Proofs of Knowledge for Non-Monotone Discrete-Log Formulae and Applications

Emmanuel Bresson and Jacques Stern

Ecole normale suprieure,
45 rue d'Ulm, 75230, Paris, France
E-mail: {Emmanuel.Bresson, Jacques.Stern}@ens.fr.

Abstract. This paper addresses the problem of defining and providing proofs of knowledge for a general class of exponentiation-based formulae. We consider general predicates built from modular exponentiations of secret values, combined by products and connected with the logical operators "AND", "OR", "NOT". We first show how to deal with non-linear combination of secret exponents. Next,we extend the work by Brands [4] to a strictly larger class of predicates, allowing a more liberal use of the logical operator "NOT". We sketch two applications by which we enhance group signatures schemes with revocation of identity and multi-signer features. Such features can be useful to protect privacy or for collabrative use of group signatures, respectively.

1 Introduction

1.1 Proof of Knowledge

Zero-knowledge has been introduced by Goldwasser, Micali and Rackoff in [23] to quantify the amount of information leaked in an interactive protocol. The (interactive) protocols are thus proved zero-knowledge when they reveal no information apart from the validity of the statement. Almost at the same time, the concept of *proof of knowledge* [21] introduced the notion of extractor for a secret and became a building block in public-key cryptography. The property of zero-knowledge is useful as soon as one wants to perform some operations with secret values without revealing them. Classical examples are authentication, identification [5, 20, 21], digital signatures [21] and group signatures [10, 18].

From a more general point of view, the idea of satisfying boolean statements (*predicates*) without leaking any information has been first introduced by Chaum *et al.* [13, 15, 22]. Numerous schemes [14] allow to combine several proofs to prove more elaborated statements about discrete logarithms; the very first only covered the case of a single equations connected by "AND" statement. In 1994, De Santis *et al.* [25] and Cramer *et al.* [19] independently discovered a general method to deal with the "OR" connective. Using their method, one can design proof systems for monotone formulae (i.e. statements without negations). An application to group signature was made in an earlier paper [18] that mentioned [19]. Both papers were based on a protocol proposed by Schoenmakers in [26]. Later, Camenisch and Stadler [11, 7] introduced a formal model for building and proving general linear relations about discrete logarithms, and combining them by the logical operators "OR" and "AND".

In 1997, Brands [4] described a general method allowing to prove *any* boolean combination of linear or affine relations on secrets exponents, including relations obtained by "NOT" operator. Note that, strictly speaking, this method does not encompass the entire class of boolean formulae obtained by connecting exponentiations of elements since one may wish to state that an exponential term differs from an element whose discrete logarithm is *unknown*. This will be precisely where the core of our paper can be found.

1.2 Contributions

Our contribution is on two points. We first explain how to prove non-linear relations on secret values and give as an example a protocol to *efficiently* prove knowledge of roots of discrete logarithms.

Next we define zero-knowledge proofs for statements involving negations (i.e. operator "NOT"), in a framework which is strictly larger than the one used in [4]. We explain why in some situations that we believe to be practical, the scheme proposed in [4] is not sufficient. As a concrete application of this particular extension, we give two examples related to group signatures. The first formalizes the work of [6, 27] to perform member revocation while the second introduces multi-signer features in that context.

2 A General Class of Exponentiation-Based Formulae

We describe here a formal class of *predicates*, seen as boolean statements built from modular exponentiations of secret values, combined by products and connected with the logical operators "AND", "OR", "NOT".

2.1 Preliminaries

We follow the notation of [4, 3]. We denote a polynomial-time prover by $\mathcal{P}$ and a possibly unlimited verifier by $\mathcal{V}$. The notation $x \in_R S$ means that x is chosen in the set S randomly; we assume uniform distribution, unless otherwise specified. Finally, we fix a cyclic group G of prime-order q (e.g. a subgroup of $\mathbb{Z}_p^*$, where p is a large prime and $q|p-1$). Note that q is made public. For any polynomial integer m, for any element y and generators $g_1, \ldots, g_m$ in G, we say that $(x_1, \ldots, x_m) \in \mathbb{Z}_q^m$ is a *representation* of y with respect to the g_i's if $y = \prod_{i=1}^m g_i^{x_i}$. Without loss of generality, we assume that the generators appearing in such a product are different.

The following lemma is a critical part of our construction.

Lemma 1 (see [17]). *Under the discrete logarithm assumption, it holds that no probabilistic polynomial-time algorithm, on input q and a randomly chosen, polynomial-sized tuple of generators $(g_1, \ldots, g_m)$, can output with non-negligible probability an element $h \in G$ and two different representations of h with respect to some of the g_i's.*

2.2 Knowledge of representations

In the remaining of this paper, we consider a polynomial-sized family of generators $g_1, \ldots, g_m$ of G, whose relative discrete logarithms are unknown. An *atomic representation-based predicate* relatively to some variables $(x_1, \ldots, x_k)$ is build from a representation of a public value y w.r.t. a subset of the basis $g_1, \ldots, g_m$:

$$(T) \qquad : \qquad y = \prod_{l \in [1,k], j \in S} g_j^{x_l} \qquad \text{where } S \subseteq [1,m], \#(S) = k.$$

To avoid using two different indices for the bases and the variables, we consider an appropriate injective function π from $[1,k]$ onto $[1,m]$ such that the predicate can be re-written as follows:

$$(T) \qquad : \qquad y = \prod_{l \in [1,k]} g_{\pi(l)}^{x_l}$$

Depending on the values of $x_1, \ldots, x_k$, the above equation may be satisfied or not, and the predicate is seen as *true* or *false* for this set of values. Proving such atomic representation predicate means proving knowledge of values x_i such that the underlying equality holds. Camenish and Stadler [11] formalize this notion through *knowledge specification* and *knowledge specification set*.

We assume that $\mathcal{P}$ holds a set of values satisfying (T). Using standard zero-knowledge interactive protocol, $\mathcal{P}$ can prove knowledge of such private values $(x_1, \ldots, x_k)$. We denote such a proof:

$$ZKPK\big[\sigma_i \ : \ T(\sigma_1, \ldots, \sigma_k)\big] = ZKPK\big[\sigma_i \ : \ y = \prod_{l \in [1,k]} g_{\pi(l)}^{\sigma_l}\big]$$

which means "some secrets σ_i are known that make T *true*".

Proofs for Monotone Formulae. The prover is able to prove knowledge of the conjunction of two predicates by running a "multiple proof" in which it commits to several values using the same random number, receives a unique challenge, and gives a set of values as an answer. A predicate containing only "AND" connectives is referred to as a *conjunctive representation predicate* or CR-predicate on a set of variables $(\sigma_1, \ldots, \sigma_v)$:

$$T(\sigma_1, \ldots, \sigma_v) \qquad : \qquad \bigwedge_{i=1}^{k} \left(y_i = \prod_{l \in S_i} g_{\pi_i(l)}^{\sigma_l} \right)$$

where for each $i \in [1, k]$, S_i is a subset of $[1, v]$ of cardinality at most m and π_i is an injective function from S_i onto $[1, m]$.

Also the technique by Cramer *et al.* [19] can be used to prove knowledge for a disjunction of atomic predicates. This technique is now classical and consists in running a simulator on the predicates for which the secrets are unknown, and in completing the proof on *known* secrets by an adequate "share" of the given challenge. Such extensions appear in the literature and details are omitted here. The reader can refer to [10]. Thus, we can generically obtain proofs for any *disjunctive-conjunctive representation predicate* (DCR-predicate) on variables $(\sigma_1, \ldots, \sigma_{v_i})_{1 \le i \le n}$ written as:

$$(T) \qquad : \qquad \bigvee_{i=1}^{n} \bigwedge_{j=1}^{k_i} \left(y_{ij} = \prod_{l \in S_{ij}} g_{\pi_{ij}(l)}^{\sigma_l} \right)$$

where for each $i \in [1, n]$ and each $j \in [1, k_i]$, S_{ij} is a subset of $[1, v_i]$ of cardinality at most m and π_{ij} is an injective function from S_{ij} onto $[1, m]$.

Remark 1. DCR-predicates capture as a particular case proofs of knowledge of discrete logarithms or representations, as well as equality of discrete logs ("AND") or knowledge of one discrete log out of n (disjunctive predicate).

They also can be used to demonstrate *roots of representation* with the technique explained in [10]. However, in section 4, we show that a much more efficient technique can be used to demonstrate such statements (*logarithmic* size instead of *linear* size).

2.3 Case where $\#(G)$ is unknown

When $\#(G)$ is not public, the answers cannot be reduced modulo q anymore, and zero-knowledge is only statistical. We take the example of $G = Q_N$, where Q_N is the set of quadratic residues modulo N. If N is the product of two safe primes $p = 2p' + 1$ and $q = 2q' + 1$, then Q_N is a cyclic group, and a random element is a generator with overwhelming probability, as shown in the following lemma:

Lemma 2. *Let p, q, p', q' be four prime numbers such that $p = 2p' + 1$ and $q = 2q' + 1$ and let $N = pq$. In that case, if an element $w \in Q_N^* \backslash \{1\}$ has order $ord(w) < p'q'$, then either* $\gcd(w - 1, N)$ *or* $\gcd(w + 1, N)$ *is a non-trivial factor of N.*

3 Case of Monotone Formulae

3.1 Proving Linear Relations

We first recall known results on demonstrating linear relations between secret values using representation-based predicates. Following Brands formalization [4], we consider a prover $\mathcal{P}$ having knowledge of some secret values $x_1, \ldots, x_k$. The following result shows how a linear relation between the x_i's can be efficiently proven without revealing any additional information.

We assume $\mathcal{P}$ has published $h = g_1^{x_1} g_2^{x_2}$ as its public key. Let a, b be some public values in $\mathbb{Z}_q$ and assume that $\mathcal{P}$ wants to prove knowledge of two secrets σ_1, σ_2 such that the following predicate (T) : $h = g_1^{\sigma_1} g_2^{\sigma_2} \wedge (b = a\sigma_1 + \sigma_2)$ is true.

It can be shown [4] that under lemma 1, is is equivalent for $\mathcal{P}$ to prove knowledge of σ_1 and σ_2 satisfying (T) or to prove knowledge of the discrete logarithm of hg_2^{-b} to the base $g_1 g_2^{-a}$. To see that, let s be this discrete logarithm. From the definition of s, we easily get $h = g_1^s g_2^{b-as}$. According to lemma 1, $\mathcal{P}$ can know at most one representation of h to the bases g_1 and g_2. Thus we have $x_1 = s$ and $x_2 = b - as$. Hence, predicate (T) is true.

Also it must be noted that $g_1 g_2^{-a}$ is generator of G with overwhelming probability; we assume that it is always the case (for instance if a is chosen by the verifier).

These proofs can then be combined using standard techniques described in the previous section. We refer the reader to [4, 11] for more details.

3.2 Extension to Quadratic Relations

We now deal with the method for proving non-linear relations. More precisely, in this section, we describe how to prove that a secret value is the product of two other secret values (or the square of another secret). Although such a technique is implicitely used in many papers, we put forward here the very conditions needed for a safe use.

Let T be a conjunctive representation predicate as defined in section 2: T : $\bigwedge_{i=1}^{n} y_i = \prod_l g_{\pi_i(l)}^{x_l}$ and consider the following, new predicate:

$$T' \quad : \quad (T) \wedge \left(x_i x_j = x_k \bmod q\right)$$

This is not a representation predicate anymore. However, we claim that $\mathcal{P}$ can still prove the validity of (T') when knowing the secrets x_l by using the following technique:

Protocol 1.

---

1. $\mathcal{V}$ sends to $\mathcal{P}$ a random generator h of G.
2. $\mathcal{P}$ chooses an integer $r \in_R \mathbb{Z}_q^*$ and sends to $\mathcal{V}$: $y' = g^{x_i} h^r$.
3. $\mathcal{P}$ performs the following proof of knowledge:

$$ZKPK\left[\sigma_1, \ldots, \sigma_k, \rho, \tau \;:\; (T) \wedge \left(y' = g^{\sigma_i} h^{\rho}\right) \wedge \left({y'}^{\sigma_j} = g^{\sigma_k} h^{\tau}\right)\right] \qquad (1)$$

We underline that it is of prime importance that $\mathcal{P}$ has no control over g and h, and that g, h are generators of G with overwhelming probability.

This protocol allows to add any non-linear relation to an arbitrary conjunctive predicate, as stated in the following theorem:

Theorem 1. *Let* (T) *be a conjunctive predicate on the variables* $(\sigma_1, \ldots, \sigma_k)$. *Then under lemma 1,* **Protocol 1** *is a zero-knowledge proof of knowledge for the predicate* $(T') := (T) \wedge \big(x_i x_j = x_k \bmod q\big)$.

Proof. We prove the following three properties.

Completeness. The verification is straightforward.

Soundness. The soundness property is computational. Let $\mathcal{E}$ be a knowledge extractor for the proof (1). We show that $\mathcal{E}$ is also a knowledge extractor for predicate (T'), provided lemma 1 holds.

Let $(z_1, \ldots, z_k, r, t)$ the secrets extracted by $\mathcal{E}$. If we raise the equation $y' = g^{z_i} h^r$ to the power of z_j, we get:

$$y'^{z_j} = g^{z_i z_j} h^{r z_j} = g^{z_k} h^t$$

Now, if g and h are randomly chosen by the verifier, lemma 1 applies and we have with overwhelming probability $z_i z_j = z_k \bmod q$. That is, returning $(z_1, \ldots, z_k)$ makes an extractor for predicate (T').

Zero-Knowledge. It is easy to see that if r is uniformly distributed over $\mathbb{Z}_q$, sending y' to the verifier does not reveal any helpful information. Also, the proof performed in the third step of the protocol does not reveal anything either. □

Corollary 1. *When running* **Protocol 1** *with* $i = j$, *one can prove that a secret exponent is the square of another secret exponent.*

This simple remark allow us to build new, efficient scheme for proving knowledge of more complex statements, including polynomial relations, as explained in the next section.

4 Application to Efficient Protocols

While proving knowledge of discrete logarithms or representation is feasible in efficient ways by classical means, proving the knowledge of a root of a discrete logarithm is much more difficult. Two ideas have been proposed so far, which remain quite inefficient or constrained to particular values.

4.1 A Generic Bit-by-bit Solution [28]

M. Stadler proposed in [28] a bit-by-bit protocol which allows to prove knowledge of the e-th root of the discrete logarithm of an element y, relatively to a base element g. That is, given $y, g \in G$ and $e > 2$, one proves knowledge of x such that $y = g^{x^e}$.

This can be done as follows. Let ℓ denote a security parameter.

Protocol 2.

---

1. $\mathcal{P}$ chooses $r_1, \ldots, r_\ell \in_R \mathbb{Z}_n^*$ and sends $t_i = g^{r_i^e}$ to $\mathcal{V}$, for $i = 1, \ldots, \ell$..
2. $\mathcal{V}$ sends a random ℓ-bit challenge c to $\mathcal{P}$.

3. $\mathcal{P}$ answers with $(s_1, \ldots, s_\ell)$ where $s_i = r_i$ if $c[i] = 0$ and $s_i = r_i/x$ otherwise.
4. $\mathcal{V}$ accepts if for all $i = 1, \ldots, \ell \ : \ t_i = \begin{cases} g^{s_i^e} \text{ if } c[i] = 0, \\ y^{s_i^e} \text{ otherwise.} \end{cases}$

4.2 An Alternative Solution for Small e only [10]

In [10], another solution acceptable for small values of e appears. The resulting proof being linear in e, the usefulness is very limited in practice. One cannot use it for $e > 160$, for instance.

Let consider another generator h of group G such that the discrete logarithm of h to the base g is unknown. The basic idea is to prove a conjunctive predicate made of some statements like $y_i = g^{\sigma_i} \wedge (\sigma_i = x \times \sigma_{i-1})$, and $y_0 = g$ such that one recursively gets $\sigma_i = x^i$. However, such a solution reveals partial information about x, namely all the intermediate values g^{x^i}. The solution consists in "blinding" these intermediate values using another generator h whose discrete logarithm to the base g is unknown (this can be done by having $\mathcal{V}$ choosing h).

More precisely, $\mathcal{P}$ performs the following two steps. In the first step, $\mathcal{P}$ computes $e-1$ random values y_1 through y_{e-1} as $y_i = h^{r_i} g^{x^i}$ where r_i is a random secret exponent. $\mathcal{P}$ sends these values to $\mathcal{V}$. In the second step, $\mathcal{P}$ proves the following representation predicate:

$$ZKPK\Big[\sigma_1, \ldots, \sigma_e, \rho : y_1 = h^{\sigma_1} g^{\rho} \quad \wedge \quad y_2 = h^{\sigma_2} y_1^{\rho} \quad \wedge \quad \ldots \\ \wedge \quad y_{e-1} = h^{\sigma_{e-1}} y_{e-2}^{\rho} \quad \wedge \quad y = h^{\sigma_e} y_{e-1}^{\rho}\Big]$$

According to this proof, and independently of lemma 1, y is actually of the form $h^{\tau} g^{\rho^e}$, where ρ and $\tau = \sigma_e + \cdots + \sigma_2 \rho^{e-2} + \sigma_1 \rho^{e-1}$ are proven to be known by $\mathcal{P}$. Also, according to lemma 1, $\mathcal{P}$ can know at most one representation of y to the bases g and h. It follows that: $\tau \equiv \rho^e \bmod q$, which means that ρ satisfies $y = g^{\rho^e}$. It is easy to see that no information is leaked by the values y_i's if the exponent r_i's are randomly and independently chosen.

4.3 Our Solution in $\mathcal{O}(\log e)$ Size

We now describe a method that can be used for larger values of e. Let ℓ be a security parameter. Our protocol can be used for any $e < 2^{\ell/2}$ and leads to proof of size $\mathcal{O}(\log e)$. Such improvment has been also suggested in [7, p. 64]. In this section, we give the first formal protocol together with a proof of security. Typically, for $\ell = 160$ our protocol outperforms both Camenish-Stadler protocol [10] and Stadler protocol [28].

The main idea is as follows. First when proving both $A = g^{\sigma}$ and $A^{\rho} = g^{\tau}$, for some values A and g, one proves that $\tau = \sigma\rho$. That is, we prove that a secret exponent is the product of two other secret exponents. Second, we note that if

$e = 2^k + 2^i + 2^j + \cdots$, one can write $x^e = x^{2^k} x^{2^i} x^{2^j} \cdots$. It follows that a statement like $y = g^{x^e}$ can be written as

$$y = g^{x^{2^k} x^{2^i} x^{2^j} \cdots}$$

Hence using the binary representation of e (which is $\mathcal{O}(\log e)$ long), we obtain a new form of the statement g^{x^e} wherein x^e is the product of some exponents. Let us now go into details.

Let k be $\lfloor \log_2 e \rfloor$ so that $e = \sum_{i=0}^{k} e[i] 2^{k-i}$, where $e[i]$ is the i-th most significant bit of e. We denote by e_j the integer made of the $j+1$ most significant bits of e, that is $\sum_{i=0}^{j} e[i] 2^{j-i}$. Note that we have $e_k = e$ and $e_0 = e[0] = 1$. Finally, $<e>$ denote the Hamming weight of e, that is $\sum_{i=0}^{k} e[i]$.

Now we perform a sequence of operations reflecting a square-and-multiply exponentiation.

Protocol 3.

---

1. $\mathcal{V}$ chooses at random a generator h of G and sends it to $\mathcal{P}$.
2. $\mathcal{P}$ chooses at random $k+1$ secret exponents r_0 through r_k, as well as $<e> -1$ random exponents s_i for all $i \in [1, k]$ such that $e[i] = 1$. Then the following values are made public:

$$\begin{array}{ll} v_i = g^{x^{e_i}} h^{r_i} & \text{for all } i \in [0.k] \\ w_i = g^{x^{2e_{i-1}}} h^{s_i} & \text{for all } i \in [1, k] \text{ such that } e[i] = 1 \end{array}$$

3. $\mathcal{P}$ performs the following proof of knowledge:

$$\begin{aligned} ZKPK\Big[& \sigma_0, \ldots, \sigma_k, \rho_0, \ldots, \rho_k \ : \ v_0 = g^{\sigma_0} h^{\rho_0} \ \wedge \\ & \Big(\bigwedge_{i=1}^{k} \left(v_i = g^{\sigma_i} h^{\rho_i} \wedge (\sigma_i = \sigma_0^{e[i]} \sigma_{i-1}^2) \right) \Big) \wedge y = g^{\sigma_k} \Big] \end{aligned} \quad (2)$$

Note that according to Theorem 1, a statement of the form $\sigma_i = \sigma_0^{e[i]} \sigma_{i-1}^2$ can be demonstrated by proving an additional representation of the form $v_i = v_{i-1}^{\sigma_{i-1}} h^{t_i}$ in case $e[i] = 0$ and two additional representations of the form $v_i = w_{i-1}^{\sigma_0} h^{t_i} \ \wedge \ w_{i-1} = v_{i-1}^{\sigma_{i-1}} h^{t'_i}$ in case $e[i] = 1$.

Theorem 2. *Let g be a generator of G and e be a $(k+1)$-bit integer of Hamming weight $< e >$. Then under lemma 1,* **Protocol 3** *is a zero-knowledge proof of knowledge of the e-th root of the discrete logarithm of y to the base g.*

Proof. We have to show the following three properties.

Completeness. Consider $\mathcal{P}$ who knows all values $x, \{r_i\}, \{s_j\}$. An easy verification ensures that for all $i \in [0, k]$, $\sigma_i = x^{e_i}$ and $\rho_i = r_i$ form a correct set of secrets for the protocol, since we have $\sigma_i = x^{e_i} = x^{e[i] + 2e_{i-1}} = x^{e[i]} (x^{e_{i-1}})^2 = \sigma_0^{e[i]} \sigma_{i-1}^2$.

Soundness. The soundness property is computational. If h is randomly chosen by the verifier, lemma 1 applies and we can consider a knowledge extractor $\mathcal{E}$ for the proof (2). Let $(z_0, \ldots, z_k, r_0, \ldots, r_k)$ the secrets extracted by $\mathcal{E}$. One can easily check that for any $i \in [1, k]$ one has: $z_i = z_0^{e_i}$ and in particular $z_k = z_0^e$. Therefore, $y = g^{z_0^e}$, that is, z_0 is the e-th root of the discrete logarithm of y to the base g.

Zero-Knowledge. It is easy to see that if the $\{r_i\}$ and $\{s_j\}$ are uniformly distributed over $\mathbb{Z}_q$, no information is leaked when revealing $v_0, \ldots, v_k$ and the w_i's. Also the proof performed in the third step of the protocol is zero-knowledge and thus does not reveal anything either. $\square$

5 How to Deny a Predicate

5.1 Motivation

We now turn to the second contribution of our work. In this section, we explain how a prover can convince a verifier that he knows some secret values $(x_1, \ldots, x_k)$ which verify some equalities but do not verify other ones. Generalization to negate several predicates is easy. The technique can be useful in several situations:

1. A user wants to prove that he did not use a particular secret or random value in some signature or encryption he produced.
2. The prover has committed to some values x_i and wants to prove $\prod_i g_i^{x_i} \neq Y$ for a public value Y (whose representation is not known).

The protocols proposed by Brands [4] can solve the first example, but not the second one. Briefly speaking, these protocols allow a prover $\mathcal{P}$ holding a secret x *namely a discrete logarithm* to convince any verifier $\mathcal{V}$ that x differs from another *known* discrete logarithm. The main (and indeed, critical) point of these protocols, is that the prover must know these two secrets in order to perform the interaction.

What if the prover wants to demonstrate a statement like: "I know the discrete log of y to the base g, and it differs from the (unknown) discrete log of z to the base h" ? Consider the situation where Alice's public key is $Y_A = g^{x_A}$ and Bob's public key is $Y_B = h^{x_B}$. Alice knows x_A but not x_B; at the same time, she can easily verify that $x_A \neq x_B$ by computing h^{x_A}, which then must differ from Y_B. This is not achievable by Brands' protocols in which one needs to know both x_A and x_B at the same time.

That's why we claim that, even for linear relations, Brands' method only applies to a restricted class of non-monotone predicates. Indeed, Brands consider only predicates where relations hold in $\mathbb{Z}_q$. Our consider the case of relations between data obtained by exponentiating these secrets, namely, relations holding in G. This clearly leads to a larger class of formulae.

The following theorem illustrate the previously known result:

Proposition 1 (Brands, [4]). *We denote by (x_1, x_2) the set of secrets known to $\mathcal{P}$ and by $h \in G$ the (public) product $g_1^{x_1} g_2^{x_2}$. Let $T(\sigma_1, \sigma_2)$ be the predicate: $h = g_1^{\sigma_1} g_2^{\sigma_2}$ and a, b be some public values in $\mathbb{Z}_q$.*

Then, under lemma 1, $\mathcal{P}$ can prove the predicate $T'(\sigma_1, \sigma_2) = T(\sigma_1, \sigma_2) \wedge (\sigma_1 \neq a + b\sigma_2)$ if and only if it is able to prove knowledge of a representation of g_1 with respect to $g_1^a h^{-1}$ and $g_1^b g_2$.

The sketch of the proof consists in considering a knowledge extractor for the representation of g_1 and to use lemma 1 to identify these representations with the trivial representation of g_1.

5.2 Equivalent Formulation of a Negation

We are based on the fact that denying an atomic predicate is equivalent to proving a conjunction of representation predicates, relative to some well-chosen public parameters. This is also the underlying idea in [4]. But our protocol uses additional values, which allow "blinding" techniques. Some examples of this technique can be found in [12, 6].

A Basic Situation. We consider Alice holding public key $Y_A = g^{x_A}$. Bob's public key is Y_B, but the corresponding secret key $X_B = \log_h Y_B$ is, of course, not known. If Alice wants to prove that she is actually Alice and not Bob, she runs the following protocol.

Protocol 4.

---

1. $\mathcal{P}$ chooses an exponent $r \in_R \mathbb{Z}_q^*$ and gives to the verifier $w = (h^{x_A}/Y_B)^r$.
2. The verifier $\mathcal{V}$ checks that $w \neq 1$.
3. $\mathcal{P}$ proves the following predicate:

$$ZKPK[\sigma, \rho, \tau \quad : \quad (Y_A = g^\sigma) \ \wedge \ (w = h^\tau / Y_B^\rho) \ \wedge \ (\tau = \sigma\rho)] \qquad (3)$$

Theorem 3. *The above protocol is a zero-knowledge proof of knowledge for the predicate $T(\sigma) \ : \ Y_A = g^\sigma \wedge (\sigma \neq \log_h Y_B)$, provided that lemma 1 holds.*

Proof. **Completeness.** Verification is straightforward.

Soundness. The soundness property is computational. Since h and Y_B are not chosen by $\mathcal{P}$, lemma 1 applies and we can consider a knowledge extractor $\mathcal{E}$ for the proof (3). Let (s, r, t) the secrets extracted by $\mathcal{E}$. According to (3), we have $t = sr$ which leads to $w = h^t Y_B^{-r} = (h^s/Y_B)^r$. Consequently, the fact that $w \neq 1$ implies $h^s \neq Y_B$, that is $s \neq \log_h Y_B$. Hence, returning s makes an extractor for predicate $Y_A = g^\sigma \wedge (\sigma \neq \log_h Y_B)$.

Zero-Knowledge. Since h^{x_A}/Y_B is a generator with overwhelming probability (in fact, this is always the case if q is a prime), w is completely random over $G \backslash \{1\}$ if r is uniformly distributed over $\mathbb{Z}_q^*$. It follows that neither w nor the zero-knowledge proof performed in the third step does reveal helpful information. □

General Linear Relations. We now consider $\mathcal{P}$ holding $(x_1, \ldots, x_k)$ and whose public key is $h = \prod_1^k g_i^{x_i}$, where π is an adequate permutation. Let $(\alpha_0, \ldots, \alpha_k)$ some fixed coefficients, Y be a public element in G (whose no representation is known) and (T) be the following predicate:

$$\neg(\log_g Y = \alpha_0 + \alpha_1 x_1 + \cdots + \alpha_k x_k \bmod q)$$

Note the validity of such predicate can clearly be checked by $\mathcal{P}$ by raising g to the power of the above linear combination. Now assume $\mathcal{P}$ wants to convince $\mathcal{V}$ of the validity of the statement without revealing the x_i's. To that goal, $\mathcal{P}$ runs the following protocol:

Protocol 5.

---

1. $\mathcal{P}$ chooses $r \in_R \mathbb{Z}_q^*$ and sends $w = (g^{\alpha_0+\alpha_1 x_1+\cdots+\alpha_k x_k}/Y)^r$ to $\mathcal{V}$.
2. The verifier $\mathcal{V}$ checks that $w \neq 1$.
3. $\mathcal{P}$ proves knowledge for the following predicate on variables $\rho, \sigma, \tau, \omega_1, \ldots, \omega_k$:

$$ZKPK\Big[\rho, \sigma, \tau, \omega_i \ : \ \Big(hg^{\alpha_0} = g^{\sigma} \prod_{i=1}^{k} \left(g^{-\alpha_i} g_i\right)^{\omega_i}\Big) \wedge \Big(w = g^{\tau}/Y^{\rho}\Big) \wedge \big(\tau = \rho\sigma\big)\Big]$$

Corollary 2. *The above* **Protocol 5** *is a proof of knowledge for the predicate* $T(\omega_i) \ : \ h = \prod(g_i^{\omega_i}) \wedge (\alpha_0 + \alpha_1\omega_1 + \cdots + \alpha_k\omega_k \neq \log_g Y \bmod q)$*, provided that lemma 1 holds.*

Proof. **Completeness.** Verification is straightforward.

Soundness. The soundness property is computational. Since g and Y are not chosen by $\mathcal{P}$, lemma 1 applies and we can consider a knowledge extractor $\mathcal{E}$ for the above proof in step 3. Let $(s, r, t, u_1, \ldots, u_k)$ the secrets extracted by $\mathcal{E}$. Since $\mathcal{P}$ cannot know two different representations of h with respect to $g_1, \ldots, g_k$, we have with overwhelming probability $s = \alpha_0 + \sum_1^k \alpha_i u_i$. Also the second statement and the third one lead to $w = g^{rs}/Y^r = (g^s/Y)^r$. From $w \neq 1$, it follows that $s = \alpha_0 + \sum_1^k \alpha_i u_i \neq \log_g Y$. Hence, returning $u_1, \ldots, u_k$ makes an extractor for predicate $h = \prod(g_i^{\omega_i}) \wedge (\alpha_0 + \alpha_1\omega_1 + \cdots + \alpha_k\omega_k \neq \log_g Y)$.

Zero-Knowledge. Since $g^{\alpha_0+\cdots+\alpha_k x_k}/Y$ is a generator of the cyclic group G with overwhelming probability, w is completely random over $G\backslash\{1\}$ if r is uniformly distributed over $\mathbb{Z}_q^*$. It follows that neither w nor the zero-knowledge proof performed in the third step does reveal helpful information. □

6 Applications

In this section, we show some applications of our technique to classical group signature schemes. We emphasize that our solutions are generic, and we define a class of group signature schemes for which we can get some additional properties. Such schemes are derived from some general schemes for large groups originally proposed in 1997 by Camenish and Stadler [10].

6.1 Group signatures

The concept of group signature, although extremely useful, appeared relatively recently in cryptography [16, 18]. It allows a member of a group to sign documents anonymously on behalf of the group, in an unlinkable but publicly verifiable way. As a feature, a group signature scheme considers a *group leader*, also called *group manager*. The group manager deals with membership, and is allowed to "open" a group signature in order to reveal the identity of the actual signer. This can be necessary in case of legal dispute. However, he is not a trusted party and the security of a group signature scheme must consider attacks involving him. To provide separability, we can consider two different entities, a *membership manager* and a *revocation manager*. The latter should be needed only to open signatures. Separability is considered in [9, 24]. However, in this paper, and to avoid confusion in the context of member exclusion, we call him the *judge* rather than the revocation manager.

A group signature scheme consists of the following five procedures:

- KEY GENERATION ($\mathcal{KG}$) : provides every player with its initial data.
- JOIN ($\mathcal{J}$) : adds a member to a group.
- SIGN ($\mathcal{S}$) allows a member to sign messages.
- VERIFY ($\mathcal{V}$) : checks validity of a group signature.
- OPEN (Ω) : allows an authority to reveal the identity of a signer.

Requirements We briefly recall the desirable security requirements for a group signature scheme. Additionally to the usual properties of correctness (i.e., a correctly generated signature is always accepted by the verification algorithm), a group signature scheme must ensure that signatures are anonymous and unlinkable: it must be computationally infeasible to find the identity of the signer within the group. Moreover, deciding whether two signatures have been issued by the same member has to be hard. Also the scheme must ensure that opening is always possible, without cheating (i.e. the judge, when revealing the signer's identity, cannot falsely accuse an honest member). The last property is coalition resistance, which have been proven for the first time in the recent scheme [1]. Coalition resistance means that a subset of dishonest members cannot generate a valid group signature which, in case of opening, would reveal an honest (non-colluding) member as the signer.

6.2 A Class of Group signature schemes

The class of group signatures that we define contains the most recent and efficient schemes [1, 8, 10]. It is characterized by the following criteria:

- Computations are made in a group $G = \langle g \rangle$ in which the discrete logarithm problem is hard.
- The *judge* holds an ElGamal public key $y = g^w$ which is an element of G. The secret key is w.

– When registering, each member chooses a secret *membership key* x and receives a *membership certificate* for which demonstration of the corresponding secret key can be done using disjunctive-conjunctive representation predicate as defined in section 2.
– A group signature on a message m consists of a verifiable ElGamal encryption, under the judge's public key, of the public membership key, as well as an *anonymous* ZK proof of knowledge for the membership certificate and the corresponding secret key. It means that the proof can be verified using only the group public key, and not a particular member's key. We generically denote by $\mathcal{T}$ the underlying predicate contained in the zero-knowledge proof, and by (A, B) the ElGamal encryption containing the signer's identity.

Definition 1 (Camenish-Stadler -derived group signatures). We call a CS-Group Signature Scheme any group signature scheme that uses algorithms matching the above features.

6.3 Member Revocation

In this section, we illustrate the use of our technique to achieve member revocation in a group signature scheme. Revocation of members consists in preventing some officially excluded members to sign document on behalf of the group (for instance, after some abusive use). The general and most delicate problem encountered is how to preserve anonymity of *past signatures* for a revoked member? For instance, if one reveals its secret key, its previously issued signatures might be linked, which is undesirable.

We provided the first solution to the member revocation problem in [6]. We explain below how it is related to our general technique. Later, Song [27] proposed a slightly different solution; the resulting scheme is more efficient (in constant size rather than in linear size), however, it *implicitly* assumes that the registration of a new member is done using an authentic, private channel between the membership manager and the joining member[1]; nevertheless such an assumption follows [1]. Note that this is not necessarily the case in [6], where the revocation mechanism is done via Zero-Knowledge proofs, and was originally designed for [10]. In this section, we propose a more abstract and general framework which allows such member revocation for all efficient recent schemes [1, 8, 10].

6.4 Our method

We can use the technique of section 5 to build a revocation feature for any CS-group signature scheme. The protocol described in section 5 can be used to demonstrate that a given fair ElGamal encryption $(A, B) = (g^r, Y_A y^r)$ does

[1] If it is not the case, any signature can be opened in a linear time (in the number of members).

not contain a fixed plaintext Y_B. To do so, it is sufficient to prove the following predicate:

$$ZKPK\Big[\rho \ : \ \rho = \log_y(B/Y) \neq \log_g A]$$

This yields to a simple mechanism of revocation. Firstly, the authority publishes via a CRL (Certificate Revocation List) the identities of revoked members. Now, the signer becomes able to show anonymously that he is not on the black-list using non-montone DCR-proofs, since a CS-group signature is partially made of an ElGamal encryption of the signer's identity.

Theorem 4. *Let $\mathcal{S}$ be a CS-group signature scheme and let us denote (A, B) the ElGamal encryption of the signer's identity, and $\mathcal{T}(\sigma_i)$ the predicate contained in the zero-knowledge proof of knowledge. Then member revocation is achieved by proving the following predicate instead of $\mathcal{T}$:*

$$ZKPK\Big[\rho, \sigma_i \ : \ (\mathcal{T}(\sigma_i)) \wedge A = g^\rho \wedge \rho \neq \log_y(B/Y)\Big] \tag{4}$$

where Y is the identity of a revoked member.

The generic transformation thus works as follows. Instead of proving $\mathcal{T}$ when signing a document, a member is required to prove (4), that is to show that he holds a membership certificate, but that he is not a revoked member. If several members are revoked, one has to prove a linear number of negations. We refer the reader to [6] for further details.

6.5 Multi-signer group signatures

We are now interested in providing multi-signer features to any CS-group signature scheme. A multi-signer mechanism for a group signature scheme consists in generating extended group signatures that can convince a verifier that a minimal number of members have actually cooperated to produce the signature, while hiding their identity at the same time. In other words, the signers (e.g., the majority among a board of directors), can prove they have cooperatively generated such a signature, while remaining anonymous. The main point lies in the fact that a group signatures being anonymous, the signers have to prove that the same group member did not sign many times.

A multi-signer (threshold) feature is of interest when a document needs to be signed by several persons who cooperate to authenticate the document while hiding their identities. For instance, several parties have to agree on a contract and prove that at least half of the group signed it, without revealing exactly who. In this section, we show how to extend our result in section 5 to correctly provide an efficient solution to such a scenario. We consider the case where two signers want to cooperate, and refer this scenario as a *double-signer mechanism.*

We first note that the problem is different from the one of member revocation. In the former, the signer has to prove that the identity contained in the signature (within the ElGamal encryption) differs from a *public* value, namely the identity of the revoked member. The subtlety in proving that two members $\mathcal{P}_1$ and $\mathcal{P}_2$ have actually signed is that one must prove that the identity in the first encryption differs from the *hidden* identity contained in the second encryption. More formally, if member $\mathcal{P}_1$ and member $\mathcal{P}_2$ have encrypted their respective identities Y_1 and Y_2 under ElGamal, using the judge's public key (g, y) they have to prove in Zero-Knowledge that $Y_1 \neq Y_2$. A first solution has been proposed in [7, p.118]. However, this solution was designed to suit the scheme by Camenisch and Stadler [10] only. Our solution is more generic, and can be applied to more efficient and secure scheme, such like [1].

We present the following protocol:

Protocol 6.

---

1. $\mathcal{P}_1$ and $\mathcal{P}_2$ choose an exponent $u \in_R \mathbb{Z}_q^*$ and compute $T = (Y_1/Y_2)^u$. They send it to $\mathcal{V}$.
2. $\mathcal{V}$ checks that $T \neq 1$.
3. $\mathcal{P}_1$ and $\mathcal{P}_2$ prove knowledge for the following predicate:

$$\left[\sigma_1, \sigma_2, \rho, \tau \ : \ A_1 = g^{\sigma_1} \wedge A_2 = g^{\sigma_2} \wedge T = (B_1/B_2)^{\rho} y^{\tau} \wedge \left(\frac{A_1}{A_2}\right)^{\rho} = g^{-\tau}\right]$$

Theorem 5. *Under lemma 1, the above protocol is a proof of knowledge of* (μ, ν) *for the predicate:* $T(\sigma_1, \sigma_2) \ : \ (A_1 = g^{\sigma_1}) \wedge (B_1 = Y_1 y^{\sigma_1}) \wedge (A_2 = g^{\sigma_2}) \wedge (B_2 = Y_2 y^{\sigma_2}) \wedge (Y_1 \neq Y_2)$

Proof. Let $(A_1, B_1) = (g^r, y^r Y_1)$ and $(A_2, B_2) = (g^s, y^s Y_2)$ be the encryptions of Y_1 and Y_2, respectively.

Completeness. Verification is easy.

Soundness. Let $\mathcal{E}$ be an extractor for the proof of knowledge performed in the third step of the protocol and (s_1, s_2, r, t) be the values output by $\mathcal{E}$. From the first, second and fourth statement, we get: $(s_1 - s_2)r = -t$. It follows that:

$$T = \left(\frac{B_1/y^{s_1}}{B_2/y^{s_2}}\right)^r$$

Consequently the fact that T differs from 1 ensures that the plaintext corresponding to (A_1, B_1) differs from the plaintext corresponding to (A_2, B_2). Hence, returning (s_1, s_2) makes a knowledge extractor for the predicate: $T(\sigma_1, \sigma_2) \ :$ $(A_1 = g^{\sigma_1}) \wedge (B_1 = Y_1 y^{\sigma_1}) \wedge (A_2 = g^{\sigma_2}) \wedge (B_2 = Y_2 y^{\sigma_2}) \wedge (Y_1 \neq Y_2)$.

Zero-Knowledge. We note that if r and s are randomly chosen, B_1/B_2 is a generator with overwhelming probability, thus T is randomly distributed over $G \backslash \{1\}$. Also the proof performed in step 3 does not leak any information neither. $\square$

Corollary 3. *Let* $\mathcal{S}$ *be a CS-group signature scheme in which a signature is made of an ElGamal encryption* (A, B) *of the signer's identity, together with a proof of knowledge for a predicate* $\mathcal{T}(\rho_i)$. *Then a double-signer mechanism is achieved by providing in the signature*

- $(A_1, B_1) = (g^{s_1}, h^{s_1} Y_1)$ *and* $(A_2, B_2) = (g^{s_2}, h^{s_2} Y_2)$ *for* $s_1, s_2 \in_R \mathbb{Z}_q^*$.
- *the following proof of knowledge:*

$$ZKPK\Big[\rho_i^1, \rho_i^2, \sigma_1, \sigma_2) \;:\; \mathcal{T}(\rho_i^1) \;\wedge\; \mathcal{T}(\rho_i^2) \;\wedge\; (A_1 = g^{\sigma_1}) \;\wedge\; (B_1 = Y_1 y^{\sigma_1}) \\ \wedge\; (A_2 = g^{\sigma_2}) \;\wedge\; (B_2 = Y_2 y^{\sigma_2}) \;\wedge\; (Y_1 \neq Y_2)\Big]$$

where Y_1, Y_2 *are the identities of two different signers.*

Proof. It is easy to see that the first two ElGamal encryptions allows the judge to recover the signers' identities. Also the zero-knowledge proof can be generated only by two different members (soundness). Finally, the signature is anonymous (apart the fact that two people have signed) and unlinkable , due to the security of ElGamal and the zero-knowledgeness. □

7 Conclusion

In this paper, we extended in several ways the class of boolean predicates that can be efficiently proved by means of zero-knowledge proofs. We showed how to compose such predicates to demonstrate polynomial relations among variables, as well as negations of predicates. Also we gave generic applications of such mechanisms to group signature schemes.

References

1. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In M. Bellare, editor, *Proc. of Crypto '00*, volume 1880 of *LNCS*, pages 255–270. Springer-Verlag, August 2000.
2. M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proc. of ACM CCS '93*, pages 62–73. ACM Press, November 1993.
3. S. A. Brands. Untraceable off-line cash in wallets with observers. In D. R. Stinson, editor, *Proc. of Crypto '93*, volume 773 of *LNCS*, pages 302–318. Springer-Verlag, August 1994.
4. S. A. Brands. Rapid demonstration of linear relations connected by boolean operators. In W. Fumy, editor, *Proc. of Eurocrypt '97*, volume 1233 of *LNCS*, pages 318–333. Springer-Verlag, May 1997.
5. G. Brassard and C. Crpeau. Non transitive transfer of confidence: a perfect zero-knowledge interactive protocol for SAT and beyond. In *Proc. of FOCS '86*, pages 188–195. IEEE Press, October 1986.
6. E. Bresson and J. Stern. Efficient revocation in group signatures. In K. Kim, editor, *Proc. of PKC '01*, volume 1992 of *LNCS*, pages 190–206. Springer-Verlag, February 2001.
7. J. Camenisch. *Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem.* PhD thesis, ETH Zürich, 1998.
8. J. Camenisch and M. Michels. A group signature scheme with improved efficiency. In K. Ohta and D. Pei, editors, *Proc. of Asiacrypt '98*, volume 1514 of *LNCS*, pages 160–174. Springer-Verlag, October 1999.

9. J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes. In M. Wiener, editor, *Proc. of Crypto '99*, volume 1666 of *LNCS*, pages 106–121. Springer-Verlag, August 1999.
10. J. Camenisch and M. Stadler. Efficient group signatures schemes for large groups. In B. Kaliski, editor, *Proc. of Crypto '97*, volume 1294 of *LNCS*, pages 410–424. Springer-Verlag, August 1997.
11. J. Camenisch and M. Stadler. Proofs systems for general statements about discrete logarithms. Technical Report TR 260, ETH Zürich, Zürich, CH, March 1997.
12. R. Canetti and S. Goldwasser. An efficient threshold PKC secure against adaptive CCA. In J. Stern, editor, *Proc. of Eurocrypt '99*, volume 1592 of *LNCS*, pages 90–106. Springer-Verlag, May 1999.
13. D. Chaum. Demonstrating that a public predicate can be satisfied without revealing any information about how. In A. M. Odlyzko, editor, *Proc. of Crypto '86*, volume 263 of *LNCS*, pages 195–199. Springer-Verlag, August 1986.
14. D. Chaum, J. H. Evertse, and J. van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In D. Chaum and W. L. Price, editors, *Proc. of Eurocrypt '87*, volume 304 of *LNCS*, pages 127–141. Springer-Verlag, May 1987.
15. D. Chaum, J. H. Evertse, J. van de Graaf, and R. Peralta. Demonstrating possession of a discrete logarithm without revealing it. In A. M. Odlyzko, editor, *Proc. of Crypto '86*, volume 263 of *LNCS*, pages 200–212. Springer-Verlag, August 1986.
16. D. Chaum and E. van Heyst. Group signatures. In D. W. Davies, editor, *Proc. of Eurocrypt '91*, volume 547 of *LNCS*, pages 257–265. Springer-Verlag, May 1992.
17. D. Chaum, E. van Heyst, and B. Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. In J. Feigenbaum, editor, *Proc. of Crypto '91*, volume 576 of *LNCS*, pages 470–484. Springer-Verlag, August 1992.
18. L. Chen and T. P. Pedersen. New group signature schemes. In A. De Santis, editor, *Proc. of Eurocrypt '94*, volume 950 of *LNCS*, pages 171–181. Springer-Verlag, May 1995.
19. R. Cramer, I. B. Damgrd, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Y. G. Desmedt, editor, *Proc. of Crypto '94*, volume 839 of *LNCS*, pages 174–187. Springer-Verlag, August 1994.
20. U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *J. of Cryptology*, 1(2):77–94, Summer 1988.
21. A. Fiat and A. Shamir. How to prove yourself : Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Proc. of Crypto '86*, volume 263 of *LNCS*, pages 186–194. Springer-Verlag, August 1986.
22. Z. Galil, S. Haber, and M. Yung. A private interactive test of a boolean predicate and minimum-knowledge public key cryptosystem. In *Proc. of FOCS '85*, pages 360–371. IEEE Press, October 1985.
23. S. Goldwasser, S. Micali, and C. W. Rackoff. Knowledge complexity of interactive proofs. In *Proc. of STOC '85*, pages 291–304. ACM Press, May 1985.
24. J. Kilian and E. Petrank. Identity escrow. In H. Krawczyk, editor, *Proc. of Crypto '98*, volume 1462 of *LNCS*, pages 169–185. Springer-Verlag, August 1998.
25. A. De Santis, G. Di Crescenzo, G. Persiano, and M. Yung. On monotone formula closure of SZK. In *Proc. of FOCS '94*, pages 454–465. IEEE Press, November 1994.
26. B. Schoenmakers. Efficient proofs of or. Manuscript, 1993.
27. D. X. Song. Practical forward secure group signature schemes. In P. Samarati, editor, *ACM CCS '01*, pages 225–234. ACM Press, November 2001.
28. M. Stadler. Publicly verifiable secret sharing. In U. M. Maurer, editor, *Proc. of Eurocrypt '96*, volume 1070 of *LNCS*, pages 190–199. Springer-Verlag, May 1996.