IET Information Security

Research Article

IET Journals
The Institution of Engineering and Technology

Sigma protocol for faster proof of simultaneous homomorphism relations

ISSN 1751-8709
Received on 26th October 2017
Revised 12th February 2019
Accepted on 4th March 2019
E-First on 13th May 2019
doi: 10.1049/iet-ifs.2018.5167
www.ietdl.org

Bagher Bagherpour[1] ✉, Ali Zaghian[1], Mahdi Sajadieh[2]

[1]*Department of Mathematics and Cryptography, Malek Ashtar University of Technology, Isfahan, Iran*
[2]*Department of Electrical and Computer Engineering, Islamic Azad University Khorasgan Branch, Isfahan, Iran*
✉ *E-mail: bagher.bagherpour@mut-es.ac.ir*

Abstract: The Σ-protocols for homomorphism relations are one of the cryptographic protocols which are used to prove knowledge of homomorphism relations. The Schnorr protocol is one of the most famous Σ-protocols used for proving knowledge of discrete logarithm (DL) relation in which the verifier essentially performs one double-exponentiation (i.e. a group computation of the form $a^x b^y$). A direct application of the Schnorr protocol for proving simultaneous knowledge of n DLs with a common base leads to a Σ-protocol in which the verifier performs n double-exponentiations. In this study, the authors propose another Σ-protocol for homomorphism relations. The proposed Σ-protocol has fast verification when is used to prove the simultaneous homomorphism relations with a common homomorphism. Also, when the DL instantiation (DL-instantiation) of the proposed Σ-protocol is used to prove simultaneous knowledge of n DLs with a common base, it leads to a Σ-protocol in which the verifier performs n+1 single-exponentiations.

A Σ-protocol [1] for a relation R, defining an NP language L_R, is a three-round proof system run by a prover and a verifier in which the prover proves knowledge of a witness for a common input $x \in L_R$. In a Σ-protocol the only message sent by the verifier is a random challenge string. Such proof systems are distinguished by their two very useful properties: special soundness, which is a strong form of proof of knowledge [2], and a special form of zero knowledge [3]. One might argue that the security provided by the special honest-verifier zero knowledge property is insufficient as it gives no immediate guarantees against malicious verifiers deviating from the protocol. Nevertheless, Σ-protocols are frequently used in constructing other cryptographic primitives and protocols. In [4, 5] an overview of Σ-protocols and some of their basic interesting properties can be found.

Consider homomorphism $\psi: \mathbb{G} \to \mathbb{H}$, where $(G, +)$ is an additive group and $\mathbb{H}$ is a multiplicative group. A Σ-protocol under homomorphism ψ is a proof of knowledge of a preimage which runs between the prover and verifier. We use Σ_ψ-protocol to denote the Σ-protocols for homomorphism relations. Informally speaking, a Σ_ψ-protocol has the property that if a prover, holding $y \in \mathbb{H}$, succeeds in convincing the verifier to accept, then the prover must know $x \in \mathbb{G}$ such that $\psi(x) = y$. From a theoretical point of view, the problem of proving knowledge of a preimage under a homomorphism is completely resolved. Because, by using the general methods that can be applied for all NP relations, there exists a Σ_ψ-protocol for any homomorphism relation [6]. However, many of these methods are not efficient from the practical point of view. Σ_ψ-protocols have a lot of applications in cryptography, for instance: in designing signatures, blind signatures, commitment schemes, non-interactive zero-knowledge proofs. Also, many of the known Σ-protocols are special instantiations of these protocols. Therefore, designing efficient Σ_ψ-protocols for homomorphism relations is desirable and very important. An interesting feature of Σ_ψ-protocols is that they can be used to prove compound statements efficiently. A Σ-protocol for proving the AND of two statements (compound-AND) is a parallel execution of two Σ-protocols in which a challenge is sent by the verifier. Unlike the compound-AND, the compound-OR construction is more complicated but still efficiently possible [7]. For an overview of Σ_ψ-protocols, see [6, 8].

In this paper, we consider proving knowledge of several (say n) preimages with a common homomorphism which proving knowledge of several discrete logarithms (DLs) with a common base is a special instantiation of it. Suppose Π is a Σ_ψ-protocol for proving knowledge of one preimage. A naive solution for constructing a Σ_ψ-protocol for proving knowledge of n preimages with a common homomorphism is to build a compound-AND construction from Π. The Schnorr protocol, proposed in 1989 [9], is one of the most famous Σ-protocols used for proving knowledge of DL relation. This protocol is DL-instantiation of the proposed Σ_ψ-protocol in [6]. If the Schnorr protocol is used for proving knowledge of n DLs, then the verification of the resulted protocol can be carried out by performing n double-exponentiations (i.e. a group computation of the form $a^x b^y$). We construct another Σ_ψ-protocol for homomorphism relations which can be applied to a more general class of Σ-protocols. Our approach leads to a faster protocol when we deal with simultaneous homomorphism relations with common homomorphism. When the DL-instantiation of the proposed Σ_ψ-protocol runs to prove simultaneous knowledge of n DLs with a common base, it leads to a Σ-protocol in which the verifier performs $n+1$ single exponentiations (i.e. a group computation of the form a^x). Consequently, if we execute DL-instantiation of our Σ_ψ-protocol for proving simultaneous knowledge of n DLs with a common base, then our verifier runs $nt_{2xp}/(n+1)t_{1xp}$ times faster than that of the Schnorr protocol when the Schnorr protocol is used for proving simultaneous knowledge of n DLs with a common base (t_{1xp} and t_{2xp} are, respectively, costs of a single- and double-exponentiation). Based on known results on fast multi-exponentiation [10–13], we show that in typical applications we achieve a gain of up to 16.9% for our verifier.

Related work: The work of [14] (see [15] for the journal version) considers proving simultaneous DL (SDL) equalities, a generalised version of the Chaum and Pedersen [16] protocol. Moreover, the security guarantee holds in the random oracle model and only for computationally bounded provers. The batch verification technique [17] is not applicable either since the completeness does not hold with probability one.

Organisation: Section 1 presents the required background on Σ-protocols. In Section 2, we introduce our protocol for homomorphism relation, and then we express the DL-instantiation

of the proposed Σ_ψ-protocol and analyse its efficiency. In the sequel of this section, we express the Diffie–Hellman instantiation (DH-instantiation) of the proposed Σ_ψ-protocol. Section 3 concludes the paper.

1 Preliminary

In this section, we introduce our notation and review the standard definition of Σ-protocols and their AND composition. Besides, we introduce some well-known Σ_ψ-protocols and the well-known Schnorr protocol. In the sequel of this section, we mention the Chaum–Pedersen protocol. We refer the interested reader to [4, 5] for an overview of Σ-protocols and some of their intriguing properties.

1.1 Notation

Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be a binary relation. For each $(x, w) \in R$, w is called a witness for x. The only restriction imposed on R is that the length of each witness w of x is at most a polynomial in $|x|$. The relation R defines a language $L_R = \{x | (x, w) \in R\}$. For finite group $\mathbb{G}$ of prime order q with generator g, define the DL relation and SDL relations with a common base g as follows, where $w, w_1, \ldots, w_n \in \mathbb{Z}_q$:

$$R_{\mathrm{DL}} = \{((\mathbb{G}, q, g, h), w) \,|\, h = g^w\},$$

$$R_{\mathrm{SDL}}(n) = \{((\mathbb{G}, q, g, h_1, \ldots, h_n), (w_1, \ldots, w_n)) \,|\, h_i = g^{w_i}, \; i = 1, \ldots, n\}.$$

Also, the Diffie–Hellman (DH) relation and simultaneous DH (SDH) relations with a common base (g, h) can be defined as

$$R_{\mathrm{DH}} = \{((\mathbb{G}, q, g, h, u, v), w) \,|\, u = g^w, v = h^w\},$$

$$R_{\mathrm{SDH}}(n) = \{((\mathbb{G}, q, g, h, (u_1, v_1), \ldots, (u_n, v_n)), (w_1, \ldots, w_n)) \,|\, u_i = g^{w_i}, v_i = h^{w_i}, i = 1, \ldots, n\},$$

where g, h are generators of $\mathbb{G}$ and $w, w_1, \ldots, w_n \in \mathbb{Z}_q$. To simplify our description, we omit the specification of the security parameter in our definitions. In the sequel of this section, we mention a brief exposition of homomorphism relations. We refer the reader to [6] for further studying on the homomorphism relations and their Σ_ψ-protocols. Let $(\mathbb{G}, +)$ be an additive group with identity element '0' and $(\mathbb{H}, .)$ be a multiplicative group with identity element '1'. A mapping $\psi: \mathbb{G} \to \mathbb{H}$ is called *homomorphism* if for very $g_1, g_2 \in \mathbb{G}$ it holds $\psi(g_1 + g_2) = \psi(g_1) \cdot \psi(g_2)$. If $\psi(g) = h$, then $g \in \mathbb{G}$ is called a *preimage* of $h \in \mathbb{H}$ under the homomorphism $\psi: \mathbb{G} \to \mathbb{H}$. If $\psi(g) = h$, then h is called the *image* of g. For given homomorphism $\psi: \mathbb{G} \to \mathbb{H}$, $\mathbb{G}$ is called the *domain* of ψ and $\mathbb{H}$ is called the *co domain* of ψ.

Definition 1: (Simultaneous homomorphism relation): Let Ψ be a collection of homomorphisms. The homomorphism relation is defined as

$$R[\psi] = \{((\psi, y), x) : \psi \in \Psi, \psi: \mathbb{G} \to \mathbb{H}, x \in \mathbb{G}, y = \psi(x)\},$$

and the simultaneous homomorphism relations with a common homomorphism ψ is defined as

$$R[\Psi] = \{((\psi, y_1, \ldots, y_n), x_1, \ldots, x_n) : \psi \in \Psi, \psi: \mathbb{G} \to \mathbb{H}, x_1, \ldots, x_n \in \mathbb{G}, y_i = \psi(x_i), i = 1, \ldots, n\}.$$

1.2 Σ-protocol

Consider a two-party protocol, for a relation R, played by a prover $\mathscr{P}$ and a verifier $\mathscr{V}$. Both $\mathscr{P}$ and $\mathscr{V}$ receive a common input $x \in L_R$ and $\mathscr{P}$ receives w, a witness of x, as an additional private input. We associate two algorithms $\mathtt{P}_1, \mathtt{P}_2$ to $\mathscr{P}$ and a deterministic algorithm $\mathtt{V}$ to $\mathscr{V}$. Let ℓ, the challenge length, be an integer and consider a three-round protocol $\Pi = ((\mathtt{P}_1, \mathtt{P}_2), \mathtt{V})$ defined by the following general template:

(i) *Announcement:* $\mathscr{P}$ executes $\mathtt{P}_1$ on common input x, private input w and random input r and obtains $(a, r) \leftarrow \mathtt{P}_1(x, w, r)$, where a is the announcement value. Then $\mathscr{P}$ sends a to $\mathscr{V}$.
(ii) *Challenge:* The verifier $\mathscr{V}$ chooses a challenge value $c \in \{0,1\}^\ell$ at random and sends it to $\mathscr{P}$.
(iii) *Response:* The prover $\mathscr{P}$ executes $\mathtt{P}_2$ on input (x, w, r, c), obtains a response value $z \leftarrow \mathtt{P}_2(x, w, r, c)$ and then sends z to $\mathscr{V}$, where r is the output of the algorithm $\mathtt{P}_1$.
(iv) *Verification:* The verifier $\mathscr{V}$ executes $\mathtt{V}$ on input (x, a, c, z) and outputs the bit $b = \mathtt{V}(x, a, c, z)$ as its decision ($b = 1$ for accept and $b = 0$ for reject).

The triple $\langle a, c, z\rangle$ of exchanged messages is called a *transcript* for (x, w). A transcript $\langle a, c, z\rangle$ is called *accepting for* x (or shortly accepting when there is no confusion) if and only if $\mathrm{V}(x, a, c, z) = 1$. A pair of accepting transcripts $\langle a, c, z\rangle$ and $\langle a, c', z'\rangle$ for x is called *colliding* if $c \neq c'$.

Definition 2: (Σ-protocol): A three-round protocol $\Pi = ((\mathtt{P}_1, \mathtt{P}_2), \mathtt{V})$ is a Σ-protocol for relation R if the following requirements hold:

- *Completeness:* If $(x, w) \in R$ and $\mathscr{P}$ and $\mathscr{V}$ follow the protocol honestly on their inputs, then every transcript $\langle a, c, z\rangle$ for (x, w) is accepting.
- *Special soundness:* There exists a polynomial-time algorithm $\mathtt{E}$, called extractor, that on input x and any pair of colliding transcripts for x, outputs a witness w such that $(x, w) \in R$.
- *Special honest verifier zero-knowledge:* There exists a probabilistic polynomial-time algorithm $\mathtt{S}$, called a simulator, which on any input $x \in L_R$ and $c \in \{0,1\}^\ell$ outputs an accepting transcript for x where c is the challenge. Furthermore, the output of $\mathtt{S}(x, c)$ is identically distributed as that of a transcript obtained when the verifier sends c as a challenge and prover runs on input (x, w) for any witness w of x.

Let $\Pi = ((\mathtt{P}_1, \mathtt{P}_2), \mathtt{V})$ be a Σ-protocol for relation R and $(x, w) \in R$. Π is a *delayed-input* protocol if prover can compute the announcement value without knowing x. The Schnorr protocol is a delayed-input protocol [18]. It can be verified that the considered protocols in this paper enjoy this property.

1.3 Well-known protocols

Let Ψ be a collection of homomorphisms with finite domain and $((\psi, y), x) \in R[\Psi]$. Protocol 1 is a Σ_ψ-protocol to prove the knowledge of preimage x for which $\psi(x) = y$. By using the homomorphism properties, it can be easily verified that the completeness, special soundness and special honest verifier zero-knowledge properties hold for Protocol 1 [6].

Protocol 1: Σ_ψ-protocol for homomorphism relation with finite domain.

- *Common input:* $(\mathbb{G}, \mathbb{H}, \psi, y \in \mathbb{H})$ is known to $\mathscr{P}$ and $\mathscr{V}$.
- *Private input:* $\mathscr{P}$ knows $x \in \mathbb{G}$ such that $y = \psi(x)$.

1. $\mathscr{P}$ chooses $r \in \mathbb{G}$ at random, sets $t = \psi(r)$ and sends t to $\mathscr{V}$.
2. $\mathscr{V}$ chooses a random challenge $c \in \{0,1\}^\ell$ and sends it to $\mathscr{P}$.
3. $\mathscr{P}$ sends s to $\mathscr{V}$, where $s = r + cx$.
4. $\mathscr{V}$ outputs 1 (i.e. accepts) if and only if $\psi(s) = ty^c$.

Protocol 2, proposed by Schnorr [9], is a well-known Σ-protocol for proving relation R_{DL}. It is an easy exercise to demonstrate that

completeness, special soundness and special honest verifier zero-knowledge properties all hold.

Protocol 2: Schnorr protocol for R_{DL}.

- *Common input:* $(\mathbb{G}, q, g, h)$ is known to $\mathscr{P}$ and $\mathscr{V}$.
- *Private input:* $\mathscr{P}$ knows $w \in \mathbb{Z}_q$ such that $h = g^w$.

1. $\mathscr{P}$ chooses a random $r \in \mathbb{Z}_q$ and sends $a = g^r$ to $\mathscr{V}$.
2. $\mathscr{V}$ chooses a random challenge $c \in \{0,1\}^\ell$ and sends it to $\mathscr{P}$.
3. $\mathscr{P}$ computes $z = r + wc \bmod q$ and sends z to $\mathscr{V}$.
4. $\mathscr{V}$ outputs 1 (i.e. accepts) if and only if $g^z = ah^c$ holds.

1.4 Proving compound statements

Σ_ψ-protocols can be used to construct new Σ_ψ-protocols for compound statements. To prove several statements, the prover simply proves all statements in parallel while the verifier sends a single challenge for all proofs. A formal proof of this claim is straightforward. Protocol 3 is a Σ_ψ-protocol to prove relation $R[\Psi]$.

Protocol 3: Σ_ψ-protocol for simultaneous homomorphism relations with finite domain.

- *Common input:* $(\mathbb{G}, \mathbb{H}, \psi, y_1, \ldots, y_n \in \mathbb{H})$ is known to $\mathscr{P}$ and $\mathscr{V}$.
- *Private input:* $\mathscr{P}$ knows $x_1, \ldots, x_n \in \mathbb{G}$ such that $y_i = \psi(x_i)$ for each $i = 1, \ldots, n$.

1. $\mathscr{P}$ chooses $r_1, \ldots, r_n \in \mathbb{G}$ at random, sets $t_i = \psi(r_i)$, for each $i = 1, \ldots, n$ and sends $t_1, \ldots, t_n$ to $\mathscr{V}$.
2. $\mathscr{V}$ chooses a random challenge $c \in \{0,1\}^\ell$ and sends it to $\mathscr{P}$.
3. $\mathscr{P}$ sends s_i to $\mathscr{V}$, where $s_i = r_i + cx_i$ for each $i = 1, \ldots, n$.
4. $\mathscr{V}$ outputs 1 (i.e. accepts) if and only if $\psi(s_i) = t_i y_i^c$ holds for every $i = 1, \ldots, n$.

Protocol 4 uses n instances of Schnorr protocol in parallel for proving knowledge of n DLs in the same base. In other words, it is a Σ-protocol for proving relation $R_{\mathrm{SDL}}(n)$. We remark that proving the OR of several statements is also possible, but more involved, as it was first shown in [7].

Protocol 4: Σ-protocol for $R_{\mathrm{SDL}}(n)$ using Protocol 2.

- *Common input:* $(\mathbb{G}, q, g, h_1, \ldots, h_n)$ is known to $\mathscr{P}$ and $\mathscr{V}$.
- *Private input:* $\mathscr{P}$ knows $w_1, \ldots, w_n \in \mathbb{Z}_q$ such that $h_i = g^{w_i}$ for each $i = 1, \ldots, n$.

1. $\mathscr{P}$ chooses $r_1, \ldots, r_n \in \mathbb{Z}_q$ at random, sets $a_i = g^{r_i}$ for each $i = 1, \ldots, n$ and sends $(a_1, \ldots, a_n)$ to $\mathscr{V}$.
2. $\mathscr{V}$ chooses a random challenge $c \in \{0,1\}^\ell$ and sends it to $\mathscr{P}$.
3. $\mathscr{P}$ sends $(z_1, \ldots, z_n)$ to $\mathscr{V}$, where $z_i = r_i + w_i c \bmod q$ for each $i = 1, \ldots, n$.
4. $\mathscr{V}$ outputs 1 (i.e. accepts) if and only if $g^{z_i} = a_i h_i^c$ holds for every $i = 1, \ldots, n$.

We quote the Chaum–Pedersen protocol for SDH relations in Protocol 5, which is a Σ-protocol to prove the relation $R_{\mathrm{SDH}}(n)$.

Protocol 5: Chaum–Pedersen protocol for relation $R_{\mathrm{SDH}}(n)$.

- *Common input:* $(\mathbb{G}, q, g, h, (u_1, v_1), \ldots, (u_n, v_n))$ is known to $\mathscr{P}$ and $\mathscr{V}$.
- *Private input:* $\mathscr{P}$ knows $w_1, \ldots, w_n \in \mathbb{Z}_q$ such that $u_i = g^{w_i}$ and $v_i = h^{w_i}$ for each $i = 1, \ldots, n$.

1. $\mathscr{P}$ chooses $r_1, \ldots, r_n \in \mathbb{Z}_q$ at random, sets $a_i = g^{r_i}$ and $b_i = h^{r_i}$ for each $i = 1, \ldots, n$ and sends $((a_1, b_1), \ldots, (a_n, b_n))$ to $\mathscr{V}$.
2. $\mathscr{V}$ chooses a random challenge $c \in \{0,1\}^\ell$ and sends it to $\mathscr{P}$.
3. $\mathscr{P}$ sends $(z_1, \ldots, z_n)$ to $\mathscr{V}$, where $z_i = r_i + cw_i \bmod q$ for each $i = 1, \ldots, n$.
4. $\mathscr{V}$ outputs 1 (i.e. accepts) if and only if $g^{z_i} = a_i u_i^c$ and $h^{z_i} = b_i v_i^c$ hold for every $i = 1, \ldots, n$.

2 Our Σ-protocols

In this section, we propose another Σ_ψ-protocol (see Protocol 6) for proving relation $R[\psi]$. Protocol 6 by itself has no advantage over Protocol 1. However, when Protocol 6 is used to construct a Σ_ψ-protocol for proving relation $R[\Psi]$, it leads to a Σ_ψ-protocol (see Protocol 7) which is more efficient than Protocol 3.

In Protocols 6 and 7, the witnesses must have multiplicative inverses. Therefore, in the definition of the homomorphism relation we need the preimages x and $x_1, \ldots, x_n$ have multiplicative inverses. This assumption does not decrease generality of the presented protocols, since proving knowledge of preimage of group identity element is trivial to handle.

Protocol 6: Our protocol for homomorphism relation with finite domain.

- *Common input:* $(\mathbb{G}, \mathbb{H}, \psi, y \in \mathbb{H})$ is known to $\mathscr{P}$ and $\mathscr{V}$.
- *Private input:* $\mathscr{P}$ knows $x \in \mathbb{G}$ such that $y = \psi(x)$.

1. $\mathscr{P}$ chooses random value $r \in \mathbb{G}$. $\mathscr{P}$ executes $\mathtt{P}_1$ on input (y, x, r) and obtains t. Then $\mathscr{P}$ sends t to $\mathscr{V}$. Algorithm $\mathtt{P}_1$ is as follows:

$$\mathtt{P}_1\colon \begin{cases} \text{input}\colon (y, x, r); \\ t \leftarrow \psi(r); \\ \text{output } t. \end{cases}$$

2. $\mathscr{V}$ chooses a random challenge $c \in \{0,1\}^\ell$ and sends it to $\mathscr{P}$.
3. $\mathscr{P}$ executes $\mathtt{P}_2$ on input (y, x, r, c) and obtains s. Then $\mathscr{P}$ sends s to $\mathscr{V}$. Algorithm $\mathtt{P}_2$ is as follows:

$$\mathtt{P}_2\colon \begin{cases} \text{input}\colon (y, x, r, c); \\ s \leftarrow x^{-1}(r + c); \\ \text{output } s. \end{cases}$$

4. $\mathscr{V}$ executes $\mathtt{V}$ on input (y, t, c, s) and accepts protocol if and only if $\mathtt{V}$ outputs 1. Algorithm $\mathtt{V}$ is as follows:

$$\mathtt{V}\colon \begin{cases} \text{input}\colon (y, t, c, s); \\ \text{if } y^s = t\psi(c), \text{then output 1}; \\ \text{else output 0}. \end{cases}$$

Protocol 7: Our protocol for simultaneous homomorphism relations with a common homomorphism and finite domain.

- *Common input:* $(\mathbb{G}, \mathbb{H}, \psi, y_1, \ldots, y_n \in \mathbb{H})$ is known to $\mathscr{P}$ and $\mathscr{V}$.
- *Private input:* $\mathscr{P}$ knows $x_1, \ldots, x_n \in \mathbb{G}$ such that $y_i = \psi(x_i)$ for each $i = 1, \ldots, n$.

1. $\mathscr{P}$ chooses random values $r_1, \ldots, r_n \in \mathbb{G}$. $\mathscr{P}$ executes $\mathtt{P}_1$ on input $((y_1, x_1, r_1), \ldots, (y_n, x_n, r_n))$ and obtains $(t_1, \ldots, t_n)$. Then $\mathscr{P}$ sends $(t_1, \ldots, t_n)$ to $\mathscr{V}$. Algorithm $\mathtt{P}_1$ is as follows:

$$\mathtt{P}_1\colon \begin{cases} \text{input}\colon ((y_1, x_1, r_1), \ldots, (y_n, x_n, r_n)); \\ t_i \leftarrow \psi(r_i), \text{ for each } i \in \{1, \ldots, n\}; \\ \text{output } (t_1, \ldots, t_n). \end{cases}$$

2. $\mathscr{V}$ chooses a random challenge $c \in \{0,1\}^\ell$ and sends it to $\mathscr{P}$.

3. $\mathscr{P}$ executes $\mathtt{P}_2$ on input $((y_1, x_1, r_1), \ldots, (y_n, x_n, r_n), c)$ and obtains $(s_1, \ldots, s_n)$. Then $\mathscr{P}$ sends $(s_1, \ldots, s_n)$ to $\mathscr{V}$. Algorithm $\mathtt{P}_2$ is as follows:

$$\mathtt{P}_2: \begin{cases} \text{input}: ((y_1, x_1, r_1), \ldots, (y_n, x_n, r_n), c); \\ s_i \leftarrow x_i^{-1}(r_i + c), \text{ for each } i \in \{1, \ldots, n\}; \\ \text{output } (s_1, \ldots, s_n). \end{cases}$$

4. $\mathscr{V}$ executes $\mathtt{V}$ on input $((y_1, \ldots, y_n), (t_1, \ldots, t_n), c, (s_1, \ldots, s_n))$ and accepts protocol if and only if $\mathtt{V}$ outputs 1. Algorithm $\mathtt{V}$ is as follows:

$$\mathtt{V}: \begin{cases} \text{input}: ((y_1, \ldots, y_n), (t_1, \ldots, t_n), c, (s_1, \ldots, s_n)); \\ \text{for each } i \in \{1, \ldots, n\} \text{ if } y_i^{s_i} = t_i \psi(c), \text{then output 1}; \\ \text{else output 0}. \end{cases}$$

.

Theorem 1: Let $\psi: \mathbb{G} \to \mathbb{H}$ be a homomorphism with finite domain and $x_1, \ldots, x_n \in \mathbb{G}$. Suppose for each $i = 1, \ldots, n$ it holds $x_i \neq 0$ and each element of G has multiplicative inverse. Then Protocol 6 and Protocol 7 are Σ_ψ-protocols for relations $R[\psi]$ and $R[\Psi]$, respectively.

Proof: We provide a proof for relation $R[\Psi]$ which is a general statement. To prove the completeness, observe that when the prover runs the protocol honestly we have

$$y_i^{s_i} = y_i^{x_i^{-1}(r_i + c)} = \psi(x_i^{-1}(r_i + c)x_i) = \psi(r_i + c) = t_i \psi(c).$$

To prove the special soundness, consider two colliding (i.e. accepting with $c \neq c'$) transcripts $\langle t_1, \ldots, t_n, c, s_1, \ldots, s_n \rangle$ and $\langle t_1, \ldots, t_n, c', s_1', \ldots, s_n' \rangle$. Given the two colliding transcripts, the extractor $\mathtt{E}$ outputs x_i such that $x_i = (c - c')/(s_i - s_i')$. Since $y_i^{s_i} = t_i \psi(c)$ and $y_i^{s'i} = t_i \psi(c')$ hold, $c \neq c'$ implies $s_i \neq s_i'$. Therefore, the extractor works fine. From $y_i^{s_i - s'i} = \psi(c - c')$, it follows that $y_i = \psi(x_i)$. To demonstrate the special honest verifier zero-knowledge property, given a challenge c, the simulator $\mathtt{S}$ selects $s_1, \ldots, s_n \in \mathbb{Z}_q$ at random, sets $t_i = y_i^{s_i} \psi(-c)$ and outputs the transcript $\langle (t_1, \ldots, t_n), c, (s_1 \ldots, s_n) \rangle$. Clearly, the output has exactly the same probability distribution as a real conversation between an honest prover and an honest verifier. □

In Table 1, we mention the computational costs of Protocol 7 and Protocol 3. We use t_h to denote the computational cost of one homomorphic operation (i.e. the computational cost of computing $\psi(x)$, where $x \in$). Also, we use t_M to denote the computation cost of one group multiplication. By Table 1, it can be said that in Protocol 7 the verifier computes $n(t_{1XP} + t_M) + t_h$ operations, while in Protocol 3 it needs to compute $n(t_{1XP} + t_M + t_h)$ operations. Since in some real-life scenario, the witnesses are used to carry out several proofs in a long period and the prover can pre-compute the modular inverses once and forever. Therefore, the computational cost of the prover in Protocol 7 is the same as that of Protocol 3. Moreover, the overall communication cost for prover of Protocol 7 is exactly that of Protocol 3.

At the follows, we express the DL-instantiation of Protocol 6 which is a Σ-protocol for proving relation R_{DL} (see Protocol 8). Protocol 8 by itself has no advantage over Protocol 2. However, when Protocol 8 is used to construct a Σ-protocol for proving relation $R_{\mathrm{SDL}}(n)$, it leads to an improved protocol (see Protocol 9), when the number of DLs is moderately large.

Table 1 Computational cost of Protocol 7 and Protocol 3

	Protocol 7	Protocol 3
computational cost for $\mathscr{V}$	$n(t_{1XP} + t_M) + t_h$	$n(t_{1XP} + t_h + t_M)$
computational cost for $\mathscr{P}$	$nt_h + nt_M$	$nt_h + nt_M$

Due to a minor technical reason, as we have mentioned before, we let all the exponent witnesses $w, w_1, \ldots, w_n$ in the definition of relations R_{DL} and $R_{\mathrm{SDL}}(n)$ lie in $\mathbb{Z}_q^\star$ instead of $\mathbb{Z}_q$. This assumption does not decrease generality of the presented protocols since proving knowledge of DL of the group identity element is trivial to handle.

Protocol 8: DL-instantiation of Homo for proving relation R_{DL}.

- *Common input:* $(\mathbb{G}, q, g, h)$ is known to $\mathscr{P}$ and $\mathscr{V}$.
- *Private input:* $\mathscr{P}$ knows $w \in \mathbb{Z}_q^\star$ such that $h = g^w$.

1. $\mathscr{P}$ chooses random value $r \in \mathbb{Z}_q$. $\mathscr{P}$ executes $\mathtt{P}_1$ on input (h, w, r) and obtains a. Then $\mathscr{P}$ sends a to $\mathscr{V}$. Algorithm $\mathtt{P}_1$ is as follows:

$$\mathtt{P}_1: \begin{cases} \text{input}: (h, w, r); \\ a \leftarrow g^r; \\ \text{output } a. \end{cases}$$

2. $\mathscr{V}$ chooses a random challenge $c \in \{0, 1\}^\ell$ and sends it to $\mathscr{P}$.
3. $\mathscr{P}$ executes $\mathtt{P}_2$ on input (h, w, r, c) and obtains z. Then $\mathscr{P}$ sends z to $\mathscr{V}$. Algorithm $\mathtt{P}_2$ is as follows:

$$\mathtt{P}_2: \begin{cases} \text{input}: (h, w, r, c); \\ z \leftarrow w^{-1}(r + c) \bmod q; \\ \text{output } z. \end{cases}$$

4. $\mathscr{V}$ executes $\mathtt{V}$ on input (h, a, c, z) and accepts protocol if and only if $\mathtt{V}$ outputs 1. Algorithm $\mathtt{V}$ is as follows: (see equation below).

Protocol 9: Proving relation $R_{\mathrm{SDL}}(n)$ using DL-instantiation of Protocol 6.

- *Common input:* $(\mathbb{G}, q, g, h_1, \ldots, h_n)$ is known to $\mathscr{P}$ and $\mathscr{V}$.
- *Private input:* $\mathscr{P}$ knows $w_1, \ldots, w_n \in \mathbb{Z}_q^\star$ such that $h_i = g^{w_i}$ for each $i = 1, \ldots, n$.

1. $\mathscr{P}$ chooses random values $r_1, \ldots, r_n \in \mathbb{Z}_q$. $\mathscr{P}$ executes $\mathtt{P}_1$ on input $((h_1, w_1, r_1), \ldots, (h_n, w_n, r_n))$ and obtains $(a_1, \ldots, a_n)$. Then $\mathscr{P}$ sends $(a_1, \ldots, a_n)$ to $\mathscr{V}$. Algorithm $\mathtt{P}_1$ is as follows:

$$\mathtt{P}_1: \begin{cases} \text{input}: ((h_1, w_1, r_1), \ldots, (h_n, w_n, r_n)); \\ a_i \leftarrow g^{r_i}, \text{ for each } i \in \{1, \ldots, n\}; \\ \text{output } (a_1, \ldots, a_n). \end{cases}$$

2. $\mathscr{V}$ chooses a random challenge $c \in \{0, 1\}^\ell$ and sends it to $\mathscr{P}$.
3. $\mathscr{P}$ executes $\mathtt{P}_2$ on input $((h_1, w_1, r_1), \ldots, (h_n, w_n, r_n), c)$ and obtains $(z_1, \ldots, z_n)$. Then $\mathscr{P}$ sends $(z_1, \ldots, z_n)$ to $\mathscr{V}$. Algorithm $\mathtt{P}_2$ is as follows:

$$\mathtt{V}: \begin{cases} \text{input}: (h, a, c, z); \\ \text{if } h^z = a g^c, \text{then output 1}; \\ \text{else output 0}. \end{cases}$$

$$\mathtt{P}_2:\begin{cases}\text{input}:((h_1,w_1,r_1),\ldots,(h_n,w_n,r_n),c);\\ z_i\leftarrow w_i^{-1}(r_i+c) \bmod q, \text{for each } i\in\{1,\ldots,n\};\\ \text{output } (z_1,\ldots,z_n).\end{cases}$$

4. $\mathscr{V}$ executes $\mathtt{V}$ on input $((h_1,\ldots,h_n),(a_1,\ldots,a_n),c,(z_1,\ldots,z_n))$ and accepts protocol if and only if $\mathtt{V}$ outputs 1. Algorithm $\mathtt{V}$ is as follows:

$$\mathtt{V}:\begin{cases}\text{input}:((h_1,\ldots,h_n),(a_1,\ldots,a_n),c,(z_1,\ldots,z_n));\\ \text{for each } i\in\{1,\ldots,n\} \text{ if } h_i^{z_i}=a_ig^c, \text{then output 1};\\ \text{else output 0};\end{cases}$$

Theorem 2: Let $\mathbb{G}$ be a finite group of prime order q with generator g. Protocol 8 and Protocol 9 are Σ-protocols for the relations R_{DL} and $R_{\mathrm{SDL}}(n)$, respectively, where $w,w_1,\ldots,w_n\in\mathbb{Z}_q^\star$.

Proof: We provide a proof for the second general statement. To see the completeness, observe that when the prover runs the protocol honestly we have

$$h_i^{z_i}=g^{w_iz}=g^{r_i+c}=g^{r_i}g^c=a_ig^c.$$

Regarding special soundness, consider two colliding (i.e. accepting with $c\neq c'$) transcripts $\langle(a_1,\ldots,a_n),c,(z_1,\ldots,z_n)\rangle$ and $\langle(a_1,\ldots,a_n),c',(z_1',\ldots,z_n')\rangle$. Given the two colliding transcripts, the extractor $\mathtt{E}$ outputs $(w_1,\ldots,w_n)$, where $w_i=((c-c')/(z_i-z')) \bmod q$. Notice that since $h_i^{z_i}=a_ig^c$ and $h_i^{z'_i}=a_ig^{c'}$ hold, $c\neq c'$ implies $z_i\neq z_i'$ and, therefore, the extractor works fine. From $h_i^{z_i-z'_i}=g^{c-c'}$ or equivalently $h_i=g^{(c-c')/(z_i-z'_i)}$, it follows that indeed $h_i=g^{w_i}$. To demonstrate the special honest verifier zero-knowledge property, given a challenge c, the simulator $\mathtt{S}$ selects $z_1,\ldots,z_n\in\mathbb{Z}_q$ at random, sets $a_i=h_i^{z_i}g^{-c}$ and outputs the transcript $\langle(a_1,\ldots,a_n),c,(z_1,\ldots,z_n)\rangle$. It is evident that the output has exactly the same probability distribution as a real conversation between an honest prover and an honest verifier. □

At follows, we compared the efficiency of Protocol 9 and Protocol 4 and show that Protocol 9 is better than Protocol 4 in term of efficiency.

Efficiency comparison: The overall communication cost for prover of Protocol 9 is exactly that of Protocol 4. The computational costs of the prover and verifier however differ. The prover of Protocol 9 needs to compute n modular inverses in addition to what the prover of Protocol 4 performs. Nevertheless, the additional cost is not significant compared with the heavier cost of group exponentiations. Moreover, in some real-life scenario, it will be the case where the witnesses are used to carry out several proofs in a long period, and so the prover can pre-compute the modular inverses once and forever. We conclude that the computational costs of provers in both protocols are almost the same and equal to performing n group exponentiations. There are $2n$ (respectively $n+1$) exponentiations in the verification of Protocol 4 (respectively Protocol 9). Nevertheless, we cannot conclude that our verifier is asymptotically twice faster. The verifier of Protocol 4 can exploit the fact that a double-exponentiation (i.e. a computation of the form a^xb^y) can be performed significantly more efficiently than two single exponentiations using the window techniques of [10–13]. A double-exponentiation requires a number of multiplications and squarings. For simplicity, we assume that multiplication and squaring have the same complexities.

In [10] (Theorem 3.4), the authors proved that the number of multiplications and squarings needed to compute $\Pi_{i=1}^d g_i^{k_i}$ is equal to

$$\frac{l}{w+(2^d-1)^{-1}}+\frac{1-2^{-d(w-1)}}{2^d-1}+l-w-1, \tag{1}$$

Table 2 Computational cost of Protocol 9 and Protocol 4

	Protocol 9	Protocol 4
computational cost for $\mathscr{V}$	$(n+1)t_{1XP}$	nt_{2XP}
computational cost for $\mathscr{P}$	$nt_{1XP}+nt_M$	$nt_{1XP}+nt_M$

where w is the window size, $l=\mathrm{Max}\{1+\lfloor\log(k_i)\rfloor:i=1,\ldots,d\}$, k_i is the exponent of g^{k_i} and d is the number of g^{k_i} in $\Pi_{i=1}^d g_i^{k_i}$. By using the relation 1, we can say that the computational cost of single-exponentiation is

$$t_{1XP}=\frac{l}{w+1}-2^{1-w}+l-w, \tag{2}$$

and the computational cost of double-exponentiation is

$$t_{2XP}=\frac{1-2^{2-2w}}{3}+\frac{3l}{3w+1}+l-w-1. \tag{3}$$

Let us to denote by T_v^9 the computational cost of verifier in Protocol 9 and by T_v^4 the computational cost of verifier in Protocol 4. Let $T=T_v^4/T_v^9$. By using Table 2, it can be said that

$$T=\frac{nt_{2XP}}{(n+1)t_{1XP}}. \tag{4}$$

Consider a setting with window size $w=1$. In this setting, we have

$$t_{1XP}=\frac{3l}{2}-2,$$

$$t_{2XP}=\frac{7l}{4}-2.$$

Therefore

$$T=\frac{n(7l-8)}{(n+1)(6l-8)}.$$

In this setting, if we use $|q|=160$, then $l=160$ and

$$T\simeq\frac{n}{n+1}\times\frac{1169}{1000}. \tag{5}$$

In order that the verifier of Protocol 9 outperforms that of Protocol 4, we must have $T>1$. The relation 5 implies that, for $n\geq 6$ the verifier of Protocol 9 outperforms that of Protocol 4. Also, for large n we have $T=1.169$ which is the best achieved improvement for $w=1$ and $l=160$. Namely, for large n, the verifier of Protocol 9 runs 1.169 times (i.e. 16.90%) faster than that of Protocol 4. If we use $|q|=256$, then $l=256$ and

$$T\simeq\frac{n}{n+1}\times\frac{1784}{1528}.$$

For $n\geq 6$, the verifier of Protocol 9 outperforms that of Protocol 4. Also, for large n the verifier of Protocol 9 runs 1.1675 times (i.e. 16.75%) faster than that of Protocol 4. Now, if we use $|q|=512$, then $l=512$ and

$$T\simeq\frac{n}{n+1}\times\frac{3576}{3064}.$$

For $n\geq 6$, the verifier of Protocol 9 outperforms that of Protocol 4. Also, for large n the verifier of Protocol 9 runs 1.1671 times (i.e. 16.71%) faster than that of Protocol 4. The improved gain for a moderate value of $n\simeq 60$ is already 15%. In Table 3, we express the minimum value of n for which the verifier of Protocol 9 runs faster than that of Protocol 4 when they run with window sizes $w=1,2,3,4$ and $l=160,256,512$. Also, we express the achieved

Table 3 Improvements by Protocol 9 for large n

w	l	Minimum value of n for which $T > 1$	Improvement for large n, %
1	160	6	16.90
	256	6	16.75
	512	6	16.71
2	160	15	7.10
	256	15	7.08
	512	15	7.07
3	160	25	4.16
	256	25	4.10
	512	25	4.04
4	160	43	2.49
	256	43	2.41
	512	43	2.32

Table 4 Computational cost of Protocol 5 and Protocol 10

	Protocol 10	Protocol 5
computational cost for $\mathscr{V}$	$2(n+1)t_{1XP}$	$2nt_{2XP}$
computational cost for $\mathscr{P}$	$2nt_{1XP} + nt_M$	$2nt_{1XP} + nt_M$

Table 5 Improvements by Protocol 10 for large n

w	l	Minimum value of n for which $T' > 1$	Improvement for large n, %
1	160	6	33.80
	256	6	33.50
	512	6	33.42
2	160	15	14.20
	256	15	14.16
	512	15	14.14
3	160	25	8.32
	256	25	8.20
	512	25	8.08
4	160	43	4.98
	256	43	4.82
	512	43	4.64

improvements for large n. By using Table 3, it can be inferred that the improvements mainly depend on the window size, when the window size increases, the improvements decreases and close to zero. Also, the minimum value of n (for which $T > 1$) increases by increasing the window size. For large window sizes ($w \geq 3$), the running time of Protocol 9 is approximately equal to the running time of Protocol 4, but for large window sizes the storage increases, because for window size w the verifier needs to pre-compute almost $n2^{2w}$ values and save them before starting protocol [10].

Now, we express DH-instantiation of Protocol 6 for proving relation $R_{\mathrm{SDH}}(n)$.

Protocol 10: Proving relation $R_{\mathrm{SDH}}(n)$ using DH-instantiation of Protocol 6.

- *Common input:* $(\mathbb{G}, q, g, h, (u_1, v_1), \ldots, (u_n, v_n))$ is known to $\mathscr{P}$ and $\mathscr{V}$.
- *Private input:* $\mathscr{P}$ knows $w_1, \ldots, w_n \in \mathbb{Z}_q^\star$ such that $u_i = g^{w_i}$ and $v_i = h^{w_i}$ for each $i = 1, \ldots, n$.

1. $\mathscr{P}$ chooses random values $r_1, \ldots, r_n \in \mathbb{Z}_q$. $\mathscr{P}$ executes $\mathtt{P}_1$ on input $(((u_1, v_1), w_1, r_1), \ldots, ((u_n, v_n), w_n, r_n))$ and obtains $((a_1, b_1), \ldots, (a_n, b_n))$. Then $\mathscr{P}$ sends $((a_1, b_1), \ldots, (a_n, b_n))$ to $\mathscr{V}$. Algorithm $\mathtt{P}_1$ is as follows:

$$\mathtt{P}_1: \begin{cases} \text{input}: (((u_1, v_1), w_1, r_1), \ldots, ((u_n, v_n), w_n, r_n)); \\ a_i \leftarrow g^{r_i}, b_i \leftarrow h^{r_i}, \text{ for each } i \in \{1, \ldots, n\}; \\ \text{output } ((a_1, b_1), \ldots, (a_n, b_n)). \end{cases}$$

2. $\mathscr{V}$ chooses a random challenge $c \in \{0,1\}^\ell$ and sends it to $\mathscr{P}$.
3. $\mathscr{P}$ executes $\mathtt{P}_2$ on input $(((u_1, v_1), w_1, r_1), \ldots, ((u_n, v_n), w_n, r_n), c)$ and obtains $(z_1, \ldots, z_n)$. Then $\mathscr{P}$ sends $(z_1, \ldots, z_n)$ to $\mathscr{V}$. Algorithm $\mathtt{P}_2$ is as follows:

$$\mathtt{P}_2: \begin{cases} \text{input}: (((u_1, v_1), w_1, r_1), \ldots, ((u_n, v_n), w_n, r_n), c); \\ z_i \leftarrow w_i^{-1}(r_i + c) \bmod q, \text{ for each } i \in \{1, \ldots, n\}; \\ \text{output } (z_1, \ldots, z_n). \end{cases}$$

4. $\mathscr{V}$ executes $\mathtt{V}$ on input $(((u_1, v_1), \ldots, (u_n, v_n)), ((a_1, b_1), \ldots, (a_n, b_n)), c, (z_1, \ldots, z_n))$ and accepts protocol if and only if $\mathtt{V}$ outputs 1. Algorithm $\mathtt{V}$ is as follows:

$$\mathtt{V}: \begin{cases} \text{input}: (((u_1, v_1), \ldots, (u_n, v_n)), ((a_1, b_1), \ldots, (a_n, b_n)), \\ c, (z_1, \ldots, z_n)); \\ \text{for each } i \in \{1, \ldots, n\} \text{ if } u_i^{z_i} = a_i g^c \text{ and } v_i^{z_i} = b_i h^c, \\ \text{then output } 1; \\ \text{else output } 0. \end{cases}$$

Theorem 3: Let $\mathbb{G}$ be a finite group of prime order q with generator g. Protocol 10 is a Σ-protocol for relation $R_{\mathrm{SDH}}(n)$, where $w, w_1, \ldots, w_n \in \mathbb{Z}_q^\star$.

Proof: The proof is similar to the proof of Theorem 2. □

Efficiency comparison: The overall communication cost for prover of Protocol 10 is the same as that of Protocol 5. By using the similar methods which we have mentioned, it can be inferred that the computational cost of $\mathscr{P}$ in Protocol 10 and Protocol 5 is equal to $2nt_{1XP} + nt_M$. The computational cost of $\mathscr{V}$ in Protocol 10 is equal to $2(n+1)t_{1XP}$, while that of $\mathscr{V}$ in Protocol 5 is equal to $2nt_{2XP}$ (Table 4). Protocol 10 and Protocol 5 can be seen as two implementations of Protocol 9 and Protocol 4, respectively. Let T_v^5 and T_v^{10} denote the computational cost of verifier in Protocol 5 and Protocol 10, respectively. Let $T' = T_v^5/T_v^{10}$. For a typical setting with $|q| = 160$, window size $w = 1$, and for large n the verifier of Protocol 10 runs $2 \times 1.169 = 2.338$ (i.e. 33.80%) times faster than that of Protocol 5. Also, for $n \geq 6$ we have $T' > 1$. Namely, for $n \geq 6$ the verifier of Protocol 10 outperforms that of Protocol 5. In Table 5, we express the minimum value of n for which $T' > 1$ when Protocol 10 and Protocol 5 run with window sizes $w = 1, 2, 3, 4$ and $l = 160, 256, 512$. Besides, the achieved improvements are expressed for large n.

3 Conclusion

In this paper, we proposed a Σ_ψ-protocol (Protocol 6) for proving knowledge of homomorphism relations. If the Protocol 6 is used to prove n homomorphism relations with a common homomorphism then it leads to a Σ_ψ-protocol (Protocol 7) with faster verification. By using the window techniques, we showed that if the DL-instantiation of Protocol 6 (Protocol 9) is used to prove the relation $R_{\mathrm{SDL}}(n)$, then it leads to a faster Σ-protocol. With window size $w = 1$, the verifier of the Protocol 9 runs 1.169 times faster than the verifier of the Schnorr protocol. This is the best achieved improvement because we showed that the improvements mainly depend on the window size and when the window size increases, the improvements decrease. Also, the DH-instantiation of Protocol 6 leads to a Σ-protocol with faster verification for proving the relation $R_{\mathrm{SDH}}(n)$. This can be seen as a generalisation of the Chaum–Pedersen [16] protocol for proving equality of two discrete

logarithms and might find applications in other areas such as design of more efficient mix-nets (e.g. see [19]). Consider the following compound-OR relation:

$$R_k = \{((y_1, \ldots, y_n), ((x_1, d_1), \ldots, (x_k, d_k))) : 1 \leq d_1 < \cdots < d_k \leq n \text{ and } y_{d_i} = \psi(x_i), i = 1, \ldots, k\},$$

where $\psi : \mathbb{G} \to \mathbb{H}$ is a homomorphism relation. In [20], the authors proposed an adaptive-input (k, n)-proof of partial knowledge by which the prover can postpone the use of simulator to third round. Since Protocol 1 and Protocol 6 are delayed-input protocols, we can construct an adaptive-input (k, n)-proof of partial knowledge for relation R_k. It can be verified that in the constructed protocol the used sub-protocols in verification are executed in parallel and the same challenge is used in many of them, therefore if we use Protocol 6 instead of Protocol 1 in the adaptive-input (k, n)-proof of partial knowledge for proving the relation R_k, then we have a protocol with fast verification. As a general result, it can be said that if protocol Π uses n instances of Protocol 1 as sub-protocols (in parallel with other sub-protocols) and we use Protocol 6 instead of Protocol 1 in Π, then this leads to a protocol with fast implementation.

4 Acknowledgments

The authors were very grateful to Dr. Shahram Khazaei for his help and suggestions. The authors also like to thank the anonymous referees for their comments which improved this paper.

5 References

[1] Cramer, R.: 'Modular design of secure yet practical cryptographic protocol'. PhD thesis, University of Amsterdam, 1996

[2] Bellare, M., Goldreich, O.: 'On defining proofs of knowledge'. Advances in Cryptology - CRYPTO'92, 12th Annual Int. Cryptology Conf., Santa Barbara, California, USA, August 1992, pp. 390–420

[3] Goldwasser, S., Micali, S., Rackoff, C.: 'The knowledge complexity of interactive proof systems', *SIAM J. Comput.*, 1989, **18**, pp. 186–208

[4] Damgård, I.: 'On Σ-protocols'. Available at http://www.cs.au.dk/ivan/Sigma.pdf, accessed 2004

[5] Hazay, C., Lindell, Y.: 'Efficient secure two-party protocols-techniques and constructions' in Basin, D., Maurer, U. (Eds.): '*Information security and cryptography*' (Springer, New York, 2010), pp. 147–175

[6] Bangerter, E.: 'Efficient zero-knowledge proofs of knowledge for homomorphisms'. PhD thesis, Ruhr-University Bochum, 2005

[7] Cramer, R., Damgård, I., Schoenmakers, B.: 'Proofs of partial knowledge and simplified design of witness hiding protocols'. Advances in Cryptology - CRYPTO'94, 14th Annual Int. Cryptology Conf., Santa Barbara, California, USA, August 1994, pp. 174–187

[8] Maurer, U.: 'Unifying zero-knowledge proofs of knowledge'. Progress in Cryptology - AFRICACRYPT 2009, Second Int. Conf. on Cryptology in Africa, Gammarth, Tunisia, June 2009, pp. 272–286

[9] Schnorr, C.P.: 'Efficient identification and signatures for smart cards'. Advances in Cryptology - EUROCRYPT'89, Workshop on the Theory and Application of Cryptographic Techniques, Houthalen, Belgium, April 1989, pp. 239–252

[10] Avanzi, R.M.: 'On multi-exponentiation in cryptography', *IACR Cryptol. ePrint Archive*, 2002, pp. 6–10

[11] Vassil, S.D., Jullien, G.A., Miller, W.C.: 'Complexity and fast algorithms for multi exponentiations', *IEEE Trans. Comput.*, 2000, **49**, pp. 141-147

[12] Menezes, A., van Oorschot, P.C., Vanstone, S.A.: '*Handbook of applied cryptography*' (CRC Press, Boca Raton, FL, USA, 1996)

[13] Möller, B.: 'Algorithms for multi-exponentiation'. 8th Annual Int. Workshop Selected Areas in Cryptography (SAC 2001), Toronto, Ontario, Canada, August 2001, pp. 165–180

[14] Chow, S.S.M., Ma, C., Weng, J.: 'Zero-knowledge argument for simultaneous discrete logarithms'. Computing and Combinatorics, 16th Annual Int. Conf., COCOON 2010, Nha Trang, Vietnam, July 2010, pp. 520–529

[15] Chow, S.S.M., Ma, C., Weng, J.: 'Zero-knowledge argument for simultaneous discrete logarithms', *Algorithmica*, 2012, **64**, pp. 246–266

[16] Chaum, D., Pedersen, T.P.: 'Wallet databases with observers'. Advances in Cryptology - CRYPTO'92, 12th Annual Int. Cryptology Conf., Santa Barbara, California, USA, August 1992, pp. 89–105

[17] Bellare, M., Juan, A.G., Rabin, T.: 'Fast batch verification for modular exponentiation and digital signatures'. Advances in Cryptology - EUROCRYPT'98, Int. Conf. on the Theory and Application of Cryptographic Techniques, Espoo, Finland, June 1998, pp. 236–250

[18] Ciampi, M., Persiano, G., Scafuro, A., *et al.*: 'Improved OR composition of Sigma protocols'. Theory of Cryptography Conf., Tel Aviv, Israel, January 2016, (LNCS), pp. 112–142

[19] Neff, A.C.: 'A verifiable secret shuffle and its application to e-voting'. Proc. of the 8th ACM Conf. on Computer and Communications Security, Philadelphia, Pennsylvania, USA, November 2001, pp. 116–125

[20] Ciampi, M., Persiano, G., Scafuro, A., *et al.*: 'Online/offline OR composition of Sigma protocols'. Advances in Cryptology - EUROCRYPT 2016, Vienna, Austria, May 2016, pp. 63–92