This Extended Abstract appears
in the Proceedings of the 17th International Conference on Practice and Theory in Public-Key Cryptography (PKC '14)
(26 – 28 March 2014, Buenos Aires, Argentina)
Hugo Krawczyk Ed., Springer-Verlag, LNCS 8383, pages 167–184.

Efficient Delegation of Zero-Knowledge Proofs of Knowledge in a Pairing-Friendly Setting

Sébastien Canard[1], David Pointcheval[2], and Olivier Sanders[1,2]

[1] Orange Labs, Applied Crypto Group, Caen, France
[2] École normale supérieure, CNRS & INRIA, Paris, France

Abstract. Since their introduction in 1985, by Goldwasser, Micali and Rackoff, followed by Feige, Fiat and Shamir, zero-knowledge proofs have played a significant role in modern cryptography: they allow a party to convince another party of the validity of a statement (proof of membership) or of its knowledge of a secret (proof of knowledge). Cryptographers frequently use them as building blocks in complex protocols since they offer quite useful soundness features, which exclude cheating players. In most of modern telecommunication services, the execution of these protocols involves a prover on a portable device, with limited capacities, and namely distinct trusted part and more powerful part. The former thus has to delegate some computations to the latter. However, since the latter is not fully trusted, it should not learn any secret information.
This paper focuses on proofs of knowledge of discrete logarithm relations sets (DLRS), and the delegation of some prover's computations, without leaking any critical information to the delegatee. We will achieve various efficient improvements ensuring perfect zero-knowledge against the verifier and partial zero-knowledge, but still reasonable in many contexts, against the delegatee.

1 Introduction

Zero-Knowledge Proofs of Knowledge. The past three decades have witnessed the emergence of several new cryptographic notions. In 1985, Goldwasser, Micali and Rackoff [16] introduced the concept of zero-knowledge interactive proofs that enable an entity, called the *prover*, to convince another entity, called the *verifier*, of the validity of a statement without revealing anything else beyond the assertion of this statement. In other words, one wants to prove that a statement is in the set of the valid statements, hence the notion of *zero-knowledge proof of membership*. They were followed by Feige, Fiat and Shamir [12] with the notion of *zero-knowledge proof of knowledge* (ZKPK) in which the prover convinces the verifier not only of the validity of a statement but also that it possesses a witness for this fact.

Since these seminal papers, many ZKPK have been introduced, such as the Schnorr's protocol [25], that provide efficient ways of proving knowledge of a discrete logarithm in finite groups with known order, and even with unknown order [14, 15]. In modern cryptography, these proofs of knowledge are heavily used for authentication but also as building blocks in more complex protocols, such as group signature schemes [1, 11, 4, 21] or Direct Anonymous Attestation (DAA) schemes [5, 3]. Indeed, such protocols usually require to prove that some public elements, relying on private values, are well-formed. For anonymous authentications, one classically wants to prove one's knowledge of a secret key related to a public key certified by a given authority, without revealing the secret key, the public key, nor the certificate itself. They can be efficiently addressed by using Schnorr-like interactive ZKPK. Moreover, these interactive proofs can be turned into non-interactive proofs or signatures using the Fiat-Shamir paradigm [13, 24], in the random oracle model [2].

Discrete-Logarithm Relation Sets. More complex protocols, such as group signature schemes or DAA schemes, involve several proofs of knowledge of discrete logarithms or of representations in a fixed or variable basis: they deal with a *Discrete-Logarithm Relation Set* (or DLRS, as defined by Kiayias, Tsiounis and Yung [20]), *i.e* a set of relations involving objects and free variables. Extensions of the Schnorr's protocol can be applied to this setting, but they require the prover to compute many exponentiations for the first round of the protocol (the commitments).

Pairing-Friendly Settings. Elliptic curves with or without pairing-friendly groups have been widely used for the past few years, since they offer many new features and provide communication-wise efficient protocols. They allow to prove complex relations with still reasonable efficiency, namely when

compared with the RSA setting. Indeed, most of the recent group signature schemes [11, 17, 4, 21] or DAA schemes [6, 10, 3] are based on groups ($\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$) of prime order with a bilinear map ($e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$).

The main interesting feature is definitely the possibility of non-interactive zero-knowledge proofs in the standard model, using the so-called Groth-Sahai methodology [18]. Unfortunately, while reducing the number of interactions is quite useful, this leads to quite costly protocols, for both the prover and the verifier. They are currently totally impractical on constrained devices.

Delegation of Computation. However, most of these complex cryptographic primitives, such as anonymous authentications and DAAs, achieve their ultimate impact when implemented on portable and mobile devices. This increases the contrast between the important needs to embed these protocols in such lightweight devices and their practical limitations when performing many exponentiations or pairing evaluations. A common way to overcome this problem is to delegate (when possible) some computations to a more powerful, but not fully trusted, delegatee as in [5, 7, 3, 8]. Since the latter entity cannot have access to secret values, most of the computations on the prover's side have to be performed by the constrained device, which reduces the benefits of server-aided cryptography. Moreover, if the DLRS involved in the protocol contains several relations or variables, the overall computational cost may remain prohibitive. One may argue that exponentiations in the first flow of Schnorr's protocol are precomputable. This is true if the basis is fixed, but when the proof is used as a building block in a more complex construction, the basis is not always fixed or known in advance (as *e.g.* in DAA schemes [5, 3]). The lack of way to efficiently delegate the prover's side of the proof of knowledge may then prevent portable devices to get access to all features of modern cryptography.

Although the delegatee might not be fully trusted, it may have access to some additional information. For example, let us consider the following setting: a SIM-card in a smartphone. This is probably the best illustration of a lightweight but fully trusted device (the SIM-card with embedded secrets) within a more powerful but partially trusted device (the smartphone with more and more powerful processors, and even co-processors). In case of group signature or anonymous authentication to a server, only the SIM-card knows the secret key to perform authentication, and no information about the identity of the actual user should leak to the server. However, while not trusted enough to learn the secret key, since it can potentially be corrupted by a virus, the smartphone anyway already knows its owner. As a consequence, the anonymity has to be enforced with respect to the server but not to the smartphone (it has other means to learn owner's identity). However, the secret key should not be leaked to neither the server nor the smartphone.

Such a SIM-card together with a smartphone issuing anonymous authentication illustrates well the relaxation on the security model that seems reasonable in practice: during delegation of computation, some additional information can be leaked to the helper until it does not help it to impersonate the real prover. We will thus provide several security models in which the delegatee might be given access to some extra knowledge. We however stress that the delegatee should remain unable to recover the secrets or to impersonate the prover, but still being able to handle a significant part of the prover's computations.

Achievements. In this paper, we provide an efficient way to delegate the prover's side of zero-knowledge proofs of knowledge for any DLRS in a group $\mathbb{G}_1$. Our method enables a delegator to use the computational power of a delegatee to prove knowledge of witnesses for any DLRS with significantly fewer computations than with the classical Schnorr's based protocol. While lifting the verification relation into $\mathbb{G}_T$, and thus involving pairing computations on the verifier's side, no pairing computations have to be performed on the prover's side (for both the delegator and the delegatee). Moreover, the computations that remain to be done by the delegator do not rely on the objects involved in the DLRS, but on a fixed basis only, they can thus all be precomputed.

By decreasing the computational cost for the constrained devices (the delegator), our work improves on the efficiency of protocols using zero-knowledge proofs of knowledge and thus enables engineers to embed complex primitives on such devices.

More precisely, we provide two constructions in which the delegator essentially computes as many exponentiations of a fixed basis as the number of secret discrete logarithms involved in the relations, whatever the number of relations is. We illustrate the effective gain on concrete examples.

2 Preliminaries

In this section, we provide a basic review of the tools that will be used throughout this paper. Namely, we recall the notations of bilinear maps and zero-knowledge proofs of knowledge together with the concept of Discrete-Logarithm Relations Sets (DLRS) and the Schnorr's protocol for such relations.

2.1 Pairing-Friendly Groups

Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be three groups of prime order p. In the following, we will use additive notations for $\mathbb{G}_1$ and $\mathbb{G}_2$, but multiplicative notations for $\mathbb{G}_T$. Elements of $\mathbb{G}_1$ will be written in uppercase ($G, X, T, \ldots$) and elements of $\mathbb{G}_2$ will be written ($\widetilde{G}, \widetilde{X}, \widetilde{T}, \ldots$). Pairing-friendly settings are defined by $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ along with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties:

1. for all $X \in \mathbb{G}_1, \widetilde{X} \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$ we have $e([a]X, [b]\widetilde{X}) = e(X, \widetilde{X})^{ab}$;
2. for $X \neq 0$ and $\widetilde{X} \neq 0$, $e(X, \widetilde{X}) \neq 1$;
3. e is efficiently computable.

We emphasize that our protocols will work in any pairing-friendly setting: in both the symmetric (*i.e.*, $\mathbb{G}_1 = \mathbb{G}_2$) and asymmetric (*i.e.*, $\mathbb{G}_1 \neq \mathbb{G}_2$) cases. In the following, the setting $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G, \widetilde{G}, e)$ defines the bilinear environment, with $\mathbb{G}_1 = \langle G \rangle$, $\mathbb{G}_2 = \langle \widetilde{G} \rangle$, and $\mathbb{G}_T = \langle e(G, \widetilde{G}) \rangle$. All the three groups being of the same prime order p.

2.2 Zero-Knowledge Proofs of Knowledge

Interactive zero-knowledge proofs of knowledge have been introduced by Goldwasser, Micali and Rackoff [16] and formalized by Feige, Fiat and Shamir [12]. We recall here the informal definition.

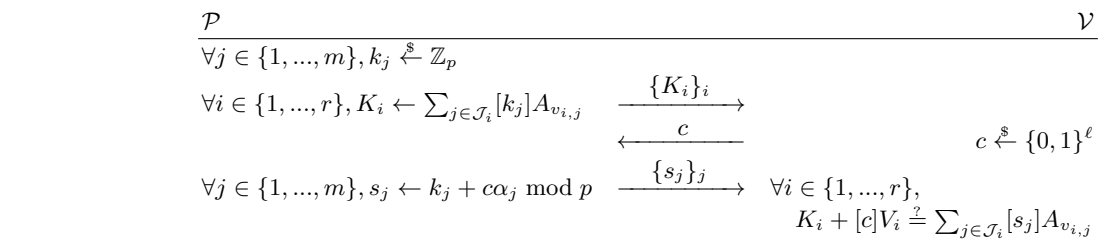
Definition 1. An interactive protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ is a zero-knowledge proof of knowledge of a private witness w for $\mathcal{P}$ that a public information Y satisfies a relation R if the three following properties are satisfied.

- **Completeness:** for an honest prover $\mathcal{P}$ with correct witness w and an honest verifier $\mathcal{V}$, the protocol succeeds with overwhelming probability.
- **Soundness:** for any prover $\widetilde{\mathcal{P}}$ that is accepted by a verifier $\mathcal{V}$ with non negligible probability, it is possible to construct a probabilistic polynomial time Turing machine $\mathcal{E}$ (called *extractor*) that can extract a valid witness w by interacting with $\widetilde{\mathcal{P}}$.
- **Zero-knowledge:** for every verifier $\mathcal{V}$, there exists a probabilistic polynomial-time Turing machine $\mathcal{S}$ (called *simulator*) that just takes Y as input and outputs a string that is indistinguishable from the transcript of the communications between an honest prover $\mathcal{P}$ with a valid witness w and $\mathcal{V}$.

The soundness property models the fact that in order to be accepted, the prover must actually know a valid witness, while the zero-knowledge property shows that the real protocol with the prover that uses the witness w does not leak more information than a simulation that does not know the witness.

2.3 Discrete-Logarithm Relations Set

Discrete-logarithm relations sets (DLRSs) were introduced by Kiayias *et al.* [20] to describe sets of relations involving secret variables that correspond to discrete logarithms. Many cryptographic protocols [22, 10, 3] require some entity to prove that some public elements (a ciphertext, a certificate, ...) relying on several secret values, are well-formed and based on a DLRS. They thus require a proof of knowledge for a DLRS. More formally, a DLRS can be defined as follows:



Setting: A group $\mathbb{G}$ of prime order p and a DLRS $\mathcal{R}$ in $\mathbb{G}$: for $A_1, \ldots, A_w, V_1, \ldots, V_r \in \mathbb{G}$, and $\mathcal{J}_1, \ldots, \mathcal{J}_r \subseteq \{1, \ldots, w\}$, the prover $\mathcal{P}$ knows variables $\alpha_1, \ldots, \alpha_m \in \mathbb{Z}_p$ such that $V_i = \sum_{j \in \mathcal{J}_i} [\alpha_j] A_{v_{i,j}}$, for $i = 1, \ldots, r$.

Fig. 1. The Extended Schnorr's Protocol for any DLRS $\mathcal{R}$

Definition 2. A DLRS $\mathcal{R}$ on the group $\mathbb{G}$ (of prime order p) with r relations over m variables and $w+r$ objects in $\mathbb{G}$ is a set of relations $R_1, \ldots, R_r$ defined over objects $A_1, \ldots, A_w, V_1, \ldots, V_r \in \mathbb{G}$ and the free variables $\alpha_1, \ldots, \alpha_m \in \mathbb{Z}_p$ where R_i, for $i = 1, \ldots, r$, is to be interpreted as: $V_i = \sum_{j \in \mathcal{J}_i} [\alpha_j] A_{v_{i,j}}$, where $\mathcal{J}_i \subseteq \{1, \ldots, m\}$ and $v_{i,j} \in \{1, \ldots, w\}$ for $i = 1, \ldots, r$ and $j \in \mathcal{J}_i$. We will write $\mathcal{R}(\alpha_1, \ldots, \alpha_m)$ to denote the conjunction of all the relations R_i on the variables $\alpha_1, \ldots, \alpha_m$.

Remark 3. The above definition is given in a group $\mathbb{G}$, but it could be in any group. In our practical applications, as we will work in pairing-friendly settings, relations could be all in $\mathbb{G}_1$ but also all in $\mathbb{G}_2$ or in both $\mathbb{G}_1$ and $\mathbb{G}_2$. In the following, we will describe our results in the group $\mathbb{G}_1$, with companion values in $\mathbb{G}_2$, and we will give evidences that it can also work in the general case.

Using these notations, a prover that knows witnesses $\alpha_1, \ldots, \alpha_m$ such that $\mathcal{R}(\alpha_1, ..., \alpha_m) = 1$ will generally use the 3-flow zero-knowledge proof of knowledge described in Figure 1 (which is easily derived from the Schnorr's protocol [25] for groups of known order). This protocol then corresponds to a proof of knowledge for a DLRS. The completeness comes from the fact that for valid witnesses $\alpha_1, \ldots, \alpha_m$ that satisfy, for all i, $V_i = \sum_{j \in \mathcal{J}_i} [\alpha_j] A_{v_{i,j}}$, then for all $i \in \{1, ..., r\}$,

$$\sum_{j \in \mathcal{J}_i} [s_j] A_{v_{i,j}} = \sum_{j \in \mathcal{J}_i} [k_j + c\alpha_j] A_{v_{i,j}} = \sum_{j \in \mathcal{J}_i} [k_j] A_{v_{i,j}} + [c] \sum_{j \in \mathcal{J}_i} [\alpha_j] A_{v_{i,j}} = K_i + [c] V_i.$$

The complexity for the prover is: $\sum_{i=1}^{r} \#\mathcal{J}_i$ multiplications by scalars in $\mathbb{G}$ and $\sum_{i=1}^{r} (\#\mathcal{J}_i - 1)$ additions in $\mathbb{G}$ to get the commitments K_i for $i \in \{1, ..., r\}$.

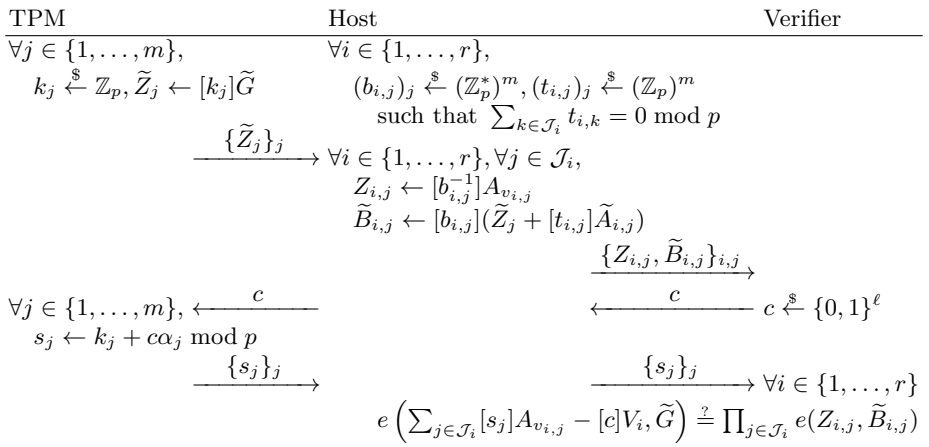
For complex DLRSs, it can represent too many computations. In the next section, we explain how to delegate such proofs of knowledge of DLRSs, where the constrained device has to compute m scalar multiplications in $\mathbb{G}_2$ to prove knowledge of $\alpha_1, \ldots, \alpha_m$ satisfying a DLRS $\mathcal{R}$ in $\mathbb{G}_1$, no matter how many relations R_i are involved in $\mathcal{R}$.

3 Delegating Proofs of Knowledge

As in [5, 7, 3], we will split the prover into a trusted device which has a limited computational power and a more powerful, but untrusted, machine. As in DAA [5] schemes, the trusted device will be called the TPM (Trusted Platform Module) and the untrusted machine will be called the host.

3.1 Our First Protocol

We consider the following situation: the TPM knows witnesses $(\alpha_1, \ldots, \alpha_m)$ for the DLRS $\mathcal{R}$, such that $\mathcal{R}(\alpha_1, \ldots, \alpha_m) = 1$, and wants to use the computational power of the host to prove knowledge of these witnesses. Since the host is not trusted, we do not want to give $(\alpha_1, \ldots, \alpha_m)$ to it (else it would be able to impersonate the TPM). However, we allow it to get access to more information than a standard verifier (see Theorem 5). This is a common requirement in DAA schemes and, more generally, in server-aided cryptography (see *e.g.* [8]).



Setting: A pairing-friendly setting $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G, \widetilde{G}, e)$ and a DLRS $\mathcal{R}$ in $\mathbb{G}_1$: for $A_1, \dots, A_w, V_1, \dots, V_r \in \mathbb{G}_1$, and $\mathcal{J}_1, \dots, \mathcal{J}_r \subseteq \{1, \dots, w\}$, the TPM knows variables $\alpha_1, \dots, \alpha_m \in \mathbb{Z}_p$ such that $V_i = \sum_{j\in\mathcal{J}_i}[\alpha_j]A_{v_{i,j}}$, for $i = 1, \dots, r$.
Notations: For $i = 1, \dots, w$, we denote $a_i \in \mathbb{Z}_p$ the discrete logarithms such that $A_i = [a_i]G$, and, for $i = 1, \dots, r$ and $j \in \mathcal{J}_i$, one computes $\widetilde{A}_{i,j} = \left[\frac{1}{a_{v_{i,j}}}\prod_{k\in\mathcal{J}_i} a_{v_{i,k}}\right]\widetilde{G}$ that are added to the public parameters (see Section 3.2 for details).
Players' inputs: The public input contains $G, \widetilde{G}, \{V_i\}_i, \{\mathcal{J}_i\}_i, \{A_j\}_j$ and the $\{\widetilde{A}_{i,j}\}_{i,j}$; The TPM additionally knows $\{\alpha_i\}_i$.

Fig. 2. Delegation of Proof of Knowledge of Witnesses for a DLRS

Intuition. Informally, we do not want the TPM to have to compute $[k_j]A_{v_{i,j}}$ for all the pairs (i, j), as in the extended Schnorr's protocol, then we essentially lift them to $\mathbb{G}_T$, by applying pairing with $\widetilde{G}$, and then the K_i's become

$$e\left(K_i, \widetilde{G}\right) = e\left(\sum_{j\in\mathcal{J}_i}[k_j]A_{v_{i,j}}, \widetilde{G}\right) = \prod_{j\in\mathcal{J}_i} e\left(A_{v_{i,j}}, [k_j]\widetilde{G}\right) = \prod_{j\in\mathcal{J}_i} e\left(A_{v_{i,j}}, \widetilde{Z}_j\right).$$

The verification $K_i \stackrel{?}{=} \sum_{j\in\mathcal{J}_i}[s_j]A_{v_{i,j}} - [c]V_i$ would then become

$$\prod_{j\in\mathcal{J}_i} e\left(A_{v_{i,j}}, \widetilde{Z}_j\right) \stackrel{?}{=} e\left(\sum_{j\in\mathcal{J}_i}[s_j]A_{v_{i,j}} - [c]V_i, \widetilde{G}\right).$$

This is the reason why the TPM can just compute $\widetilde{Z}_j = [k_j]\widetilde{G}$, for $k = 1, \dots, m$.

A First Note. However, it cannot directly send these values to the verifier. Otherwise, the zero-knowledge property obtained by our protocol would not be equivalent to the one of the initial Extended Schnorr's protocol, from the verifier's view: from $\widetilde{Z}_j = [k_i]\widetilde{G}$ and $s_j = k_j - c\alpha_j \bmod p$, one would be able to compute $[c^{-1}]\left(\widetilde{Z}_j - [s_j]\widetilde{G}\right) = [c^{-1}][c\alpha_j]\widetilde{G} = [\alpha_j]\widetilde{G}$. This might be too much information about α_j. These values are thus just sent to the host who will compute blinded versions $Z_{i,j} \leftarrow [b_{i,j}^{-1}]A_{v_{i,j}}$ and $\widetilde{B}_{i,j} \leftarrow [b_{i,j}](\widetilde{Z}_j + [t_{i,j}]\widetilde{A}_{i,j})$, with random scalars $(b_{i,j})_{i,j}$ and $(t_{i,j})_{i,j}$ and additional elements $(\widetilde{A}_{i,j})_{i,j}$ (defined in Figure 2), so that for any i,

$$\prod_{j\in\mathcal{J}_i} e\left(A_{v_{i,j}}, \widetilde{Z}_j\right) = \prod_{j\in\mathcal{J}_i} e\left(Z_{i,j}, \widetilde{B}_{i,j}\right) / \prod_{j\in\mathcal{J}_i} e\left(A_{v_{i,j}}, [t_{i,j}]\widetilde{A}_{i,j}\right)$$

where the latter denominator is equal to, with $c_i = \prod_{k\in\mathcal{J}_i} a_{v_{i,k}}$,

$$\prod_{j\in\mathcal{J}_i} e\left([a_{v_{i,j}}]G, [t_{i,j}/a_{v_{i,j}}]\prod_{k\in\mathcal{J}_i}[a_{v_{i,k}}]\widetilde{G}\right) = e\left(G, \left[\left(\sum_{j\in\mathcal{J}_i} t_{i,j}\right)c_i\right]\widetilde{G}\right).$$

By choosing $(t_{i,j})_{i,j}$ such that $\sum_{j\in\mathcal{J}_i} t_{i,j} = 0 \bmod p$, it is equal to $1_{\mathbb{G}_T}$.

A Second Note. If one just uses the factors $(b_{i,j})_{i,j}$, but not $(t_{i,j})_{i,j}$, the values $(Z_{i,j})_{i,j}$ and $(\widetilde{B}_{i,j})_{i,j}$ would reveal to much information too. Let us consider any pair (i,j) such that $j \in \mathcal{J}_i$ and $k = v_{i,j}$: $e(Z_{i,j}, \widetilde{B}_{i,j}) = e(A_k, \widetilde{Z}_j)$, and thus

$$\left(e(Z_{i,j}, \widetilde{B}_{i,j})/e(A_k, [s_j]\widetilde{G})\right)^{1/c} = e\left(A_k, [c^{-1}]\left(\widetilde{Z}_j - [s_j]\widetilde{G}\right)\right) = e\left(A_k, \alpha_j \widetilde{G}\right).$$

Then, $e(A_k, \widetilde{G})^{\alpha_j}$ would leak, which is again too much information about α_j.

In the case of a singleton $\mathcal{J}_i = \{j\}$, $V_i = [\alpha_j]A_k$ indeed leaks this information too, but in case of larger sets, such information does not leak, and thus should not leak from the proof either.

Description. These blinding factors $(b_{i,j})_{i,j}$ and $(t_{i,j})_{i,j}$ will make the protocol zero-knowledge from the verifier's view (as formally proven in Section 4). This leads to the 3-flow protocol described on Figure 2, that enables the TPM to prove knowledge of $(\alpha_1, \ldots, \alpha_m)$ with fewer computations than in the extended Schnorr's protocol (see Figure 1).

Example I. Let us consider the following example:

$$\begin{array}{rcr} V_1 = [\alpha_1]A_1 & \ldots & V_q = [\alpha_1]A_q \\ V_{q+1} = [\alpha_2]A_{q+1} & \ldots & V_{q+s} = [\alpha_2]A_{q+s} \\ V_{q+s+1} = [\alpha_1]A_{q+s+1} + [\alpha_2]A_{q+s+n+1} & \ldots & V_{q+s+n} = [\alpha_1]A_{q+s+n} + [\alpha_2]A_{q+s+2n} \end{array}$$

Using the extended Schnorr's protocol described on Figure 1, one would require $q + s + 2n$ multiplications by scalars in $\mathbb{G}_1$ (group exponentiations) and n additions in $\mathbb{G}_1$ from the TPM. With our protocol (see Figure 2), the TPM has to compute only 2 multiplications by scalars in $\mathbb{G}_2$ (group exponentiations).

3.2 Additional Computations

One might have noted that the public parameters must now contain several $\widetilde{A}_{i,j}$ that may not be known in practice. However, in most cases, there is no need of additional values. First, when $\mathcal{J}_i = \{j\}$ is a singleton, $\widetilde{A}_{i,j} = \widetilde{G}$. Second, when $\mathcal{J}_i = \{\alpha, \beta\}$ is a pair, and $v_{i,\alpha} = u$ and $v_{i,\beta} = v$, then $\widetilde{A}_{i,\alpha} = [a_v]\widetilde{G}$ and $\widetilde{A}_{i,\beta} = [a_u]\widetilde{G}$. Thus, $\widetilde{A}_{i,\alpha} = A_v$ and $\widetilde{A}_{i,\beta} = A_u$ in the case of symmetric pairing (*i.e.*, $\mathbb{G}_1 = \mathbb{G}_2$). Our above Example I involves singletons and pairs only, and thus the $\widetilde{A}_{i,j}$ can be easily publicly computed. However, in Section 5, we provide another delegation protocol that does not present these limitations, and can thus be used in more situations.

3.3 Computational Cost

Since the TPM is considered to be far less powerful than the host and the verifier, we want to decrease its computational load even if it involves a slight increase of work for the host and for the verifier. Let us evaluate the computational cost for each party (see Table 1):

- the TPM has to compute m multiplications by a scalar in $\mathbb{G}_2$ (one *per* variable α_i), which are moreover all precomputable. Its computational cost is thus independent of the number of relations, which can be very useful when a variable is involved in many relations (as in our above Example I);
- the host has to compute $\sum_{i=1}^{r} \#\mathcal{J}_i$ multiplications by a scalar in $\mathbb{G}_1$ and at most the same number of additions in $\mathbb{G}_2$ and twice as many multiplications by a scalar in $\mathbb{G}_2$;
- the verifier has to compute $\sum_{i=1}^{r} \#\mathcal{J}_i$ additions in $\mathbb{G}_1$, $r + \sum_{i=1}^{r} \#\mathcal{J}_i$ multiplications by a scalar in $\mathbb{G}_1$, $r + \sum_{i=1}^{r} \#\mathcal{J}_i$ pairings, and some multiplications in $\mathbb{G}_T$.

Table 1. Complexity Comparisons

	Prover		Verifier
	TPM	Host	
Ext. Schnorr	$JM + (J-r)A$		$JM + (J-r)A$
Example I	$(q+s+2n)M + nA$		$(q+s+2n)M + nA$
Example II	$7M + 2A$		$7M + 2A$
Example III	$9M + 3A$		$9M + 3A$
Figure 2	mM_2	$J(M_1 + 2M_2 + A_2)$	$J(M_1 + A_1 + P + M_T)$ $+r(M_1 + P - M_T)$
Example I	$2M_2$	$(q+s+2n)(M_1 + 2M_2 + A_2)$	$(2q+2s+3n)(M_1 + P)$ $+(q+s+2n)A_1 + nM_T$
Example II	$2M_2$	$7(M_1 + 2M_2 + A_2)$	$12(M_1 + P) + 7A_1 + 2M_T$
Example III	$6M_2$	$9(M_1 + 2M_2 + A_2)$	$15(M_1 + P) + 9A_1 + 3M_T$
Figure 3	mM_2	$J(2M_1 + 2M_2 + A_2 + A_1)$ $-rA_1$	$(J+r)(M_1 + A_1 + P)$ $+(J-r)M_T$
Example I	$2M_2$	$(q+s+2n)(2M1 + 2M_2 + A_2)$ $+nA_1$	$(2q+2s+3n)(M_1 + A_1 + P)$ $+nM_T$
Example II	$2M_2$	$7(2M_1 + 2M_2 + A_2) + 2A_1$	$12(M_1 + A_1 + P) + 2M_T$
Example III	$6M_2$	$9(2M_1 + 2M_2 + A_2) + 3A_1$	$15(M_1 + A_1 + P) + 3M_T$

Generic DLRS: m secret scalars, r relations each involving J_i elements respectively for $i = 1, \ldots, r$, and thus globally $J = \sum J_i$.
For the extended Schnorr, all computations have to be done by the TPM itself.
A, A_1, A_2 denote point additions in $\mathbb{G}$, $\mathbb{G}_1$, $\mathbb{G}_2$ respectively;
M, M_1, M_2 denote point multiplications by a scalar in $\mathbb{G}$, $\mathbb{G}_1$, $\mathbb{G}_2$ respectively;
M_T denotes multiplication in $\mathbb{G}_T$; P denotes a pairing.

3.4 More Examples

We now provide some concrete examples, with comparisons of the complexity computations on Table 1: Extended Schnorr is the natural 3-round protocol between a prover and a verifier, while the two other protocols are the delagated protocols proposed above (in Section 3) and below (in Section 5). One can note that our protocols with delagation drastically reduce the computational cost for the TPM with respect to the Prover in the basic protocol. To this aim, one can indeed use $\mathbb{G}_2$ as the efficient group and $\mathbb{G}_1$ as the less efficient group in the pairing-friendly setting.

Example II. In 2007, Shacham [26] described an encryption scheme based on the DLIN assumption. This is a Cramer-Shoup variant of the linear encryption, where the first triple is a linear tuple used for masking the plaintext in the fourth element, while the last element helps to verify validity with a hash proof system (see also [19]). With the public parameters $(G_1, G_2, G_3) \in \mathbb{G}_1^3$ and the public key $(H_1, H_2, C_1, C_2, D_1, D_2) \in \mathbb{G}_1^6$ and a collision-resistant hash function $\mathcal{H}$, to encrypt a message $M \in \mathbb{G}_1$, one computes, for random scalars $\alpha_1, \alpha_2 \in \mathbb{Z}_p$:

$$\begin{pmatrix} U_1 = [\alpha_1]G_1, \quad U_2 = [\alpha_2]G_2, \quad U_3 = [\alpha_1 + \alpha_2]G_3, \\ E = M + [\alpha_1]H_1 + [\alpha_2]H_2, \qquad V = [\alpha_1](C_1 + [u]D_1) + [\alpha_2](C_2 + [u]D_2) \end{pmatrix}$$

where $u = \mathcal{H}(U_1, U_2, U_3, E) \in \mathbb{Z}_p$. We may need to prove, as in [9], that (U_1, U_2, U_3, E, V) is a valid ciphertext. Since 2 secret variables (α_1 and α_2) are involved in the 4 relations to be checked for ciphertext validity (on U_1, U_2, U_3, and V), our protocol only requires 2 multiplications by a scalar from the TPM.

Example III. In [23], the authors provided a group signature with message-dependent opening (GS-DMO) scheme secure in the random oracle model. With the public parameters $(U, V, G, H) \in \mathbb{G}_1^4$, to issue a signature σ, one has to prove knowledge of $\alpha, \beta, x, \delta_1, \delta_2, \delta_3 \in \mathbb{Z}_p$ such that:

$$\begin{pmatrix} T_1 = [\alpha]U, & T_2 = [\beta]V, & T_3 = [\alpha + \beta]H, \\ 0 = [x]T_1 - [\delta_1]U, & 0 = [x]T_2 - [\delta_2]V, & 0 = [x]T_5 - [\delta_3]G \end{pmatrix}$$

where $T_1, T_2, T_3, T_5 \in \mathbb{G}_1$ are part of the signature σ. Since 6 secret variables are involved in these relations, our protocol only requires 6 multiplications by a scalar from the TPM.

3.5 Security Properties

The protocol described on Figure 2 may actually be divided in two parts: a proof of knowledge between $\mathcal{P}$ (TPM + host) and $\mathcal{V}$ (verifier) and a proof of knowledge between $\mathcal{P}$ (TPM) and $\mathcal{V}$ (host). We consider the security of each part in the following theorems, which proofs are provided in Section 4.

Theorem 4. *The protocol described on Figure 2 is a 3-move zero-knowledge proof of knowledge of the witnesses $\alpha_1, ..., \alpha_m$ between $\mathcal{P}$ (TPM + host) and $\mathcal{V}$ (verifier), where the description of $\mathcal{R}$ is the unique auxiliary input.*

The first theorem essentially shows that this proof of knowledge does not leak any information outside the host. But one may wonder if the host learns a lot of information. This is the goal of the second theorem below that says that the host just learns $\{[\alpha_i]\widetilde{G}\}_i$, which is not enough to impersonate the TPM later.

Theorem 5. *The protocol described on Figure 2 is a 3-move zero-knowledge proof of knowledge of the witnesses $\alpha_1, ..., \alpha_m$ between $\mathcal{P}$ (TPM) and $\mathcal{V}$ (host), where the auxiliary input contains the description of $\mathcal{R}$ and the additional values $\{[\alpha_i]\widetilde{G}\}_i$.*

3.6 Discussions

Honest Verifier Zero-Knowledge. As usual, this protocol is actually a zero-knowledge proof of knowledge if the challenge c is selected from $\{0,1\}^\ell$ and the proof is repeated k times with ℓ logarithmically bounded in the security parameter and $2^{k\ell}$ super-polynomial. If one wants the soundness in one execution only, which implies 2^ℓ to be super-polynomial, then the protocol is no longer zero-knowledge but *honest-verifier zero-knowledge* only.

Precomputation. As already noticed, if computations of a party are independent of external values, they can be prepared and stored in advance. This is the case of the elements $\widetilde{Z}_j$ computed by the TPM.

For example let us consider the `Sign` protocol of the DAA scheme from [3, page 32]. The TPM has to prove knowledge of its secret key s involved in two relations (namely $K = [s]J$ and $W = [s]S$). Since the authors use the standard Schnorr's protocol, this leads to 2 multiplications by a scalar for the TPM, one of which (the one involving J) has to be computed *online* because J is determined by the *basename* submitted by the verifier. Using our protocol, the TPM only has to compute one multiplication by a scalar, and it can even be precomputed, since the basis $\widetilde{G}$ is a public parameter.

We even emphasize that these precomputations (the group elements $\widetilde{Z}_j$) can even be sent to the host. The TPM just has to store the scalars k_j, or even a seed (and some index), as off-line precomputed coupons [15].

Extra Inputs. In the Theorem 5, we allow the host to learn the elements $[\alpha_j]\widetilde{G}$ for all $j \in \{1, \ldots, m\}$. In the DAA scheme considered above, this means that the host can learn $[s]\widetilde{G}$, which does not endanger the security properties.

Indeed, the non-frameability property of their scheme is based on the fact that the adversary does not know s. However, recovering s from both $[s]G$ and $[s]\widetilde{G}$ is not known to be much easier than recovering s from $[s]G$ alone. As a consequence, the non-frameability still holds.

However, one could argue that this additional information helps to break the anonymity property. But as already remarked, one does not require to enforce anonymity of the TPM with respect to the host, since the latter already knows which TPM is inserted (or even sees the signature which is sent outside). And as explained in [7], in DAA schemes and in server-aided version of group signatures, the host is not adversarially-controlled in the anonymity experiment, but just for the impersonation or frameability.

More General Relations. The protocol described on Figure 2 only considers relations in $\mathbb{G}_1$. But as already said, our protocol would work the same way if all relations were in $\mathbb{G}_2$, by simply swapping the role of $\mathbb{G}_1$ and $\mathbb{G}_2$ in our protocol described in Figure 2.

However, one could have to prove knowledge of variables involved in relations in both $\mathbb{G}_1$ and $\mathbb{G}_2$. In such a case the host would need to know a commitment in $\mathbb{G}_2$ (to compute the proof for relations in $\mathbb{G}_1$) and one in $\mathbb{G}_1$ (for the relations in $\mathbb{G}_2$). The computational cost for the TPM would then depend on the type of the pairing. For pairings of Type 1 or Type 2, the computational cost will remain the same because of the isomorphism. For pairings of Type 3 (without any efficient isomorphism), the TPM would have to compute the values in both groups, and thus with a multiplication by a scalar in $\mathbb{G}_1$ and a multiplication by a scalar in $\mathbb{G}_2$ for each variable involved in both $\mathbb{G}_1$ and $\mathbb{G}_2$. In any case, the computational cost remains independent of the number of relations.

4 Security Proofs

We now formally prove the two above theorems. Completeness and soundness will be similar for both, but the zero-knowledge property will involve two different simulators.

4.1 Completeness

It follows from the construction explained in Section 3.1: The verifier checks whether

$$e\left(\sum_{j\in\mathcal{J}_i}[s_j]A_{v_{i,j}} - [c]V_i, \widetilde{G}\right) \stackrel{?}{=} \prod_{j\in\mathcal{J}_i} e(Z_{i,j}, \widetilde{B}_{i,j}).$$

Since, for all $i \in \{1,\ldots,r\}$, $V_i = \sum_{j\in\mathcal{J}_i}[\alpha_j]A_{v_{i,j}}$ and for all $j \in \{1,\ldots,m\}$, $s_j = k_j + c\alpha_j \bmod p$, then $\sum_{j\in\mathcal{J}_i}[s_j]A_{v_{i,j}} = \sum_{j\in\mathcal{J}_i}[k_j + c\alpha_j]A_{v_{i,j}} = \sum_{j\in\mathcal{J}_i}[k_j]A_{v_{i,j}} + [c]V_i$, and one easily verifies that both sides are equal to $e\left(\sum_{j\in\mathcal{J}_i}[k_j]A_{v_{i,j}}, \widetilde{G}\right)$, which proves the completeness.

4.2 Soundness

Let $\{Z_{i,j}, \widetilde{B}_{i,j}\}_{i,j}$ be the values sent to the verifier at the first flow. If the adversary (trying to impersonate $\mathcal{P}$ (TPM + host)) can answer successfully with probability significantly greater than $1/2^\ell$, then it can send $\{s_j\}_j$ and $\{s'_j\}_j$ for two different challenges c and c': $\forall i \in \{1,\ldots,r\}$,

$$e\left(\sum_{j\in\mathcal{J}_i}[s_j]A_{v_{i,j}} - [c]V_i, \widetilde{G}\right) = \prod_{j\in\mathcal{J}_i} e(Z_{i,j}, \widetilde{B}_{i,j}) = e\left(\sum_{j\in\mathcal{J}_i}[s'_j]A_{v_{i,j}} - [c']V_i, \widetilde{G}\right),$$

which leads to $e\left(\sum_{j\in\mathcal{J}_i}[s_j - s'_j]A_{v_{i,j}} - [c-c']V_i, \widetilde{G}\right) = 1_{\mathbb{G}_T}$ and thus, from the non-degeneracy of the pairing, $\sum_{j\in\mathcal{J}_i}[s_j - s'_j]A_{v_{i,j}} - [c-c']V_i = 0_{\mathbb{G}_1}$. As a consequence, $\alpha_j = (s_j - s'_j)/(c-c')$ for $j = 1,\ldots,m$, we have $V_i = \sum_{j\in\mathcal{J}_i}[\alpha_j]A_{v_{i,j}}$ for $i = 1,\ldots,r$. This is thus a solution to the DLRS $\mathcal{R}$.

4.3 Zero-Knowledge w.r.t. the Host

For Theorem 5, we assume the host already knows (or can learn, as explained above) $T_j = [\alpha_j]\widetilde{G}$, $\forall j \in \{1,\ldots,m\}$. The simulator operates as follows:

- it first selects $c \stackrel{\$}{\leftarrow} \{0,1\}^\ell$ and $\{s_j\}_j \stackrel{\$}{\leftarrow} \mathbb{Z}_p$;
- it computes: $\widetilde{Z}_j \leftarrow [s_j]\widetilde{G} - [c]T_j$, for all $j \in \{1,\ldots,m\}$;
- it then outputs $\{\widetilde{Z}_j\}_j$, and waits for the challenge and rewinds in case of incorrect guess of c;
- it eventually answers $\{s_j\}_j$.

This is statistically indistinguishable from transcripts generated during a real protocol between the TPM and the host. Since the initial guess for c is perfectly hidden in $\{\widetilde{Z}_j\}_j$, the probability of successful simulation is $1/2^\ell$, which is non-negligible for a logarithmic value ℓ. For a larger ℓ, it remains honest-verifier zero-knowledge.

4.4 Zero-Knowledge w.r.t. the Verifier

For Theorem 4, the verifier just knows the public parameters: $G, \widetilde{G}, \{A_j\}_j, \{V_i\}$. The simulator operates as follows:

- it first selects $c \xleftarrow{\$} \{0,1\}^\ell$ and $\{s_j\}_j \xleftarrow{\$} \mathbb{Z}_p$;
- it computes $K_i \leftarrow \sum_{j\in\mathcal{J}_i}[s_j]A_{v_{i,j}} - [c]V_i$, for all $i \in \{1,\ldots,r\}$;
- it additionally selects, for $i \in \{1,\ldots,r\}$ and $j \in \mathcal{J}_i$, $u_{i,j} \xleftarrow{\$} \mathbb{Z}_p^*$ and $U_{i,j} \xleftarrow{\$} \mathbb{G}_1\backslash\{0_{\mathbb{G}_1}\}$, such that $\sum_{j\in\mathcal{J}_i} U_{i,j} = K_i$ (which conditions the last $U_{i,j}$);
- it then computes, for $i \in \{1,\ldots,r\}$ and $j \in \mathcal{J}_i$, $Z_{i,j} = [u_{i,j}^{-1}]U_{i,j}$ and $\widetilde{B}_{i,j} = [u_{i,j}]\widetilde{G}$;
- it then outputs $\{Z_{i,j}, \widetilde{B}_{i,j}\}_{i,j}$, and waits for the challenge and rewinds in case of incorrect guess of c;
- it eventually answers $\{s_j\}_j$.

A problem can occur with the above simulation if some elements get zero while it is not allowed. But the large order of the groups makes this problem to happen with negligible probability only. We exclude these bad cases in the following.

In order to prove the zero-knowledge property, we need to show that our simulated tuples are indistinguishable from the tuples generated during a real protocol, for the verifier. In a real protocol, the verifier sees: $\{Z_{i,j}, \widetilde{B}_{i,j}\}, c, \{s_j\}_j$, where $Z_{i,j} = [b_{i,j}^{-1}]A_{v_{i,j}} = [a_{v_{i,j}}/b_{i,j}]G$ for random non-zero scalars $b_{i,j}$, and $\widetilde{B}_{i,j} = [b_{i,j}](\widetilde{Z}_j + [t_{i,j}]\widetilde{A}_{i,j}) = [b_{i,j}/a_{v_{i,j}} \cdot (k_j a_{v_{i,j}} + t_{i,j}\prod_{k\in\mathcal{J}_i} a_{v_{i,k}})]\widetilde{G}$ for random scalars $t_{i,j}$, such that $\sum_{j\in\mathcal{J}_i} t_{i,j} = 0 \bmod p$.

Let us denote $u'_{i,j} = (b_{i,j}/a_{v_{i,j}}) \cdot (k_j a_{v_{i,j}} + t_{i,j}\prod_{k\in\mathcal{J}_i} a_{v_{i,k}})$, for $i = 1,\ldots,r$ and $j \in \mathcal{J}_i$. Then $\widetilde{B}_{i,j} = [u'_{i,j}]\widetilde{G}$. Since the $b_{i,j}$'s are independent random scalars, the $u'_{i,j}$'s are also independent random scalars, and thus follow the same distribution as the $u_{i,j}$'s.

With such a notation and $d_i = \prod_{k\in\mathcal{J}_i} a_{v_{i,k}}$, we have $Z_{i,j} = [(u'_{i,j})^{-1}(k_j a_{v_{i,j}} + t_{i,j}d_i)]G$. Let us denote $U'_{i,j} = [k_j a_{v_{i,j}} + t_{i,j}d_i]G$. Since the $t_{i,j}$ are random scalars with the unique constraint that $\sum_{j\in\mathcal{J}_i} t_{i,j} = 0 \bmod p$, for $i = 1,\ldots,r$, then the $U'_{i,j}$'s are random elements in $\mathbb{G}_1$ with the constraint that, for $i = 1,\ldots,r$,

$$\sum_{j\in\mathcal{J}_i} U'_{i,j} = [\sum_{j\in\mathcal{J}_i} k_j a_{v_{i,j}}]G = \sum_{j\in\mathcal{J}_i}[s_j - c\alpha_j]A_{v_{i,j}} = \sum_{j\in\mathcal{J}_i}[s_j]A_{v_{i,j}} - [c]V_i = K_i.$$

As a consequence, in the real protocol execution, for $i \in \{1,\ldots,r\}$ and $j \in \mathcal{J}_i$, $Z_{i,j} = [(u'_{i,j})^{-1}]U'_{i,j}$ and $\widetilde{B}_{i,j} = [u'_{i,j}]\widetilde{G}$, where the $u'_{i,j}$'s and $U'_{i,j}$'s follow the same distributions as the $u_{i,j}$'s and $U_{i,j}$'s generated by our simulator.

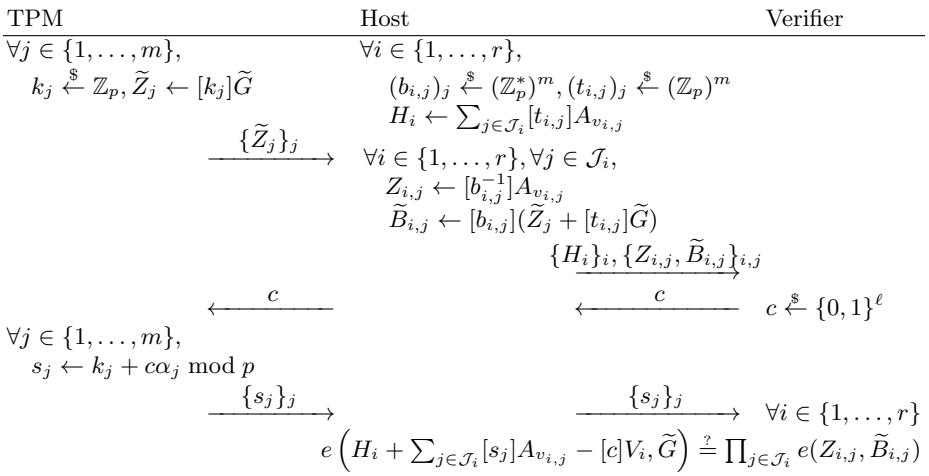
5 Delegating with Weaker Assumptions

5.1 Description

As said in Section 3.2, our first protocol required the knowledge of the elements $\widetilde{A}_{i,j}$. In many applications, such as our first example, this is not a strong requirement. However, in some other cases, this can be a problem. We thus now provide another protocol for the same delegation from the TPM to the host, with just a slight increase of the computations for the host, but without any additional information. The main difference with our first protocol is that the Host now needs to additionally compute the H_i's which permit to blind the $\widetilde{A}_{i,j}$'s. This protocol is described on Figure 3 and the obtained efficiency is given in Table 1.

5.2 Security Results

Theorem 6. *The protocol described on Figure 3 is a 3-move zero-knowledge proof of knowledge of the witnesses $\alpha_1, ..., \alpha_m$ between $\mathcal{P}$ (TPM + host) and $\mathcal{V}$ (verifier), where the description of $\mathcal{R}$ is the unique auxiliary input.*



Setting: A pairing-friendly setting $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G, \widetilde{G}, e)$ and a DLRS $\mathcal{R}$ in $\mathbb{G}_1$: for $A_1, \dots, A_w, V_1, \dots, V_r \in \mathbb{G}_1$, and $\mathcal{J}_1, \dots, \mathcal{J}_r \subseteq \{1, \dots, w\}$, the TPM knows variables $\alpha_1, \dots, \alpha_m \in \mathbb{Z}_p$ such that $V_i = \sum_{j\in\mathcal{J}_i}[\alpha_j]A_{v_{i,j}}$, for $i = 1, \dots, r$.
Players' inputs: The public input contains $G, \widetilde{G}, \{V_i\}_i, \{\mathcal{J}_i\}_i, \{A_j\}_j$; The TPM knows $\{\alpha_i\}_i$.

Fig. 3. Delegation of Proof of Knowledge of Witnesses for a DLRS (without additional information)

As for Theorems 4 and 5, the first theorem essentially shows that this proof does not leak any information outside the host, and the next one says that the host just learns $\{[\alpha_i]\widetilde{G}\}_i$, which is not enough to impersonate the TPM later.

Theorem 7. *The protocol in Figure 3 is a 3-move zero-knowledge proof of knowledge of the witnesses $\alpha_1, ..., \alpha_m$ between $\mathcal{P}$ (TPM) and $\mathcal{V}$ (host), where the auxiliary input contains the description of $\mathcal{R}$ and the additional values $\{[\alpha_i]\widetilde{G}\}_i$.*

5.3 Proofs of the Theorems

Completeness. The verifier checks, for $i = 1, \dots, r$, $e(H_i + \sum_{j\in\mathcal{J}_i}[s_j]A_{v_{i,j}} - [c]V_i, \widetilde{G}) = \prod_{j\in\mathcal{J}_i} e(Z_{i,j}, \widetilde{B}_{i,j})$, where

$$e\left(H_i + \sum_{j\in\mathcal{J}_i}[s_j]A_{v_{i,j}} - [c]V_i, \widetilde{G}\right) = e\left(H_i + \sum_{j\in\mathcal{J}_i}[k_j]A_{v_{i,j}}, \widetilde{G}\right)$$

and

$$\prod_{j\in\mathcal{J}_i} e(Z_{i,j}, \widetilde{B}_{i,j}) = \prod_{j\in\mathcal{J}_i} e(A_{v_{i,j}}, \widetilde{Z}_j + [t_{i,j}]\widetilde{G}) = \prod_{j\in\mathcal{J}_i} e(A_{v_{i,j}}, [k_j + t_{i,j}]\widetilde{G})$$
$$= e\left(\sum_{j\in\mathcal{J}_i}[t_{i,j}]A_{v_{i,j}} + \sum_{j\in\mathcal{J}_i}[k_j]A_{v_{i,j}}, \widetilde{G}\right) = e\left(H_i + \sum_{j\in\mathcal{J}_i}[k_j]A_{v_{i,j}}, \widetilde{G}\right).$$

Soundness. The proof is similar to the one in Section 4 since everything was on the left-hand side of the verification equation, that remains the same plus a constant H_i.

5.4 Zero-Knowledge w.r.t. the Host

The protocol between the TPM and the host is the same as the first protocol, and thus the security analysis is the same as in Section 4.

5.5 Zero-Knowledge w.r.t. the Verifier

As in Section 4, the verifier just knows the public parameters: $G, \widetilde{G}, \{A_j\}_j, \{V_i\}$. The simulator operates as follows:

- it first selects $c \xleftarrow{\$} \{0,1\}^\ell$ and $\{s_j\}_j \xleftarrow{\$} \mathbb{Z}_p$;
- it computes $K_i \leftarrow \sum_{j\in\mathcal{J}_i}[s_j]A_{v_{i,j}} - [c]V_i$, for all $i \in \{1,\ldots,r\}$;
- it additionally selects, for $i \in \{1,\ldots,r\}$ and $j \in \mathcal{J}_i$, $u_{i,j} \xleftarrow{\$} \mathbb{Z}_p^*$ and $U_{i,j} \xleftarrow{\$} \mathbb{G}_1\backslash\{0_{\mathbb{G}_1}\}$, with no constraint;
- it then computes, for $i \in \{1,\ldots,r\}$, $H_i = \sum_{j\in\mathcal{J}_i} U_{i,j} - K_i$ and for $j \in \mathcal{J}_i$, $Z_{i,j} = [u_{i,j}^{-1}]U_{i,j}$ and $\widetilde{B}_{i,j} = [u_{i,j}]\widetilde{G}$;
- it then outputs $\{H_i\}_i, \{Z_{i,j}, \widetilde{B}_{i,j}\}_{i,j}$, and waits for the challenge and rewinds in case of incorrect guess of c;
- it eventually answers $\{s_j\}_j$.

As in Section 4, a problem can occur with the above simulation if some elements gets zero while it is not allowed. But the large order of the groups makes this problem to happen with negligible probability only. We exclude these bad cases in the following analysis.

In a real protocol, the verifier sees: $\{H_i\}_i, \{Z_{i,j}, \widetilde{B}_{i,j}\}, c, \{s_j\}_j$, where $H_i = \sum_{j\in\mathcal{J}_i}[t_{i,j}]A_{v_{i,j}} = \sum_{j\in\mathcal{J}_i}[t_{i,j}a_{v_{i,j}}]G$, for random scalars $t_{i,j}$, $Z_{i,j} = [b_{i,j}^{-1}]A_{v_{i,j}} = [a_{v_{i,j}}/b_{i,j}]G$ for random non-zero scalars $b_{i,j}$, and $\widetilde{B}_{i,j} = [b_{i,j}](\widetilde{Z}_j + [t_{i,j}]\widetilde{A}_{i,j}) = [b_{i,j}/a_{v_{i,j}} \cdot (k_j a_{v_{i,j}} + t_{i,j}\prod_{k\in\mathcal{J}_i} a_{v_{i,k}})]\widetilde{G}$.

Let us denote $u'_{i,j} = (b_{i,j}/a_{v_{i,j}}) \cdot (k_j a_{v_{i,j}} + t_{i,j}\prod_{k\in\mathcal{J}_i} a_{v_{i,k}})$, for $i = 1,\ldots,r$ and $j \in \mathcal{J}_i$. Then $\widetilde{B}_{i,j} = [u'_{i,j}]\widetilde{G}$. Since the $b_{i,j}$'s are independent random scalars, the $u'_{i,j}$'s are also independent random scalars, and thus follow the same distribution as the $u_{i,j}$'s.

With such a notation and $d_i = \prod_{k\in\mathcal{J}_i} a_{v_{i,k}}$, we have $Z_{i,j} = [(u'_{i,j})^{-1}(k_j a_{v_{i,j}} + t_{i,j}d_i)]G$. Let us denote $U'_{i,j} = [k_j a_{v_{i,j}} + t_{i,j}d_i]G$. Since the $t_{i,j}$ are random scalars, then the $U'_{i,j}$'s are random elements in $\mathbb{G}_1$. Eventually,

$$\sum_{j\in\mathcal{J}_i} U'_{i,j} = [\sum_{j\in\mathcal{J}_i} k_j a_{v_{i,j}} + t_{i,j}d_i]G = K_i + \sum_{j\in\mathcal{J}_i}[t_{i,j}]A_{v_{i,j}} = K_i + H_i.$$

As a consequence, in the real protocol execution, for $i \in \{1,\ldots,r\}$, $H_i = \sum_{j\in\mathcal{J}_i} U_{i,j} - K_i$, and for $j \in \mathcal{J}_i$, $Z_{i,j} = [(u'_{i,j})^{-1}]U'_{i,j}$ and $\widetilde{B}_{i,j} = [u'_{i,j}]\widetilde{G}$, where the $u'_{i,j}$'s and $U'_{i,j}$'s follow the same distributions as the $u_{i,j}$'s and $U_{i,j}$'s generated by our simulator.

Acknowledgments

This work was supported in part by the French ANR-12-INSE-0014 SIMPATIC Project and in part by the European Commission through the FP7-ICT-2011-EU-Brazil Program under Contract 288349 SecFuNet.

References

1. Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer, May 2003.
2. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, November 1993.
3. David Bernhard, Georg Fuchsbauer, Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi. Anonymous attestation with user-controlled linkability. *Int. J. Inf. Sec.*, 12(3):219–249, 2013.
4. Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Get shorty via group signatures without encryption. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10: 7th International Conference on Security in Communication Networks*, volume 6280 of *Lecture Notes in Computer Science*, pages 381–398. Springer, September 2010.
5. Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 04: 11th Conference on Computer and Communications Security*, pages 132–145. ACM Press, October 2004.
6. Ernie Brickell, Liqun Chen, and Jiangtao Li. Simplified security notions of direct anonymous attestation and a concrete scheme from pairings. *Int. J. Inf. Sec.*, 8(5):315–330, 2009.
7. Sébastien Canard, Iwen Coisel, Giacomo de Meulenaer, and Olivier Pereira. Group signatures are suitable for constrained devices. In Kyung Hyune Rhee and DaeHun Nyang, editors, *ICISC 10: 13th International Conference on Information Security and Cryptology*, volume 6829 of *Lecture Notes in Computer Science*, pages 133–150. Springer, December 2010.

8. Sébastien Canard, Iwen Coisel, Julien Devigne, Cécilia Gallais, Thomas Peters, and Olivier Sanders. Toward generic method for server-aided cryptography. In Sihan Qing, Jianying Zhou, and Dongmei Liu, editors, *ICICS*, volume 8233 of *Lecture Notes in Computer Science*, pages 373–392. Springer, 2013.
9. Julien Cathalo, Benoît Libert, and Moti Yung. Group encryption: Non-interactive realization in the standard model. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 179–196. Springer, 2009.
10. Liqun Chen, Dan Page, and Nigel P. Smart. On the design and implementation of an efficient daa scheme. In Dieter Gollmann, Jean-Louis Lanet, and Julien Iguchi-Cartigny, editors, *CARDIS*, volume 6035 of *Lecture Notes in Computer Science*, pages 223–237. Springer, 2010.
11. Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In Phong Q. Nguyen, editor, *Progress in Cryptology - VIETCRYPT 06: 1st International Conference on Cryptology in Vietnam*, volume 4341 of *Lecture Notes in Computer Science*, pages 193–210. Springer, September 2006.
12. Uriel Feige, Amos Fiat, and Adi Shamir. Zero knowledge proofs of identity. In Alfred V. Aho, editor, *STOC*, pages 210–217. ACM, 1987.
13. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, August 1987.
14. Marc Girault. An identity-based identification scheme based on discrete logarithms modulo a composite number (rump session). In Ivan Damgård, editor, *Advances in Cryptology – EUROCRYPT'90*, volume 473 of *Lecture Notes in Computer Science*, pages 481–486. Springer, May 1990.
15. Marc Girault, Guillaume Poupard, and Jacques Stern. On the fly authentication and signature schemes based on groups of unknown order. *Journal of Cryptology*, 19(4):463–487, October 2006.
16. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In Robert Sedgewick, editor, *STOC*, pages 291–304. ACM, 1985.
17. Jens Groth. Fully anonymous group signatures without random oracles. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 164–180. Springer, December 2007.
18. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, April 2008.
19. Dennis Hofheinz and Eike Kiltz. Secure hybrid encryption from weakened key encapsulation. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 553–571. Springer, August 2007.
20. Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Traceable signatures. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 571–589. Springer, May 2004.
21. Benoît Libert, Thomas Peters, and Moti Yung. Group signatures with almost-for-free revocation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 571–589. Springer, August 2012.
22. Toru Nakanishi, Hiroki Fujii, Yuta Hira, and Nobuo Funabiki. Revocable group signature schemes with constant costs for signing and verifying. In Stanislaw Jarecki and Gene Tsudik, editors, *Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 463–480. Springer, 2009.
23. Kazuma Ohara, Yusuke Sakai, Keita Emura, and Goichiro Hanaoka. A group signature scheme with unbounded message-dependent opening. In Kefei Chen, Qi Xie, Weidong Qiu, Ninghui Li, and Wen-Guey Tzeng, editors, *ASIACCS*, pages 517–522. ACM, 2013.
24. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
25. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, August 1990.
26. Hovav Shacham. A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. *IACR Cryptology ePrint Archive*, 2007:74, 2007.