Proofs of Knowledge with Several Challenge Values

Grzegorz Stachowiak

Institute of Computer Science, University of Wrocław, Joliot-Curie 15, 50-383 Wrocław, Poland (gst@ii.uni.wroc.pl)

Abstract. In this paper we consider the problem of increasing the number of possible challenge values from 2 to s in various zero-knowledge cut and choose protocols. First we discuss doing this for graph isomorphism protocol. Then we show how increasing this number improves efficiency of protocols for double discrete logarithm and e-th root of discrete logarithm. Double discrete logarithm protocol is potentially a very useful tool for constructing complex cryptographic protocols. The improvement given by our paper is 2-4 times in terms of both time complexity and transcript size.

1 Introduction

Zero-knowledge proof protocols were introduced by Goldwasser, Micali and Rackoff in [8]. These proof were intended to prove validity of statements interactively without revealing any other information. Almost at the same time the concept of *proof of knowledge* [7] introduced the notion of *knowledge extractor*. Property of zero-knowledge is useful when one wants to perform operations on secret values, without revealing them. Classical examples are authentication, identification and digital signatures.

In our proofs of knowledge we have two persons: Prover and Verifier or Peggy and Vic denoted $\mathcal{P}$ and $\mathcal{V}$ respectively. $\mathcal{P}$ proves $\mathcal{V}$ her knowledge of some secret x for which a Boolean formula $\varphi(x)$ is satisfied. We assume, that $\mathcal{P}$ and $\mathcal{V}$ are polynomial time. We say that the proof is *zero-knowledge* if it discloses nothing about the secret to $\mathcal{V}$. If this requires choosing $\mathcal{V}$'s challenges at random this proof is *honest verifier zero-knowledge (HVZK)*. Such proof of knowledge is denoted

$$PK[x : \varphi(x)].$$

Basic properties of zero-knowledge protocol are

completeness A person knowing $x : \varphi(x) = \text{true}$ can always pass the protocol.

soundness A person who can pass the protocol with non-negligible probability must know $x : \varphi(x) = \text{true}$.

zero-knowledge Nothing about the secret x can be concluded from the data exchanged during the protocol.

In zero-knowledge protocols $\mathcal{P}$ proves her knowledge of x responding correctly to some challenge c received from $\mathcal{V}$. Majority of zero-knowledge protocols are *cut and choose* ones. In these protocols there are two possible challenges c – typically 0 and 1. A person not knowing the secret x can respond to at most one of them, so her probability of success is $1/2$. To make impossible for her to pass the protocol, it is necessary to repeat it multiple times. If we repeat this protocol κ times to reduce the chance of passing the protocol without the knowledge of x to $1/2^\kappa$. A very classical example of such a protocol is one for graph isomorphism [8].

Thus we can measure the computational complexity of a zero-knowledge protocol as a function of two parameters: λ and κ. The first of them λ is the size of the problem i.e. the size of the graphs for graph isomorphism protocol or the length of the numbers for number theoretic protocols. The second κ, is the security parameter denoting cheating probability at most $1/2^\kappa$.

There are protocols, that admit multiple values of challenge – more than $1/2^\kappa$ for any reasonable κ. $\mathcal{E}$ can respond to at most one of them which gives her chance of success smaller than $1/2^\kappa$ in a single iteration. Very well known example of such a protocol is Okamoto protocol [12]. Another example is HVZK Schnorr protocol [14].

Zero-knowledge proofs behaving like Okamoto protocol are much more desired than cut and choose ones. But such efficient protocols were found only for a limited class of problems. Nevertheless a general rule can be formulated as follows: the more possible values of challenge, the more efficient the protocol is. Our aim in this paper is to increase the efficiency of some existing zero-knowledge protocols by increasing the number of possible challenges.

In section 2 we discuss increasing this number for graph isomorphism protocol. But we are mainly interested in improving two other protocols. They are Stadler [16] protocols for double discrete logarithm, and for e-th root of discrete logarithm. Numerous more complicated cryptographic protocols are based on these two zero-knowledge proofs. These zero-knowledge proofs are cut and choose ones, so they are quite inefficient. Our improvements increase efficiency of our protocols $O(\log \kappa)$ times. In terms of practical improvement for $\kappa = 160$ it is 2-4 times.

The double discrete logarithm protocol and e-th root of discrete logarithm protocol were originally constructed for publicly verifiable secret sharing (PVSS) [16, 15]. They are also used in group signatures [3, 1], divisible e-cash [11, 4, 13] and verifiable escrow [10].

2 Graph isomorphism.

In this section we increase the number of challenge values in zero-knowledge proof of knowledge of graph isomorphism. After increasing the number of challenges this protocol is no longer a proof of knowledge, but we prove that it can be still applied for purposes like identification or digital signatures. We prove that this protocol has the property of *weak soundness*, which states that nobody who knows only $\varphi(\cdot)$, can learn how to impersonate the prover with nonnegligible probability. In this section we take a natural assumption of graph isomorphism problem intractability.

Graph isomorphism example is intended to illustrate what advantages increasing the number of possible challenge values gives. This example also shows that there are protocols, other than described in the next sections, for which the number of challenges can be increased from 2 to s. It would be interesting to find methods to increase the number of challenges for other proofs of knowledge. The protocol for graph isomorphism was constructed in [8]. In this protocol $\mathcal{P}$ proves that she knows the isomorphism between G_1 and G_2. This protocol is presented in Fig. 1.

1. $\mathcal{P}$ sends $\mathcal{V}$ random G isomorphic to G_1
2. $\mathcal{V}$ responds with challenge $c \in \{1, 2\}$
3. $\mathcal{P}$ sends the isomorphism $\varphi_c : G \to G_c$

Fig. 1. Classical protocol for graph isomorphism

In modified protocol for graph isomorphism shown in Fig. 2 $\mathcal{P}$ knows isomorphisms between $G_1, G_2, \ldots, G_s$, that are images of some graph G_0 by isomorphisms.

1. $\mathcal{P}$ sends $\mathcal{V}$ random G isomorphic to G_1
2. $\mathcal{V}$ responds with challenge $c \in \{1, 2, \ldots, s\}$
3. $\mathcal{P}$ sends the isomorphism $\varphi_c : G \to G_c$.

Fig. 2. Modified protocol for graph isomorphism.

Now we discuss the properties of modified protocol.

Completeness. Follows from the formulation of the protocol.

Zero-Knowledge. The simulator for this protocol is almost identical as one for the classical graph isomorphism protocol. Note that even if $\mathcal{V}$ is dishonest we can remove from the simulator transcript the rounds in which c does not agree with $\mathcal{V}$'s choice.

Weak Soundness. We claim that an opponent not knowing any of the isomorphisms can pass the protocol for at most one challenge c unless the graph isomorphism problem (i.e. the version for a pair of graphs) is easy. Assume on the contrary that in a round she can respond to two different challenges with a non-negligible probability p. We show how she can solve graph isomorphism problem with probability $p/2$. Let us have two isomorphic graphs H_1 and H_2 for which we want to find an isomorphism. We can generate $G_1, G_2, \ldots, G_s$ as the images of H_1 and H_2 by random isomorphisms. We do it so that $s/2$ of them are images of H_1, the other $s/2$ are images of H_2, and they are shuffled randomly. With probability p the opponent can respond successfully to some challenges c and d. In such case she is able to find an isomorphism between G_c and G_d. The probability that G_c and G_d are images of different H_i's is at least $1/2$. Thus with probability at least $p/2$ using opponent's method to pass the protocol, one can find an isomorphism between H_1 and H_2.

Complexity. The probability of $\mathcal{E}$'s success in a single iteration is at most $1/s$. The probability of her success in k iterations is at most $1/s^k$. For example in the classical protocol to achieve probability of $\mathcal{E}$'s success $1/2^{100}$ we need 100 iterations. When we apply our protocol for $s = 10$ we get this probability after only 30 iterations.

Note that modified protocol is not efficient zero-knowledge proof that $G_1, G_2, \ldots, G_s$ are isomorphic. If $s = 10$, then the probability that fixed c is not a challenge during $k = 30$ iterations at least once is close to $1/e^3$. So there is a big chance that G_c is not isomorphic to other graphs and it is not verified.

3 Preliminaries for number-theoretic protocols

In the rest of the paper we present protocols, that are related to discrete logarithm problem. They are protocols for double discrete logarithm and e-th root of discrete logarithm. First we remind some results of previous authors. From now on we consider elements of $\mathbb{Z}_p^*$ where p is a large prime. We take the standard DL (discrete logarithm) assumption, that computing $\log_g h$ is infeasible. We begin with Lemma from [5]. This lemma is used in the analysis of our protocols, and is the reason why in our protocols we implicitly require, that whenever g, h are different elements of $\mathbb{Z}_p^*$, $\mathcal{P}$ has no control over them and in particular does not know $\log_g h$.

Lemma 1. *Let $g_1, g_2, \ldots, g_k$ be elements of $\mathbb{Z}_p^*$ of order q. Under the DL assumption it holds that no probabilistic polynomial-time algorithm can output with non-negligible probability two different tuples $(u_1, u_2, \ldots, u_k)$ and $(v_1, v_2, \ldots, v_k)$ such that*

$$g_1^{u_1} g_2^{u_2} \cdots g_k^{u_k} = g_1^{v_1} g_2^{v_2} \cdots g_k^{v_k}.$$

In the paper we also apply well-known protocol for proving equality of powers [6]. Let $g, G, h \in \mathbb{Z}_p^*$. This protocol is presented in Fig. 3 and can be described as

$$PK\left[(x, X_1, X_2) : Z_1 = g^x h^{X_1}, Z_2 = G^x h^{X_2}\right].$$

This protocol admits multiple challenges and one its iteration is enough to be sure, that the prover indeed has the secret key for which the statement is true. This protocol can be used to prove nonlinear relations. Assume, that $\mathcal{P}$ knows

1. $\mathcal{P}$ chooses at random r, R_1, R_2 and sends $t_1 = g^r h^{R_1}, t_2 = G^r h^{R_2}$,
2. $\mathcal{V}$ responds with $c \in \mathbb{Z}_q$,
3. $\mathcal{P}$ sends $y = r + xc, Y_1 = R_1 + X_1 c, Y_2 = R_2 + X_2 c$,
4. $\mathcal{V}$ checks if $g^y h^{Y_1} = Z_1^c t_1, G^y h^{Y_2} = Z_2^c t_2$.

Fig. 3. Protocol for equality of powers.

secret x_1, x_2, X_1, X_2 such that $z_1 = g^{x_1} h^{X_1}$ and $z_2 = g^{x_2} h^{X_2}$. In such a case she can prove that $z_3 = g^{x_1 x_2} h^{X_3}$ for some X_3 performing the protocol

$$PK\left[(x_2, X_2, X_4) : z_2 = g^{x_2} h^{X_2}, z_3 = Z_1^{x_2} h^{X_4}\right].$$

Using this protocol it is also easy to construct a protocol (see [3] for details)

$$PK\left[(x, X_1, X_2, \ldots, X_S) : z_1 = g^x h^{X_1}, z_2 = g^{x^2} h^{X_2}, \ldots, z_s = g^{x^s} h^{X_s}\right].$$

Now we remind security assumptions related to our protocols, that were introduced in [9].

s**-Power Decisional Diffie-Hellman** (s-PDDH) assumption. No polynomial-time algorithm can distinguish between the following two distributions with non-negligible advantage over a random guess:

- Distribution: $\left(g^x, g^{x^2}, g^{x^3}, \ldots, g^{x^s}\right)$ where g is known and x is chosen at random and
- Distribution: $(g_1, g_2, \ldots, g_s)$ where $g_1, g_2, \ldots, g_s$ are chosen at random.

s**-Power Computational Diffie-Hellman** (s-PCDH) assumption. No probabilistic polynomial-time algorithm can compute g^{x^s} given $g^x, g^{x^2}, \ldots, g^{x^{s-1}}$ with non-negligible probability.

In this paper we can use a weaker assumption.

s**-Power Discrete Logarithm** (s-PDL) assumption. No probabilistic polynomial-time algorithm can compute x given $g^x, g^{x^2}, \ldots, g^{x^s}$ with non-negligible probability.

Note that some nice properties of DL assumption are inherited by s-PDL assumption. One of them is first/last bit security of x, that can be proved in the same way as for DL assumption.

In the paper we assume s-PDL for x being in two special forms: $x = h^y$ and $x = y^e$. In many natural cases s-PDL for x in such a form can be reduced to s-PDL for general x.

If $x = h^y$ we can consider the case where g has order $q = 2Q + 1$, h has order Q and q, Q are primes. In this case the probability, that a random x has form h^y is $1/2$. And a polynomial algorithm solving s-PDL problem for $x = h^y$ with nonnegligible probability p solves this problem for random x with probability $p/2$.

If $x = y^e$ we can assume, that $g \in \mathbb{Z}_p$ and $e \perp p-1$. In this case we can compute $e^{-1} \bmod (p-1)$ using Euclidean algorithm. Thus for any x we have $x = \left(x^{e^{-1}}\right)^e$. So solving s-PDL problem for $x = y^e$ is equivalent to solving this problem for general x.

4 Protocols for double discrete logarithm

Now we formulate a new proof of knowledge of double discrete logarithm. Actually we formulate its very simple version, whose security is based on s-PDL assumption. Zero-knowledge version, that is based only on DL assumption is presented later on. Let $g \in \mathbb{Z}_p^*$ (of order q) and $h \in \mathbb{Z}_q$ of order n. We require, that n does not have small divisors. The protocol can be denoted as

$$PK\left[x : z = g^{h^x}\right].$$

It should be mentioned, that in this protocol no information about h^x should be disclosed. This zero-knowledge proof has numerous applications in more complex cryptographic protocols. Classical Stadler [16] protocol for this problem is presented in Fig. 4.

1. $\mathcal{P}$ chooses r at random and sends $t = g^{h^r}$,
2. $\mathcal{V}$ responds with $c \in \{0, 1\}$,
3. If $c = 0$ then $\mathcal{P}$ sends $y = r$ else she sends $y = r - x$,
4. If $c = 0$ $\mathcal{V}$ checks whether $t = g^{h^y}$ else he checks whether $t = z^{h^y}$.

Fig. 4. Stadler protocol for double discrete logarithm.

Stadler protocol admits two challenge values: 0 and 1. Our protocol presented in Fig. 5 admits $s+1$ challenge values, so it requires fewer iterations.

Now we analyze our protocol.

Completeness. Follows from the formulation of the protocol.

Soundness. Nobody who does not know β can pass phase 1. Suppose that in some iteration of phase 2 we can respond to two challenges c and d. We show, that we can compute x such that $z = g^{h^x}$. Indeed we have

$$t = g^{\beta^c h^{y_c}} = g^{\beta^d h^{y_d}},$$

thus

$$\beta^c h^{y_c} = \beta^d h^{y_d},$$

and

$$\beta = h^{(y_c - y_d)/(d-c)}.$$

Phase 1

1. $\mathcal{P}$ sends $z_0 = g = g^{h^0}, z = z_1 = g^{h^x}, z_2 = g^{h^{2x}}, z_3 = g^{h^{3x}}, \ldots, z_s = g^{h^{sx}}$,
2. $\mathcal{P}$ uses the zero-knowledge protocol for nonlinear relations to prove that she knows β, such that $z_1 = g^{\beta}, z_2 = g^{\beta^2}, \ldots, z_s = g^{\beta^s}$.

Phase 2 (repeat k times)

1. $\mathcal{P}$ chooses r at random and sends $t = g^{h^r}$,
2. $\mathcal{V}$ responds with $c \in \{0, 1, 2, \ldots, s\}$,
3. $\mathcal{P}$ sends $y = r - cx$,
4. $\mathcal{V}$ checks if $t = z_c^{h^y}$.

Fig. 5. Our protocol for double discrete logarithm.

So we can take $x = (y_c - y_d)/(d - c)$.

Security. We prove that under s-PDL assumption it is intractable for the opponent to compute h^x. Assume that an opponent can compute $X = h^x$ from the communication in the protocol in polynomial time with nonnegligible probability. She can break s-PDL assumption for $X = h^x$ putting s-PDL input as $z_1, z_2, z_3, \ldots, z_s$. She then uses standard simulator of phase 1 and a simulator of phase 2 which is almost the same as for Stadler protocol to produce the amount of protocol transcripts needed by the opponent to find X.

5 Protocols for e-th root of discrete logarithm

The proof of knowledge e-th root of discrete logarithm can be specified as

$$PK\left[x : z = g^{x^e}\right].$$

We assume that $g \in \mathbb{Z}_p$ of order n. We present an improved protocol for proving the knowledge of the e-th root of discrete logarithm where e does not have small divisors $(2, 3, \ldots, s)$. We also assume that $e \perp \phi(n)$, so all elements of $\mathbb{Z}_n$ can be expressed as x^e.

We formulate very simple version of this protocol, whose security is based on s-PDL assumption. Zero-knowledge version can be formulated in the same way as double discrete logarithm protocol in the next section.

Our protocol is based on cut and choose Stadler [16] protocol that admits two values of challenge: 0 and 1. So it requires κ iterations for security parameter κ. Note, that his probability does not depend on e. There are also protocols for this problem described in [3, 2] that outperform Stadler protocol for small e. The protocol from [2] is better than Stadler protocol for $\log e \leq 160$ if $\kappa = 160$. Stadler protocol is shown in Fig. 6.

Our protocol presented in Fig. 7 admits $s+1$ challenge values. Since it requires fewer iterations, the protocol from [2] remains competitive only for $\log e \leq 60$ when $\kappa = 160$.

1. $\mathcal{P}$ chooses r at random and sends $t = g^{r^e}$,
2. $\mathcal{V}$ responds with $c \in \{0, 1\}$,
3. If $c = 0$ then $\mathcal{P}$ sends $y = r$ else she sends $y = r/x$,
4. If $c = 0$ $\mathcal{V}$ checks whether $t = g^{y^e}$ else he checks whether $t = z^{y^e}$.

Fig. 6. Stadler protocol for e-th root of discrete logarithm

Phase 1

1. $\mathcal{P}$ sends $z_0 = g, z = z_1 = g^{x^e}, z_2 = g^{x^{2e}}, z_3 = g^{x^{3e}}, \ldots, z_s = g^{x^{se}}$,
2. $\mathcal{P}$ uses the zero-knowledge protocol for nonlinear relations to prove that she knows β, such that $z_1 = g^{\beta}, z_2 = g^{\beta^2}, \ldots, z_s = g^{\beta^s}$.

Phase 2 (repeat k times)

1. $\mathcal{P}$ chooses r at random and sends $t = g^{r^e}$,
2. $\mathcal{V}$ responds with $c \in \{0, 1, 2, \ldots, s\}$,
3. $\mathcal{P}$ sends $y = r/x^c$,
4. $\mathcal{V}$ checks if $t = z_c^{y^e}$.

Fig. 7. Our protocol for e-th root of discrete logarithm.

The analysis of this protocol is very similar to one of the protocol for double discrete logarithm.

Completeness. Follows from the formulation of the protocol.

Soundness. Nobody who does not know β can pass phase 1. Suppose, that in some iteration of phase 2 we can respond to two challenges c and d. We show, that we can compute x such that $z = g^{x^e}$. Indeed we have

$$t = g^{\beta^c y_c^e} = g^{\beta^d y_d^e},$$

so

$$\beta^c y_c^e = \beta^d y_d^e,$$

and

$$\beta^{d-c} = (y_c/y_d)^e .$$

Since $d - c \perp e$ Euclidean Algorithm can find $a, b : a(d - c) + be = 1$. Thus we can compute

$$x = (y_c/y_d)^a \beta^b.$$

It is easy to check, that $x^e = \beta$.

Security. We prove that under s-PDL assumption it is intractable for the opponent to compute x^e. Assume that an opponent can compute $X = x^e$ from the communication in the protocol in polynomial time with nonnegligible probability. She can break general s-PDL assumption (see the assumption about e) putting s-PDL input as $z_1, z_2, z_3, \ldots, z_s$. She then uses standard simulator of phase 1 and a simulator of phase 2 which is almost the same as for Stadler protocol to produce the amount of protocol transcripts needed to find X.

6 Randomized versions of protocols and signatures

In this section we present more secure version of the protocol for double discrete logarithm. This version does not rely on s-PDL assumption and does not require honest verifier. A very similar protocol for e-th root of discrete logarithm can also be formulated. We skip this second protocol in this paper due to its similarity to the protocol described in this section. These secure versions are almost as efficient as the simple versions from the previous sections.

Secure version of protocol for double discrete logarithm is presented in Fig. 8. In this protocol we have (unrelated) elements $g, G, g_1, g_2, \ldots, g_k$ in $\mathbb{Z}_p^*$ of order q.

This protocol is written to produce a signature of knowledge according to Fiat-Shamir heuristic [7] and uses secure hash functions $\mathcal{H}_l$ producing l-bit outputs. This protocol can be described as

$$PK\left[(x, x') : z = g^{h^x} G^{x'}\right]$$

Now we analyze protocol from Fig. 8.

Completeness. Follows from the formulation of the protocol.

Soundness. The soundness property is computational If $\mathcal{P}$ does not know $\beta, x', X_0, X_1, \ldots, X_s$ such that

$$z = g^{\beta} G^{x'},\ t'_s = G^{\sum \beta^{s-j} X_j}$$

and for $j \in \{0, 1, 2, \ldots, s\}$

$$t_j = \left(t'_{j-1}\right)^{\beta} G^{X_j},$$

then she is not able to respond to challenge C. Responding to challenge C means sending $Y_1, \ldots, Y_s, Y, y, Y'$. Having above equalities we can compute, that

$$t_0 = G^{X_0} \prod_{i=1}^{k} g_i^{-h^{y_i} \beta^{-c_i}}$$

Assume $\mathcal{P}$ can respond to another challenge c'. In such a case

$$t_0 = G^{X'_0} \prod_{i=1}^{k} g_i^{-h^{y'_i} \beta^{-c'_i}}$$

For some i we have $c_i \neq c'_i$. Thus under DL assumption from Lemma 1

$$h^{y'_i} \beta^{-c'_i} = h^{y_i} \beta^{-c_i}.$$

1. Let $k = \lceil \kappa / \log_2(s+1) \rceil$. For random $r_1, \ldots, r_k, X_0$ compute

$$t_0 = G^{X_0} \left(\prod_{i=1}^{k} g_i^{h^{r_i}} \right)^{-1} .$$

2. Let $c = \mathcal{H}_\kappa (m \| t_0)$. There are unique $c_i \in \{0, \ldots, s\}$ such that

$$c = c_1 + c_2(s+1) + c_3(s+1)^2 + \cdots + c_k(s+1)^{k-1}.$$

3. For all $i \in \{1, 2, \ldots, k\}$ compute $y_i = r_i + c_i \cdot x$.
4. Choose $X_1, \ldots, X_s$ at random and compute $t'_0, \ldots, t'_s$ and $t_1, \ldots, t_s$:

$$t'_j = t_j \prod_{i:c_i=j} g_i^{h^{y_i}} = t_j \left(\prod_{i:c_i=j} g_i^{h^{r_i}} \right)^{h^{jx}} , \; t_j = \left(t'_{j-1}\right)^{h^x} G^{X_j}.$$

5. Choose R at random. For $j \in \{1, 2, \ldots, s\}$ choose R_j at random and compute

$$T_j = \left(t'_{j-1}\right)^{R} G^{R_j}.$$

   Choose r, R' at random and compute $T = g^R G^r, T' = G^{R'}$.
6. Let $C = \mathcal{H}_\lambda (m \| t_0 \| t_1 \| \cdots \| t_s \| T_1 \| \cdots \| T_s \| T \| T')$.
7. Compute $Y = R + Ch^x, y = r + Cx', Y' = R' + C \sum h^{(s-j)x} X_j$ and values $Y_j = R_j + C \cdot X_j$ for $j \in \{1, 2, \ldots, s\}$.
8. The signature on m has the form

$$t_0 \| t_1 \| \cdots \| t_s \| T_1 \| \cdots \| T_s \| T \| T' \| y_1 \| \cdots \| y_k \| Y_1 \| \cdots \| Y_s \| Y \| y \| Y'.$$

Fig. 8. Secure version of our protocol for double discrete logarithm formulated as a signature scheme for message m.

1. Let $c = \mathcal{H}_\kappa (m \| t_0)$. There are unique $c_i \in \{0, \ldots, s\}$ such that

$$c = c_1 + c_2(s+1) + c_3(s+1)^2 + \cdots + c_k(s+1)^{k-1}.$$

   Let $C = \mathcal{H}_\lambda (m \| t_0 \| t_1 \| \cdots \| t_s \| T_1 \| \cdots \| T_s \| T \| T')$.
2. For $j \in \{0, 1, 2, \ldots, s\}$ compute $t'_j = t_j \prod_{i:c_i=j} g_i^{h^{y_i}}$.
3. Verify that $T'(t'_s)^C = G^{R'}, Tz^C = g^Y G^y$ and $T_j t_j^C = \left(t'_{j-1}\right)^Y G^{Y_j}$, for $j \in \{1, 2, \ldots, s\}$.

Fig. 9. Verification of signature on m obtained by protocol from Fig. 8.

So $\beta = h^{(y_i - y'_i)/(c_i - c'_i)}$ and $x = (y_i - y'_i)/(c_i - c'_i)$.

Zero-Knowledge. We prove zero-knowledge for version of the protocol, whose challenges are produced by $\mathcal{V}$ as soon as he receives all arguments used in hash functions. The data revealed in steps 1-4 i.e. $t_0, \ldots, t_s$ and $y_1, \ldots, y_k$ are uniformly distributed in their domains, so they leak no information. The proof of equality of powers from steps 5-7 in Figure 8 is known to be zero-knowledge, so it also leaks no information.

7 Complexity of protocols

We concentrate on the protocols for double discrete logarithm. The protocols for e-th root of discrete logarithm behave the same when e is large. If κ is a parameter and λ is fixed, the complexity is $O(\kappa/\log\kappa)$. But we are not interested in asymptotics but in practical execution time and transcript length. We assume that λ is some constant (e.g. 2048) that assures intractability of number-theoretic problems and $\kappa = 160$. We express the *length* of the proof as the number of long integers that are exchanged in the protocol. The *time* complexity can be measured as the number of power operations (for x, y compute x^y) on pairs of long numbers, since they have the biggest cost. The complexities as functions of s and k are in Fig. 10. In Fig. 11 we substitute the values s and k that minimize the times and lengths of the proofs for $\kappa = 160$ to the formulas from Fig. 10. We should mention, that the protocol in Fig. 8 can give even more advantages as the length of the transcript than specified in Fig. 11. It could be the case when $n \ll p$ (similarly as in DSA), because the only information repeated k times has the length of n.

	length	time $\mathcal{P}$	time $\mathcal{V}$
Stadler [16]	$2k$	$2k$	$2k$
protocol Fig. 5	$2k+s+1$	$2k+s$	$2k+s$
protocol Fig. 8	$k+3s+6$	$2k+4s+5$	$2k+3s+5$

Fig. 10. Complexities of double discrete logarithm protocols as functions of k and s.

	length	time $\mathcal{P}$	time $\mathcal{V}$
Stadler [16]	320	320	320
protocol Fig. 5 ($s = 21, k = 36$)	94	93	93
protocol Fig. 8 ($s = 8, k = 51$)	81	140	131

Fig. 11. Complexities of protocols for double discrete logarithm for $\kappa = 160$.

Acknowledgment

Author wishes to thank Paweł Zalewski for valuable remarks that improved the presentation.

References

1. Giuseppe Ateniese, Dawn Xiaodong Song, Gene Tsudik: Quasi-Efficient Revocation in Group Signatures. Financial Cryptography 2002: 183-197
2. Emmanuel Bresson, Jacques Stern: Proofs of Knowledge for Non-monotone Discrete-Log Formulae and Applications. ISC 2002: 272-288
3. Jan Camenisch, Markus Stadler: Efficient Group Signature Schemes for Large Groups (Extended Abstract). CRYPTO 1997: 410-424
4. Sébastien Canard, Aline Gouget: Divisible E-Cash Systems Can Be Truly Anonymous. EUROCRYPT 2007: 482-497
5. David Chaum, Eugène van Heijst, Birgit Pfitzmann: Cryptographically Strong Undeniable Signatures, Unconditionally Secure for the Signer. CRYPTO 1991: 470-484
6. David Chaum, Torben P. Pedersen: Wallet Databases with Observers. CRYPTO 1992: 89-105
7. Amos Fiat, Adi Shamir: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. CRYPTO 1986: 186-194
8. Shafi Goldwasser, Silvio Micali, Charles Rackoff: The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract) STOC 1985: 291-304
9. Philippe Golle, Stanisław Jarecki, Ilya Mironov: Cryptographic Primitives Enforcing Communication and Storage Complexity. Financial Cryptography 2002: 120-135
10. Wenbo Mao: Verifiable Escrowed Signature. ACISP 1997: 240-248
11. Toru Nakanishi, Yuji Sugiyama: Unlinkable Divisible Electronic Cash. ISW 2000: 121-134
12. Tatsuaki Okamoto: Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. CRYPTO 1992: 31-53
13. Paweł Pszona, Grzegorz Stachowiak: Unlinkable Divisible Digital Cash without Trusted Third Party. eprint 2007/216
14. Claus-Peter Schnorr: Efficient Identification and Signatures for Smart Cards. CRYPTO 1989: 239-252
15. Berry Schoenmakers: A Simple Publicly Verifiable Secret Sharing Scheme and Its Application to Electronic Voting. CRYPTO 1999: 148-164
16. Markus Stadler: Publicly Verifiable Secret Sharing. EUROCRYPT 1996: 190-199.