due to the partial-witness-independent property of Π_v and the binding property of *Com*, this gives us a chance to break the *DL* assumption.

Proof: **Completeness**. Straightforward.

Concurrent (perfect) Zero Knowledge. The analysis is very similar to the analysis presented in Ref.[11]. Here we omit the tedious proof and just provide some intuition. As usual, we can construct a simulator S that extracts all secret keys corresponding to those public keys registered by the malicious verifier from the executions of Π_v and then uses them as witnesses to complete executions of Π_p in expected polynomial time. To prove the concurrent zero knowledge property, we need to show that the output of S is indistinguishable from the real interactions. This can be done by constructing a non-uniform hybrid simulator *HS* and showing the output of *HS* is indistinguishable from both the output of S and the real interaction. *HS* runs as follows. Taking as inputs all these secret keys and all the witnesses of statements in interactions, *HS* computes commitments (at P step 1) exactly as S does but uses the same witnesses as the honest provers do in the executions of Π_p. It is easy to see that the output of the hybrid simulator is indistinguishable from both the transcripts of real interactions (because of the computational-hiding property of *Com*) and the output of S (because of the witness indistinguishability of Π_p), therefore, we prove the output of S is indistinguishable from the real interactions.

If *Com* enjoys perfect-hiding property and the Σ-protocol for L satisfies special honest verifier *perfect zero* knowledge, then it is easy to see that Π_p is a perfect witness indistinguishable proof, so protocol 3 satisfies *perfect* zero knowledge.

Concurrent Soundness. We prove this property by contradiction. Assume that the protocol does not satisfy

the concurrent soundness property, thus there is a s-concurrently malicious prover P^*, concurrently interacting with V, makes the verifier accept a false statement $x\notin L$ with non-negligible probability p. We now construct an algorithm A that breaks the DL assumption with non-negligible probability. A runs as follows. On inputting a 4-tuple discrete log challenge 4-tuple (p,q,g,h') (i.e., given description of the group, find x' such that $h'=g^{x'}$), A randomly chooses a bit b and a number $x_b\in Z_q$, then A registers $pk=(p,q,g,h_0,h_1)$ as the public key, where $h_b=g^{x_b}$, $h_{1-b}=h'$. A guesses a session number $j\in\{1,\ldots,s\}$ (assumes that in session j, P^* will cheat the verifier successfully on a false statement $x\notin L$. Note that the event that this guess is correct happens with probability $1/s$). □

Now, A interacts with P^* as honest verifier (note that A knows the secret key $sk=x_b$ corresponding to the public key pk) for all but the j-th session. Specifically, A employs the following extraction strategy:

1. A acts as the honest verifier in the first time it involves in the j-th session. That is, it completes Π_v using $sk=x_b$ as a secret key, and sends a random challenge e to the P^* in V step 2. At the end of this session A get an accepting transcript (a,e,z) of Π_p;
2. A rewinds P^* to the point of beginning of V step 2 in the j-th session, and it sends a random challenge e' $(e'\neq e)$ to P^* again. When A gets another accepting transcript (a,e',z') of Π_v at V step 3, it ends and computes the valid witness w' for the statement that $x\in L$ or C is a commitment to one of the two discrete logs of h_b and h_{1-b} from the two transcripts (a,e,z) and (a,e',z') of Π_p, and outputs them. Otherwise, A plays V step 2 again.

We denote this extraction process with $EXTRA$.

We first note that the witness w' must satisfy $w'=(y,r)$, $C=Com(y,r)$ and $h_b=g^y\vee h_{1-b}=g^y$ because $x\notin L$. If $h'=h_{1-b}=g^y$, A breaks the DL assumption, otherwise, y satisfies $h_b=g^y$, that is the secret key we know, A fails. Next we claim that A succeeds in breaking the DL assumption with non-negligible probability.

Assume otherwise, except with at most a *negligible* probability, A fails. Then we can construct a non-uniform algorithm A' to break the witness indistinguishability of Π_v or the computational binding of the commitment scheme Com. The non-uniform algorithm A' takes as auxiliary input (h_0,h_1,x_0,x_1) (with input both secret keys) and interacts with P^* under the public key $pk=(p,q,g,h_0,h_1)$. It performs the following experiment:

1. *Simulation* (*until A' receives the first message a of Π_p in the j-th session*). A' acts exactly as the A. Without loss of generality, let A' uses x_0 as witness in all executions of Π_v that are completed before P step 1 of the j-th session. Once A' receives the first message a of Π_p in the j-th session, it splits this experiment and continues independently in the following games:
2. *Extracting Game* 0: A' continues the above simulation and uses the same extraction strategy of A. In particular, it runs as follows:
   1) Continuing to simulate: A' uses x_0 as witness in all executions of Π_v that takes place during this game;
   2) Extracting: if A' obtains an accepting transcript (a,e,z) at the end of the first run of Π_p in the j-th session, it rewinds to the point of beginning of V step 2 in the j-th session and replays this round by sending another random challenge e' $(e'\neq e)$ until he gets another accepting transcript (a,e',z') of Π_p, and then A' outputs a valid witness.
3. *Extracting Game* 1: A' repeats Extracting Game 0 but A' uses x_1 as witness in all executions of Π_v during this game (i.e., those executions of Π_v completed after the P step 1 in the j-th session). At the end of this game, A' will obtain two accepting transcripts (a,e,z), (a,e',z') and output a valid witness. Note that Π_v is *partial-witness-independent* (so we can decide to use which witness at the last (third) step of Π_v), A' can choose witness at its desire to complete any execution of Π_v that is completed after the P step 1 in the

j-th session during this game.

We denote by EXP_0 the *Simulation* in stage 1 described above with its first continuation *Extracting Game* 0, similarly, denote by EXP_1 the same *Simulation* with its second continuation *Extracting Game* 1.

Note that the P^*'s view in EXP_0 is identical to its view in *EXTRA* in which *A* uses x_0 (b=0) as witness in all executions of Π_v, so the outputs of A' at the end of EXP_0 are identical to the outputs of *A* taking x_0 as the secret key in *EXTRA*. That is, except with negligible probability, A' outputs a valid witness $w_0'=(x_0,r_0)$.

Consider *A*'s behavior in *EXTRA* when it uses x_1 (b=1) as the secret key. The behavior differs from the behavior of A' in EXP_1 in those executions of Π_v that are completed before the *P* step 1 in the *j*-th session: A' uses x_0 as witness in all those executions, while *A* uses x_1 as witness. However, P^* cannot tell these two cases apart because Π_v is witness indistinguishable and all those executions of Π_v have not been rewound during both *EXTRA* and EXP_1 (note that A' does not rewind past the *P* step 1 in the *j*-th session in the whole experiment). Thus, we can claim that at the end of EXP_1, A' outputs a valid witness $w_1'=(x_1,r_1)$.

In the above experiment conducted by A', the first message *a* sent by P^* in the *j*-th session contains a commitment *C* and this message *a* (therefore *C*) remains unchanged during the above whole experiment. Clearly, with non-negligible probability, A' outputs two valid witnesses $w_0'=(x_0,r_0)$ and $w_1'=(x_1,r_1)$ from the above two games such that the following holds: $h_0=g^{x_0}$, $h_1=g^{x_1}$, C=Com(x_0,r_0) and C=Com(x_1,r_1). This contradicts the computational-binding property of the scheme *Com*.

In sum, we have proved that if *Com* enjoys computational-binding and Π_v is a witness indistinguishable protocol with *partial-witness-independence* property, then *A* succeeds in breaking the *DL* assumption. In another words, if the *DL* assumption holds, it is infeasible for P^* to cheat an honest verifier in the concurrent settings with non-negligible probability.

Acknowledgement Yi Deng thanks Giovanni Di Crescenzo for many helpful discussions, and Ivan Visconti for discussion on the proof of security in Ref.[11] and pointing out that our argument satisfies perfect zero knowledge property.

References:

[1] Goldwasser S, Micali S, Rackoff C. The knowledge complexity of interactive proof systems. SIAM. Journal Computing, 1989, 18(1):186−208.

[2] Micali S, Wigderson A. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. Journal of ACM, 1991,38(3):691−729.

[3] Dwork C, Naor M, Sahai A. Concurrent zero-knowledge. In: Vitter J, ed. Proc. of the ACM Symp. on Theory of Computing. New York: ACM Press, 1998. 409−418.

[4] Canetti R, Kilian J, Petrank E, Rosen A. Concurrent zero-knowledge requires $\tilde{\Omega}(\log n)$ rounds. In: Vitter J, *et al*., eds. Proc. of the ACM Symp. on Theory of Computing. New York: ACM Press, 2001. 570−579.

[5] Damgard I. Efficient concurrent zero-knowledge in the auxiliary string model. In: Preneel B, ed. Advances in Cryptology-EUROCYPT. LNCS 1807, Berlin: Springer-Verlag, 2000. 174−187.

[6] Barak B. How to go beyond the black-box simulation barrier. In: Proc. of the IEEE Conf. on Foundation of Computer Science. IEEE Computer Society Press, 2001. 106−115.

[7] Canetti R, Goldreich O, Goldwasser S, Micali S. Resettable zero knowledge. In: Yao F, *et al*., eds. Proc. of the ACM Symp. on Theory of Computing. New York: ACM Press, 2000, 235−244.

[8] Di Crescenzo G, Ostrovsky R. On concurrent zero knowledge with preprocessing. In: Wiener M, ed. Advances in Cryptology-Crypto. LNCS1666, Berlin: Springer-Verlag, 1999. 485−502.

[9] Micali S, Reyzin L. Soundness in the public-key model. In: Kilian J, ed. Advances in Cryptology-Crypto. LNCS 2139, Berlin: Springer-Verlag, 2001. 542−565.

[10] Di Crescenzo G, Persiano G, Visconti I. Constant round resettable zero knowledge with concurrent soundness in the bare public-key model. In: Franklin M, ed. Advances of Cryptology-Crypto. LNCS 3152, Berlin: Springer-Verlag, 2004. 237−253.

[11] Di Crescenzo G, Visconti I. Concurrent zero knowledge in the public-key model. In: Italiano GF, ed. Proc. of the Int'l Colloquium on Automata, Languages and Programming. LNCS 3580, Berlin: Springer-Verlag, 2005. 816−827.1

[12] Zhao Y. Concurrent/Resettable zero knowledge with concurrent soundness in the bare public-key model and its applications. Report 2003/265, Cryptology ePrint Archive. http://eprint.iacr.org/2003/265

[13] Visconti I. Efficient zero knowledge on the Internet. In: Bugliesi M, *et al*., eds. Proc. of the Int'l Colloquium on Automata, Languages and Programming. LNCS 4052, Berlin: Springer-Verlag, 2006. 22−33.

[14] Guillou LC, Quisquater JJ. A practical zero-knowledge protocol fitted to security microprocessors minimizing both transmission and memery. In: Günther CG, ed. Advance in Cryptology-EUROCRYPT. LNCS 330, Berlin: Springer-Verlag, 1988. 123−128.

[15] Schnorr CP. Efficient signature generation for smart cards. Journal of Cryptology, 1991,4(3):239−252.

[16] De Santis A, Di Crescenzo G, Persiano G, Yung M. On monotone formaula close of SZK. In: Proc. of the IEEE Conf. on Foundation of Computer Science. IEEE Computer Society Press, 1994. 454−465.

[17] Pedersen TP. Non-Interactive and information-theoreticl secure verifiable secret sharing. In: Feigenbaum J, ed. Advances in Cryptology-Crypto. LNCS 576, Berlin: Springer-Verlag, 1991. 129−140.

[18] Feige U, Shamir A. Witness indistinguishability and witness hiding protocols. In: Ortiz H, ed. Proc. of the ACM Symp. on Theory of Computing. New York: ACM Press, 1990. 416−426.

DENG Yi was born in 1977. He is a candidate for Ph.D. at State Key Laboratory of Information Security, Institute of Software, the Chinese Academy of Sciences. His research interest lies in foundation of cryptography, especially zero knowledge.

LIN Dong-Dai was born in 1964. He is now a full time research professor and deputy director of State Key Laboratory of Information Security, Institute of Software, the Chinese Academy of Sciences. His current research areas are cryptology, information security, grid computing and symbolic computations.