Theory and application of computationally independent one-way functions: Interactive proof of ability - Revisited

Sabyasachi Dutta* and Kouichi Sakurai

Faculty of Information Science and Electrical Engineering,
Kyushu University
saby.math@gmail.com,sakuraicsce2009g@gmail.com

Abstract. We introduce[1] the concept of computationally independent pair of one-way functions (CI-OWF). We also provide two rich classes of examples of such functions based on standard assumptions. We revisit two-party interactive protocols for proving possession of computational power and existing two-flow challenge-response protocols. We analyze existing protocols for proof of computation power and propose a new two-flow protocol using CI-OWF based on *square* Diffie-Hellman problem.

Keywords: one-way functions, zero knowledge proof, computational independence, proof of computation ability.

1 Introduction

1.1 Background

Interactive proof systems were developed by Goldwasser et al. [15] to prove membership in a language. In such a system, a *prover* and a *verifier* interacts so that at the end of the protocol the prover convinces the verifier (with overwhelming probability) that indeed the input is a member of some prefixed language. A *zero-knowledge* interactive proof between a prover and a verifier was introduced to capture the scenario that after the interaction, the verifier is convinced of the fact that the input belongs to a language but gains no more knowledge. Later, zero-knowledge *proof of knowledge* was introduced in which a polynomial-time prover not only proves membership in a zero knowledge manner but also provides evidence of the fact that he really possesses/knows a *witness* for some predicate about the input. Feige et al. [11] and Tompa et al. [28] gave formal definitions and implementations of a zero-knowledge (interactive) proof of knowledge system. However, what we are interested in this paper is not exactly the knowledge/possession of witness for proving a statement but to "prove the ability to perform a computational task". Such a scenario was first considered by Yung [30] and Koyama [20] independently and later by Bellare et al. [4].

* S. Dutta is financially supported under the NICT International Exchange Program.

1 This article was accepted & presented at the International Conference on Mathematics & Computing (ICMC 2019)

1.2 Motivation

Suppose a prover wants to prove that it has the power to "invert" a one-way function $f : D \longrightarrow R$ (may or may not be trapdoor). One intuitive example is to consider an asymmetric encryption scheme in which the prover may or may not have the secret key. However, the prover does not want to reveal the secret algorithm that he has found to break the scheme. A verifier wanting to judge the veracity of the claim may query the prover with a value $y = f(x) \in R$ with a randomly chosen $x \in D$. The prover can invert y to find a value x' such that $y = f(x')$. One important thing to note is that it may be possible that the prover has an algorithm where the functionality is hard-wired and he only gets to invert f on inputs of his choice. On the other hand, while proving to a verifier the prover must not reveal such an x' (non-triviality).

1.3 Related Work

Initial two works [30, 20] are along the same line. Both of them considered the scenario when someone possesses some extraordinary power to solve a *hard* problem and he wants to "prove this ability" to a verifier. Yung [30] considered the power to solve integer factorization and gave a "zero-knowledge proof of computational ability" protocol by using two zero-knowledge proof of knowledge protocols as subroutines. The author also gave a protocol for proving computational power for "samplable verifiable" problem by using two perfect zero-knowledge proof protocols as subroutines. The techniques are novel which help to extend "proofs of knowledge" to "proofs of proving computational power" but the complexity of the protocols are very high. Later Bellare et al. [4] formalized the idea and provided theoretical constructions.
On the other hand Koyama [20] proposed a novel idea to prove "directly" the computational power that a prover possesses. He proposed protocols with only two flows of message transfer between a prover and a verifier which rapidly decreased the cost of the protocol. On the downside, the protocols remain no longer zero-knowledge protocols as claimed by the author. However, the idea of reducing communication complexity just by using two flows of message transfer was novel. Similar idea was also considered by Stinson et al. [27] to build an efficient zero-knowledge identification protocol. Their scheme is secure in the random oracle model and is based on non-standard "knowledge of exponent" problem and standard computational Diffie-Hellman problem. Some more works [1, 29] followed the idea of Koyama [20] in different scenarios.

Our Contribution

We first revisit Koyama's protocol in Section 3 which acts as a motivation to our work. We analyze the protocols and show some flaws. We then introduce notion of *computationally-independent* one-way functions (CI-OWF) which can be useful to construct interactive protocols for proving computational power. We argue that since in a two-flow message transfer protocol achieving "zero-knowledge" is impossible [12], one has to make a trade-off between the level of security and cost of communication. With moderate and practical security, one can build proof of computational ability using CI-OWF. Moreover, all the existing works consider scenarios like "inverting a one-way function" to be a hard problem and the prover has the power to invert. But there can be hard computational problems which may not require inversion of one-way functions e.g., computational Diffie-Hellman Problem. Studying a pair of one-way functions as a building block for constructing cryptographic protocols is not new. The work of Goldwasser et al. [16] constructs a pair of claw-free one way functions to build general signature schemes. Later the concept was studied in [8, 10, 21, 26].

2 Preliminaries

We mention some basic results that will be needed in the paper.

Variants of Diffie-Hellman Problem

Let us recall the some relevant notions required for our work. Let G be a cyclic group of order n and generator g. The two well-known "hard" problems are the following:

- **The Discrete Logarithm Problem**(DLP): On input $g, g^x \in G$, compute $x \in \mathbb{Z}_n$.
- **The Computational Diffie-Hellman Problem**(CDH): On input $g, g^x, g^y \in G$ with $x, y \in \mathbb{Z}_n$, compute g^{xy}.

It is not hard to see that if an adversary is able to solve DLP then he can also solve CDH, i.e., DLP $\Longrightarrow$ CDH. The converse has been shown to hold for some cases [6, 22] but in general, it is yet to be proved that the problems are indeed equivalent. Numerous cryptographic primitives have been developed based on the DLP and CDH assumptions. Chaum et al. [7] gave a "proof-of-possession" of discrete logarithm in a zero-knowledge manner. In their work they have provided a protocol which enables a *prover* to convince a *verifier* that he really possesses discrete log without actually revealing the value to the verifier.

Maurer and Wolf [23] defined various types of CDH *oracles* and proved their equivalence. One of the interesting type is called *square* Diffie-Hellman problem (SDHP) where the corresponding oracle returns g^{x^2} when given inputs g and g^x. Bao et al. [2] proved the equivalence of CDH and SDH when the underlying group is of prime-order. Many more variations of CDH and their connections to DLP have been studied in the literature. For example, l-weak DHP [24], *square root* DHP (SRDHP) [19], equivalence between SDHP and SRDHP [25], to name a few. In the following we clearly state the problems that are needed in this work.

- **The Square DH Problem**(SDH): On input $g, g^x \in G$, compute g^{x^2}.

- **The Square root DH Problem**(SRDH): On input $g, g^x \in G$, compute y such that $g^{y^2} = g^x$.
- **The l-weak DH Problem**(l-wDH): On input $g, g^x, \dots, g^{x^l} \in G$, compute $g^{x^{l+1}}$.

3 Revisiting the protocols of Koyama [20]

Koyama [20] considered a scenario where a cryptanalyst wants to convince a potential buyer that the former has indeed invented an efficient code-breaking algorithm without actually revealing the algorithm to the buyer. To be more precise, suppose a cryptanalyst has found an efficient algorithm to break an encryption scheme (Enc, Dec) completely. So, given any ciphertext $c = E(m)$ he is able to find the underlying message m. Now the cryptanalyst acts as a *prover* and a potential buyer acts as a *verifier*. The main goal of the prover is to demonstrate the breaking of the cryptosystem without revealing any knowledge about the algorithm and the message m to the verifier. Therefore, basic requirements of such a *secure* demonstrating protocol are as follow: it must be "non-cheatable" and revealing no knowledge. Formally, the protocol must be:

1. (*Complete*) if the proof is correct then the verifier should accept prover's claim with probability 1.
2. (*Sound*) if the proof is *incorrect* then verifier should reject prover's claim with overwhelming probability.
3. (*zero-knowledge*) the proof does not yield knowledge such as value of m and the code-breaking method.

Based on the above requirements described in [20], the author built two protocols to securely and directly demonstrate the possession of code-breaking algorithm by a cryptanalyst to a potential buyer. First protocol corresponds to breaking the RSA cryptosystem and the second protocol corresponds to solving the DLP on a cyclic group.

The idea of Koyama was to decompose the underlying encryption function Enc into two one-way functions f and g such that $Enc(m) = g_o f(m)$. Writing the encryption function in this cleverly manner, a generic protocol (Algorithm 1) was given (Protocol 1 in [20]):

This above mentioned general form of protocol relies primarily on the following three requirements:
(i) f, g are both one-way functions, (ii) computing f^{-1} is no easier than computing E^{-1} and (iii) g^{-1} is computationally no easier than computing E^{-1}.

Based on the requirements and observations Koyama gave the following protocol (Algorithm 2) for the discrete logarithm problem. The *prover* wants to demonstrate the power of solving DLP to a verifier. Let G be a cyclic group of order n with g as one of the generators.

Koyama gave another protocol (Algorithm 3) to demonstrate the power of a prover that he can attack RSA cryptosystem.

Algorithm 1 General Form of Direct Protocol [20]

```
1: procedure Agreement between Prover & Verifier
2:     Prover and Verifier agree on functions $Enc, f, g$ such that $Enc(m) = g(f(m))$.
3:     Prover and Verifier share a random input $C$ in the ciphertext space.
4: procedure Computation by Prover
5:     Prover computes $m$ from $C$ using his efficient attacking method as $m = E^{-1}(C)$.
6:     Prover computes $P = f(m)$ easily and sends $P$ to the Verifier.
7: procedure Verification by the Verifier
8:     Upon receiving $P$, Verifier checks $g(P) = C$ or not.
9:     If equality holds then the Verifier accepts the claim that Prover can break the target
       public-key cryptosystem. Otherwise, the Verifier rejects.
```

Algorithm 2 Direct Protocol for DLP [20]

```
1: procedure Agreement between Prover & Verifier
2:     Prover and Verifier agree on base value $g$ and modulus $n$.
3:     Prover and Verifier share a random input $C \in G$.
4: procedure Randomness chosen by Verifier
5:     Verifier randomly generates $R$ relatively prime to $n - 1$.
6:     Computes $S$ such that $RS = 1 \ mod(n - 1)$.
7:     Computes $k = g^R$ and sends $k$ to the Prover.
8: procedure Computation by the Prover
9:     Prover uses his efficient algorithm to compute $m$ such that $g^m = C$.
10:    Computes $P = k^m$ and sends $P$ to the Verifier.
11: procedure Verification by the Verifier
12:    Verifier computes $P^S$ and checks $P^S = C$ or not.
13:    If equality holds then the Verifier accepts. Otherwise, the Verifier rejects.
```

Algorithm 3 Direct Protocol for RSA [20]

```
1: procedure Agreement between Prover & Verifier
2:     Prover and Verifier agree on public key $e = km$, $k, m \geq 3$ and RSA modulus $n$
       (whose factorization is unknown to both).
3:     Prover and Verifier share a random input $C \in \{0, 1, \dots, n - 1\}$.
4: procedure Computation by the Prover
5:     Prover uses his efficient algorithm to compute $M$ from $(c, e, n)$ such that $M^e = C$.
6:     Computes $P = M^k \ mod \ n$ and sends $P$ to the Verifier.
7: procedure Verification by the Verifier
8:     Verifier computes $P^m$ and checks $P^m = C$ or not.
9:     If equality holds then the Verifier accepts. Otherwise, the Verifier rejects.
```

4 Analysis of Koyama's Protocols

We show some flaws present in Koyama's direct protocols. Both the protocols do not satisfy conditions for zero-knowledge. We further claim that the protocol for the DLP is not sound. That is, a cheating prover is always able to make an honest verifier accept. In this regard, we again mention that solving DLP implies solving CDH but the converse is yet to be proved as true. Although there are a number of works [6, 22] that shows evidence to the fact that possibly CDH and DLP are equivalent, but it is not proved for all cyclic groups. An adversary attacking CDH problem is thus less powerful adversary.
Suppose $\mathcal{A}$ is an adversary who is able to solve CDH problem and not DLP. The adversary $\mathcal{A}$ can cheat an honest verifier into making him believe that $\mathcal{A}$ has the power of solving DLP in the following manner:

1. After receiving $k = g^R$ from the verifier (line 7 of Algorithm 2), $\mathcal{A}$ possesses $(g, C = g^m, k = g^R)$.
2. $\mathcal{A}$ computes $g^{mR} \longleftarrow (g, g^m, g^R)$ using his algorithm to solve CDHP.
3. $\mathcal{A}$ sends g^{mR} to the verifier who then computes (line 12 of Algorithm 2) $(g^{mR})^S = g^m = C$.
4. Since the equality holds, the verifier *accepts* (line 13 of Algorithm 2).

We now have the following theorem.

Theorem 1. *The protocol (Algorithm 2) for DLP does not satisfy the soundness property. Hence, the protocol is not secure.*

Remark 1. We want to make a remark at this point that although we have shown that there is a direct attack to Koyama's protocol for DLP, the protocol is also not a zero-knowledge protocol. The verifier comes to know about a S-th root of a randomly chosen element C of G.

Theorem 2. *The protocol (Algorithm 3) for RSA is not secure. More specifically, the protocol lacks zero-knowledge property.*

Proof. We simply note that the scheme is not zero-knowledge, as claimed by Koyama [20]. The verifier gets a random m th root of C, which makes the scheme insecure even against an honest verifier.

We conclude that, Koyama's "zero-knowledge" requirement is not exactly the same as standard zero-knowledge requirement. It is but a loose version of the standard one. The main requirement was not to leak the message m and to keep the design of the attacking algorithm completely hidden.

5 Computationally-independent one way functions

Building cryptographic tools on the basis of existence of two one-way functions is not new in the literature. Goldwasser et al. [16] considered *claw-free* permutations to build signature schemes resisting *chosen-message attack*. Later, Damgard [8] constructed collision-free hash functions based on the claw-free permutations and Russell [26] showed existence of collision-free hash functions is equivalent to the existence of claw-free pairs of pseudo-permutations. Krawczyk et al. [21] provided non-interactive trapdoor commitments based on claw-free permutations and lastly, Dodis et al. [10] showed that claw-free pair of permutations deliver more security than trapdoor permutations when building full-domain hash like signature schemes.

We begin with a proposal to defining computationally-independent one-way functions drawing motivation from Koyama's work and justify the reasons behind it. Our aim is also to define a pair of one-way functions based on which a moderately secure "proof of computational power" can be built leaking only minimal amount of information.

5.1 On the definition of computationally-independent one way function

Definition 1. *A pair of one-way functions f, g is called computationally-independent one way functions if they satisfy the following two conditions*

- *(CI-a) The function $H(x) = (f(x), g(x))$ is still one-way.*
- *(CI-b) Given $f(x_0)$ for randomly chosen x_0, it is hard to compute $g(x_0)$ and given $g(y_0)$ for randomly chosen y_0, it is hard to compute $f(y_0)$.*

In the above definition the first condition viz. (CI-a) ensures the fact that nothing about the input is revealed when both the functional outputs are obtained, which is a basic requirement. This condition is called the one-wayness of the pair. The second condition guarantees the computational independence of the functions i.e. it is hard to compute one function given the output of the other function. To understand the necessity of the second condition let us consider the following example.

Example 1. Let us choose the functions $f(x) = x^2 \ mod \ n$ and $g(x) = x^4 \ mod \ n$, where $n = pq$ is a product of two large safe primes so that we are in a RSA group. Now we can see that (CI-a) is automatically satisfied because computing a pre-image would be equivalent to finding integer square root modulo n which is widely believed to be a hard problem. However, computing $g(x)$ from $f(x)$ is easy and when intended to be applied as a proof of work/computing power protocol, a cheating verifier can easily fool an honest verifier.

Remark 2. We point out that the two conditions (CI-a) and (CI-b) are independent. One does not imply the other. In Example 1 we see that although (CI-a) holds, (CI-b) does not hold true. Therefore, (CI-a) does not imply (CI-b). To see that the reverse implication is also false we consider the following Example 2.

Example 2. Let us consider the functions $f(x) = x^2 \ mod \ n$ and $g(x) = x^3 \ mod \ n$ where n is a product of two large safe primes. It is easy to see that given a randomly chosen input $x_0 \in \mathbb{Z}_n^*$ it is easy to compute $f(x_0)$ and $g(x_0)$ both. Individually, they are one-way. Given the value of one function it is hard to compute the value of the other one. For example, if one can efficiently find the value of $(g(x_0))$ from x_0^2 then by division $\frac{x_0^3}{x_)^2}$ yields x_0 and thus there exists an efficient algorithm to extract integer square roots modulo n. Similarly, the hardness of computing $f(x)$ from $g(x)$ follows. But given the value $H(x_0) = (f(x_0), g(x_0))$, it is easy to compute the pre-image x_0 by simply dividing $g(x_0)$ by $f(x_0)$. Thus the function H is not one-way anymore.

We now give two examples of family of computationally-independent one-way functions based on RSA problem and variants of Diffie-Hellman problems.

Example 3. Let us consider the family of functions with RSA modulus $n = pq$, $\mathcal{F} = \{x^2 \ mod \ n, x^3 \ mod \ n, \ldots, x^s \ mod \ n, \ldots\}$. Let us choose the functions $f(x) = x^4 \ mod \ n$ and $g(x) = x^6 \ mod \ n$ from the above family $\mathcal{F}$. Now we can see that (CI-a) is satisfied because from $x^4 \ mod \, n$ and $x^6 \ mod \, n$ one can compute $x^2 \ mod \ n$, but producing a pre-image is as hard as finding a square root in the RSA group. The idea of the proof is as follows:
Suppose there is an adversary $\mathcal{A}$ who can successfully break the one-wayness of $H(x) = (x^4 \ mod \ n, x^6 \ mod \ n)$. We can now construct an adversary $\mathcal{B}$ having an oracle access to $\mathcal{A}$ to solve the problem of finding square root. $\mathcal{B}$ is given a quadratic-residue y in the RSA group and is asked to find a square root of it. $\mathcal{B}$ computes (y^2, y^3) and passes the tuple to $\mathcal{A}$. $\mathcal{B}$ outputs the number that he receives from $\mathcal{A}$. The success probability of $\mathcal{B}$ is equal to the success probability of $\mathcal{A}$.

Now it is easy to prove that computing $g(x)$ from $f(x)$ is hard. An adversary who can break the above hardness with non-negligible success probability ϵ can be used as an oracle to compute square root an element with success probability roughly $\frac{\epsilon}{4}$. Also, computing $f(x)$ from $g(x)$ is hard because if one can compute $f(x)$ from $g(x)$ then he can actually extract cube root of x^6.

We can now state the following two theorems.

Theorem 3. *Let $\mathcal{F} = \{x^k \ mod \ n\}_{k \geq 2}$ denote a family of one-way functions over $\mathbb{Z}_n^*$ with RSA modulus $n = pq$, where p, q are two large safe primes of same size. Let $f(x) = x^s \ mod \, n$ and $g(x) = x^t \ mod \, n$ be two functions from the family such that $gcd(s,t) = d > 1$ and neither s divides t nor t divides s. Then the pair $(f(x), g(x))$ gives computationally-independent one-way functions.*

Proof. Since $\gcd(s,t) = d > 1$, let us write $s = dm_1$ and $t = dm_2$ where $m_1, m_2 > 1$ are two integers.
Let us first prove that the function $H(x) = (f(x), g(x))$ is still one-way. We will show that if there is an adversary $\mathcal{A}$ who can invert $H(x)$ with a non-negligible probability then there is an adversary $\mathcal{B}$ with oracle access to $\mathcal{A}$ can solve d-th root problem in the RSA group. Suppose a challenger $\mathcal{C}$ chooses a random element r from $\mathbb{Z}_n^*$, computes $r^d = y$ and sends y to $\mathcal{B}$ to find a d-th root of y. $\mathcal{B}$ computes (y^{m_1}, y^{m_2}) and queries the pair to $\mathcal{A}$. If $\mathcal{A}$ outputs r' then $\mathcal{B}$ also

outputs the same number. It can be easily seen that the success probability of $\mathcal{B}$ of finding a d-th root is also non-negligible.

To prove the second condition, i.e. the computational independence we observe that computing $g(x) = x^t$ from $f(x) = x^s$ is equivalent to extracting a m_1-th root of x^s and conversely, computing $f(x) = x^s$ from $g(x) = x^t$ is equivalent to extracting a m_2-th root of x^t.

This completes the proof.

Theorem 4. *Let $\mathcal{F} = \{x^k \ mod \ n\}_{k\geq 2}$ denote a family of one-way functions over $\mathbb{Z}_n^*$ with RSA modulus $n = pq$, where p, q are two large safe primes of same size. Any two functions x^a mod n and x^b mod n from the family such that a, b are relatively prime fails to be a pair of computationally-independent one-way functions.*

Proof. As a, b are relatively prime therefore one can efficiently find, using Euclid's algorithm, two integers u, v such that $ua + vb = 1$. Thus, finding x from $x^a \ mod \ n$ and $x^b \ mod \ n$ becomes easy using the identity $(x^a)^u + (x^b)^v = x$. Hence (CI-a) does not hold.

Another result that is based on the computational square Diffie-Hellman Problem is stated below.

Theorem 5. *Let G be a cyclic group of order p and generator g. Let f, h be two functions defined by $f(x) = g^x$ and $h(x) = g^{x^2}$ for all $x \in \mathbb{Z}_p^*$. Then the pair $(f(x), h(x)$ gives a computationally-independent pair of one-way functions.*

Proof. The one-wayness of $H(x) = (f(x), h(x))$ follows from hardness of solving the l-weak DHP (*see* Section 2). Since g is a generator of the cyclic group G, the function g^x where $x \in \{0, 1, \dots, p-1\}$ is an injective function. Thus there is only one x which satisfies the tuple $(g, g^x, g^{x^2}, g^{x^3})$. Therefore an adversary who can find a pre-image of $H(x)$ will be able to solve 2-weak DHP.
As for (CI-b), it is not very hard to see that computing $h(x)$ from $f(x)$ is the SDH problem and the converse is at least as hard as SRDH (*see* Section 2).

5.2 On Leakage of information from CI-OWF

So far we have discussed the definition of computationally-independent one-way functions and gave two rich classes of examples. In this section, we discuss on the amount of information leaked (about the input) from the pair $(f(x), g(x))$, where $f(x)$ and $g(x)$ are CI-OWF. The condition (CI-b) in Definition 1 requires that computing the value of $f(x)$ (resp. $(g(x))$) from $g(x)$ (resp. $(f(x))$ is hard. Therefore, when both the values are available, it is rather expected to have some non-trivial knowledge about x. From the output values one can compute every efficiently computable function with input $(f(x), g(x))$. We ask the most important question: how much leakage about the input x is observed from the values $f(x)$ and $g(x)$?

From the condition (CI-a) of Definition 1 the leakage cannot be x itself or something from which it is easy to derive x. Thus the leakage can be thought as the output of some one-way leakage function L when evaluated at x, i.e., $L(x)$.

Theorem 6. *The leakage obtained from $f(x) = x^s \bmod n$ and $g(x) = x^t \bmod n$ satisfying all the conditions of Theorem 3 can be written as $L(x) = x^d \bmod n$ where d is the greatest common divisor of s and t.*

Proof. The proof follows from the fact that d can be written as $d = us + vt$, a linear combination of s, t using Euclid algorithm where u, v are integers.

Theorem 7. *The leakage obtained from $f(x) = g^x$ and $h(x) = g^{x^2}$ satisfying all conditions of Theorem 5 can be written as $L(x) = g^{Q(x)}$, where $Q(x)$ is a quadratic polynomial in x.*

Proof. Observing g, g^x, g^{x^2} one can choose $a, b, c \in \{0, 1, \ldots, p-1\}$ and can compute $g^c.(g^x)^b.(g^{x^2})^a = g^{ax^2+bx+c}$.

6 Protocol for proving ability

In this section we discuss two-flow protocols (one challenge from the verifier and one response from the prover) for proving computational ability with the help of CI-OWF (f, g). One fully secure way to implement a protocol is to follow the footsteps of Yung [30]. However, we are interested in two-flow protocols.
The broad idea for constructing such protocols is as follows: the *verifier* sends a challenge in the form of $f(r)$ to the *prover* and prover responds back with $g(r)$. This way the prover establishes the fact that he has computational power to solve the "hard" problem of computing $g(r)$ from $f(r)$, where r is chosen randomly by the verifier. We note that it includes the scenario of "the power of inverting f" (which was the original motivation for defining proof of ability [20, 30]) as well as "the power to extract $g(r)$ from $f(r)$ without explicitly computing r" capturing a notion of malleability.

We now describe modifications of Koyama's protocols to convert them into two-flow protocols assuming an *honest* verifier. We modify our requirements for the protocols as follows:

1. (*Completeness*) if the proof is correct then the verifier should accept prover's claim with probability 1.
2. (*Soundness*) if the proof is *incorrect* then verifier should reject prover's claim with overwhelming probability.
3. (*privacy*) the proof does not reveal the input r (to an eavesdropper or a dishonest verifier) and the "powerful" algorithm remains completely hidden.

We have already shown that the protocol of Koyama for DLP (Algorithm 2) is not sound. We propose the following protocol based on the hardness of Square DH (SDH) problem. Note that in a prime order group CDH problem and SDH problem are equivalent [2]. We recall that SDH assumption says that given g and g^x it is hard to compute g^{x^2}, where x is randomly chosen element from $\mathbb{Z}_p^*$. Suppose a prover wants to prove "securely" his ability to solve the SDH problem. The following simple protocol based on the CI-OWF (g^x, g^{x^2}) provides a solution to this problem in presence of an honest verifier.

Algorithm 4 Direct Protocol for SDH problem

```
 1: procedure Agreement between Prover & Verifier
 2: Prover and Verifier agree on base value $g$ and modulus $n$.
 3: procedure Challenge by Verifier
 4: Verifier randomly generates $r$ and computes $g^r = C$.
 5: Sends $C$ to the Prover.
 6: procedure Computation by the Prover
 7: Prover uses his efficient algorithm to compute $g^{r^2} = R$ from $C$.
 8: Sends $R$ to the Verifier.
 9: procedure Verification by the Verifier
10: Verifier computes $C^r$ and checks $C^r = R$ or not.
11: If equality holds then the Verifier accepts. Otherwise, the Verifier rejects.
```

Conclusion

We revisited the interactive protocols to prove one's computational power to a verifier in a secure manner. We considered two-flow protocols with moderate/practical security. We introduced the notion of computationally-independent one-way functions and two rich classes of examples based on standard assumptions. We analyzed the two-flow protocols of Koyama [20] and showed some flaws inherent in the protocols. Last we described a concrete two-flow protocol on the basis of newly introduced computationally-independent one-way functions. Finding further examples of such family of functions remain an interesting open issue.

References

1. Bao, F., Lee,C.-C. and Hwang, M.-S, "Cryptanalysis and improvement on batch verifying multiple RSA digital signatures", Applied Mathematics and Computation 172(2): 1195-1200 (2006).
2. Bao, F., Deng, R. H. and Zhu, H., "Variations of Diffie-Hellman Problem", ICICS'03: 301-312 (2003).
3. Bellare, M. and Goldreich, O., "On Defining Proofs of Knowledge", CRYPTO'92: 390-420 (1992).
4. Bellare, M. and Goldreich, O., "Proving Computational Ability", Studies in Complexity and Cryptography 2011: 6-12 (2011).
5. Blum, M. and Kannan, S., "Designing Programs That Check Their Work", STOC'89: 86-97 (1989).
6. den Boer, B., "Diffie-Hillman is as Strong as Discrete Log for Certain Primes", CRYPTO'88: 530-539 (1988).
7. Chaum, D., Evertse, J.-H., van de Graaf, J. and Peralta, R., "Demonstrating Possession of a Discrete Logarithm Without Revealing It", CRYPTO'86: 200-212 (1986).
8. Damgard, I., "Collision Free Hash Functions and Public Key Signature Schemes", EUROCRYPT'87: 203-216 (1987).
9. Diffie, W. and Hellman, M. E., "New directions in cryptography", IEEE Trans. Inform. Theory 22(6), 644-654 (1976).

10. Dodis, Y. and Reyzin, L., “On the Power of Claw-Free Permutations”, SCN’02: 55-73 (2002).
11. Feige, U., Fiat, A. and Shamir, A., “Zero-Knowledge Proofs of Identity”, J. Cryptology 1(2): 77-94 (1988).
12. Goldreich, O. and Kahan, A., “How to Construct Constant-Round Zero-Knowledge Proof Systems for NP”, J. Cryptology 9(3): 167-190 (1996).
13. Goldreich, O. and Krawczyk, H., “On the Composition of Zero-Knowledge Proof Systems”, SIAM J. Comput. 25(1): 169-192 (1996).
14. Goldreich, O., Micali, S. and Wigderson, A., “Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design (Extended Abstract)”, FOCS’86: 174-187 (1986).
15. Goldwasser, S., Micali, S. and Rackoff, C., “The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract)”, STOC’85: 291-304 (1985).
16. Goldwasser, S., Micali, S. and Rivest, R. L., “A “Paradoxical” Solution to the Signature Problem (Extended Abstract), FOCS’84: 441-448 (1984).
17. Hastad, J., “Solving Simultaneous Modular Equations of Low Degree”, SIAM J. Comput. 17(2): 336-341 (1988).
18. Joux, A., Naccache, D. and Thom, E., “When e-th Roots Become Easier Than Factoring”, ASIACRYPT’07: 13-28 (2007).
19. Konoma, C., Mambo, M. and Shizuya H., “The computational difficulty of solving cryptographic primitive problems related to the discrete logarithm problem”, IEICE Trans. Fundam. Electron. Commun. Comput. Sci. E88-A-1, 81-88 (2005).
20. Koyama, K., “Direct Demonstration of the Power to Break Public-Key Cryptosystems”, AUSCRYPT 1990: 14-21, (https://link.springer.com/chapter/10.1007/BFb0030346) (1990).
21. Krawczyk, H. and Rabin, T., “Chameleon Signatures”, NDSS’00: 143-154 (2000).
22. Maurer, U. M., “Towards the Equivalence of Breaking the Diffie-Hellman Protocol and Computing Discrete Algorithms”, CRYPTO’94: 271-281 (1994).
23. Maurer, U. and Wolf, S., “DiffieHellman oracles”, CRYPTO’96, LNCS 1109, 268-282 (1996).
24. Mitsunari, S., Sakai, R. and Kasahara, M., “A new traitor tracing”, IEICE Trans. Fundam. Electron. Commun. Comput. Sci, E85-A-2, 481-484 (2002).
25. Roh, D. and Hahn, S. G., “The square root Diffie-Hellman problem”, Des. Codes Cryptography 62(2): 179-187 (2012).
26. Russell, A., “Necessary and Sufficient Conditions For Collision-Free Hashing”, CRYPTO’92: 433-441 (1992).
27. Stinson, D. R. and Wu, J., “An efficient and secure two-flow zero-knowledge identification protocol”, J. Mathematical Cryptology 1(3): 201-220 (2007).
28. Tompa, M. and Woll, H., “Random Self-Reducibility and Zero Knowledge Interactive Proofs of Possession of Information”, FOCS’87: 472-482 (1987).
29. Verheul, E. R. and Van Tilborg, H. C. A, “Cryptanalysis of ’Less Short’ RSA Secret Exponents” Appl. Algebra Eng. Commun. Comput. 8(5): 425-435 (1997).
30. Yung, M., “Zero-Knowledge Proofs of Computational Power” (Extended Summary), EUROCRYPT’89: 196-207 (1989).