Xue R, Li NH, Li JT. Algebraic construction for zero-knowledge sets. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 23(2): 166–175 Mar. 2008

Algebraic Construction for Zero-Knowledge Sets

Rui Xue[1] (薛　锐), Ning-Hui Li[2] (李宁辉), and Jiang-Tao Li[3] (李江滔)

[1] *The State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences Beijing 100080, China*

[2] *Department of Computer Science, Purdue University, West Lafayette, U.S.A.*

[3] *Software and Solutions Group, Intel Corporation, Hillsboro, U.S.A.*

E-mail: rxue@is.iscas.ac.cn; ninghui@cs.purdue.edu; jiangtao.li@intel.com

Revised January 3, 2008.

Abstract Zero knowledge sets is a new cryptographic primitive introduced by Micali, Rabin, and Kilian in FOCS 2003. It has been intensively studied recently. However all the existing ZKS schemes follow the basic structure by Micali *et al.* That is, the schemes employ the Merkle tree as a basic structure and mercurial commitments as the commitment units to nodes of the tree. The proof for any query consists of an authentication chain. We propose in this paper a new algebraic scheme that is completely different from all the existing schemes. Our new scheme is computationally secure under the standard strong RSA assumption. Neither mercurial commitments nor tree structure is used in the new construction. In fact, the prover in our construction commits the desired set without any trapdoor information, which is another key important difference from the previous approaches.

Keywords zero knowledge set, Merkle tree, accumulator strong RSA assumption, random oracle

1 Introduction

Zero knowledge set (ZKS) is a new cryptographic primitive introduced by Micali, Rabin, and Kilian[1] in FOCS 2003. It allows a prover to commit an arbitrary finite set S and later on, for any string x, the prover responds with a proof whether $x \in S$ or $x \notin S$ without revealing any knowledge other than these membership assertions. While zero knowledge proof of membership is now a routine in cryptography, the biggest challenge in ZKS lies in the non-membership proof for potentially infinite many elements without leaking more information about the committed set.

It is not hard to see that a ZKS scheme could be implemented with zero knowledge proof for the general NP languages. However the approaches there are very involved and far from practice. Micali, Rabin, and Kilian[1] give an efficient construction of zero-knowledge sets using the Merkle hash tree[2] and the Pedersen commitment scheme[3]. The key point to reduce the complexity is the use of "faked" Pedersen commitment with trapdoors to deal with nonmembers of the committed set. These "faked" commitment is developed later into mercurial commitment[4].

The main tools in [1] is Merkle's authentication tree and Peterson's commitment. After hashing strings in $\{0,1\}^*$ into $\{0,1\}^k$. Each string in the target set S has a fixed position in the binary tree of depth k. They are separately committed by the Pedersen commitments. These commitments are then (separately) hashed with their siblings as it does in the Merkle's tree. Recursively, these hashed values are committed with the Pedersen commitments. Here we have to pay attention to the siblings that are not in S (in the leaves) or that has not committed value during the hash process. In this case, a faked Pedersen commitment will be used to commit a random value there. This faked Pedersen commitment has a property that, with trapdoor, it can be revealed into any desired value. The commitment in the root is the commitment for the set S.

A proof for an element $x \in S$ in the above construction consists of an authentication sequence as does in the Merkle tree. Together with the witnesses of the commitments at the nodes along the path from x to

Regular Paper

This work is supported by NSF of USA under Grant Nos. IIS-0430274, and CCR-0325951, and sponsors of CERIAS. Rui Xue is partially supported by the Fund of the China Scholarship Council, partially by National Natural Science Foundation of China under Grant No. 60773029, National Grand Fundamental Research 973 Program of China under Grant No. 2007CB311202, and the National High Technology Research and Development 863 Program of China under Grant No. 2006AA01Z427.

Most of this work was done while Jiang-Tao Li and Rui Xue were at Purdue University.

the root are all real Pedersen commitments. A proof for $x \notin S$ consists of only authentication sequence without revealing any information of the witnesses.

The research works about ZKS followed are all under the frame of Merkle tree. These tree-based schemes have potential weakness that it is hard to update whenever the committed set are updated.

Our Contributions. In this paper, we propose a new algebraic ZKS scheme. It is able to implement in common reference string model. The prover and verifier will share a common random string or random parameter, that enables the prover to commit the set and to provide proofs for enquired elements by the verifier later. The idea is that to map each element into unique prime numbers. The committed elements are multiplied into one element and then committed. The committed elements can be proved by prover and non-committed elements also can be proved to be NOT in the committed. The soundness is guaranteed under the strong RSA assumption. The proof corresponding to the queries makes use of efficient zero knowledge proof of knowledge in exponentiations, which is secure in the random oracle model. Our scheme is efficient in commitment and storage phases as well as in communication complexity comparing to the previous approaches.

Related Work. Micali, Rabin, and Kilian[1] introduced the notion of zero-knowledge set. They gave a construction using the Merkle hash tree and the Pedersen[3] commitment scheme introduced above. Their construction uses "fake" Pedersen commitment with trapdoors to deal with nonmembers of the committed set. Chase *et al.*[4] abstracted the commitment construction in [1] and introduced the notion of mercurial commitments. It extends the general commitment scheme with a "soft" commitment in addition to the ordinary "hard" commitment. A soft commitment can be decommitted as any value, that is, soft decommitment is not binding. The authors implemented the mercurial commitment with several cryptographic primitives. Recently, Liskov[5] considered the updatable zero-knowledge sets. Ostrovsky *et al.*[6] investigated the general database queries for committed one, but not to assume the privacy in general. The privacy keeping scheme will employ the zero knowledge proof for general NP statement and by no means efficient. Catalano *et al.*[7] showed that Mercurial commitments is equivalent to trapdoor commitment and the existence of non-interactive zero-knowledge sets is equivalent to the existence of collision-resistant hash functions. Gennaro and Micali[8] recently extended the ZSK to get independent ZKS so that to defense the man-in-the-middle attacks.

Our scheme is an accumulator like construction. Accumulators were first introduced by Benaloh and de Mare[9] as a method to condense a set of values into one short accumulator, such that there is a short witness for each value that has been accumulated. In the mean time, it is infeasible to find a witness for a value that has not been accumulated. Barić and Pfitzmann[10] proposed a construction of a collision-resistant accumulator under the strong RSA assumption. Camenisch and Lysyanskaya[11] further proposed a dynamic accumulator in which elements can be efficiently added into or removed from the accumulator. Accumulators have been widely used for many applications[9−12] such as membership testing, time stamping, authenticated directory, and certificate revocation. Jiang-Tao Li *et al.*[13] introduced the notion of universal accumulator.

In a universal accumulator scheme[13], one can accumulate a set of elements such that each value has either a membership proof or a non-membership proof. However, the universal accumulator scheme proposed in [13] cannot be used directly for ZKS because, given a set of values to be accumulated, the accumulator reveals information about the set. In [14] a new definition for statistical zero-knowledge sets is proposed and a new scheme is constructed using trapdoor DDH groups. The construction there is based on the assumption of knowledge of exponentiation, and trapdoor DDH groups, but secure in standard model. The construction in this paper is based on standard assumption and with the cost that it is secure in the random oracle model.

Organizations. The rest of this paper is organized as follows. In Section 2, we give our notations and security assumptions. In Section 3, we give out the formal definition of zero knowledge sets. Section 4 introduces several cryptographic primaries that will be used in our main construction in Section 5. They are cited from various sources in the literatures. The contexts in Subsection 4.3 mainly come from [14]. In Section 5, we present our construction for zero knowledge sets. Section 6 is dedicated to the proof of zero knowledge property of our scheme. We conclude our paper in the last section.

2 Notations and Assumptions

2.1 Notations

In the rest of this paper, we use the following notations. Let $n = pq$ be an RSA modulus. Function $\phi(n)$ denotes the Euler totient function, and $\mathbb{Z}_n^*$ the set of all integers less than n and relative prime to n. We use QR_n to denote the set of quadratic residues modular n.

A negligible function, denoted by $\mathrm{neg}(k)$, represents any function that vanishes faster than the inverse of any fixed positive polynomial. That is, for every polynomial $p(x)$ and for all large enough integers n, $\mathrm{neg}(n) < 1/p(n)$. If S is a probability space, then the probability assignment $x \leftarrow_R S$ means that an element x is chosen at random according to S. If F is a finite set, then $x \leftarrow_R F$ denotes that x is chosen uniformly from F. If p is a predicate and $S_1, S_2, \ldots, S_m$ are probability spaces, then the notation $\mathtt{Pr}[x_1 \leftarrow_R S_1, \ldots, x_m \leftarrow_R S_m\colon\ p(x_1, x_2, \ldots, x_m)]$ denotes the probability that $p(x_1, \ldots, x_m)$ will be true after the ordered execution of the probabilistic assignments $x_1 \leftarrow_R S_1, \ldots, x_m \leftarrow_R S_m$.

We use notation used by Camenisch and Stadler in [15] for the various zero-knowledge proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms. For instance,

$$\mathrm{PK}\{(\alpha, \beta) : y = g^{\alpha} h^{\beta} \wedge (u \leqslant \alpha \leqslant v)\}$$

denotes a zero-knowledge proof of knowledge of integers α and β, such that $y = g^{\alpha} h^{\beta}$ holds, where $u \leqslant \alpha \leqslant v$.

2.2 Security Assumptions

The security of our construction is based on the strong RSA assumption, which assumes that it is infeasible to solve the following problem: given an RSA modulus n and a random $x \leftarrow_R \mathbb{Z}_n^*$, find $e > 1$ and $y \in \mathbb{Z}_n^*$ such that $y^e = x \bmod n$. The strong RSA was introduced by Barić and Pfitzmann[10] and has been used in proving the security of many cryptographic schemes (e.g., [16–18]). It can be formally stated as follows.

Assumption 1 (Strong RSA Assumption). *For every probabilistic polynomial-time algorithms A,*

$$\mathtt{Pr}[n \leftarrow G(1^k), x \leftarrow_R \mathbb{Z}_n, (y, e) \leftarrow A(n, x) : \\ y^e = x \pmod n \wedge 1 < e < n] = \mathrm{neg}(k),$$

where $G(1^k)$ is a algorithm that generates an RSA modulus n of size k, and $\mathrm{neg}(k)$ is a negligible function.

Our security proofs also use the following lemma[18,19]:

Lemma 1. *For any integer n, given integers $u, v \in \mathbb{Z}_n^*$ and $a, b \in \mathbb{Z}$, such that $u^a = v^b \bmod n$ and $\gcd(a, b) = 1$, one can efficiently compute $x \in \mathbb{Z}_n^*$ such that $x^a = v \bmod n$.*

To see the correctness of this lemma, observe that, as $\gcd(a, b) = 1$, one can use the extended Euclid algorithm to find $c, d \in \mathbb{Z}$ such that $bd = 1 + ac$. Let $x = (u^d v^{-c} \bmod n)$, then

$$x^a = (u^a)^d v^{-ac} = (v^b)^d v^{-ac} = v \pmod n.$$

Random Oracle Model. In our proofs, we use the random oracle model, which is an idealized security model by Bellare and Rogaway[20] to analyze the security of certain natural cryptographic constructions. Roughly speaking, a random oracle is a function $H\colon\ X \to Y$ chosen uniformly at random from the set of all functions $\{h\colon\ X \to Y\}$ (we assume Y is a finite set). An algorithm can query the random oracle at any point $x \in X$ and receive the value $H(x)$ in response. Random oracles are used to model cryptographic hash functions such as SHA-1. Note that security in the random oracle model does not imply security in the real world. Nevertheless, the random oracle model is a useful tool for validating natural cryptographic constructions.

Zero Knowledge Proof of Knowledge. Our construction uses zero-knowledge proof of knowledge for discrete logarithm. Let n be an RSA modulus, g be an element in $\mathbb{Z}_n$, and $y = g^x \bmod n$, one can efficiently prove the knowledge of x given g, y, and n using the protocol in [15, 17]. We denote such a protocol runs as $\mathrm{PK}\{\alpha\colon\ y = g^{\alpha} \bmod n\}$. Note that we can convert an interactive zero-knowledge protocol to a non-interactive one using Fiat-Shamir heuristic[21] under the random oracle model.

3 Definition of Zero-Knowledge Sets

A zero-knowledge set scheme consists of four probabilistic polynomial-time algorithms: setup, commit, prove, verify, where setup is to set up the public parameters, commit generates a commitment of an arbitrary set, prove gives non-interactive proof for any queries, and verify verifies the proofs generated by prove. A secure zero-knowledge set scheme satisfies the following three properties.

- *Completeness*: for any set S, for any $x \in \{0,1\}^*$, an honest prover who correctly generates a commitment to S can always convince the verifier that x is in S if $x \in S$ or x is not in S if $x \notin S$.
- *Soundness*: given a commitment to a set S formed by a (malicious) prover, no probabilistic polynomial-time prover could find a string x and prove that x is in S and simultaneously prove x is not in S.
- *Zero-Knowledge*: there exists a probabilistic polynomial-time simulator sim such that for any finite set $S \subseteq \{0,1\}^*$, no polynomial-time distinguisher can tell whether she is interacting with an honest prover committed to S, or interacting with the simulator sim who has only oracle access to the set S.

Among the preceding four algorithms, setup is computed jointly by the prover and the verifier, commit and prove are executed by the prover, algorithm verify is executed by the verifier. Given a security parameter k, the setup algorithm setup outputs a random k-bit RSA modulus n and a generator g such that the factorization of n is known to neither the prover nor the verifier.

Our scheme, as in [1] to generate parameters used in Paderson like commitment, has a parameter generation procedure in order to generate an RSA modulus n, which factorization is hard and is unknown to all polynomial time algorithm including the prover and the verifier. We denote the set of such kind of RSA modula as RSA, to denote its subset of all elements with length k as $RSA(k)$, an element in it as n. It will be used during the commitment and proof. This procedure processes according to shared reference string σ with respect to secure parameter k. In practice this procedure could be replaced by an agreement to use a public well known "hard" to factor integer (like RSA challenge numbers.)

The formal definition given here is objecting to our scheme, following [1], as follows.

Definition 1. *A zero-knowledge set has four probabilistic polynomial-time algorithms* (setup, commit, prove, verify)*, and it satisfies three secure properties.*

The four algorithms are as follows.

• setup. This polynomial-time algorithm is to generate system parameters for commitment and proofs. In our scheme, they are a random RSA modulus n and an element $g \in \mathbb{Z}_n^*$, where the length of n is k and the factorization of n is unknown. That is,

$$(n, g) \leftarrow_R \mathsf{setup}(1^k, \sigma)$$

where $n \in RSA(k)$, $g \in \mathbb{Z}_n$ and $RSA(k)$ is the set of RSA modulus of length k.

• commit. Given any set S, a security parameter k, and system parameters (n, g), commit outputs a commitment c. That is,

$$(c, SK) \leftarrow_R \mathsf{commit}(1^k, \sigma, S, (n, g)).$$

• prove. This algorithm is a probabilistic polynomial-time algorithm that, given the input $(S, 1^k, \sigma, \mathsf{commit}, SK)$, and an additional input $x \in \{0,1\}^*$, computes a proof π_x about x. That is,

$$\pi_x \leftarrow_R \mathsf{prove}(1^k, \sigma, c, SK, x).$$

• verify. Given $(1^k, \sigma, c, x, \pi_x)$ as input, verify verifies the proof with respect to whether $x \in S$ or $x \notin S$. If the proof is valid and $x \in S$, then it outputs 1; else if the proof is valid and $x \notin S$ then it outputs 0; if the proof is invalid, it outputs $\perp$.

The three security properties they enjoyed are as follows.

• $\mathsf{Completeness}$. For all set S and $\forall x \in S$ such that the probability of following event is 1.

$$\begin{aligned}&\{\sigma \leftarrow_R \{0,1\}^{k^c};\\ &(n, g) \leftarrow \mathsf{setup}(1^k, \sigma);\\ &(c, SK) \leftarrow \mathsf{commit}(1^k, \sigma, S, (n, g));\\ &\pi_x \leftarrow \mathsf{prove}(\sigma, c, SK, x):\\ &\mathsf{verify}(\sigma, c, x, \pi_x) \neq \perp\}.\end{aligned}$$

• $\mathsf{Soundness}$. $\forall x \in \{0,1\}^*$ and for any probabilistic polynomial-time algorithm P', the probability of the following event is negligible.

$$\begin{aligned}&\{\sigma \leftarrow_R \{0,1\}^{k^c};\\ &(n', c', \pi_1', \pi_2') \leftarrow P'(1^k, \sigma):\\ &n' \in RSA(k) \wedge\\ &\mathsf{verify}(1^k, \sigma, c', x, \pi_1') = 0 \wedge\\ &\mathsf{verify}(1^k, \sigma, c', x, \pi_2') = 1\}.\end{aligned}$$

• $\mathsf{Zero\text{-}knowledge}$. There exists a polynomial time simulator sim such that for any algorithm Adv, all sufficient large integer $k > 0$, a set S. The following two views are computational undistinguishable: $view(k) \approx view'(k)$, where $view$ is

$$\begin{aligned}&\{\sigma \leftarrow_R \{0,1\}^{k^c}; (n, g) \leftarrow \mathsf{setup}(1^k, \sigma);\\ &(c, SK) \leftarrow \mathsf{commit}(1^k, \sigma, S, (n, g));\\ &(x_1, s_1) \leftarrow \mathsf{Adv}(1^k, \sigma, c);\\ &\pi_{x_1} \leftarrow \mathsf{prove}(1^k, \sigma, c, SK, x_1);\\ &(x_2, s_2) \leftarrow \mathsf{Adv}(1^k, \sigma, c, s_1, \pi_{x_1});\\ &\pi_{x_2} \leftarrow \mathsf{prove}(1^k, \sigma, c, SK, x_2);\\ &\cdots : n, g, c, x_1, \pi_1, x_2, \pi_2, \ldots\}\end{aligned}$$

and $view'$ is

$$\begin{aligned}&\{(\sigma', n', g', c', SK') \leftarrow \mathsf{sim}(1^k);\\ &(x_1, s_1') \leftarrow_R \mathsf{Adv}(1^k, \sigma', c', n');\\ &\pi_{x_1}' \leftarrow \mathsf{sim}(1^k, \sigma', c', SK', x_1);\\ &(x_2, s_2') \leftarrow \mathsf{Adv}(1^k, \sigma', c', n', s_1', \pi_{x_1}');\\ &\pi_{x_2}' \leftarrow \mathsf{sim}(1^k, \sigma', c', SK', x_2);\\ &\cdots : n', g', c', x_1, \pi_1', x_2, \pi_2', \ldots\}.\end{aligned}$$

4 Building Blocks

Before we present our construction to the zero-knowledge sets, we review two building blocks that will be used in our scheme.

4.1 Generation of Shared RSA Modulus

In our construction, the prover and the verifier share common parameters: (n, g), where n is the product of two large safe primes p, q. Neither the prover nor the verifier knows the factorization of n. The following are three possible approaches to realize this.

The first possibility is to use interactive proof preprocessing model[22]. Before the commitment and proofs procedure, prover and verifier, by employing the protocol by Algesheimer, Camenisch and Shoup in [23], will interactively produce a shared RSA modulus n that is a production of two safe primes p, q, where the factorization of n is unknown to anyone.

The second possibility is to generalize RSA modulus using the shared random string σ. Sander[24] first suggested an accumulator with unknown modulus, and worked out an algorithm to generate the shared RSA-UFO. A number n is called an RSA-UFO if n has at least two large prime factors p, q such that no one could, in polynomial time, find the factorization of $n = n_1 \cdot n_2$ and $p|n_1$ and $q|n_2$. This algorithm allows prover, according to the shared random string, to generate such a modulus in polynomial time of k, $\log(1/\epsilon)$, where k is the secure parameter, and $\epsilon > 0$.

By using this approach, we have to base the security of our scheme on the so called *strong RSA-UFO assumption*, which is the same as usual strong RSA assumption except that the latter use RSA-UFO modulus rather than usual RSA modulus.

The third possibility is to use those published by the third party (such as those published as RSA challenges). We could even assume that there is a server that takes secure parameter k and the random string σ as input and randomly to generate in black-box an RSA modulus n of length k as production of two safe primes. The commitment and the proofs will be independent of the server.

4.2 Zero-Knowledge Proof of DL

The zero-knowledge proof of discrete logarithm is used in many literatures[15,25,26]. Our scheme will adapt the following case of proof of discrete logarithm.

Proof of knowledge was defined in [27]. Proofs of discrete logarithms are more important block in many cryptographic systems ranging from identification to voting systems. It allows a prover to convince a verifier that he knows a solution of some kinds of discrete logarithms. The proof should have the following properties: completeness means an honest prover of knowing the solutions can successfully convince the verifier. Soundness properties mean a cheating prover of not knowing the solution can successfully convince verifier with negligible probability. And the verifier cannot obtain useful information about the solution that prover knows.

Let n be an RSA modulus that the discrete logarithm problem is hard in $\mathbb{Z}_n$. Element g is an element in $\mathbb{Z}_n$. The discrete logarithm of an element $y \in \mathbb{Z}_n$ to the base g is the unique integer $x \in \mathbb{Z}_n$ for which $y = g^x \pmod n$.

The conventional proof of the knowledge of the discrete logarithm of $y = g^x$ to the base g processes as follows[15]:

prover computes

$$r \in_R \mathbb{Z}_n, \quad t = g^r, \quad c = \mathcal{H}(g, y, t),\ v = r - cx.$$

The proof is (c, v). The verifier checks the validity of

$$t = g^v y^c, \quad c = \mathcal{H}(g, y, t).$$

By assumption that $\mathcal{H}$ is a truly random function, the protocol can be proved to be computational zero-knowledge in random oracle model.

This protocol is denoted conventionally as PK$\{\alpha$: $y = g^\alpha\}$.

4.3 Mapping Strings to Primes

Our construction requires that the elements in the universe (a finite subset of which will be committed to) be represented by prime numbers. This needs to map any strings into primes, with the following two properties.

Unambiguity: no two strings map to the same prime number, and each string has exactly one prime number that it maps to.

Efficiency: the prime number corresponding to a string can be computed in time polynomial in the length of the string.

Although several approaches for representing bit strings as primes are suggested in the accumulator literature[10,16], these mappings are not unambiguous and not suffices for the purposes here. So instead we use the simpler mappings suggested in [14] that have the desired properties.

We shall consider the problem of mapping binary strings of length k into prime numbers, where k is the security parameter (and all efficient operations should be in time polynomial in k, with failure probability negligible in k). Consider the following mapping: first map the given binary string unambiguously to a sufficiently large integer t; then map it to *the smallest prime number greater than* t^2.

This mapping relies on a widely-believed conjecture on the density of prime numbers.

Conjecture 1 (Cramer-Granville Conjecture[28]). *$p_{n+1} - p_n = O(\log^2 p_n)$, where p_n stands for the n-th largest prime number.*

By this conjecture, for sufficiently large t, there is a prime in the range $[t^2, (t+1)^2)$. This guarantees unambiguity. Further, it also guarantees that finding the smallest prime number greater than t^2 can be done in time polynomial in $\log t$ (using a polynomial time primality test to test each integer starting from t^2, until a prime is found).

5 Construction for ZKS

Our scheme, compared to the scheme from Micali *et al.*[1], has the advantages that it does not use mercurial commitments[4], and it does not employ tree structure which makes it efficiently updatable. Another feature of this construction is that it does not possess a trapdoor during the construction, which is completely deferent from previous construction. And our complexity is similar to the scheme in [1] (cf. also [4]).

5.1 Construction

We use two hash functions H_0 and H_1 in the construction. Let H be the hash function mapping each string of length $k+1$ to a prime number, H can be constructed using the techniques in Subsection 4.3. We define the hash functions H_0 to be $H_0(x) = H(x\|0)$ and H_1 to be $H_1(x) = H(x\|1)$, which are hashes from $\{0,1\}^k$ to primes.

Construction 1 (Zero-Knowledge Sets). The construction has the following four algorithms.

1. setup. The prover and the verifier jointly set up the public parameters (n, g). They take the security parameter k as the input and generate a k-bit RSA modulus n which is the product of two safe primes. They also randomly choose a $g \leftarrow_R QR_n$.

2. commit. To commit a set $S = \{x_1, x_2, \ldots, x_m\}$, the prover randomly chooses a k-bit binary string y and computes

$$u = H_1(y)H_0(x_1)H_0(x_2)\cdots H_0(x_m),$$
$$c = g^u \bmod n,$$

where c is the commitment for the set S. The prover keeps (S, u, y) privately for generating proofs in the future.

3. prove. When the prover receives a query about an element $x \in \{0,1\}^*$, it will respond depending on whether $x \in S$:

• If $x \in S$. The prover computes $p_x = H_0(x)$, $u_x = u/p_x$, $c_x = g^{u_x} \bmod n$, and sends c_x to the verifier.

• If $x \notin S$. The prover computes $p_x = H_0(x)$, chooses a pair of integers $(a, b) \in \mathbb{Z} \times \mathbb{Z}$ such that $au + bp_x = 1$, and computes $c_x = c^a \bmod n$ and $d = g^{-b} \bmod n$. The prover sends c_x, d, and proof π

$$\mathrm{PK}\{(\alpha, \beta) : (c_1 = c^\alpha \ \wedge \ d = g^\beta)\}$$

to the verifier.

4. verify. When the verifier queries about an element x, it receives a message from the prover. The format of the message depends on whether x is in the committed set or not. The verifier does the following to check.

• The message is of the form (c_x), i.e., the prover asserts that $x \in S$. The verifier computes $p_x = H_0(x)$ and checks

$$c_x^{p_x} = c \pmod n.$$

It accepts if the preceding equation holds.

• The message is of the form (c_x, d, π), i.e., the prover asserts that $x \notin S$. The verifier computes $p_x = H_0(x)$ and checks whether

$$c_x = d^{p_x} g \pmod n$$

and that π is a proof that the prover knows a and b such that $c_x = c^a \pmod n$ and $d = g^{-b} \pmod n$. The verifier accepts if both checks succeed.

5.2 Completeness and Soundness Properties

In this subsection, we show that our construction is secure, i.e., our construction is complete, sound, and zero-knowledge scheme.

Completeness. For the completeness property, we need to show that if an honest prover can always convince the verifier about the membership of the queried element.

Theorem 1. *Scheme 1 satisfies the completeness property.*

Proof. For any finite set S and any string x, the prover can prove the membership correctly. Given a set $S = \{x_1, \ldots, x_m\}$, let u be computed as in the commitment phase of our construction and p_x be $H_0(x)$. If $x \in S$, then $p_x | u$, and the prover can compute c_x such that $c_x^{p_x} = g^u = c$. If $x \notin S$, then $\gcd(p_x, u) = 1$ with overwhelming probability. The reason is that $p_x \notin \{H_0(x_1), \ldots, H_0(x_m)\}$, and by the definition of H_0 and H_1, if H is a random hash function mapping each string to a prime, then $\mathtt{Pr}[p_x \in \{H_1(x) : x \in \{0,1\}^*\}]$ is negligible. Therefore, p_x is relatively prime to u, the prover can find a and b such that $au + bp_x = 1$. The prover can effectively convince the verifier that $x \notin S$. $\square$

Soundness. The soundness property guarantees that the prover cannot lie about the membership of any string x. That is, it should neither allow to prove a nonmember x of X to be a member in X, nor prove a member to be nonmember. We have the following.

Theorem 2. *Under the strong RSA assumption, Construction* 1 *satisfies the soundness property.*

Proof. For an adversary prover, who, given $n \in RSA(k)$, $g \in_R \mathbb{Z}_n$, with input $(1^k, \sigma, S)$, outputs the commitment c.

If the adversary can generate two proofs for some $x \in \{0,1\}^*$ that $x \in S$ and $x \notin S$, i.e., there are c_x, c_1, d, π such that

$$c_x^{p_x} = c \ (\bmod n), \quad c_1 = g \cdot d^{p_x} \ (\bmod n)$$

and

$$\pi = \mathrm{PK}\{(\alpha, \beta) : c_1 = c^{\alpha} \wedge d = g^{\beta} \bmod n\}$$

which means adversary could find α, β such that

$$c^{\alpha p_x} = g^{1+\beta p_x}.$$

From Shamir's Lemma 1, we get $y, e \in \mathbb{Z}_n$ such that $y^e = g$, which break the strong RSA assumption. □

6 Zero-Knowledge Property

Theorem 3. *The Construction* 1 *is a computational zero-knowledge scheme in the random oracle model.*

For the proof of the zero-knowledge property, we construct a polynomial time algorithm sim to simulate a real proof procedure, such that a view generated by sim is computationally indistinguishable from a view in an actual protocol run.

The main idea of constructing the simulator sim is as follows: given security parameter k, and an oracle for answering membership queries for a finite set S, the simulator sim generates two safe primes of length $k/2$, whose product is n'; sim also chooses an element $g' \in QR_{n'}$, where $QR_{n'}$ is the set of all quadratic residues in $\mathbb{Z}_{n'}^*$. The simulator sim then chooses a prime $p' \in \{0, 1, \ldots, \lfloor n/4 \rfloor\}$ at random, and computes $c' = g^{p'}$. Suppose polynomial $q(\cdot)$ is a a bounding polynomial for queries. Simulator sim chooses a string $\sigma' \in \{0,1\}^{O(q(k))}$ at random for shared random string. The output is (σ', c', p'), where p' is the secret to be used in proof. Whenever an element $x \in S$ is enquired, simulator sim will, after enquiring oracle S, produce $p'' = H_0(x)$ and compute $c'^{1/p''}$ with the knowledge of the factorization of n', to output $c'^{1/p''}$ as the answer. If an element $x \notin S$ is enquired, simulator does the same as the prover does in Construction 1, though with different parameters. In either case, if an element y satisfying $H_0(y) = p''$ is enquired, then output nothing. This, however, will happen with negligible probability.

Simulator sim. On input k, S, where S is an oracle to sim and k is security parameter. Simulator sim proceeds as follows.

1) *Parameters Setting Up.* Let $q(\cdot)$ be a polynomial bounding the number of queries. The simulator sim starts by uniformly selecting a string $\sigma \in \{0,1\}^{O(q(k))}$ to be used as the shared random string during the querying and answering process.

The simulator sim chooses two safe primes p', q' of length $k/2$, computes $n' = p'q'$, and chooses a generator $g' \in QR_{n'}$. It also chooses another prime $p'' \in \{0, 1, \ldots, \lfloor n'/4 \rfloor\}$ and computes $c' = g'^{p''}$. The simulator outputs $(\sigma', n', g', c', p'')$.

2) *Simulating the Proofs.* For a sequence of adaptive chosen elements $x_1, x_2, \ldots, x_m, \ldots$, the simulator sim conducts a sequence of proofs $\pi'_1, \pi'_2, \ldots, \pi'_m$ correspondingly.

The simulator sim imitates the proof of membership (or nonmembership) of x in S as follows: when receiving a query about the element x_i, let $p_x = H_0(x_0)$, sim queries the oracle for S, and

• if x is in S, sim computes $c_x = c'^{1/p_x} \bmod n'$ using the knowledge of $\phi(n')$, and outputs c_x as the proof;

• if x is not in S, sim computes (a', b') such that $a'p'' + b'p_x = 1$, and computes $c'^{a'} \bmod n'$ and $d' = g'^{(-b')} \bmod n'$. It then outputs (c', d', π') as the nonmembership proof, where π' is a zero-knowledge proof of knowledge $\mathrm{PK}\{a', b' | \ c'_1 = c'^{a'} \wedge d' = g'^{b'}\}$.

We use $view_t$ to denote the view of interacting with the actual protocol in Construction 1 and $view'_t$ to denote the view generated by the simulator sim, where t is the number of queries one makes.

These two views are given as follows.

$$\begin{aligned} view_t = \{ & \sigma \leftarrow_R \{0,1\}^{k^c}; (n, g) \leftarrow \mathsf{setup}(1^k, \sigma); \\ & (c, SK) \leftarrow \mathsf{commit}(1^k, \sigma, S, (n, g)); \\ & (x_1, s_1) \leftarrow \mathsf{Adv}(1^k, \sigma, c); \\ & \pi_{x_1} \leftarrow \mathsf{prove}(1^k, \sigma, c, SK, x_1); \\ & (x_2, s_2) \leftarrow \mathsf{Adv}(1^k, \sigma, c, s_1, \pi_{x_1}); \\ & \pi_{x_2} \leftarrow \mathsf{prove}(1^k, \sigma, c, SK, x_2); \\ & \cdots : n, g, c, x_1, \pi_1, x_2, \pi_2, \ldots, x_t, \pi_t \}, \end{aligned}$$

$$\begin{aligned} view'_t = \{ & (\sigma', n', g', c', SK') \leftarrow \mathsf{sim}(1^k); \\ & (x_1, s'_1) \leftarrow_R \mathsf{Adv}(1^k, \sigma', c', n'); \\ & \pi'_{x_1} \leftarrow \mathsf{sim}(1^k, \sigma', c', SK', x_1); \\ & (x_2, s'_2) \leftarrow \mathsf{Adv}(1^k, \sigma', c', n', s'_1, \pi_{x_1}); \end{aligned}$$

$$\pi'_{x_2} \leftarrow \mathsf{sim}(1^k, \sigma', c', SK', x_2);$$
$$\cdots : n', g', c', x_1, \pi'_1, x_2, \pi'_2, \ldots, x_t, \pi'_t\}.$$

We now show that $view_t$ and $view'_t$ are computationally indistinguishable, provided that $H_0(x_i) \neq p''$ for all $i \leqslant t$, which holds with overwhelming probability.

Lemma 2. *For any polynomial time algorithm D and any $t \in O(q(k))$, if $H_0(x_i) \neq p''$ for all $i \leqslant t$, the following is negligible over k.*

$$|\mathtt{Pr}[D(view_t,) = 1] - \mathtt{Pr}[D(view'_t) = 1]| \tag{1}$$

where the probabilities are over the coin toss of D and σ, σ' which $view_t$ and $view'_t$ depend on.

Proof. We denote $D(view_t) = 1$ as $A(t)$, and $D(view'_t) = 1$ as $A'(t)$, then

$$\begin{aligned}
&\big|\mathtt{Pr}[D(view_t) = 1] - \mathtt{Pr}[D(view'_t) = 1]\big| \\
&= \big|\mathtt{Pr}[A(t)|A(t-1)] \cdot \mathtt{Pr}[A(t-1)] \\
&\quad - \mathtt{Pr}[A'(t)|A'(t-1)] \cdot \mathtt{Pr}[A'(t-1)]\big| \\
&\leqslant \big|\mathtt{Pr}[A(t)|A(t-1)] \\
&\quad - \mathtt{Pr}[A'(t)|A'(t-1)]\big| \cdot \mathtt{Pr}[A(t-1)] \\
&\quad + \big|\mathtt{Pr}[A(t-1)] - \mathtt{Pr}[A'(t-1)]\big| \cdot \mathtt{Pr}[A'(t)|A'(t-1)],
\end{aligned} \tag{2}$$

To prove the proposition, the following claim is used.

Claim 1. *The following formulae α_1, α_2 are all negligible functions in k:*

(a) *$\alpha_1 = |p_1 - p_2|$, where $p_1 = \mathtt{Pr}[D(n, g, c, 1^k) = 1]$ and $p_2 = \Pr[D(n', g', c', 1^k) = 1]$.*

(b) *$\alpha_2 = \big|\mathtt{Pr}[\pi_t = \delta] - \mathtt{Pr}[\pi'_t = \delta]\big|$, where δ is an element $1 \leqslant \delta \leqslant \max\{|QR_n|, |QR_{n'}|\}$ if $x_t \in S$, and a tuple (h_1, h_2)*[1] *if $x_t \notin S$, where $0 < h_1, h_2 \leqslant \max(|QR_n|, |QR_{n'}|)$.*

With the claim we proceed with the proof of Lemma 2. Suppose, for the sake of contradiction, that (1) is not negligible, then there exists a polynomial time algorithm D such that for some values t, (1) is not negligible. Let t_0 be the smallest such number, then

$$\big|\mathtt{Pr}[A(t_0 - 1, k)] - \mathtt{Pr}[A'(t_0 - 1, k)]\big| = \text{neg}(k) \tag{3}$$

and (a) of Claim 1 is the base case. And (b) of Claim 1 implies

$$\begin{aligned}
&\big|\mathtt{Pr}[A(t_0, k)|A(t_0 - 1, k)] \\
&\quad - \mathtt{Pr}[A'(t_0, k)|A'(t_0 - 1, k)]\big| = \text{neg}(k).
\end{aligned} \tag{4}$$

These together imply that (2) is negligible and hence (1) is also negligible in k. This ends proof of Lemma 2. □

We now prove Claim 1.

(a) We need to prove that, for any $r \in \{1, 2, \ldots, \max\{|QR_n|, |QR_{n'}|\}\}$

$$\big|\mathtt{Pr}[c = r] - \mathtt{Pr}[c' = r]\big| = \text{neg}(k).$$

This is true because $|n| = |n'| = k$ and g and g' can be considered to be the generators of QR_n and $QR_{n'}$, respectively (readers can refer to [29] for more details). In this case, $|QR_n| = \phi(n)/2$, $QR_{n'} = \phi(n')/2$. Then $\mathtt{Pr}[c = r] = 2/\phi(n)$ and $\mathtt{Pr}[c' = r] = 2/\phi(n')$. Since n and n' are safe integers of length k, $\phi(n)$ and $\phi(n')$ are integers of length 2^{k-3}; therefore,

$$\begin{aligned}
\big|\mathtt{Pr}[c = r] - \mathtt{Pr}[c' = r]\big| &= \Big|\frac{2}{\phi(n)} - \frac{2}{\phi(n')}\Big| \\
&= \Big|\frac{2(\phi(n') - \phi(n))}{\phi(n)\phi(n')}\Big| \leqslant \Big|\frac{2 \cdot 2^{k-3}}{2^{k-4} 2^{k-4}}\Big| \\
&= \frac{1}{2^{(k-6)}} = \text{neg}(k).
\end{aligned}$$

(b) The case that $x_t \in S$ can be proved in ways similar to the proof in part (a).

For the case $x = x_t \notin S$, the tuple (h_1, h_2) satisfies $0 < h_1, h_2 \leqslant \max(|QR_n|, |QR_{n'}|)$. The real proof in Construction 1 outputs c_x, d satisfying $c_x = c^a = g^{ua}$, $d = g^{-b}$ such that $au + bp_x = 1$.

We notice that for any a, there is a prime p such that $p \nmid au$ and au is prime to p. Then there exists b such that $au + bp = 1$. While for any $b \in \{1, \ldots, |QR_n|\}$ that is relatively prime to u, there exists an integer $a \in \mathbb{Z}_n$ such that $au + bq = 1$. That is, a ranges over $\{1, \ldots, |QR_n|\}$, and b ranges over all integers in $\{1, \ldots, |QR_n|\}$ that are relatively prime to u and has $\phi(u) + |QR_n|/u$ possibilities.

$$\mathtt{Pr}[\pi_t = (h_1, h_2)] = \frac{1}{|QR_n|\Big(\phi(u) + \dfrac{|QR_n|}{u}\Big)}.$$

Similarly,

$$\mathtt{Pr}[\pi'_t = (h_1, h_2)] = \frac{1}{|QR_{n'}|(\phi(p'') + \dfrac{|QR_{n'}|}{p''}\Big)}.$$

We thus have

$$\begin{aligned}
&\big|\mathtt{Pr}[\pi_t = (h_1, h_2)] - \mathtt{Pr}[\pi'_t = (h_1, h_2)]\big| \\
&= \Big|\frac{1}{|QR_n|(\phi(u) + |QR_n|/u)} \\
&\quad - \frac{1}{|QR_{n'}|(\phi(p'') + |QR_{n'}|/p'')}\Big|
\end{aligned}$$

[1]Here we treat the zero-knowledge proof of discrete logarithm as a black box, without consideration further.

$$\leqslant \Big|\frac{\alpha\cdot\beta}{|QR_n|^2\cdot|QR_{n'}|^2}\Big|$$

$$\leqslant \Big|\frac{\max(|QR_n|^3,|QR_{n'}|^3)}{\min(|QR_n|^4,|QR_{n'}|^4)}\Big|$$

$$\leqslant \frac{(2^{(k-2)})^3}{(2^{(k-4)})^4}=\frac{2^{3k-6}}{2^{4k-8}}=\frac{1}{2^{k-2}}=\mathrm{neg}(k).$$

where $\alpha = \max(|QR_n|,|QR_{n'}|)$ and $\beta = \max(|QR_n|^2,|QR_{n'}|^2)$.

This proves Claim 1.

7 Conclusion and Future Work

A new zero knowledge set construction based on the accumulators is proposed. Our scheme is completely different from the existing ones in that ours is an algebraic construction without tree, without trapdoor, and with no mercury commitment components. It is efficient in commitment and storage. In addition, the length of proof is much shorter than previous schemes.

The future work includes extending our scheme to have independent properties as defined in [8], investigating the update efficiency, and extending it to implement more general queries other than just membership queries.

References

[1] Micali S, Rabin M, Kilian J. Zero-knowledge sets. In *Proc. the 44th Annual IEEE Symposium on Foundations of Computer Science* (*FOCS*), Cambridge, MA, USA, 2003, p.80.

[2] Merkle R C. A certified digital signature. In *Proc. Advances in Cryptology—CRYPTO'89*, Brassard G (ed.), Santa Barbara, California, United States, *Lecture Notes in Computer Science*, Vol. 435, Springer-Verlag, 1990, 20–24 Aug., 1989, pp.218–238.

[3] Pedersen T P. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proc. Advances in Cryptology — CRYPTO'91*, Santa Barbara, California, USA, 1992, pp.129–140.

[4] Chase M, Healy A, Lysyanskaya A, Malkin T, Reyzin L. Mercurial commitments with applications to zero-knowledge sets. In *Proc. Advances in Cryptology — EUROCRYPT'05*, Aarhus, Denmark, 2005, pp.422–439.

[5] Liskov M. Updatable zero-knowledge databases. In *Proc. Advances in Cryptology — ASIACRYPT'05*, Chennai, India, 2005, pp.174–198.

[6] Ostrovsky R, Rackoff C, Smith A. Efficient consistency proofs for generalized queries on a committed database. In *Proc. ICALP*, Turku, Finland, 2004, pp.1041–1053.

[7] Catalano D, Dodis Y, Visconti I. Mercurial commitments: Minimal assumptions and efficient constructions. In *Proc. Theory of Cryptography — TCC'06*, *Lecture Notes in Computer Science*, Vol.3876, New York, Springer-Verlag, 2006, pp.120–144.

[8] Gennaro R, Micali S. Independent zero-knowledge sets. In *Proc. ICALP* (2), Venice, Italy, 2006, pp.34–45.

[9] Benaloh J C, de Mare M. One-way accumulators: A decentralized alternative to digital signatures. In *Proc. Advances in Cryptology — EUROCRYPT'93*, Lofthus, Norway, 1994, pp.274–285.

[10] Barić N, Pfitzmann B. Collision-free accumulators and fail-stop signature schemes without trees. In *Proc. Advances in Cryptology — EUROCRYPT'97*, Konstanz, Germany, 1997, pp.480–494.

[11] Camenisch J, Lysyanskaya A. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology — CRYPTO'02*, Santa Barbara, California, USA, 2002, pp.61–76.

[12] Goodrich M T, Tamassia R, Hasic J. An efficient dynamic and distributed cryptographic accumulator. In *Proc. the 5th International Conference on Information Security*, London, UK, 2002, pp.372–388.

[13] Li J T, Li N H, Xue R. Universal accumulators with efficient nonmembership proofs. In *Proc. Cryptography and Network Security* (*ACNS07*), *Lecture Notes in Computer Science*, Vol. 4521, Zhuhai, China, Springer-Verlag, 2007, pp.253–269.

[14] Prabhakaran M, Xue R. Statistically Hiding Sets. Submitted to *ICALP 2008*, 2007.

[15] Camenisch J, Stadler M. Efficient group signature schemes for large groups. In *Proc. Advances in Cryptology — CRYPTO'97*, Santa Barbara, California, USA, 1997, pp.410–424.

[16] Gennaro R, Halevi S, Rabin T. Secure hash-and-sign signatures without the random oracle. In *Proc. Advances in Cryptology — EUROCRYPT'99*, Prague, Czech Republic, 1999, pp.123–139.

[17] Fujisaki E, Okamoto T. Statistical zero knowledge protocols to prove modular polynomial relations. In *Proc. Advances in Cryptology — CRYPTO'97*, Santa Barbara, California, Aug. 1997, pp.16–30.

[18] Cramer R, Shoup V. Signature schemes based on the strong RSA assumption. In *Proc. the 6th ACM Conference on Computer and Communications Security* (*CCS*), Singapore, Nov. 1999, pp.46–51.

[19] Shamir A. On the generation of cryptographically strong pseudorandom sequences. *ACM Transactions on Computer Systems,* 1983 1(1): 38.

[20] Bellare M, Rogaway P. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. ACM Conference on Computer and Communications Security*, Alexandria, VA, USA, 1993, pp.62–73.

[21] Fiat A, Shamir A. How to prove yourself: Practical solutions to identification and signature problems. In *Proc. Advances in Cryptology — CRYPTO'86*, Santa Barbara, California, USA, 1987, pp.186–194.

[22] Santis A D, Micali S, Persiano G. Non-interactive zero-knowledge proof systems. In *Proc. RYPTO'87*, Pomerance C (ed.), Santa Barbara, California, United States, *Lecture Notes in Computer Science*, Springer-Verlag, 1988, Vol. 293, pp.52–72.

[23] Algesheimer J, Camenisch J, Shoup V. Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. In *Proc. Advances in Cryptology — CRYPTO'02*, Santa Barbara, California, USA, 2002, pp.417–432.

[24] Sander T. Efficient accumulators without trapdoor extended abstracts. In *Proc. ICICS*, Sydney, Australia, Varadharajan V, Mu Y (eds.), *Lecture Notes in Computer Science*, Vol. 1726, Springer, 1999, pp.252–262.

[25] Chaum C, Evertse J H, van de Graaf J, Peralta R. Demonstrating possession of a discrete logarithm without revealing

it. In *Proc. Advances in Cryptology — CRYPTO'86*, Santa Barbara, California, USA, 1987, pp.200–212.

[26] Schnorr C P. Efficient signature generation by smart cards, *Journal of Cryptology*, 1991, 4: 161–174.

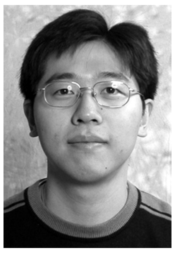
[27] Feige U, Fiat A, Shamir A. Zero knowledge proofs of identity. *Journal of Cryptology*, 1988, 1(2): 77–94.

[28] Granville A. Harold Cramér and the distribution of prime numbers. *Scandanavian Actuarial Journal*, 1995, 1: 12–28.

[29] Shoup V. Practical threshold signatures. In *Proc. Advances in Cryptology — EUROCRYPT'00*, Bruges, Belgium, 2000, pp.207–220.

Rui Xue received his Ph.D. degree in 1999 from Beijing Normal University. As a visiting scholar, he did research work in Purdue University and University of Illinois at Urban-Champaign during 2005 to 2007. He is currently an associate research professor at State Key Laboratory of Information Security, Institute of Software, and Chinese Academy of Science. His research interests include cryptographic protocols analysis, computational cryptography, and formal methods in cryptography and computer science. He is a senior member of China Computer Federation.

Ning-Hui Li received the B.Eng. degree in computer science from the University of Science and Technology of China in 1993, and the M.Sc. and Ph.D. degrees in computer science from New York University, in 1998 and 2000 respectively. He is currently an assistant professor in computer science at Purdue University. Prior to joining Purdue University in 2003, he was a research associate at Computer Science Department, Stanford University. Dr. Li's research interests are in security and privacy in information systems, with a focus on access control. He has worked on projects on trust management, automated trust negotiation, role-based access control, online privacy protection, privacy-preserving data publishing, and operating system access control. He has published more than 60 technical papers in refereed journals and conference proceedings and has served on the Program Committees of more than three dozen international conferences and workshops. Dr. Li is a senior member of the IEEE, and a member of the ACM and the USENIX Association.



Jiang-Tao Li received his B.S. degree from the University of Science and Technology of China in 1999. He obtained his M.S. and Ph.D. degrees from Purdue University in 2002 and 2006, respectively. He currently works at Intel Corporation as a security architect. His research interests are in the area of applied cryptography and privacy.