Rapid Demonstration of Linear Relations Connected by Boolean Operators

Stefan Brands

DigiCash, Kruislaan 419, NL-1098 VA Amsterdam, The Netherlands.
E-mail: brands@digicash.com

Abstract. Consider a polynomial-time prover holding a set of secrets. We describe how the prover can rapidly demonstrate any satisfiable boolean formula for which the atomic propositions are relations that are linear in the secrets, without revealing more information about the secrets than what is conveyed by the formula itself. Our protocols support many proof modes, and are as secure as the Discrete Logarithm assumption or the RSA/factoring assumption.

1 Introduction

Consider a polynomial-time prover that has committed to a vector of secrets and wants to demonstrate that the secrets satisfy some satisfiable formula from propositional logic, where the atomic propositions are relations that are linear in the secrets. An example formula is

$$\big((5x_1 - 3x_2 = 5) \text{ AND } (2x_2 + 3x_3 = 7)\big) \text{ OR } \big(\text{NOT}(x_1 + 4x_3 = 5)\big),$$

where $(x_1, \ldots, x_k)$ is the prover's vector of secrets. The prover does not want to reveal any more information about its secrets than what is conveyed by the formula itself. Can a truly practical protocol for this task be constructed?

In this paper we will show that truly practical protocols exist, assuming the intractability of the Discrete Logarithm problem or the RSA/factoring problem. Our protocols can be performed in all manner of proof modes, including four-move zero-knowledge proofs, three-move witness-hiding proofs, interactive or non-interactive signed proofs that are provably secure in the random oracle model, limited-show proofs, multi-prover proofs, and blinded and restrictively blinded signed proofs.

Our results are organized as follows. Section 2 discusses preliminary notions and reviews basic results. Related work is discussed in Section 3. In Section 4 we introduce our techniques for rapidly demonstrating linear relations connected by boolean operators. We conclude in Section 5.

2 Preliminaries

Throughout this paper, the polynomial-time prover and the (not necessarily polynomial-time) verifier are denoted by $\mathcal{P}$ and $\mathcal{V}$, respectively. The symbol "$\leftarrow$" is used to denote assignment, and $|\cdot|$ denotes binary length. The symbol "$\in_{\mathcal{R}}$" and the word "random" indicate an independent and uniformly random

W. Fumy (Ed.): Advances in Cryptology - EUROCRYPT '97, LNCS 1233, pp. 318-333, 1997.

choice, and we allow distributions that are computationally indistinguishable for polynomially bounded $\mathcal{V}$ and statistically indistinguishable for unbounded $\mathcal{V}$. Whenever we say that $\mathcal{P}$ is able to prove knowledge, we imply the existence of a knowledge extractor that outputs a witness when having oracle access to $\mathcal{P}$.

Our techniques can be based either on the Discrete Logarithm assumption or on the RSA/factoring assumption. We now discuss preliminary notions and basic cryptographic results for these two settings.

2.1 Discrete Logarithm Setting

Set-up. $\mathcal{P}$ and $\mathcal{V}$ initially agree on a cyclic group of order q, denoted by G_q, where q is an integer. Efficient algorithms must be available for recognizing, testing equivalence of, and multiplying numbers in G_q. Without loss of generality it is assumed that q uniquely identifies G_q. Additionally, $k \geq 1$ generators, $g_1, \ldots, g_k$, of G_q, are agreed on; we call $(g_1, \ldots, g_k)$ a generator-tuple. From now on, an integer in the Discrete Log setting is said to be "small" if it is polynomial in $|q|$, and "large" otherwise.

Using the terminology of [6, 7], a representation of a number $h \in G_q$ *with respect to* $(g_1, \ldots, g_k)$ is a vector of numbers, $(x_1, \ldots, x_k)$, such that

$$h = \prod_{i=1}^{k} g_i^{x_i},$$

where $x_1, \ldots, x_k$ are in $\mathbb{Z}_q$.

Intractability of Collision-Finding. For the security of $\mathcal{V}$ it is important that $\mathcal{P}$ cannot know more than one representation of the same number, since it serves as a commit on $\mathcal{P}$'s secrets. For the purpose of the following proposition, which has been proved by Chaum, van Heijst and Pfitzmann [14] for constant k, and by [6, page 16] more generally[1] for all small k, we assume that q is generated according to a probabilistic polynomial-time algorithm (the "DL-instance generator") that, on input a security parameter, outputs a triple (q, g, h), where g and h are generators of G_q.

Proposition 1. *Consider the case that q, as output by the DL-instance generator, is always a prime, and k is small. Assuming that the Discrete Logarithm problem is intractable over the DL-instance generator, there cannot exist a polynomial-time algorithm that, on input q (having a distribution that is indistinguishably close to that of q induced by the DL-instance generator) and a randomly chosen generator-tuple $(g_1, \ldots, g_k)$, outputs with non-negligible probability of success a number $h \in G_q$ and two different representations of h.*

[1] Bellare, Goldreich and Goldwasser [2] noted that the reduction can be modified to achieve a success probability for the Discrete-Logarithm finder that is within a constant factor of that of the collision-finding oracle, instead of being inversely proportionate to k. Specifically, their modification achieves a constant factor $1/2$, instead of $2/k$. Note, however, that the optimization mentioned in [6, page 17] already achieves this (the constant factor is $1/2 + 1/(2k)$).

In case the invulnerable DL-instance generator outputs composite q's, it may be easy to find collisions. In particular, as noted by Chaum et al. [13, page 13], if r is a small prime factor of q then one can easily find collisions, by raising the generators in the generator-tuple to the power q/r and computing their relative Discrete Logarithms in the subgroup of order r. As in Stinson [32, page 239], one can alleviate this situation if q/r is a large prime (or a composite that is infeasible to factor) by restricting the elements $x_1, \ldots, x_k$ of a representation to be in $\mathbb{Z}_{q/r}$. Alternatively, we can consider a DL-instance generator that outputs q's that have only large prime factors; finding collisions then requires one to break the Discrete Logarithm problem in G_q or to factor q. In addition, as in Brickell and McCurley [12], one can let the g_i's be generators of a non-trivial subgroup of G_q.

To guarantee $\mathcal{V}$'s security one should generate the elements of the set-up in accordance with the probability distributions of the appropriate DL-instance generator, depending on which form of q's one is interested in. The set-up should be generated by $\mathcal{V}$ itself, in a mutually random fashion between $\mathcal{V}$ and $\mathcal{P}$, by a party trusted by $\mathcal{V}$ or liable for security breaks by $\mathcal{P}$, or in any other manner that ensures that $\mathcal{P}$ cannot find collisions for the generated instance.

Proving Knowledge. Our results in Section 4 can be based on *any* proof of knowledge (see Bellare and Goldreich [1]) of a representation. For practical purposes we are interested in highly efficient protocols that offer a wide range of *proof modes*. The following generic protocol enables $\mathcal{P}$, for any m with $1 \leq m \leq k$, to demonstrate knowledge of a representation, $(x_1, \ldots, x_m)$ (its *secret key*), of a number $h \in G_q$ (its *public key*) with respect to a generator-tuple, $(g_1, \ldots, g_m)$. We assume for the moment that q is a prime.

Step 1. $\mathcal{P}$ generates at random m numbers $w_1, \ldots, w_m \in_{\mathcal{R}} \mathbb{Z}_q$, and sends $a \leftarrow \prod_{i=1}^{m} g_i^{w_i}$ to $\mathcal{V}$.

Step 2. $\mathcal{P}$ computes m responses, responsive to a challenge $c \in \mathbb{Z}_{2^t}$, according to $r_i \leftarrow cx_i + w_i \bmod q$, for $i = 1, \ldots, m$, and sends them to $\mathcal{V}$. The process of generating c and the size of t determine the proof mode of the protocol; in the appendix several proof modes of particular relevance are discussed.

Step 3. $\mathcal{V}$ accepts if and only if $h^{-c} \prod_{i=1}^{m} g_i^{r_i} = a$.

One can also consider the above protocol for q's that are not prime, and in particular for all the forms discussed in the preceding subsubsection. Of course, this has ramifications with respect to the proof modes and/or the intractability assumptions needed for security.

Rapid Computations. Rapid evaluation of $g_1^{x_1} \cdots g_m^{x_m}$ can be performed using simultaneous repeated squaring (see Knuth [23, exercise 27, page 465]). For efficiency one can, for all 2^m subsets of $\{g_1, \ldots, g_m\}$, precompute the product of the g_i's in the subset, and store the products in a table. With $1 < l \leq m$, the product $\prod_{i=1}^{m} g_i^{x_i}$ can be computed using $d = \lceil m/l \rceil$ precomputed tables, using simultaneous repeated squaring for each of the d sub-products and multiplying the sub-product results. Several variations and optimizations of this basic technique are known in the literature. For example, one can process $j > 1$ exponent bits at once; the size of the precomputed table then increases by a factor $2^{(j-1)m}$, while the workload decreases by approximately a factor j.

2.2 RSA Setting

Set-up. $\mathcal{P}$ and $\mathcal{V}$ initially agree on a group $\mathbb{Z}_n^*$, where $n = pq$ and p and q are distinct primes. They also agree on an integer, v. Additionally, $k \geq 1$ numbers, $g_1, \ldots, g_k$, all in $\mathbb{Z}_n^*$, are agreed on. From now on, an integer in the RSA setting is said to be "small" if it is polynomial in $|n|$, and "large" otherwise.

A *representation* of a number h in $\mathbb{Z}_n^*$, *with respect to* $(g_1, \ldots, g_k, v)$, is a vector of numbers, $(x_1, \ldots, x_k, x_{k+1})$, such that

$$h = (\prod_{i=1}^{k} g_i^{x_i}) x_{k+1}^v \bmod n,$$

where x_{k+1} is in $\mathbb{Z}_n^*$ and $x_1, \ldots, x_k$ are in $\mathbb{Z}_v$.

Intractability of Collision-Finding. For the security of $\mathcal{V}$ it is important that at least $\mathcal{P}$ cannot know more than one representation of the same number. For the purpose of the following two propositions, we assume that n is generated according to a probabilistic polynomial-time algorithm (the "RSA-instance generator") that, on input a security parameter and an integer v, outputs a pair $(n, y \in \mathbb{Z}_n^*)$. For any integer $v \geq 2$, we can consider the problem of extracting v-th roots modulo n; this is called the RSA problem for that particular v.

Proposition 2. *Suppose that v is a prime that is co-prime to $\varphi(n)$, and that k is small. Assuming that the RSA problem for v is intractable over the RSA-instance generator, there cannot exist a polynomial-time algorithm that, on input n (having a distribution that is indistinguishably close to that of n induced by the RSA-instance generator) and randomly chosen $(g_1, \ldots, g_k)$, outputs with non-negligible probability of success a number $h \in \mathbb{Z}_n^*$ and two different representations of h with respect to $(g_1, \ldots, g_k, v)$.*

Sketch of proof. To compute $y^{1/v} \bmod n$, on input $y \in \mathbb{Z}_n^*$, construct each g_i as $y^{r_i} s_i^v \bmod n$, for $r_i \in_{\mathcal{R}} \mathbb{Z}_v$ and $s_i \in_{\mathcal{R}} \mathbb{Z}_n^*$. If the oracle output is correct, a relation of the form $y^t = u^v \bmod n$ can be computed, for known $t \in \mathbb{Z}_v$ and $u \in \mathbb{Z}_n^*$, and from this $y^{1/v} \bmod n$ can be computed.

Note that if p and q are random primes of equal size, then a random element in $\mathbb{Z}_n^*$ has small order with negligible probability; see Håstad, Schrift and Shamir [21, Proposition 1].

Other choices of v are possible as well. For example, in case v is small and *not* co-prime to $\varphi(n)$, according to Ohta and Okamoto [25, Theorem 1] it is as hard to compute v-th roots as to factor the modulus. By restricting the g_i's and x_{k+1} in the definition of a representation to v-th residues, one can prove the difficulty of finding collisions for v's of this particular form in a likewise manner. The following result shows that this also holds for large v of a special form.

Proposition 3. *Consider the case in which $v = 2^l$, for any integer l, and the RSA-instance generator always outputs Blum integers (i.e., p and q are congruent to 3 mod 4). Furthermore, restrict the number x_{k+1} in a representation to be a quadratic residue. Assuming that the factoring problem is intractable over the*

RSA-instance generator, there cannot exist a polynomial-time algorithm that, on input n (having a distribution that is indistinguishably close to that of n induced by the RSA-instance generator) and randomly chosen quadratic residues $(g_1, \ldots, g_k)$, outputs with non-negligible probability of success a number $h \in \mathbb{Z}_n^*$ and two different representations of h with respect to $(g_1, \ldots, g_k, v)$.

Proving Knowledge. Our results in Section 4 can be based on any proof of knowledge of a representation. The following generic protocol is very efficient and offers a wide range of proof modes. For any m with $0 \leq m \leq k$, the protocol enables $\mathcal{P}$ to demonstrate knowledge of a representation, $(x_1, \ldots, x_m, x_{m+1})$, of a number $h \in \mathbb{Z}_n^*$ with respect to $(g_1, \ldots, g_m, v)$. For the moment, we assume that v is a prime that is co-prime to $\varphi(n)$.

Step 1. $\mathcal{P}$ generates at random m numbers $w_1, \ldots, w_m \in_{\mathcal{R}} \mathbb{Z}_v$, and a number $w_{m+1} \in_{\mathcal{R}} \mathbb{Z}_n^*$. $\mathcal{P}$ computes $a \leftarrow g_1^{w_1} \cdots g_m^{w_m} w_{m+1}^v \bmod n$, and sends a to $\mathcal{V}$.

Step 2. $\mathcal{P}$ computes $m+1$ responses, responsive to a challenge $c \in \mathbb{Z}_{2^t}$, according to $r_i \leftarrow cx_i + w_i \bmod v$, for $1 \leq i \leq m$, and

$$r_{m+1} \leftarrow \left(\prod_{i=1}^{m} g_i^{cx_i + w_i \operatorname{div} v}\right) \cdot x_{m+1}^c w_{m+1} \bmod n,$$

and sends them to $\mathcal{V}$.

Step 3. $\mathcal{V}$ accepts if and only if $\left(\prod_{i=1}^{m} g_i^{r_i}\right) \cdot r_{m+1}^v = h^c a \bmod n$.

All the proof modes for the proof of knowledge discussed in the Discrete Logarithm setting apply here as well, with the obvious modifications. If $m = 0$ we have the Guillou-Quisquater protocol [20] and if $m = 1$ the Okamoto protocol [26, page 39].

By making minor adjustments to the above protocol, we can use v's that are not prime and/or not co-prime to $\varphi(n)$. In all these cases, one has to restrict the set from which the g_i's and x_{m+1} and w_{m+1} are chosen, to avoid leakage of information about $\mathcal{P}$'s representation; similar adjustments as discussed in the preceding subsubsection can be made. Note, however, that if v is a large composite with a small prime factor, u, and it is feasible to randomly generate u-th residues without knowing a u-th root, then $\mathcal{P}$ can convince $\mathcal{V}$ in the three-move protocol with non-negligible success probability (specifically, $1/u$ if v/u has no small prime factors, and larger otherwise) without knowing a representation of h with respect to $(g_1, \ldots, g_k, v)$; for these v's, another protocol should be used.

3 Related Work

A constant-round zero-knowledge argument for our task can be constructed by properly reducing the boolean formula that is to be demonstrated to an instance of the NP-complete language Directed Hamiltonian Cycle, and applying the zero-knowledge argument of knowledge of Feige and Shamir [18]. However, techniques such as this are not practical, because they amount to encoding the statement into a boolean circuit and using commitments for each gate.

By restricting q in the Discrete Logarithm setting to be a prime, one can define a relation, $R = R_{q,(g_1,\ldots,g_k),(\alpha_1,\ldots,\alpha_k)}$, for any q, for any generator-tuple $(g_1,\ldots,g_k)$ and any vector of coefficients $(\alpha_1,\ldots,\alpha_k) \in (\mathbb{Z}_q)^k \setminus \{0\}^k$, as follows:

$$\big((h,b),\,(x_1,\ldots,x_k)\big) \in R \quad \Leftrightarrow \quad h = \prod_{i=1}^{k} g_i^{x_i} \text{ and } b = \sum_{i=1}^{k} \alpha_i x_i \bmod q$$

The corresponding language is easily seen to be random self-reducible. In the RSA setting, for v a prime that is co-prime to $\varphi(n)$, a random self-reducible language can be defined in a similar manner. Now, by applying the construction of Tompa and Woll [33] one gets a perfect zero-knowledge proof of knowledge for both languages, but these protocols use binary-valued challenges and require polynomially many rounds. For the special case $k = 3$ and $b = x_1 + x_2 + mx_3 \bmod q$, and b is an undeniable signature of $\mathcal{P}$ on a message m of the (unlimited powerful) $\mathcal{V}$, this construction has been used for signature confirmation by Chaum et al. [14]. We remark that it is straightforward to improve the protocols, by using a large challenge domain and prepending a move in which the verifier commits to its challenge (note that this improvement has been overlooked by Chaum et al. [14]), but the resulting protocols remain less efficient than ours. Moreover, our protocols facilitate many other proof modes.

De Santis, Di Crescenzo, Persiano and Yung [17] show how to prove any monotone formula over a random self-reducible language (Cramer, Damgård and Schoenmakers [15] independently discovered virtually the same technique). If a monotone formula has m logical connectives, then this technique requires the prover to perform m proofs of knowledge, one for each sub-formula. In contrast, our "AND" technique has the property that the communication complexity for both the prover and the verifier slightly *decreases* as the number of "AND" connectives increases, and the computational complexity is virtually unaffected. Moreover, the technique of De Santis et al. uses binary-valued challenges, and thus polynomially many repetitions are needed.

Furthermore, the technique of De Santis et al. and Cramer et al. applies only to monotone boolean formula, while we have a very efficient "NOT" technique. Chaum et al. [14] describe a perfect zero-knowledge protocol for the "NOT" of their special relation, $b = x_1 + x_2 + mx_3 \bmod q$, but this protocol inherently works for binary-valued challenges only. Moreover, in each iteration the signer must compute seven commitments, requiring many exponentiations in G_q, and it is unclear how to efficiently construct other proof modes.

Another important difference is in the scenario that is considered. Namely, De Santis et al. and Cramer et al. consider a situation in which there are many different public keys, and $\mathcal{P}$ demonstrates (in zero-knowledge) that it knows the secret keys corresponding to some of these. In contrast, we are concerned with the situation in which the prover knows a *single* public key, and demonstrates that its secret key satisfies a certain formula.

4 Demonstrating Boolean Formulae for Linear Relations

In this section we describe our proof techniques for "AND," "NOT" and "OR" connectives, respectively, and then show how to combine them in order to demon-

strate arbitrary boolean formulae. Without loss of generality we base our discussions on the Discrete Logarithm setting, and for the RSA setting describe only the necessary adaptations. Note that if $k = 1$, then $\mathcal{V}$ can verify for any boolean formula directly whether the secret of $\mathcal{P}$ satisfies it, and so from now on we only consider the case $k \geq 2$.

4.1 Formulae with only "AND" connectives

We first consider the situation in which $\mathcal{P}$ has to demonstrate a satisfiable formula with zero or more "AND" connectives. At the outset, $\mathcal{P}$ has committed to a set of secrets, $(x_1, \ldots, x_k)$, by sending a number $h \in G_q$ to $\mathcal{V}$, where $(x_1, \ldots, x_k)$ is a representation of h with respect to $(g_1, \ldots, g_k)$. Without loss of generality, we assume that $\mathcal{P}$ has to demonstrate to $\mathcal{V}$ that this representation satisfies the following system of $l \geq 1$ independent linear relations:

$$\begin{pmatrix} \alpha_{11} & \ldots & \alpha_{1,k-l} & 1 & 0 & \ldots & 0 \\ \alpha_{21} & \ldots & \alpha_{2,k-l} & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{l1} & \ldots & \alpha_{l,k-l} & 0 & 0 & \ldots & 1 \end{pmatrix} \begin{pmatrix} x_{\pi(1)} \\ x_{\pi(2)} \\ \vdots \\ x_{\pi(k)} \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_l \end{pmatrix} \bmod q. \tag{1}$$

The coefficients α_{ij}, for $1 \leq i \leq l$ and $1 \leq j \leq k - l$, are elements of $\mathbb{Z}_q$, and $\pi(\cdot)$ is a permutation of $\{1, \ldots, k\}$. The corresponding boolean formula is:

$$\begin{aligned} &(b_1 = \alpha_{11} x_{\pi(1)} + \cdots + \alpha_{1,k-l} x_{\pi(k-l)} + x_{\pi(k-l+1)} \bmod q) \text{ AND } \ldots \\ &\quad \ldots \text{ AND } (b_l = \alpha_{l1} x_{\pi(1)} + \cdots + \alpha_{l,k-l} x_{\pi(k-l)} + x_{\pi(k)} \bmod q). \end{aligned} \tag{2}$$

Note that the atomic proposition is the special case $l = 1$.

Our technique for demonstrating formula (2) is based on the following result.

Proposition 4. *$\mathcal{P}$ can demonstrate knowledge of a representation of*

$$h \left(\prod_{i=1}^{l} g_{\pi(k-l+i)}^{b_i} \right)^{-1}$$

with respect to

$$\left(g_{\pi(1)} \prod_{i=1}^{l} g_{\pi(k-l+i)}^{-\alpha_{i1}}, \ \cdots, \ g_{\pi(k-l)} \prod_{i=1}^{l} g_{\pi(k-l+i)}^{-\alpha_{i,k-l}} \right)$$

if and only if it knows a set of secrets that satisfies the formula (2).

The proof follows straightforwardly, by considering the relations that are satisfied by the output of the knowledge extractor. Note that the tuple in Proposition 4 is a generator-tuple with overwhelming probability in case q is a prime and the prover selects the matrix entries, α_{ij}, and can always be guaranteed to be so when the matrix entries are determined by $\mathcal{V}$.

We can efficiently implement the protocol by using the proof of knowledge for Discrete Logarithm representations described in Section 2. An important benefit of using this protocol is that one can expand the resulting expressions, so that $\mathcal{P}$ and $\mathcal{V}$ can use a single precomputed table for simultaneous repeated squaring, *independent* of the particular formula that is demonstrated. The resulting (generic) protocol steps are as follows:

Step 1. $\mathcal{P}$ generates at random $k-l$ numbers, $w_1, \ldots, w_{k-l} \in_{\mathcal{R}} \mathbb{Z}_q$, and computes

$$a \leftarrow \prod_{i=1}^{k-l} g_{\pi(i)}^{w_i} \prod_{i=1}^{l} g_{\pi(k-l+i)}^{-\sum_{j=1}^{k-l} \alpha_{ij} w_j}.$$

$\mathcal{P}$ then sends a to $\mathcal{V}$.

Step 2. $\mathcal{P}$ computes a set of responses, responsive to a challenge number c in $\mathbb{Z}_{2^t}$, as follows:

$$r_i \leftarrow c x_{\pi(i)} + w_i \bmod q, \quad \forall i \in \{1, \ldots, k-l\}.$$

$\mathcal{P}$ then sends $(r_1, \ldots, r_{k-l})$ to $\mathcal{V}$.

Step 3. $\mathcal{V}$ computes

$$r_{k-l+i} \leftarrow c b_i - \sum_{j=1}^{k-l} \alpha_{ij} r_j \bmod q, \quad \forall i \in \{1, \ldots, l\},$$

and accepts if and only if

$$a = h^{-c} \prod_{i=1}^{k} g_{\pi(i)}^{r_i}.$$

The particular proof mode of the protocol is "inherited" from the mode in which the underlying proof of knowledge is performed, and a further discussion is therefore omitted here. Note, however, that special care must be taken for signed proofs: the transcript of a protocol execution is always convincing of the fact that $\mathcal{P}$ knows a set of secrets corresponding to h, but convinces of its conformity with the demonstrated formula only when a uniquely identifying description of the demonstrated formula is hashed along (or when the α_{ij}'s and the b_i's are all restricted to sets that are negligible in the range of the hash function).

To base the proof on the RSA/factoring problem, consider $\mathcal{P}$ having to prove the system of linear relations (1), but with "mod v" replacing "mod q." We assume that $\mathcal{P}$ has committed to a set of secrets, $(x_1, \ldots, x_k)$, by sending $h \leftarrow g_1^{x_1} \cdots g_k^{x_k} x_{k+1}^v \bmod n$ to $\mathcal{V}$, for some x_{k+1} in $\mathbb{Z}_n^*$.

Proposition 5. *For any integer $v \geq 2$, $\mathcal{P}$ can prove knowledge of a representation of*

$$h \left(\prod_{i=1}^{l} g_{\pi(k-l+i)}^{b_i} \right)^{-1} \bmod n$$

with respect to

$$\left(g_{\pi(1)} \prod_{i=1}^{l} g_{\pi(k-l+i)}^{-\alpha_{i1}} \bmod n, \ \ldots, \ g_{\pi(k-l)} \prod_{i=1}^{l} g_{\pi(k-l+i)}^{-\alpha_{i,k-l}} \bmod n, \ v \right).$$

if and only if it knows a set of secrets that satisfies the formula.

By using the efficient proof of knowledge for the RSA setting described in Section 2, again expanding the resulting expressions, a single precomputed table can be used for simultaneous repeated squaring; of course, one then also inherits the limitations in the range of choices for v.

4.2 Formulae with only "NOT" connectives

We next study the situation in which $\mathcal{P}$ has to demonstrate that a linear relation does *not* hold, without revealing more information than required. The situation at the outset is as in Subsection 4.1. This time, $\mathcal{P}$ has to demonstrate to $\mathcal{V}$ that its representation satisfies the formula

$$\mathrm{NOT}\big(x_{\pi(1)} = \alpha_1 + \alpha_2 x_{\pi(2)} + \cdots + \alpha_k x_{\pi(k)} \bmod q\big). \tag{3}$$

The coefficients α_i, for $1 \leq i \leq k$, are elements of $\mathbb{Z}_q$. Clearly, the permutation $\pi(\cdot)$ can always be defined to interchange at most two elements and leave the rest unchanged.

Our technique for demonstrating formula (3) is based on the following result.

Proposition 6. *Let q be a prime. $\mathcal{P}$ can prove knowledge of a representation of $g_{\pi(1)}$ with respect to*

$$\Big(g_{\pi(1)}^{\alpha_1} h^{-1}, g_{\pi(1)}^{\alpha_2} g_{\pi(2)}, \ \cdots, \ g_{\pi(1)}^{\alpha_k} g_{\pi(k)}\Big)$$

if and only if it knows a set of secrets that satisfies the formula (3).

Sketch of proof. With $(y_1, \ldots, y_k)$ denoting the representation output by the knowledge extractor, if $y_1 = 0$ then a non-trivial representation of 1 has been found and hence the Discrete Logarithm problem is tractable, and if $y_1 \neq 0$ then the representation satisfies formula (3).

Proposition 7. *If the proof of knowledge performed by $\mathcal{P}$ in the preceding Proposition is witness indistinguishable, then it is impossible for $\mathcal{V}$ (even with unlimited computing power) to learn any information about the difference between $x_{\pi(1)}$ and $\alpha_1 + \alpha_2 x_{\pi(2)} + \cdots + \alpha_k x_{\pi(k)} \bmod q$.*

Sketch of proof. Denoting the representation known to $\mathcal{P}$ by $(z_1, \ldots, z_k)$ and the difference by ϵ, observe that $z_1 = 1/\epsilon \bmod q$, and so information about ϵ is leaked if and only if information about z_1 is leaked. Since $k \geq 2$, for each $z_1 \in \mathbb{Z}_q$ there is a representation containing that z_1; and because there are equally many (namely, q^{k-2}) such representations for each z_1 and the protocol is witness-indistinguishable, no information about ϵ leaks.

If q is not a prime, then the inverse of the difference number, ϵ, is not guaranteed to exist. If q is a composite that is hard to factor then zero-divisors cannot be found and so nothing is lost, and in other cases we can force the existence of an inverse by making additional assumptions about the coefficients in (3) and/or about the representation of $\mathcal{P}$.

Applying the efficient proof of knowledge for the Discrete Logarithm setting described in Section 2, the following practical protocol results:

Step 1. $\mathcal{P}$ generates at random k numbers, $w_1, \ldots, w_k \in_{\mathcal{R}} \mathbb{Z}_q$, and computes

$$a \leftarrow h^{-w_1} g_{\pi(1)}^{\sum_{i=1}^{k} \alpha_i w_i} \prod_{i=2}^{k} g_{\pi(i)}^{w_i}.$$

$\mathcal{P}$ then sends a to $\mathcal{V}$.

Step 2. Let ϵ denote $(\alpha_1 + \sum_{i=2}^{k} \alpha_i x_{\pi(i)}) - x_{\pi(1)} \bmod q$, and let $\delta = \epsilon^{-1} \bmod q$. $\mathcal{P}$ computes a set of responses, responsive to a challenge number c in $\mathbb{Z}_{2^t}$, as follows:

$$r_1 \leftarrow c\delta + w_1 \bmod q,$$
$$r_i \leftarrow c x_{\pi(i)} \delta + w_i \bmod q, \quad \forall i \in \{2, \ldots, k\}.$$

$\mathcal{P}$ then sends $(r_1, \ldots, r_k)$ to $\mathcal{V}$.

Step 3. $\mathcal{V}$ accepts if and only if

$$a = h^{-r_1} g_{\pi(1)}^{-c + \sum_{i=1}^{k} \alpha_i r_i} \prod_{i=2}^{k} g_{\pi(i)}^{r_i}.$$

As before, this protocol inherits the proof modes of the protocol described in Section 2. Note that signed proofs convince only of the demonstrated formula if the α_i's are hashed along or if they are restricted to be in small sets.

Our technique can also be based on the RSA/factoring problem. Consider $\mathcal{P}$ having to prove formula (3), with "mod v" replacing "mod q," and having committed to $(x_1, \ldots, x_k)$ using $h \leftarrow g_1^{x_1} \cdots g_k^{x_k} x_{k+1}^v \bmod n$, for x_{k+1} in $\mathbb{Z}_n^*$.

Proposition 8. *If v is a prime (or a composite that is hard to factor), then $\mathcal{P}$ can prove knowledge of a representation of $g_{\pi(1)}$ with respect to*

$$\left(g_{\pi(1)}^{\alpha_1} h^{-1} \bmod n, g_{\pi(1)}^{\alpha_2} g_{\pi(2)} \bmod n, \ \ldots, \ g_{\pi(1)}^{\alpha_k} g_{\pi(k)} \bmod n, \ v\right),$$

if and only if it knows a set of secrets that satisfies the formula.

Of course, if $v = 2$ then all boolean formula are monotone and one can do without this technique.

4.3 Formulae with only "OR" connectives

We now show how $\mathcal{P}$ can demonstrate that at least one of two linear relations holds, without revealing which one; this technique is an application of the "OR" technique of De Santis et al. and Cramer et al., although the scenario is different. The situation at the outset is again as in Subsection 4.1. This time $\mathcal{P}$ has to demonstrate to $\mathcal{V}$ that the representation known to it satisfies the formula

$$\left(x_{\pi(1)} = \alpha_1 + \sum_{i=2}^{k} \alpha_i x_{\pi(i)} \bmod q\right) \text{ OR } \left(x_{\rho(1)} = \beta_1 + \sum_{i=1}^{k} \beta_i x_{\rho(i)} \bmod q\right). \quad (4)$$

The coefficients α_i and β_i, for $1 \leq i \leq k$, are elements of $\mathbb{Z}_q$, and $\pi(\cdot)$ and $\rho(\cdot)$ are permutations of $\{1, \ldots, k\}$ that can always be defined to interchange at most two elements each.

If (and only if) the first linear relation holds, then $\mathcal{P}$ can compute, for any challenge c_1, responses $(r_2, \ldots, r_k)$ such that

$$a_1 = h^{-c_1} g_{\pi(1)}^{c_1 \alpha_1 + \sum_{i=2}^{k} \alpha_i r_i} g_{\pi(2)}^{r_2} \cdots g_{\pi(k)}^{r_k},$$

where

$$a_1 = g_{\pi(1)}^{\sum_{i=2}^{k} \alpha_i w_i} \prod_{i=2}^{k} g_{\pi(i)}^{w_i}$$

for random $w_2, \ldots, w_k$ in $\mathbb{Z}_q$. Likewise, if (and only if) the second linear relation holds, then $\mathcal{P}$ can compute, for any challenge c_2, responses $(s_2, \ldots, s_k)$ such that

$$a_2 = h^{-c_2} g_{\rho(1)}^{c_2\beta_1 + \sum_{i=2}^{k} \beta_i s_i} g_{\rho(2)}^{s_2} \cdots g_{\rho(k)}^{s_k},$$

where

$$a_2 = g_{\rho(1)}^{\sum_{i=2}^{k} \beta_i v_i} g_{\rho(2)}^{v_2} \cdots g_{\rho(k)}^{v_k},$$

for random $v_2, \ldots, v_k$ in $\mathbb{Z}_q$. To demonstrate formula (4), we have $\mathcal{P}$ choose one of the two challenges, c_1 or c_2, at random by itself, so that it can anticipate that challenge by calculating a suitable a_i from the self-chosen challenge and a set of randomly self-chosen "responses." To ensure that $\mathcal{P}$ cannot choose the other challenge by itself as well, $\mathcal{P}$ must use challenges c_1 and c_2 such that, say, the bitwise exclusive-or of c_1 and c_2 is equal to the supplied challenge, c. (Of course, "simulation" is needed only for those sub-formulae that do not hold; if both sub-formulae would be true, $\mathcal{P}$ can do without a self-chosen challenge.)

This technique can straightforwardly be generalized to a formula with more than one "OR" connective, and as before an efficient implementation can be obtained by using the proof of knowledge of Section 2. A description based on the RSA/factoring problem is straightforward, and hence omitted.

4.4 Putting it all together

We now show how to combine the basic demonstration techniques, in order to demonstrate arbitrary satisfiable formulae from propositional logic, where the atomic propositions are linear relations over $\mathbb{Z}_q$. We hereto first show how to combine the techniques of Subsections 4.1 and 4.2 in order to demonstrate any satisfiable formula from propositional logic that has zero or more "AND" connectives and at most one "NOT" connective; these formulae play a central role in combining the basic techniques.

A consistent system consisting of linear relations and one linear inequality can be written as a system of linear relations by introducing a difference term, denoted by ϵ. By appropriate substitution, the system can then be represented by the matrix equation

$$\begin{pmatrix} \alpha_{11} & \ldots & \alpha_{1,k-l} & 1 & 0 & \ldots & 0 \\ \alpha_{21} & \ldots & \alpha_{2,k-l} & 0 & 1 & \ldots & 0 \\ \vdots & & \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{l1} & \ldots & \alpha_{l,k-l} & 0 & 0 & \ldots & 1 \end{pmatrix} \begin{pmatrix} x_{\pi(1)} \\ x_{\pi(2)} \\ \vdots \\ x_{\pi(k)} \end{pmatrix} = \begin{pmatrix} b_1 - f_1\epsilon \\ b_2 - f_2\epsilon \\ \vdots \\ b_l - f_l\epsilon \end{pmatrix} \bmod q, \tag{5}$$

where $f_1, \ldots, f_l$ are numbers in $\mathbb{Z}_q$. (Clearly, one of the f_i's can always be 1.)

Our technique for demonstrating the boolean formula that corresponds to the system (5) is based on the following result.

Proposition 9. *$\mathcal{P}$ can prove knowledge of a representation of*

$$\prod_{i=1}^{l} g_{\pi(k-l+i)}^{f_i}$$

with respect to

$$\left(h^{-1}\prod_{i=1}^{l} g_{\pi(k-l+i)}^{b_i},\ g_{\pi(1)}\prod_{i=1}^{l} g_{\pi(k-l+i)}^{-\alpha_{i1}},\ \cdots,\ g_{\pi(k-l)}\prod_{i=1}^{l} g_{\pi(k-l+i)}^{-\alpha_{i,k-l}}\right).$$

if and only if it knows a set of secrets that satisfies the system (5).

As in Proposition 7, ϵ is information-theoretically hidden if the proof is witness-indistinguishable, provided that $l < k$. If $k = l$, then $\mathcal{V}$ can check the validity of the system (5) directly from $\mathcal{P}$'s public key, without interacting with $\mathcal{P}$; computing ϵ then is as hard as breaking the Discrete Logarithm problem, and ϵ has at least $O(\log |q|)$ bits that are simultaneously hard-core.

We are now prepared to describe our general technique. Any boolean formula, F, can be expressed in the form

$$\mathrm{F} = Q_1 \text{ AND } \cdots \text{ AND } Q_m, \tag{6}$$

where each sub-formula, Q_i, has the format R_{i1} OR $\cdots$ OR R_{i,m_i}, and each subsub-formulae, R_{ij}, is a formula from propositional logic that connects linear relations over $\mathbb{Z}_q$ by zero or more "AND" connectives, at most one "NOT" connective and no other logical connectives. We have just seen how to demonstrate R_{ij}, and by using the technique of Subsection 4.3 we can have $\mathcal{P}$ demonstrate a single sub-formula, Q_i. To prove the formula F, $\mathcal{P}$ needs to demonstrate the validity of all m sub-formulae, $Q_1, \ldots, Q_m$. Hereto the corresponding m proofs can all be performed in parallel, responsive to the same challenge.

An optimization is sometimes possible, depending on the complexity of F. Namely, a system of the form (5) can be interpreted as corresponding to an atomic proposition. To demonstrate knowledge for this atomic proposition, our techniques have $\mathcal{P}$ demonstrate knowledge of a secret key corresponding to a "distorted" public key, with respect to a "distorted" generator tuple. We can now apply the monotone formula technique of De Santis et al. and Cramer et al. to prove monotone boolean formulae over these atomic propositions. In particular, the restrictions according to which $\mathcal{P}$ generates its self-chosen challenges from the supplied challenge can be dictated in accordance with the secret-sharing construction of Benaloh and Leichter [4] for the access structure defined by the dual of the formula F (see Cramer et al. [15] for details). In other words, expressing F in a more compact form than (6) may lead to a more efficient protocol.

A further optimization is for $\mathcal{V}$ to batch-process verification relations that correspond to atomic formulae that are connected by "AND" operators; this can be done similarly to the technique of Naccache, M'Raïhi, Raphaeli and Vaudenay [24] for batch verification of DSA signatures.

A description of the above techniques based on the difficulty of factoring or computing RSA-roots poses no particular difficulties, and is hence omitted.

5 Conclusion

An interesting problem is to extend the set of atomic propositions beyond linear relations. True practicality requires constant-round proofs of knowledge for which the computation and communication complexity are linearly dependent on the number of secrets of $\mathcal{P}$ and the size of its public key, but independent of the parameters specifying the atomic proposition or anything else. The following approaches do not satisfy this criterion:

- The technique of Damgård [16] can be adapted in order to demonstrate atomic formulae of the form

$$x_1^{a_1} + \alpha_2 x_2^{a_2} + \cdots + \alpha_k x_k^{a_k} = \alpha_1 \bmod q,$$

  but this requires $\mathcal{P}$ to perform $O(\sum_{i=1}^{k} a_i)$ separate basic proofs of knowledge and proofs of equality of discrete logarithms;
- Brickell, Chaum, Dåmgard and Van de Graaf [11] showed how to prove that an exponent is in an interval, but their protocol inherently requires binary challenges (and thus polynomially many iterations), and moreover the proof must be performed for a substantially larger interval in order to avoid leakage of information; and
- The protocol of Pfitzmann [28] for demonstrating multiplications in zero-knowledge also inherently requires binary challenges.

Moreover, in all three cases the number of available proof modes is seriously limited. It is an open problem to construct truly practical protocols for atomic propositions of the above forms.

Our techniques have many practical applications. For example, they can be used to implement the confirmation and the disavowal protocols of Chaum et al. [14] more efficiently (the speed-up is polynomial). The main motivation, however, for devising the techniques in this paper has been to construct all manner of practical privacy-protecting credential mechanisms; this is the subject of a forthcoming paper.

References

1. M. Bellare and O. Goldreich. On defining proofs of knowledge. In E. F. Brickell, editor, *Advances in Cryptology–CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer-Verlag, 1992.
2. M. Bellare, O. Goldreich, and S. Goldwasser. Incremental cryptography: The case of hashing and signing. In Y. G. Desmedt, editor, *Advances in Cryptology–CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 216–233. Springer-Verlag, 1994.
3. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73, Fairfax, 1993. ACM Press.
4. J. Benaloh and J. Leichter. Generalized secret sharing and monotone functions. In S. Goldwasser, editor, *Advances in Cryptology–CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 27–35. Springer-Verlag, 1988.

5. M. Blum, A. De Santis, S. Micali, and G. Persiano. Noninteractive zero-knowledge. *SIAM J. Computing*, 20(6):1084–1118, December 1991.
6. S. Brands. An efficient off-line electronic cash system based on the representation problem. Technical Report CS-R9323, Centrum voor Wiskunde en Informatica, April 1993.
7. S. Brands. Untraceable off-line cash in wallets with observers. In D. R. Stinson, editor, *Advances in Cryptology–CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 302–318. Springer-Verlag, 1994.
8. S. Brands. More on restrictive blind issuing of secret-key certificates in parallel mode. Technical Report CS-R9534, Centrum voor Wiskunde en Informatica, March 1995.
9. S. Brands. Restrictive blind issuing of secret-key certificates in parallel mode. Technical Report CS-R9523, Centrum voor Wiskunde en Informatica, March 1995.
10. S. Brands. Restrictive blinding of secret-key certificates. In L. C. Guillou and J.-J. Quisquater, editors, *Advances in Cryptology–EUROCRYPT '95*, volume 921 of *Lecture Notes in Computer Science*, pages 231–247. Springer-Verlag, 1995.
11. E. F. Brickell, D. Chaum, I. B. Damgård, and J. van de Graaf. Gradual and verifiable release of a secret. In C. Pomerance, editor, *Advances in Cryptology–CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 156–166. Springer-Verlag, 1988.
12. E. F. Brickell and K. S. McCurley. An interactive identification scheme based on discrete logarithms and factoring. *Journal of Cryptology*, 5(1):29–39, 1992.
13. D. Chaum, E. van Heijst, and B. Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. Technical report, University of Karlsruhe, February 1991. Interner Bericht 1/91.
14. D. Chaum, E. van Heijst, and B. Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. In J. Feigenbaum, editor, *Advances in Cryptology–CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 470–484. Springer-Verlag, 1992.
15. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Y. G. Desmedt, editor, *Advances in Cryptology–CRYPTO '94*, Lecture Notes in Computer Science, pages 174–187. Springer-Verlag, 1994.
16. I. B. Damgård. Practical and provably secure release of a secret. In T. Helleseth, editor, *Advances in Cryptology–EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 200–217. Springer-Verlag, 1994.
17. A. De Santis, G. D. Crescenzo, G. Persiano, and M. Yung. On monotone formula closure of SZK. In *Proc. 35th IEEE Symp. on Foundations of Comp. Science*, pages 454–465, Santa Fe, 1994. IEEE Transactions on Information Theory.
18. U. Feige and A. Shamir. Zero-knowledge proofs of knowledge in two rounds. In G. Brassard, editor, *Advances in Cryptology–CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 526–544. Springer-Verlag, 1990.
19. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. Odlyzko, editor, *Advances in Cryptology–CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer-Verlag, 1987.
20. L. C. Guillou and J.-J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessors minimizing both transmission and memory. In C. Günther, editor, *Advances in Cryptology–EUROCRYPT '88*, Lecture Notes in Computer Science, pages 123–128. Springer-Verlag, 1988.
21. J. Håstad, A. Schrift, and A. Shamir. The discrete logarithm modulo a composite hides $o(n)$ bits. *JCSS*, 47(3):376–404, 1993.

22. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In U. Maurer, editor, *Advances in Cryptology–EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 143–154. Springer-Verlag, 1996.
23. D. E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*, pages 441–462. Addison-Wesley Publishing Company, 2 edition, 1981. ISBN 0-201-03822-6.
24. D. Naccache, D. M'Raïhi, S. Vaudenay, and D. Raphaeli. Can D.S.A. be improved? – complexity trade-offs with the digital signature standard. In A. D. Santis, editor, *Advances in Cryptology–EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 77–85. Springer-Verlag, 1995.
25. K. Ohta and T. Okamoto. A modification of the Fiat-Shamir scheme. In S. Goldwasser, editor, *Advances in Cryptology–CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 232–243. Springer-Verlag, 1988.
26. T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In E. F. Brickell, editor, *Advances in Cryptology–CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53. Springer-Verlag, 1992.
27. T. Okamoto and K. Ohta. Divertible zero knowledge interactive proofs and communtative random self-reducibility. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology–EUROCRYPT '89*, volume 434 of *Lecture Notes in Computer Science*, pages 134–149. Springer-Verlag, 1989.
28. B. Pfitzmann. ZKP in $\mathbb{Z}_p$ or $\mathbb{Z}_{2^\sigma}$. Unpublished manuscript, April 1991.
29. D. Pointcheval and J. Stern. Provably secure blind signature schemes. In K. Kim and T. Matsumoto, editors, *Advances in Cryptology–ASIACRYPT '96*, 1163, pages 252–265. Springer-Verlag, 1996.
30. D. Pointcheval and J. Stern. Security proofs for signature schemes. In U. Maurer, editor, *Advances in Cryptology–EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer-Verlag, 1996.
31. C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, 1991.
32. D. R. Stinson. *Cryptography; theory and practice*. CRC Press, 1 edition, 1995. ISBN 0-8493-8521-0.
33. M. Tompa and H. Woll. Random self-reducibility and zero knowledge interactive proofs of possession of information. Technical Report RC 13207 (#59069), IBM, October 1987.

A Proof Modes

If c is chosen at random by $\mathcal{V}$, and 2^t is small, then the protocol must be repeated polynomially many times in order for $\mathcal{P}$'s proof to be convincing with overwhelming probability. Sequential repetitions result in a zero-knowledge proof, while parallel repetitions are not zero-knowledge unless preceded by an initial step in which $\mathcal{V}$ commits to its challenges (the commit must be unconditionally secure for $\mathcal{P}$ in case $\mathcal{V}$ is unbounded); alternatively, the challenges are determined in a mutually random fashion by $\mathcal{P}$ and $\mathcal{V}$.

If 2^t is large then $\mathcal{P}$ is convincing with overwhelming probability, without repetitions. The case $m = 1$ is the Schnorr proof of knowledge [31]; this is widely believed to be witness hiding, although no proof of this is known. In case $m \geq 2$ the protocol is non-trivially witness indistinguishable and provably witness

hiding, and the case $m = 2$ is Okamoto's proof of knowledge [26, page 36]. The protocol can be made zero-knowledge in the manner described above.

The protocol can be performed as a signed proof, meaning that the transcript of the protocol execution is convincing evidence that $\mathcal{P}$ has performed a protocol execution. Following Fiat and Shamir [19], the challenge c is hereto computed as a one-way hash (implying that 2^t is large) of at least a. The hash-function must be such that it is infeasible to obtain more signed proofs than the number of protocol executions that $\mathcal{P}$ has engaged in ("unforgeability"). In addition, h and a message may be hashed along; in the latter case the signed proof serves as a digital signature of $\mathcal{P}$ on the message. The signed proof consist of $(r_1, \ldots, r_m)$, one of a and c, and any (other) information hashed in order to compute c; moreover, h must be included in case it is not associated with $\mathcal{P}$. The Schnorr signature scheme [31] (resp. the Okamoto signature scheme [26, page 46]) is the special case in which $\mathcal{P}$ determines c by itself, signed proofs serve as digital signatures, and $m = 1$ (resp. $m = 2$).

If we model the hash function as a random oracle (see Bellare and Rogaway [3]) and $\mathcal{P}$ computes c in Step 2 by itself, then the unforgeability of signed proofs is guaranteed for all $m \geq 1$, assuming the Discrete Logarithm assumption; see Pointcheval and Stern [30]. In particular, this holds also if $\mathcal{V}$ supplies a message, possibly adaptively chosen based on previous protocol executions, that is hashed along by $\mathcal{P}$.

In signed proof mode, it may be desirable to let $\mathcal{V}$ instead of $\mathcal{P}$ determine c, for example to enable $\mathcal{V}$ to obtain a blinded signed proof (it is straightforward to apply the blinding technique of Okamoto and Ohta [27]). In the random oracle model, the unforgeability of signed proofs for which c determines the challenge is guaranteed for all $m \geq 2$, assuming the Discrete Logarithm assumption and provided that $\mathcal{P}$ engages in no more than logarithmically many protocol executions; see Pointcheval and Stern [29].

Other proof modes are available as well. For example, one can perform the protocol as a non-interactive zero-knowledge proof (see Blum, De Santis, Micali and Persiano [5]), a limited-show proof, a designated verifier proof (see Jakobsson, Sako and Impagliazzo [22]), or a multi-prover proof. As an example of the latter proof mode, consider i parties that have each committed to their own secret, x_i, by publishing $h_i = g_1^{x_i} g_2^{y_i}$, for randomly chosen y_i; by taking h to be the product of appropriate powers of the h_i's, they can jointly demonstrate formulae pertaining to their secrets (without revealing them to any other party), by combining their responses in accordance with the formula that is demonstrated.

Finally, we note that the protocol can be modified in order to issue a signed proof that can be blinded only restrictively, by using the techniques of [10]. Hereto $\mathcal{P}$ and $\mathcal{V}$ perform the blinded signed proof with respect to a combination of $\mathcal{P}$'s public key and $\mathcal{V}$'s public key. In addition to the properties of unforgeability and independence of the signed proof and $\mathcal{P}$'s view, it can be proved under the Discrete Logarithm assumption that part of the representation of $\mathcal{V}$ remains invariant under $\mathcal{V}$'s blinding operations. In the random oracle model, this holds even if polynomially many verifiers, each with a different public key, conspire, provided that protocol executions are performed sequentially; for parallel executions, slight modifications are required, and the security can only be argued heuristically (see [9, 8]).