

How to prove list membership in logarithmic time

*Liesje Demuynck
Bart De Decker*

Report CW 470, December 2006



Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

How to prove list membership in logarithmic time

Liesje Demuyneck
Bart De Decker

Report CW470, December 2006

Department of Computer Science, K.U.Leuven

Abstract

In this paper we propose a novel technique for proving in zero-knowledge that a committed value is on a public list. Our technique combines the concept of a hashtree with well-known zero-knowledge proofs of knowledge. The resulting proof protocol has a time complexity logarithmic in the size of the list. Its soundness property is guaranteed under the discrete logarithm assumption, while the prover's secret information is protected even in the face of a computational unbounded adversary. Applications of our technique can be found in membership revocation for group signature schemes, anonymous credential systems and identity escrow schemes.

Keywords : privacy and anonymity, commitments, zero-knowledge proofs, membership revocation

CR Subject Classification : E.3 [Data]: Data Encryption – Public key cryptosystems

How to prove list membership in logarithmic time

Liesje Demuynck
K.U.Leuven, Dept. of Comp. Sc
Celestijnenlaan 200A
B-3001 Heverlee, Belgium
Liesje.Demuynck@cs.kuleuven.be
www.cs.kuleuven.be

Bart De Decker
K.U.Leuven, Dept. of Comp. Sc
Celestijnenlaan 200A
B-3001 Heverlee, Belgium
Bart.DeDecker@cs.kuleuven.be
www.cs.kuleuven.be

1 Introduction

When developing privacy-sensitive applications, the use of commitments and corresponding zero-knowledge proofs is often indispensable. A commitment can be seen as the digital analogue of a “non-transparent sealed envelope”. On the one hand, it enables a committer to hide the committed value $s \in S$ (non-transparency property), while on the other hand, it is ensured that this value cannot be modified after commitment (sealed property). At a later time, the commitment can be opened by the committer, revealing the value s and convincing the verifier that s was indeed the committed value.

In many situations, the need arises for a guarantee that the committer really knows the secret encoded into a commitment, or that this secret satisfies a particular property. For this, zero-knowledge proofs can be used. These proofs allow a committer to demonstrate certain properties about committed values, without allowing the verifier to find out anything more than what the committer willingly reveals. Among others, they enable the demonstration of multiplicative relations $ab = c$ in S , or the equality of two committed values encoded into different commitments [5, 9].

In this paper, we present a perfect honest-verifier zero-knowledge proof of knowledge that a committed value is on a public list L . The protocol makes use of a hashtree and has a time complexity logarithmic in the size of the list. Its construction shows how existing techniques can be adapted and combined in order to achieve the resulting complexity gain. The protocol is secure under the well-accepted discrete logarithm (DL) assumption and guarantees unconditional privacy even against a computationally unbounded verifier. Applications can be found in group signature schemes [1, 11], identity escrow schemes [15] and anonymous credential systems [2, 3].

Related work. Over the past years, many commitment schemes have been proposed [7, 14, 18]. These schemes differ mainly in the structure of the commitment’s message space S (which may range from a finite group to even integers) and the intractability assumptions on which security is based. In this paper we will use the commitment scheme proposed by Pedersen [18], which is secure under the discrete logarithm assumption.

Starting with Schnorr [20] in 1991, many zero-knowledge proofs for proving both the knowledge and properties of discrete logarithms, have been introduced. The most straightforward technique

for proving that a committed value is on a public list, uses ideas proposed by [12] to prove a relation of the form $\forall x_i \in L (x = x_i)$, where x is the committed value and $L = \{x_1, \dots, x_l\}$ the public whitelist. The main drawback of this solution is that it requires time linear in l .

Better solutions exist, and make use of dynamic accumulators [4, 17]. The main idea of these constructions is to collapse multiple elements into a single value, named an accumulator. When an element is added to the accumulator, it is associated with a special value (named a witness), such that it is infeasible to find a witness for elements not incorporated into the accumulator. Also, each witness is dependent on the current accumulator value, and must be updated whenever an element is added to, or deleted from the accumulator.

Dynamic accumulators can be used to prove list membership for a public list L of size l . In a setup phase, all elements of the list are added to the accumulator. This process takes l steps and results both in an initial witness for each listed element, and the specification of an “accumulator history”, needed for updating the initial witnesses. The accumulator, its history and the initial witnesses are then published. To prove that his committed value is on the list, a prover uses the relevant initial witness and accumulator history to compute a valid witness, and then proves in zero-knowledge that this is indeed a witness for his committed value. The witness can be computed in advance and requires, in the worst case, a time linear in l . On the other hand, the zero-knowledge proof can be done in constant time.

As our solution’s setup phase also requires a complexity linear in the size of L , the main advantage in using accumulators is the constant time complexity of a membership proof. However, current proposals for dynamic accumulator schemes rely on rather non-standard intractability assumptions, such as the strong RSA assumption in [4] and the q -strong Diffie Hellman assumption in [17]. Furthermore, in [4], the proofs of knowledge are only statistical zero-knowledge and the list of accumulatable values is limited to prime numbers.

A problem related to ours is the construction of zero-knowledge sets [8, 16]. A scheme for zero-knowledge sets enables a prover to commit to an arbitrary set D of strings such that no additional information about D can be deduced from the commitment (not even the size of D is revealed). In a later stage, the prover demonstrates, in zero-knowledge and for a known value x , statements of the form $x \in D$ or $x \notin D$. While in zero-knowledge set schemes, it is proven that a known value belongs to a secret list, our solution focusses on secret values belonging to a known list. Hence, both techniques are complementary to each other.

Our solution makes use of various known techniques described in literature. More precisely, the hashfunction described in section 3.1 is an adaptation of the collision resistant function discussed among others by Chaum et al. [10] and Brands [2]. Next to this, the proof of an exponentiation with hidden exponent in Section 3.2 is based on ideas of Camenisch et al. [5].

Organization. Section 2 briefly reviews a few well-known schemes for commitments and zero-knowledge proofs. These schemes will be needed as basic building blocks in our construction. The protocol itself is presented in Section 3. Finally, Section 4 describes and evaluates some extensions to the system.

2 Basic Tools

We now introduce the building blocks on which our construction is based: the Pedersen commitment scheme and zero-knowledge proofs of knowledge of committed values satisfying certain

properties. In the remainder of the paper, all interactive protocols are executed between a polynomially bounded prover \mathcal{P} and a verifier \mathcal{V} with unbounded computing powers.

The Pedersen commitment scheme [18] is unconditionally hiding and computationally binding under the discrete logarithm assumption in a group G_q of prime order q . Its setup consists of the decision on a suitable group G_q and the selection of a random generator tuple $(g_1, g_2) \in_{\mathcal{R}} G_q^2$. A commitment $C = \text{Com}_{(g_1, g_2)}(x; r)$ for input $x \in \mathbb{Z}_q$ and randomness $r \in_{\mathcal{R}} \mathbb{Z}_q$ is then created as $C = g_1^x g_2^r$. An important note on the scheme is that $\log_{g_2} g_1$ and $\log_{g_1} g_2$ must be hidden *only* from the committer. In particular, the knowledge of any of those values may be used by \mathcal{V} to improve the efficiency of zero-knowledge protocols concerning committed values.

By applying traditional zero-knowledge protocols for demonstrating knowledge and properties of DL representations, \mathcal{P} can prove properties of her commitments to \mathcal{V} . Among others, the following statements can be demonstrated.

- **Knowledge of an opening containing public values [2, Chapter 3].** \mathcal{P} can demonstrate to \mathcal{V} her knowledge of values $(x, r) \in \mathbb{Z}_q^2$, such that $C = \text{Com}_{(g_1, g_2)}(x; r)$ for $C, g_1, g_2 \in G_q$. In the course of the process, she can disclose the committed value x while hiding r . Based on the notation of Camenisch and Stadler [6], we abbreviate this protocol as $PK\{(\rho, \chi) : C = \text{Com}_{(g_1, g_2)}(\chi; \rho)\}$ or $PK\{(\rho) : C = \text{Com}_{(g_1, g_2)}(x; \rho)\}$, depending on whether or not value x is revealed by \mathcal{V} . (Greek letters represent the values that are unknown to \mathcal{V} .)
- **Knowledge and equality of committed values [9].** Let C_1, C_2, g_1, g_2, g_3 and g_4 be elements in G_q , \mathcal{P} can prove to \mathcal{V} her knowledge of values x, r_1 and r_2 in G_q , such that $C_1 = \text{Com}_{(g_1, g_2)}(x; r_1)$ and $C_2 = \text{Com}_{(g_3, g_4)}(x; r_2)$. This protocol will be abbreviated as $PK\{(\chi, \rho_1, \rho_2) : C_1 = \text{Com}_{(g_1, g_2)}(\chi, \rho_1) \wedge C_2 = \text{Com}_{(g_3, g_4)}(\chi, \rho_2)\}$. It can be used to demonstrate multiplicative relations between committed values. For example, based on commitments $C_a = \text{Com}_{(g_1, g_2)}(a; r)$, $C_b = \text{Com}_{(g_1, g_2)}(b; s)$ and $C_c = \text{Com}_{(g_1, g_2)}(c; t)$. \mathcal{P} can prove the relation $c = ab \pmod q$ by performing the protocols $PK\{(\alpha, \rho) : C_a = \text{Com}_{(g_1, g_2)}(\alpha; \rho)\}$ and $PK\{(\beta, \sigma, \delta) : C_b = \text{Com}_{(g_1, g_2)}(\beta; \sigma) \wedge C_c = \text{Com}_{(C_a, g_2)}(\beta; \delta)\}$. As will be seen shortly, both protocols can be combined into one single proof protocol by using AND connections.
- **AND connections and OR connections.** All previous formulae can be combined, either by OR connectives [12, 19], by AND connectives, or by combinations of both. Let $F_1(x_{1,1}, \dots, x_{1,n_1}), \dots, F_m(x_{m,1}, \dots, x_{m,n_m})$ be formulae concerning secrets $(x_{i,1}, \dots, x_{i,n_i})$ ($i = 1, \dots, m$). The protocols are denoted $PK\{(\chi_{1,1}, \dots, \chi_{m,n_m}) : F_1(\chi_{1,1}, \dots, \chi_{1,n_1}) \vee \dots \vee F_m(\chi_{m,1}, \dots, \chi_{m,n_m})\}$ for OR connectives and $PK\{(\chi_{1,1}, \dots, \chi_{m,n_m}) : F_1(\chi_{1,1}, \dots, \chi_{1,n_1}) \wedge \dots \wedge F_m(\chi_{m,1}, \dots, \chi_{m,n_m})\}$ for AND connectives.

Under the discrete logarithm assumption in G_q , all these protocols are perfect honest-verifier zero-knowledge and perfect witness indistinguishable proofs of knowledge. They can be made concurrent zero-knowledge at virtually no overhead by using techniques proposed by Damgård [13].

3 A proof that a committed value is on a list

Let $L = \{x_1, \dots, x_l\}$ be a list of publicly known values, and let $C_{h_0} = \text{Com}_{(g_1, g_2)}(h_0; s_0)$ be a commitment to a value $h_0 \in L$. Before performing the proof, \mathcal{P} and \mathcal{V} agree on a suitable hashfunction H and auxiliary function f , and create a binary hashtree \mathcal{H} . The leaves of this tree consist of the

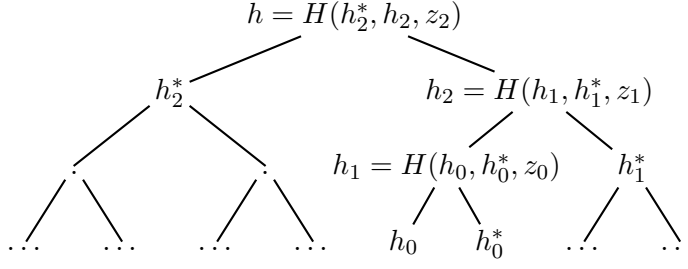


Figure 1: an example hashtree \mathcal{H} for a list L of 8 elements and a path (h_0, h_1, h_2, h) from $h_0 \in L$ to rootnode h

elements $x_i \in L$. The intermediate nodes $h_p = H(h_l, h_r, f(h_l, h_r))$ are formed as the hash H of their left child h_l , right child h_r and auxiliary value $f(h_l, h_r)$. If the size l of L is not a power of 2, the remaining leaves are filled with a standard value. To prove that her value h_0 is on the list, \mathcal{P} now has to show that she knows a way up to the public root h of the tree. Let $w = \lceil \log_2 l \rceil$ and denote the path from h_0 up to root h as (h_0, \dots, h_{w-1}, h) , the respective sibling nodes on this path as $(h_0^*, \dots, h_{w-1}^*)$, and the needed auxiliary values as (z_0, \dots, z_{w-1}) . The proof protocol essentially boils down to a proof of knowledge of hidden values h_i, h_i^*, z_i ($i = 0, \dots, w - 1$) such that h_0 is encoded into C_{h_0} and the equation

$$(h_1 = H(h_0, h_0^*, z_0) \vee h_1 = H(h_0^*, h_0, z_0)) \wedge \dots \wedge (h = H(h_{w-1}, h_{w-1}^*, z_{w-1}) \vee h = H(h_{w-1}^*, h_{w-1}, z_{w-1}))$$

holds. An example hashtree for a list L of 8 elements is depicted in figure 3.

Next we describe how this proof can be established. Sections 3.1 and 3.2 present two building blocks in the final protocol; a suitable collision intractable hashfunction H and a protocol for proving knowledge of committed values x and c such that $x = a^c \pmod q$ for a public value a . Section 3.3 then describes how to perform one step in the hashtree, while section 3.4 describes the final protocol.

3.1 A collision intractable hashfunction

To construct the hashtree, a suitable hashfunction H is needed. H must be a collision intractable function mapping values x_1, x_2 and z , with x_1, x_2 any two values in \mathbb{Z}_q and z uniquely specifying (x_1, x_2) , to a new value in \mathbb{Z}_q . Furthermore, H must allow efficient zero-knowledge proofs.

As a basis for our construction, we employ the collision intractable hashfunction of [10]. This function is defined as $H' : \mathbb{Z}_q^2 \rightarrow G_q : (x_1, x_2) \mapsto g_1^{x_1} g_2^{x_2}$. In order to construct a new function H mapping values to \mathbb{Z}_q , (x_1, x_2) is split into smaller values and the mapping is extended to be defined over multiple bases belonging to a subgroup of \mathbb{Z}_q . We now formally define this resulting family of functions $\{H_i(\cdot)\}$ over an instance generator (I, D) . Algorithm I generates a value i uniquely specifying $H_i(\cdot)$ and algorithm D returns a value x from $H_i(\cdot)$'s domain D_i . The instance generator for $\{H_i(\cdot)\}$ is defined as follows.

- On input a security parameter k , algorithm I generates primes q and q' , with q' of binary length k , $q' | q - 1$ and $q' > m$ for $m = 2^{\lceil \frac{|q|}{2} \rceil}$ and $|q|$ the binary length of q . I then randomly

picks four elements g_a, g_b, g_c and $g_d \in_{\mathcal{R}} G_{q'}$, where $G_{q'}$ is the unique q' -order subgroup of \mathbb{Z}_q^* . The final output of I is a tuple $(q, q', g_a, g_b, g_c, g_d)$.

- D receives I 's output $t = (q, q', g_a, g_b, g_c, g_d)$ and outputs an arbitrary tuple $(x_1, x_2, z) \in D_t$. Here, D_t is defined as $\{(x_1, x_2, z) \in \mathbb{Z}_q \times \mathbb{Z}_q \times \mathbb{Z}_m^4 \mid z = (z_a, z_b, z_c, z_d) \wedge x_1 \equiv z_a m + z_b \pmod{q} \wedge x_2 \equiv z_c m + z_d \pmod{q}\}$. Note that for any $(x_1, x_2) \in \mathbb{Z}_q^2$, a suitable $z \in \mathbb{Z}_m^4$ can easily be found.

A hashfunction H_t for $t = (q, q', g_a, g_b, g_c, g_d)$ is now defined as

$$\begin{aligned} H_t : D_t &\rightarrow G_{q'} \\ (x_1, x_2, (z_a, z_b, z_c, z_d)) &\mapsto g_a^{z_a} g_b^{z_b} g_c^{z_c} g_d^{z_d} \pmod{q}. \end{aligned}$$

The following result shows that under the discrete logarithm assumption in $G_{q'}$, function H_t is collision intractable.

Proposition 1. *Under the assumption that the partial outputs q and q' of I specify a group $G_{q'}$ in which the discrete logarithm problem is hard, the family of functions $\{H_i(\cdot)\}_{i \in (q, q', g_a, g_b, g_c, g_d)}$ is collision intractable over (I, D) .*

Proof. In order to prove this Proposition, we employ the following property [2, Section 2.3.2].

Lemma 1. *Let q, q' be primes and let (g_a, g_b, g_c, g_d) be a random tuple in $G_{q'}^4$ with $G_{q'}$ the q' -order subgroup of \mathbb{Z}_q . Let \mathcal{B} be a polynomial time adversary who, on input $(q, q', (g_a, g_b, g_c, g_d))$ outputs, with non-negligible success probability ϵ , two tuples $(z_a, z_b, z_c, z_d) \neq (z'_a, z'_b, z'_c, z'_d)$ in $\mathbb{Z}_{q'}$ such that $g_a^{z_a} g_b^{z_b} g_c^{z_c} g_d^{z_d} = g_a^{z'_a} g_b^{z'_b} g_c^{z'_c} g_d^{z'_d}$. \mathcal{B} can be used to break the discrete logarithm assumption in $G_{q'}$.*

Suppose that on input $t = (q, q', g_a, g_b, g_c, g_d)$, a polynomial time adversary \mathcal{A} can find, with non-negligible success probability θ , two different inputs (x_1, x_2, z) and $(x'_1, x'_2, z') \in D_t$ such that $H_t(x_1, x_2, z) = H_t(x'_1, x'_2, z')$. Adversary \mathcal{B} can then be constructed as follows.

1. \mathcal{B} receives input $(q, q', (g_a, g_b, g_c, g_d))$ and feeds $(q, q', g_a, g_b, g_c, g_d)$ to \mathcal{A} .
2. \mathcal{B} receives tuples (x_1, x_2, z) and (x'_1, x'_2, z') from \mathcal{A} . She checks whether $(x_1, x_2, z), (x'_1, x'_2, z') \in D_t$, whether $(x_1, x_2, z) \neq (x'_1, x'_2, z')$ and whether $H_t(x_1, x_2, z) = H_t(x'_1, x'_2, z')$. If so, she outputs $(z_a \pmod{q'}, z_b \pmod{q'}, z_c \pmod{q'}, z_d \pmod{q'})$ and $(z'_a \pmod{q'}, z'_b \pmod{q'}, z'_c \pmod{q'}, z'_d \pmod{q'})$, where $z = (z_a, z_b, z_c, z_d)$ and $z' = (z'_a, z'_b, z'_c, z'_d)$. Otherwise, \mathcal{B} aborts the protocol.

It can be seen that $z \neq z' \pmod{m}$. In the case of $(x_1, x_2) = (x'_1, x'_2)$, this is trivially true. In the other case, note that (x_1, x_2) and (x'_1, x'_2) are uniquely determined by z and z' respectively. Hence, an equality $z = z'$ would contradict the fact that $(x_1, x_2) \neq (x'_1, x'_2)$.

From this fact and the observation that $(z \pmod{m}) \pmod{q'} = z \pmod{q'}$, it can be deduced that $z \neq z' \pmod{q'}$. Hence, \mathcal{B} is a polynomial time algorithm which can be used to break the discrete logarithm assumption in $G_{q'}$. \mathcal{B} 's success probability ϵ equals \mathcal{A} 's success probability θ . \square

Note that the definition of (I, D) may be adapted to choose z in other ways as described above, without affecting the collision intractability of H . More precisely, D_i may be defined as (x_1, x_2, z) using any choice for constructing $z = (z_a, z_b, z_c, z_d)$, the only restrictions are that $z_a, z_b, z_c, z_d < q'$, and the existence of an onto function g such that $g(z_a, z_b, z_c, z_d) = (x_1, x_2)$. The explicit choice

$ \begin{aligned} & u_{(-1)} = 1 \\ & \text{for}(i = 0; i \leq c -1; i++)\{ \\ & \quad \text{if } (c_i = 0) \ u_i = u_{i-1} \\ & \quad \text{else } u_i = u_{i-1} \cdot a^{2^i} \\ & \} \\ & x = u_{ c -1} \end{aligned} $

Figure 2: Algorithm for computing $x = a^c$ with $c = \sum_{i=0}^{|c|-1} c_i 2^i$ for $c_i \in \{0, 1\}$ and $|c| = \lceil \log_2 c \rceil$. The algorithm uses intermediate results u_i for $i = -1, \dots, |c| - 1$.

of z in our construction is related to the possibility to, and the efficiency of, the zero-knowledge protocols for proving list membership.

Also, function f used in the construction of \mathcal{H} can be any function mapping $(x_1, x_2) \in \mathbb{Z}_q$ to a value z such that $(x_1, x_2, z) \in D_t$. A suitable choice for f is the following.

$$f : \mathbb{Z}_q^2 \rightarrow \mathbb{Z}_m^4 : (x_1, x_2) \mapsto (x_1 \text{ div } m, x_1 \text{ mod } m, x_2 \text{ div } m, x_2 \text{ mod } m)$$

3.2 Proving exponentiations with a hidden exponent

In the context of proving safe-prime products, Camenisch et al. [5] introduced a protocol for demonstrating the relation $a^c = x \text{ mod } N$ between committed values a, c, x and N . A simplified version of this protocol is presented in this section. In our protocol, value a is assumed to be public, while c and x are still committed values. The protocol is also tailored to prove a relation $a^c = x \text{ mod } q$, where \mathbb{Z}_q is the commitment scheme's message space. As a consequence, our protocol is perfect zero-knowledge, while the original technique is only statistical zero-knowledge. Next to this, the prover enjoys unconditional privacy, even in the face of a computational unbounded verifier engaging into any number of arbitrary interleaved protocols.

The proof makes use of the well-known exponentiation method depicted in figure 3.2. Let \mathcal{P} 's publicly known commitments to x and c be $C_x = \text{Com}_{(g_1, g_2)}(x; r_x)$ and $C_c = \text{Com}_{(g_1, g_2)}(c; r_c)$ respectively, with r_x and $r_c \in_{\mathcal{R}} \mathbb{Z}_q$. Let $|q|$ be the binary length of q . \mathcal{P} and \mathcal{V} engage into the following protocol.

1. \mathcal{P} generates the auxiliary commitments to conduct the proof. She generates commitments $C_{c_i} = \text{Com}_{(g_1, g_2)}(c_i; d_i)$ with $d_i \in_{\mathcal{R}} \mathbb{Z}_q$ ($i \in \{0, \dots, |q| - 1\}$) for all of the bits c_i in the $|q|$ -bit representation of c (note that $|q|$ forms an upper bound on the binary length of elements in \mathbb{Z}_q). Next to this, \mathcal{P} generates commitments $C_{u_i} = \text{Com}_{(g_1, g_2)}(u_i; w_i)$ ($i \in \{0, \dots, |q| - 2\}$) with $w_i \in_{\mathcal{R}} \mathbb{Z}_q$, to each intermediate value in the exponentiation algorithm. All commitments are then sent to the verifier.
2. The following proof of knowledge is executed. \mathcal{V} accepts only if she accepts the proof.

$$\begin{aligned}
& PK\{(\sigma_1, (\sigma_{i,j}, \tau_{i,j})_{i \in \{0, \dots, |q|-1\}, j \in \{0,1\}})\} : \\
& (C_c^{-1} \prod_{i=0}^{|q|-1} C_{c_i}^{2^i}) = \text{Com}_{(g_1, g_2)}(0; \sigma_1) \wedge \tag{1} \\
& [(C_{c_0} = \text{Com}_{(g_1, g_2)}(0; \sigma_{0,0}) \wedge C_{u_0} = \text{Com}_{(g_1, g_2)}(1; \tau_{0,0})) \vee \\
& \quad (C_{c_0} = \text{Com}_{(g_1, g_2)}(1; \sigma_{0,1}) \wedge C_{u_0} = \text{Com}_{(g_1, g_2)}(a; \tau_{0,1})] \wedge \\
& [(C_{c_1} = \text{Com}_{(g_1, g_2)}(0; \sigma_{1,0}) \wedge C_{u_1} C_{u_0}^{-1} = \text{Com}_{(g_1, g_2)}(0; \tau_{1,0})) \vee \\
& \quad (C_{c_1} = \text{Com}_{(g_1, g_2)}(1; \sigma_{1,1}) \wedge C_{u_1} C_{u_0}^{-a^2} = \text{Com}_{(g_1, g_2)}(0; \tau_{1,1}))] \\
& \quad \wedge \dots \wedge \\
& [(C_{c_{|q|-1}} = \text{Com}_{(g_1, g_2)}(0; \sigma_{|q|-1,0}) \wedge C_x C_{u_{|q|-2}}^{-1} = \text{Com}_{(g_1, g_2)}(0; \tau_{|q|-1,0})) \vee \\
& \quad (C_{c_{|q|-1}} = \text{Com}_{(g_1, g_2)}(1; \sigma_{|q|-1,1}) \wedge C_x C_{u_{|q|-2}}^{-a^{2^{|q|-1}}} = \text{Com}_{(g_1, g_2)}(0; \tau_{|q|-1,1}))] \} \tag{2}
\end{aligned}$$

Equation 1 proves the correctness of the relation $c = \sum_{i=0}^{|q|-1} c_i 2^i$, with $c, c_0, \dots, c_{|q|-1}$ the values committed into $C_c, C_{c_0}, \dots, C_{c_{|q|-2}}$ and $C_{c_{|q|-1}}$ respectively. Afterwards, value $x = a^c \bmod q$ is constructed step by step using equations 2. Each step in the construction proves a relation of the form $[(c_i = 0 \wedge u_i = u_{i-1}) \vee (c_i = 1 \wedge u_i = u_{i-1} a^{2^i})]$. This is exactly one step in the exponentiation algorithm of Figure 3.2.

The following property can easily be shown to hold, based on the properties of the original protocol of Camenisch et al. [5].

Proposition 2. *Under the discrete log assumption in G_q , the above protocol is a perfect honest-verifier zero-knowledge proof of knowledge of committed values x and c in \mathbb{Z}_q such that $x \equiv a^c \bmod q$.*

Proof. Completeness and zero-knowledge are trivially true. A simulator picks random values $C_{c_0}, \dots, C_{c_{|q|-1}}, C_{u_0}, \dots, C_{u_{|q|-2}}$ in G_q and simulates protocol step 2 by employing the simulator for the proofs of knowledge in Section 2.

To demonstrate soundness, we employ the knowledge extractor \mathcal{K} for the proofs of knowledge in Section 2. Let \mathcal{K} 's output be a tuple $(\sigma_1, (\sigma_{i,j}, \tau_{i,j})_{i \in \{0, \dots, |q|-1\}, j \in \{0,1\}})$ satisfying relations 1 and 2. Note that this output tuple must be the same as known by \mathcal{P} . If this were not the case, then \mathcal{P} and \mathcal{K} together would have two openings for at least one of the commitments used during the proof protocol. This would enable them to break the discrete logarithm assumption in G_q .

Suppose that we have bits $c_0, \dots, c_{|q|-1}$ such that

$$C_{c_i} = \text{Com}_{(g_1, g_2)}(c_i; \sigma_{i, c_i}) \wedge C_{u_i} = \text{Com}_{(C_{u_{i-1}}, g_2)}(a^{(2^i c_i)}; \tau_{i, c_i}) \tag{3}$$

for $i = 0, \dots, |q|-1$, $C_{u_{(-1)}} = g_1$ and $C_{u_{|q|-1}} = C_x$. Remark that $c_0, \dots, c_{|q|-1}$ can easily be deduced from \mathcal{K} 's output. Equation 3 enables us to recursively compute a tuple $(x, r_x) = (a^{\sum_{i=0}^{|q|-1} 2^i c_i} \bmod q, \sum_{i=0}^{|q|-1} (a^{\sum_{j=i+1}^{|q|-1} 2^j c_j} \tau_{i, c_i}) \bmod q)$ such that $C_x = \text{Com}_{(g_1, g_2)}(x; r_x)$. Next to this, Equation 1 enables us to compute a tuple $(c, r_c) = (\sum_{i=0}^{|q|-1} 2^i c_i \bmod q, -\sigma_1 + \sum_{i=0}^{|q|-1} 2^i \sigma_{i, c_i} \bmod q)$ such that $C_c = \text{Com}_{(g_1, g_2)}(c; r_c)$.

Hence, we have deduced values c, r_c, x and r_x in \mathbb{Z}_q such that $C_c = \text{Com}_{(g_1, g_2)}(c; r_c)$, $C_x = \text{Com}_{(g_1, g_2)}(x; r_x)$ and $x = a^c \bmod q$. This concludes the proof. \square

We also have the following new property.

Proposition 3. *Consider a computationally unbounded verifier \mathcal{V} and an honest prover \mathcal{P} . Consider any number of arbitrary interleaved executions of the protocol for the same or different values a, C_x and C_c . Whatever information \mathcal{V} can compute afterwards, she can also compute it using her a-priori information and the fact that \mathcal{P} knows values r_x, c and r_c in \mathbb{Z}_q such that $C_c = \text{Com}_{(g_1, g_2)}(c; r_c)$ and $C_x = \text{Com}_{(g_1, g_2)}(a^c; r_x)$.*

Proof. Let S be \mathcal{V} 's view on one execution of the protocol. For each tuple $(c, a^c) \in \mathbb{Z}_q^2$, there is exactly one tuple $(r_c, r_x) = (\log_{g_2}(C_c g_1^{-c}), \log_{g_2}(C_x g_1^{-a^c})) \in \mathbb{Z}_q^2$ such that $C_x = \text{Com}_{(g_1, g_2)}(a^c; r_x)$ and $C_c = \text{Com}_{(g_1, g_2)}(c; r_c)$. Based on the combined tuple (c, r_c, a^c, r_x) , there is exactly one set of random choices that \mathcal{P} could have made during the execution of the protocol such that \mathcal{V} 's resulting view is S .

\mathcal{V} 's view S on the protocol is a tuple $(C_{c_0}, \dots, C_{c_{|q|-1}}, C_{u_0}, \dots, C_{u_{|q|-1}}, S_1)$, where S_1 is \mathcal{V} 's view on step 2 of the protocol. \mathcal{P} 's set of random choices X is a tuple $(d_0, \dots, d_{|q|-1}, w_0, \dots, w_{|q|-2}, X_1)$, where X_1 contains \mathcal{P} 's random choices during step 2 of the protocol. Values d_i ($i = 0, \dots, |q| - 1$) and w_j ($j = 0, \dots, |q| - 2$) are uniquely determined as $d_i = \log_{g_2}(C_{c_i} g_1^{-c_i})$ and $w_i = \log_{g_2}(C_{u_i} g_1^{-u_i})$. It remains to show that X_1 is uniquely determined by S .

We first spell out the complete proof protocol in full detail. Complement bit \bar{c}_i is defined as $\bar{c}_i = 1 - c_i$ for $i \in \{0, \dots, |q| - 1\}$.

1. \mathcal{P} picks random values $vs_0, vs_{i,j_i}, vt_{i,j_i}, e_{i,k_i}, rs_{i,k_i}, rt_{i,k_i} \in_{\mathcal{R}} \mathbb{Z}_q$ for all $i \in \{0, \dots, |q| - 1\}$, $j_i = c_i$ and $k_i = \bar{c}_i$. She then computes

$$\begin{aligned} as_0 &= g_2^{vs_0} \\ as_{i,j_i} &= g_2^{vs_{i,j_i}} \quad (\forall i \in \{0, \dots, |q| - 1\}, j_i = c_i) \\ at_{i,j_i} &= g_2^{vt_{i,j_i}} \quad (\forall i \in \{0, \dots, |q| - 1\}, j_i = c_i) \\ as_{i,k_i} &= (C_{c_i} g_1^{-\bar{c}_i})^{-e_{i,k_i}} g_2^{rs_{i,k_i}} \quad (\forall i \in \{0, \dots, |q| - 1\}, k_i = \bar{c}_i) \\ at_{i,k_i} &= (C_{u_i} C_{u_{i-1}}^{-a^{(2^i \bar{c}_i)}})^{-e_{i,k_i}} g_2^{rt_{i,k_i}} \quad (\forall i \in \{0, \dots, |q| - 1\}, k_i = \bar{c}_i). \end{aligned}$$

Values $as_0, as_{i,j}, at_{i,j}$ for all $i \in \{0, \dots, |q| - 1\}$ and $j \in \{0, 1\}$ are sent to \mathcal{V} .

2. \mathcal{V} picks a random value $e \in_{\mathcal{R}} \mathbb{Z}_q$ and sends it to \mathcal{P} .

3. \mathcal{P} computes the following values.

$$\begin{aligned} rs_0 &= e \left(\sum_{i=0}^{|q|-1} d_i 2^i - r_c \right) + vs_0 \pmod q \\ e_{i,j_i} &= e - e_{i,k_i} \pmod q \quad (\forall i \in \{0, \dots, |q| - 1\}, j_i = c_i, k_i = \bar{c}_i) \\ rs_{i,j_i} &= e_{i,j_i} d_i + vs_{i,j_i} \pmod q \quad (\forall i \in \{0, \dots, |q| - 1\}, j_i = c_i) \\ rt_{i,j_i} &= e_{i,j_i} (w_i - a^{(2^i c_i)} w_{i-1}) + vt_{i,j_i} \pmod q \quad (\forall i \in \{0, \dots, |q| - 1\}, j_i = c_i \text{ and } w_{-1} = 0) \end{aligned}$$

Values $e_{i,1}, rs_0, rs_{i,j}, rt_{i,j}$ for all $i \in \{0, \dots, |q| - 1\}$, $j \in \{0, 1\}$, are then sent to \mathcal{V} .

4. \mathcal{V} computes $e_{i,0} = e - e_{i,1} \bmod q$ for all $i \in \{0, \dots, |q| - 1\}$ and accepts the protocol if the following equations holds.

$$\begin{aligned}
(C_c^{-1} \prod_{i=0}^{|q|-1} C_{c_i}^{2^i})^{-e} g_2^{rs_0} &\stackrel{?}{=} as_0 \\
C_{c_i}^{-e_{i,0}} g_2^{rs_{i,0}} &\stackrel{?}{=} as_{i,0} \quad (\forall i \in \{0, \dots, |q| - 1\}) \\
(C_{c_i} g_1^{-1})^{-e_{i,1}} g_2^{rs_{i,1}} &\stackrel{?}{=} as_{i,1} \quad (\forall i \in \{0, \dots, |q| - 1\}) \\
(C_{u_i} C_{u_{i-1}}^{-1})^{-e_{i,0}} g_2^{rt_{i,0}} &\stackrel{?}{=} at_{i,0} \quad (\forall i \in \{0, \dots, |q| - 1\}) \\
(C_{u_i} C_{u_{i-1}}^{-a^{2^i}})^{-e_{i,1}} g_2^{rt_{i,1}} &\stackrel{?}{=} at_{i,1} \quad (\forall i \in \{0, \dots, |q| - 1\})
\end{aligned}$$

\mathcal{V} 's view on step 2 of the protocol is $S_1 = (A, E, R)$ with $A = (as_0, (as_{i,j}, at_{i,j})_{i \in \{0, \dots, |q|-1\}, j \in \{0,1\}})$, $E = (e)$ and $R = (e_{0,1}, \dots, e_{|q|-1,1}, rs_0, (rs_{i,j}, rt_{i,j})_{i \in \{0, \dots, |q|-1\}, j \in \{0,1\}})$. \mathcal{P} 's random choices X_1 comprise the tuple $(vs_0, (vs_{i,j_i}, vt_{i,j_i}, e_{i,k_i}, rs_{i,k_i}, rt_{i,k_i})_{i \in \{0, \dots, |q|-1\}, j_i = c_i, k_i = \bar{c}_i})$. These values are uniquely determined as follows.

$$\begin{aligned}
vs_0 &= rs_0 - e \left(\sum_{i=0}^{|q|-1} d_i 2^i - r_c \right) \bmod q \\
vs_{i,j_i} &= rs_{i,j_i} - e_{i,j_i} d_i \bmod q \quad (\forall i \in \{0, \dots, |q| - 1\}, j_i = c_i) \\
vt_{i,j_i} &= rt_{i,j_i} - e_{i,j_i} (w_i - a^{(2^i c_i)} w_{i-1}) \bmod q \quad (\forall i \in \{0, \dots, |q| - 1\}, j_i = c_i) \\
e_{i,k_i} &= e - e_{i,j_i} \bmod q \quad (\forall i \in \{0, \dots, |q| - 1\}, k_i = \bar{c}_i, j_i = c_i) \\
rs_{i,k_i} &= \log_{g_2}(as_{i,k_i} (C_{c_i} g_1^{-\bar{c}_i})^{e_{i,k_i}}) \quad (\forall i \in \{0, \dots, |q| - 1\}, k_i = \bar{c}_i) \\
rt_{i,k_i} &= \log_{g_2}(at_{i,k_i} (C_{u_i} C_{u_{i-1}}^{-a^{(2^i \bar{c}_i)}})^{e_{i,k_i}}) \quad (\forall i \in \{0, \dots, |q| - 1\}, k_i = \bar{c}_i)
\end{aligned}$$

Note that in these equations values $e_{i,0}$ for $i = 1, \dots, |q| - 1$ are uniquely determined as $e_{i,0} = e - e_{i,1} \bmod q$.

We conclude the proof with the observation that our argumentation remains valid even when multiple protocol executions are arbitrary interleaved. \square

Note that this proof protocol implicitly includes the well-known proof technique of Schoenmakers [2, p129] for proving that a committed value c lies in an interval $[0, 2^{|q|} - 1]$. This technique consists of first committing to every bit c_i of c , and then proving that (1) these committed values c_i are either 0 or 1 and (2) they constitute a binary representation of c .

Hence, by limiting the number of bits c_i and the number of intermediate values u_i in the proof protocol to k and $k - 1$ respectively for a public value k . The proof additionally demonstrates that value c lies within an interval $[0, 2^k - 1]$. The resulting protocol for an exponentiation with a hidden exponent which lies in an interval $[0, 2^k - 1]$ will be abbreviated as $PK\{(\varsigma, \rho_c, \rho_x) : C_c = \text{Com}_{(g_1, g_2)}(\varsigma; \rho_c) \wedge C_x = \text{Com}_{(g_1, g_2)}(a^\varsigma, \rho_x) \wedge \varsigma \in [0, 2^k - 1]\}$. It has similar security properties as those stated in Propositions 2 and 3.

3.3 Proving one step in the hashtree

The protocol of Section 3.2 can now be used to prove the relation between two children and their parent in the hashtree. Let H_t for $t = (q, q', g_a, g_b, g_c, g_d)$ be a hashfunction as defined in Section 3.1. \mathcal{P} must now demonstrate her knowledge of values $h, x_1, x_2, z_a, z_b, z_c$ and z_d , hidden into commitments $C_h, C_{x_1}, C_{x_2}, C_{z_a}, C_{z_b}, C_{z_c}$ and C_{z_d} respectively, such that relations $z = (z_a, z_b, z_c, z_d)$ and $(h = H_t(x_1, x_2, z) \vee h = H_t(x_2, x_1, z))$ are satisfied. Let the prover's publicly known commitments to h, x_1, x_2 and z_i for $i \in \{a, b, c, d\}$ be $C_h = \text{Com}_{(g_1, g_2)}(h; r_h)$, $C_{x_1} = \text{Com}_{(g_1, g_2)}(x_1; r_1)$, $C_{x_2} = \text{Com}_{(g_1, g_2)}(x_2; r_2)$ and $C_{z_i} = \text{Com}_{(g_1, g_2)}(z_i; r_i)$ respectively, with $r_h, r_1, r_2, r_a, r_b, r_c, r_d \in_{\mathcal{R}} \mathbb{Z}_q$. The proof protocol is as follows.

1. \mathcal{P} computes values $a_1 = g_a^{z_a} \bmod q$, $a_2 = g_b^{z_b} \bmod q$, $a_3 = g_c^{z_c} \bmod q$, $a_4 = g_d^{z_d} \bmod q$, $e_1 = a_1 a_2 \bmod q$ and $e_2 = e_1 a_3 \bmod q$. She then generates commitments $C_{a_1} = \text{Com}_{(g_1, g_2)}(a_1; b_1)$, $C_{a_2} = \text{Com}_{(g_1, g_2)}(a_2; b_2)$, $C_{a_3} = \text{Com}_{(g_1, g_2)}(a_3; b_3)$, $C_{a_4} = \text{Com}_{(g_1, g_2)}(a_4; b_4)$ and $C_{e_1} = \text{Com}_{(g_1, g_2)}(e_1; f_1)$, $C_{e_2} = \text{Com}_{(g_1, g_2)}(e_2; f_2)$, where b_1, b_2, b_3, b_4, f_1 and f_2 are random values in \mathbb{Z}_q . All commitments are sent to \mathcal{V} .
2. The following proof of knowledge is executed, \mathcal{V} accepts only if she accepts the proof.

$PK\{(\zeta_a, \zeta_b, \zeta_c, \zeta_d, \rho_a, \rho_b, \rho_c, \rho_d, \alpha_2, \alpha_3, \alpha_4, \beta'_1, \beta'_2, \beta'_3, \beta'_4, \beta_2, \beta_3, \beta_4, \tau_1, \tau_2, \tau_3, t_4, t_5, t_6, t_7) :$

$$(C_{z_a} = \text{Com}_{(g_1, g_2)}(\zeta_a; \rho_a) \wedge C_{a_1} = \text{Com}_{(g_1, g_2)}(g_a^{\zeta_a}; \beta'_1) \wedge \zeta_a \in [0, 2^{|m|} - 1]) \wedge \quad (4)$$

$$(C_{z_b} = \text{Com}_{(g_1, g_2)}(\zeta_b; \rho_b) \wedge C_{a_2} = \text{Com}_{(g_1, g_2)}(g_b^{\zeta_b}; \beta'_2) \wedge \zeta_b \in [0, 2^{|m|} - 1]) \wedge \quad (5)$$

$$(C_{z_c} = \text{Com}_{(g_1, g_2)}(\zeta_c; \rho_c) \wedge C_{a_3} = \text{Com}_{(g_1, g_2)}(g_c^{\zeta_c}; \beta'_3) \wedge \zeta_c \in [0, 2^{|m|} - 1]) \wedge \quad (6)$$

$$(C_{z_d} = \text{Com}_{(g_1, g_2)}(\zeta_d; \rho_d) \wedge C_{a_4} = \text{Com}_{(g_1, g_2)}(g_d^{\zeta_d}; \beta'_4) \wedge \zeta_d \in [0, 2^{|m|} - 1]) \wedge \quad (7)$$

$$C_{a_2} = \text{Com}_{(g_1, g_2)}(\alpha_2; \beta_2) \wedge C_{e_1} = \text{Com}_{(C_{a_1}, g_2)}(\alpha_2; \tau_1) \wedge \quad (8)$$

$$C_{a_3} = \text{Com}_{(g_1, g_2)}(\alpha_3; \beta_3) \wedge C_{e_2} = \text{Com}_{(C_{e_1}, g_2)}(\alpha_3; \tau_2) \wedge \quad (9)$$

$$C_{a_4} = \text{Com}_{(g_1, g_2)}(\alpha_4; \beta_4) \wedge C_h = \text{Com}_{(C_{e_2}, g_2)}(\alpha_4; \tau_3) \wedge \quad (10)$$

$$\begin{aligned} & [(C_{z_a}^m C_{z_b} C_{x_1}^{-1} = \text{Com}_{(g_1, g_2)}(0; \tau_4) \wedge C_{z_c}^m C_{z_d} C_{x_2}^{-1} = \text{Com}_{(g_1, g_2)}(0; \tau_5)) \vee \\ & (C_{z_a}^m C_{z_b} C_{x_2}^{-1} = \text{Com}_{(g_1, g_2)}(0; \tau_6) \wedge C_{z_c}^m C_{z_d} C_{x_1}^{-1} = \text{Com}_{(g_1, g_2)}(0; \tau_7))] \end{aligned} \quad (11)$$

We now give a brief overview of the clauses in protocol step 2. Steps 4 to 7 are executions of the subprotocol of Section 3.2. They demonstrate to \mathcal{V} that $a_1 = g_a^{z_a} \bmod q$, $a_2 = g_b^{z_b} \bmod q$, $a_3 = g_c^{z_c} \bmod q$ and $a_4 = g_d^{z_d} \bmod q$, where values a_i ($i = 1, \dots, 4$) are encoded into C_{a_i} and values $z_i \in [0, 2^{|m|} - 1]$ ($i \in \{a, b, c, d\}$) are encoded into C_{z_i} . Step 8 proves to \mathcal{V} that C_{e_1} encodes the product $e_1 = a_1 a_2 \bmod q$. Similarly, step 9 proves that C_{e_2} encodes $e_2 = e_1 a_3 \bmod q$ and step 10 proves that C_h encodes $h = e_2 a_4 \bmod q$. Hence, value h is constructed as $h = a_1 a_2 a_3 a_4 = g_a^{z_a} g_b^{z_b} g_c^{z_c} g_d^{z_d}$. Finally, during step 11, \mathcal{P} demonstrates to \mathcal{V} the clause $(x_1 = z_a m + z_b \bmod q \wedge x_2 = z_c m + z_d \bmod q) \vee (x_2 = z_a m + z_b \bmod q \wedge x_1 = z_c m + z_d \bmod q)$, where x_1 and x_2 are encoded into C_{x_1} and C_{x_2} respectively. As a result, we can conclude that either $h = H(x_1, x_2, (z_a, z_b, z_c, z_d))$ or $h = H(x_1, x_2, (z_a, z_b, z_c, z_d))$.

Remark that prover \mathcal{P} demonstrates twice her knowledge of the commitment opening information for values C_{a_2}, C_{a_3} and C_{a_4} . This might seem odd at first, but is necessary for technical reasons. To see this, note that equations 4 to 7 are demonstrated using the protocol of Section 3.2.

Although this protocol demonstrates \mathcal{P} 's knowledge of the opening information for a commitment, it does not enable \mathcal{P} to prove equality relations concerning this opening information. Therefore, an additional proof of knowledge is needed.

Proposition 4. *Under the discrete logarithm assumption in G_q , the above protocol is a perfect honest-verifier zero-knowledge proof of knowledge of committed values $h, x_1, x_2, z_a, z_b, z_c$ and z_d in \mathbb{Z}_q such that $h = H(x_1, x_2, (z_a, z_b, z_c, z_d))$ or $h = H(x_2, x_1, (z_a, z_b, z_c, z_d))$.*

Proof. Completeness and zero-knowledge are trivially true. A simulator chooses random values $C_{a_1}, \dots, C_{a_4}, C_{e_1}$ and C_{e_2} in G_q and simulates protocol step 2 by employing the simulator for the proofs of knowledge in Section 2.

To demonstrate soundness, we employ the knowledge extractors of the proofs of knowledge in Sections 2 and 3.2. Let their output be a tuple $(\zeta_a, \zeta_b, \zeta_c, \zeta_d, \rho_a, \rho_b, \rho_c, \rho_d, \beta'_1, \beta'_2, \beta'_3, \beta'_4, \alpha_2, \alpha_3, \alpha_4, \beta_2, \beta_3, \beta_4, \tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6, \tau_7)$ satisfying relations 4 to 11. Note that this output tuple must be the same tuple as known by \mathcal{P} . If this were not the case, then \mathcal{P} and the knowledge extractor together would have two openings for at least one of the commitments used in the proof protocol. This would enable them to break the discrete logarithm assumption. For similar reasons, the equalities $\alpha_2 = g_b^{\zeta_b} \bmod q, \alpha_3 = g_c^{\zeta_c} \bmod q, \alpha_4 = g_d^{\zeta_d} \bmod q$ and $\beta_i = \beta'_i \bmod q$ for $i \in \{2, 3, 4\}$ must also hold.

Suppose, without loss of generalization, that part 1 of equation 11 holds true, i.e. $(C_{z_a}^m C_{z_b} C_{x_1}^{-1} = \text{Com}_{(g_1, g_2)}(0; \tau_4) \wedge C_{z_c}^m C_{z_d} C_{x_2}^{-1} = \text{Com}_{(g_1, g_2)}(0; \tau_5))$. We will now show how to extract values r_h, r_1 and r_2 such that $C_h = \text{Com}_{(g_1, g_2)}(h; r_h)$ for $h = g_a^{\zeta_a} g_b^{\zeta_b} g_c^{\zeta_c} g_d^{\zeta_d} \bmod q, C_{x_1} = \text{Com}_{(g_1, g_2)}(x_1; r_1)$ for $x_1 = \zeta_a m + \zeta_b \bmod q$, and $C_{x_2} = \text{Com}_{(g_1, g_2)}(x_2; r_2)$ for $x_2 = \zeta_c m + \zeta_d \bmod q$.

From equations 10, 9, 8 and 4, commitment C_h can be opened as follows.

$$\begin{aligned}
C_h &\stackrel{(10)}{=} C_{e_2}^{\alpha_4} g_2^{\tau_3} \\
&\stackrel{(9)}{=} C_{e_1}^{\alpha_3 \alpha_4} g_2^{\tau_2 \alpha_4 + \tau_3} \\
&\stackrel{(8)}{=} C_{a_1}^{\alpha_2 \alpha_3 \alpha_4} g_2^{\tau_1 \alpha_3 \alpha_4 + \tau_2 \alpha_4 + \tau_3} \\
&\stackrel{(4)}{=} g_1^{\zeta_a} g_2^{\alpha_2 \alpha_3 \alpha_4} g_2^{\beta'_1 \alpha_2 \alpha_3 \alpha_4 + \tau_1 \alpha_3 \alpha_4 + \tau_2 \alpha_4 + \tau_3} \\
&= \text{Com}_{(g_1, g_2)}(g_a^{\zeta_a} g_b^{\zeta_b} g_c^{\zeta_c} g_d^{\zeta_d} \bmod q; \beta'_1 g_b^{\zeta_b} g_c^{\zeta_c} g_d^{\zeta_d} + \tau_1 g_c^{\zeta_c} g_d^{\zeta_d} + \tau_2 g_d^{\zeta_d} + \tau_3 \bmod q)
\end{aligned}$$

Furthermore, as $C_{z_i} = \text{Com}_{(g_1, g_2)}(\zeta_i; \rho_i)$ for all $i \in \{a, b, c, d\}$, the opening information for commitments C_{x_1} and C_{x_2} can be computed as

$$\begin{aligned}
C_{x_1} &= \text{Com}_{(g_1, g_2)}(\zeta_a m + \zeta_b \bmod q; \rho_a m + \rho_b - \tau_4 \bmod q), \text{ and} \\
C_{x_2} &= \text{Com}_{(g_1, g_2)}(\zeta_c m + \zeta_d \bmod q; \rho_c m + \rho_d - \tau_5 \bmod q).
\end{aligned}$$

Hence, we have extracted values $h, x_1, x_2, z_a, z_b, z_c$ and z_d in \mathbb{Z}_q , encoded into commitments $C_h, C_{x_1}, C_{x_2}, C_{z_a}, C_{z_b}, C_{z_c}$ and C_{z_d} respectively, such that $h = H(x_1, x_2, (z_a, z_b, z_c, z_d))$ or $h = H(x_2, x_1, (z_a, z_b, z_c, z_d))$. This concludes the proof. \square

Analogous to Proposition 3, we can again show that an unbounded verifier controlling the execution of different protocols in an arbitrary way, cannot learn anything more from the protocol than what is explicitly proven by \mathcal{P} .

Proposition 5. Consider a computationally unbounded verifier \mathcal{V} and an honest prover \mathcal{P} . Consider any number of arbitrary interleaved executions of the protocol for the same or different values $C_h, C_{x_1}, C_{x_2}, C_{z_a}, C_{z_b}, C_{z_c}$ and C_{z_d} . Whatever information \mathcal{V} can compute afterwards, she can also compute it using her a-priori information and the fact that \mathcal{P} knows values $h, x_1, x_2, z_a, z_b, z_c, z_d$ and $r_h, r_1, r_2, r_a, r_b, r_c, r_d$ in \mathbb{Z}_q such that

$$\begin{aligned} C_h &= \text{Com}_{(g_1, g_2)}(h; r_h), \\ C_{x_1} &= \text{Com}_{(g_1, g_2)}(x_1; r_1), \\ C_{x_2} &= \text{Com}_{(g_1, g_2)}(x_2; r_2), \\ C_{z_i} &= \text{Com}_{(g_1, g_2)}(z_i; r_i) \quad \forall i \in \{a, b, c, d\}, \text{ and} \\ &(h = H(x_1, x_2, z) \text{ or } h = H(x_2, x_1, z)) \text{ for } z = (z_a, z_b, z_c, z_d) \end{aligned}$$

We briefly sketch the proof of this Proposition, it is analogous to the proof of Proposition 3.

Let S be \mathcal{V} 's view on one execution of the protocol. For each tuple $(h, x_1, x_2, z_a, z_b, z_c, z_d)$ satisfying the relation $(h = H(x_1, x_2, z) \vee h = H(x_2, x_1, z))$ with $z = (z_a, z_b, z_c, z_d)$, there exists exactly one tuple $(r_h, r_1, r_2, r_a, r_b, r_c, r_d)$ such that $C_h = \text{Com}_{(g_1, g_2)}(h; r_h)$, $C_{x_i} = \text{Com}_{(g_1, g_2)}(x_i; r_i)$ for $i \in \{1, 2\}$ and $C_{z_j} = \text{Com}_{(g_1, g_2)}(z_j; r_j)$ for $j \in \{a, b, c, d\}$. Namely $r_h = \log_{g_2}(C_h g_1^{-h})$, $r_{x_i} = \log_{g_2}(C_{x_i} g_1^{-x_i})$ for $i \in \{1, 2\}$ and $r_{z_j} = \log_{g_2}(C_{z_j} g_1^{-z_j})$ for $j \in \{a, b, c, d\}$. Based on the combined tuple $(h, x_1, x_2, z_a, \dots, z_d, r_h, r_1, r_2, r_a, \dots, r_d)$, there is exactly one set of random choices that \mathcal{P} could have made during the execution of the protocol such that \mathcal{V} 's resulting view is S . The proof of this statement uses similar techniques as used in the proof of Proposition 3 and will not be repeated. Finally, we note that this argumentation remains valid even when multiple protocol executions are arbitrary interleaved.

In the remainder, this protocol will be abbreviated by adding a clause $\text{Hashstep}_H(C_h, C_{x_1}, C_{x_2}, C_{z_a}, C_{z_b}, C_{z_c}, C_{z_d})$ to the statement that is to be proven. H refers to the hashfunction that is used, while $C_h, C_{x_1}, C_{x_2}, C_{z_a}, C_{z_b}, C_{z_c}, C_{z_d}$ are commitments to the output and input values of an equation $h = H(x_1, x_2, (z_a, z_b, z_c, z_d))$.

3.4 Proving that a committed value is on a list

Suppose we have a list L of length l , a hashtree \mathcal{H} for L based on a hashfunction H , and a commitment $C_{h_0} = \text{Com}_{(g_1, g_2)}(h_0; s_0)$ with $h_0 \in L$. We are now ready to show how \mathcal{P} can prove list membership to \mathcal{V} . Set $w = \lceil \log_2 l \rceil$ and let (h_0, \dots, h_{w-1}, h) be the path from h_0 up to root h , $(h_0^*, \dots, h_{w-1}^*)$ be the respective sibling nodes on this path and (z_0, \dots, z_{w-1}) with $z_i = (z_{i,a}, z_{i,b}, z_{i,c}, z_{i,d})$ for $i \in \{0, \dots, w-1\}$ be the auxiliary values. The protocol is as follows.

1. \mathcal{P} generates random values $s_1, \dots, s_{w-1}, s_0^*, \dots, s_{w-1}^*, s_h$ in \mathbb{Z}_q and computes commitments $C_{h_i} = \text{Com}_{(g_1, g_2)}(h_i; s_i)$ for all $i \in \{1, \dots, w-1\}$, $C_{h_i^*} = \text{Com}_{(g_1, g_2)}(h_i^*; s_i^*)$ for all $i \in \{0, \dots, w-1\}$, and $C_h = \text{Com}_{(g_1, g_2)}(h; s_h)$. She also generates commitments $C_{z_{i,j}} = \text{Com}_{(g_1, g_2)}(z_{i,j}; s_{i,j})$ with randomness $s_{i,j} \in_{\mathcal{R}} \mathbb{Z}_q$ for all $i \in \{0, \dots, w-1\}$ and $j \in \{a, b, c, d\}$. All commitments are sent to \mathcal{V} .

2. \mathcal{P} and \mathcal{V} engage into the following proof protocol. \mathcal{V} accepts only if she accepts the proof.

$$\begin{aligned}
PK\{(\sigma_h) : \\
& \text{HashStep}_H(C_{h_1}, C_{h_0}, C_{h_0^*}, C_{z_{0,a}}, C_{z_{0,b}}, C_{z_{0,c}}, C_{z_{0,d}}) \wedge \dots \wedge \\
& \text{HashStep}_H(C_h, C_{h_{w-1}}, C_{h_{w-1}^*}, C_{z_{(w-1),a}}, C_{z_{(w-1),b}}, C_{z_{(w-1),c}}, C_{z_{(w-1),d}}) \wedge \\
& C_h = \text{Com}_{(g_1, g_2)}(h; \sigma_h) \}
\end{aligned}$$

Proposition 6. *Under the discrete logarithm assumption in G_q , the protocol is a perfect honest-verifier zero-knowledge proof of knowledge of a value h_0 , committed into C_{h_0} , such that $h_0 \in L$.*

Completeness and zero-knowledge can easily be seen to hold. Soundness follows from the soundness of the both intermediate Hashstep protocols and the proof that C_h is a commitment to root h of \mathcal{H} , the binding property of the Pedersen commitment scheme and the collision intractability property of H .

The following property can also be shown to hold. It results from Propositions 3 and 5.

Proposition 7. *Consider a computationally unbounded verifier \mathcal{V} and an honest prover \mathcal{P} . Consider any number of arbitrary interleaved executions of the protocol for the same or different lists L and commitments C_{h_0} . Whatever information \mathcal{V} can compute afterwards, she can also compute it using her a-priori information and the fact that \mathcal{P} knows values h_0 and s_0 with $C_{h_0} = \text{Com}_{(g_1, g_2)}(h_0; s_0)$ and $h_0 \in L$.*

4 Evaluation and extensions

Efficiency analysis. We are now ready to analyze the efficiency of this membership proof. Hereto, we evaluate the communication complexity and the number of exponentiations in G_q . We neglect multiplications and additions, as their demand on computational resources is many orders of magnitude smaller. Furthermore, we assume \mathcal{V} to be in possession of $\log_{g_1} g_2$. Note that the latter does not affect the prover's security.

Recall that $|q|$ denotes the bitlength of q . Based on the observation that $|m| = |q|/2$, the prover performs an average of $(21|q| + 40)\lceil \log_2 l \rceil + 1$ exponentiations. All of these computations can be precomputed using the technique of error correction factors in Brands [2, Section 4.5.2]. \mathcal{V} in turn performs $(18|q| + 29)\lceil \log_2 l \rceil + 2$ exponentiations in G_q . Furthermore, $(12|q| + 22)\lceil \log_2 l \rceil + 1$ elements in G_q , and $(10|q| + 18)\lceil \log_2 l \rceil + 2$ elements in \mathbb{Z}_q are communicated between the participants.

Table 1 gives a comparison of the complexity of our scheme with that of the naive technique of proving a statement of the form $\bigvee_{x_i \in L} (x \neq x_i)$. The bitsize of the parameters is set to $|q'| = 800$ and $|q| = 1600$. Values in G_q are represented using 1601 bits. This is achieved by taking as G_q a subgroup of \mathbb{Z}_p for primes p and q such that $p = 2q + 1$. The best complexity measures are underlined. It is clear that for small to relatively large list sizes, the naive linear approach is still the best solution. However, for very large lists of size 300 000 or more, our hashtree technique is recommendable.

n -ary trees. Instead of using a binary tree for the whitelist, one might think of employing n -ary trees for a value $n > 2$. Hashfunction H and its instance generator (I, D) can easily be adapted to suit this requirement. In particular, to construct a hashfunction $H_t^{(n)}$ suitable for creating n -ary

Table 1: Complexity comparison of our hashtree technique with the naive technique of proving a statement of the form $\bigvee_{i=1}^l (x \neq x_i)$ for different sizes of l , $|q'| = 800$ and $|q| = 1600$.

l	expon. \mathcal{P}		expon. \mathcal{V}		commun.	
	naive	hashtree	naive	hashtree	naive	hashtree
1000	<u>2998</u>	336 401	<u>3000</u>	288 292	<u>0.38 MB</u>	67.24 MB
10 000	<u>29 998</u>	470 961	<u>30 000</u>	403 608	<u>3.81 MB</u>	94.13 MB
100 000	<u>299 998</u>	571 881	<u>300 000</u>	490 095	<u>38.16 MB</u>	114.30 MB
200 000	<u>599 998</u>	605 521	600 000	<u>518 924</u>	<u>76.31 MB</u>	121.03 MB
300 000	899 998	<u>639 161</u>	900 000	<u>547 753</u>	<u>114.48 MB</u>	127.75 MB
400 000	1 199 998	<u>639 161</u>	1 200 000	<u>547 753</u>	152.63 MB	<u>127.75 MB</u>
500 000	1 499 998	<u>639 161</u>	1 500 000	<u>547 753</u>	190.79 MB	<u>127.75 MB</u>

trees, algorithm I is adapted to output a tuple $t = (q, q', g_{1,a}, g_{1,b}, \dots, g_{n,a}, g_{n,b})$, where q, q' are defined as in Section 3.1 and $g_{1,a}, \dots, g_{n,b} \in_{\mathcal{R}} G_{q'}$. Furthermore, algorithm D is modified to pick an arbitrary value (x_1, \dots, x_n, z) from domain $D_t^{(n)} = \{(x_1, \dots, x_n, z) \in \mathbb{Z}_q \times \dots \times \mathbb{Z}_q \times (Z_m)^{2n} \mid z = (z_{1,a}, z_{1,b}, \dots, z_{n,a}, z_{n,b}) \wedge x_i \equiv (z_{i,a}m + z_{i,b}) \pmod{q} (\forall i \in \{1, \dots, n\})\}$. The resulting function $H_t^{(n)}$ is defined as

$$H_t^{(n)} : D_t^{(n)} \rightarrow G_{q'}$$

$$(x_1, \dots, x_n, (z_{1,a}, z_{1,b}, \dots, z_{2,a}, z_{2,b})) \mapsto (g_{1,a})^{z_{1,a}} (g_{1,b})^{z_{1,b}} \dots (g_{n,a})^{z_{n,a}} (g_{n,b})^{z_{n,b}}.$$

It can be easily shown that $H_t^{(n)}$ is a collision intractable hashfunction.

The protocols of section 3 can be generalized to conduct a membership proof using n -ary trees. A hashstep then consists of a proof of knowledge of committed values $h, x, x_2, \dots, x_n, z_{1,a}, \dots, z_{n,b}$, such that $z = (z_{1,a}, z_{1,b}, \dots, z_{n,a}, z_{n,b})$ and $(h = H(x, x_2, \dots, x_n, z) \vee h = H(x_2, x, x_3, \dots, x_n) \vee \dots \vee h = H(x_2, \dots, x_n, x, z))$. For a list of length l , the prover performs an average of $(2n^2 + 10, 5n|q| + 20n - 8) \lceil \log_n l \rceil + 1$ exponentiations in G_q . Verifier \mathcal{V} performs $(2n^2 + 9n|q| + 13n - 5) \lceil \log_n l \rceil + 2$ exponentiations in G_q , and a total number of $(n^2 + 6n|q| + 11n - 4) \lceil \log_n l \rceil + 1$ elements in G_q and $(n^2 + 9n - 4 + 5n|q|) \lceil \log_n l \rceil + 2$ elements in \mathbb{Z}_q are communicated.

Hence, we can see a trade-off between the increase in work for each hashstep and the decrease of the number of hashsteps. In specific cases for l , choosing an $n > 2$ can lead to better performance measures. However, in general, performance benefits for an $n > 2$ are rather small and sometimes even non-existent as $\log_n l$ decreases only slowly for increasing n .

Other commitment schemes. The proof protocols can be adapted to fit other commitment schemes, such as the following RSA-based commitment scheme [2]. Given an RSA modulus N , prime $v < N$ with $\gcd(v, \phi(N)) = 1$ and random value $g_1 \in_{\mathcal{R}} \mathbb{Z}_N^*$. A commitment $C_x = \text{Com}_{(g_1, v)}(x; r_x)$ to a value $x \in \mathbb{Z}_v$ is created as $C_x = g_1^x r_x^v \pmod{N}$ for $r_x \in_{\mathcal{R}} \mathbb{Z}_N^*$. The scheme is unconditionally hiding and computationally binding under the RSA assumption. The proofs of knowledge will be somewhat less efficient than in the case of DL based commitments, however, as the prover has to perform twice as many on-line multi-exponentiations. On the other hand, the verifier may be provided with the trapdoor information $v^{-1} \pmod{\phi(N)}$. This allows for formula 1 in

Section 3.2 and the formula $C_h = \text{Com}_{(g_1, g_2)}(h; \sigma_h)$ in Section 3.4, which in the RSA-based setting have a structure $PK\{(\alpha) : C = \alpha^v\}$ for values $C, \alpha \in \mathbb{Z}_N^*$, to be removed from the zero-knowledge proof protocol. Instead, the user sends α to \mathcal{V} , who accepts if $\alpha = C^{v^{-1}}$ and if she accepts the remaining proofs. Note that this extra information does not help \mathcal{V} , as she could have computed it herself from the correctness of the proven formula.

The integer commitment scheme of Damgård and Fujisaki [14] could also be used for creating the commitments and conducting the proofs. In this case, however, we have to rely on an extra security assumption (for example, the strong RSA assumption). Furthermore, the system is only statistical zero-knowledge, and the proof protocols need to be extended to prove relations in \mathbb{Z}_q instead of in \mathbb{Z} .

5 Acknowledgement

This research was executed when the first author was visiting McGill University, School of Computer Science in Montreal (Canada) under the guidance of Dr. Stefan Brands. Liesje Demuyneck is supported by a research assistantship and travel credit from the Fund for Scientific Research, Flanders (Belgium). The authors would like to thank Dr. Stefan Brands and Mohamed Layouni for fruitful discussions and comments on the subject.

References

- [1] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO*, pages 255–270, 2000.
- [2] S. A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Cambridge, MA, USA, 2000.
- [3] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*, pages 93–118, 2001.
- [4] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO*, pages 61–76, 2002.
- [5] Jan Camenisch and Markus Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *EUROCRYPT*, pages 107–122, 1999.
- [6] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In *CRYPTO*, pages 410–424, 1997.
- [7] Dario Catalano, Rosario Gennaro, Nick Howgrave-Graham, and Phong Q. Nguyen. Paillier’s cryptosystem revisited. In *ACM Conference on Computer and Communications Security*, pages 206–214, 2001.
- [8] Melissa Chase, Alexander Healy, Anna Lysyanskaya, Tal Malkin, and Leonid Reyzin. Mercurial commitments with applications to zero-knowledge sets. In *EUROCRYPT*, pages 422–439, 2005.
- [9] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *CRYPTO*, pages 89–105, 1992.

- [10] David Chaum, Eugène van Heijst, and Birgit Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. In *CRYPTO*, pages 470–484, 1991.
- [11] David Chaum and Eugène van Heyst. Group signatures. In *EUROCRYPT*, pages 257–265, 1991.
- [12] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO*, pages 174–187, 1994.
- [13] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT*, pages 418–430, 2000.
- [14] Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *ASIACRYPT*, pages 125–142, 2002.
- [15] Joe Kilian and Erez Petrank. Identity escrow. In *CRYPTO*, pages 169–185, 1998.
- [16] Silvio Micali, Michael O. Rabin, and Joe Kilian. Zero-knowledge sets. In *FOCS*, pages 80–91, 2003.
- [17] Lan Nguyen. Accumulators from bilinear pairings and applications. In *CT-RSA*, pages 275–292, 2005.
- [18] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, pages 129–140, 1991.
- [19] Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano, and Moti Yung. On monotone formula closure of szk . In *FOCS*, pages 454–465, 1994.
- [20] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.