Zero-knowledge Argument for Polynomial Evaluation with Application to Blacklists[*]

Stephanie Bayer and Jens Groth[**]

University College London
{s.bayer,j.groth}@cs.ucl.ac.uk

Abstract. Verification of a polynomial's evaluation in a secret committed value plays a role in cryptographic applications such as non-membership or membership proofs. We construct a novel special honest verifier zero-knowledge argument for correct polynomial evaluation. The argument has logarithmic communication cost in the degree of the polynomial, which is a significant improvement over the state of the art with cubic root complexity at best. The argument is relatively efficient to generate and very fast to verify compared to previous work. The argument has a simple public-coin 3-move structure and only relies on the discrete logarithm assumption.
The polynomial evaluation argument can be used as a building block to construct zero-knowledge membership and non-membership arguments with communication that is logarithmic in the size of the blacklist. Non-membership proofs can be used to design anonymous blacklisting schemes allowing online services to block misbehaving users without learning the identity of the user. They also allow the blocking of single users of anonymization networks without blocking the whole network.

Keywords: Zero-knowledge argument, discrete logarithm, polynomial evaluation, anonymous blacklisting, membership and non-membership proofs.

1 Introduction

In many cryptographic applications a party wants to prove possession of a secret value u that fulfills a certain property. Since polynomials are widely used a natural question is for instance given a polynomial $P(X)$ and a value v whether the secret value u satisfies $P(u) = v$ in a prime order field $\mathbb{Z}_p$.

We propose a special honest verifier zero-knowledge argument of knowledge for two committed values u, v satisfying $P(u) = v$ for a given polynomial $P(X)$ of degree D. The argument has the following properties:

[*] ©IACR 2013. This article is the final version submitted by the authors to the IACR and to Springer-Verlag on March 8, 2013. The version published by Springer-Verlag is available at http://dx.doi.org/10.1007/978-3-642-38348-9_38

[**] The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 307937 and the Engineering and Physical Sciences Research Council grant EP/G013829/1.

– It is based on the discrete logarithm assumption in a prime order group
– It has a standard 3-round public coin structure and a common reference string with just a few group elements
– Communication grows logarithmically in the degree of the polynomial
– Both the prover and the verifier are computationally efficient
– We have a working implementation, which gives us real life performance data

As a concrete application of our polynomial evaluation argument we will look at blacklisting anonymous users, which is a notoriously difficult cryptographic problem. Anonymizing networks such as Tor [35] allow a user to hide her IP address and are used by a number of groups including undercover police agents, abuse victims and citizens living under dictatorships. During the Arab Spring for instance the Tor network experienced a spike in users from Libya and Egypt [13]. However, anonymous access to services can also lead to abuse. Wikipedia for instance allows anonymous postings, but blocks the IP address of misbehaving users. This crude solution means that if one user of Tor abuses Wikipedia, all users whose traffic comes out of the same Tor relay with this IP-address are blocked. So one misbehaving user causes many innocent users to be punished. Johnson et al. [20] suggested the Nymble system to deal with this problem. In this alternative solution IP-addresses are not blocked but instead each user anonymously proves that she has not been blacklisted. Using the polynomial evaluation argument we construct a non-membership proof, which enables a user to efficiently prove that she has not been blacklisted.

We can also use our polynomial evaluation argument to construct efficient membership proofs. Membership proofs are useful when operating a whitelist access control system, or in applications such as e-voting or e-auctions where users want to prove that their votes are valid or their bids belong to a set of approved values.

1.1 Our Contribution

Our main contribution is an efficient special honest verifier zero-knowledge argument of knowledge for two secret committed values $u, v \in \mathbb{Z}_p$ satisfying $P(u) = v$ for a given polynomial $P(X) \in \mathbb{Z}_p[X]$, where p is a prime. We work over an order p group $\mathbb{G}$ and use the Pedersen commitment scheme, i.e., a commitment to u is of the form $g^u h^r$ for some $r \in \mathbb{Z}_p$. Given the coefficients of the polynomial $P(X) = \sum_{i=0}^{D} a_i X^i$ and two Pedersen commitments our zero-knowledge argument can demonstrate knowledge of openings of the commitments to values u and v such that $P(u) = v$.

Our polynomial evaluation argument is highly efficient. The communication complexity is $O(\log D)$ group and field elements, which is very small compared to the statement size of D field elements. The prover computes $O(\log D)$ exponentiations and $O(D \log D)$ multiplications in $\mathbb{Z}_p$, and the verifier calculates $O(\log D)$ exponentiations and $O(D)$ multiplications in $\mathbb{Z}_p$. The constants in the expressions are small and the argument very efficient in practice as illustrated by a concrete implementation. We refer to Sections 3 and 5 for further details on the efficiency and a comparison with previous solutions.

The argument has a simple 3-move public coin structure: the prover sends a message, the verifier picks a challenge uniformly at random from $\mathbb{Z}_p$, and the prover answers the challenge. It has perfect completeness, perfect special honest verifier zero-knowledge[1], and computational soundness, which is based on the discrete logarithm assumption in $\mathbb{G}$.

The discrete logarithm assumption is one of the most fundamental and well-studied cryptographic assumptions. There are several types of prime order groups where the discrete logarithm assumption is believed to hold, for instance an order p subgroup of $\mathbb{Z}_q^*$ where q is a large prime, or a group of points on an elliptic curve. There are examples of elliptic curve groups where group elements are roughly $|p|$ bits and the best known attacks have a complexity of $\Omega(\sqrt{p})$ group operations. In such groups a communication complexity of $O(k \log k)$ bits suffices to get a security level of 2^{-k} when $D = \mathrm{poly}(k)$ and $|p| = O(k)$.[2]

Based on the polynomial evaluation argument we then construct zero-knowledge arguments for membership and non-membership with logarithmic communication complexity. More precisely, given a Pedersen commitment c and a list of values $\mathcal{L} = \{\lambda_1, \ldots, \lambda_D\}$ we give a zero-knowledge argument of knowledge that c is a commitment to $u \in \mathcal{L}$ in the case of a membership proof or $u \notin \mathcal{L}$ in the case of a non-membership proof. Following Brands et al. [4] we do this by computing the polynomial $P(X) = \prod_{i=1}^{D}(X - \lambda_i)$ and demonstrating $P(u) = 0$ in the case of a membership proof or $P(u) \neq 0$ in the case of a non-membership proof. With our polynomial evaluation argument this requires only $O(\log D)$ communication, which is much smaller than the size of the list.

1.2 Related Work

Polynomial Evaluation Arguments. Given two committed values u, v we give a zero-knowledge argument that $P(u) = v$ for a public polynomial $P(X)$ of degree D. Kilian [21] gave a communication efficient argument for circuit satisfiability and several other general purpose zero-knowledge arguments for NP-languages exist. However, since they are not tailored for the discrete logarithm setting using them would require a costly NP-reduction.

In the prime order group setting there are already several general zero-knowledge techniques for the satisfiability of arithmetic circuits that can demonstrate the correctness of a polynomial evaluation. Using Cramer and Damgård [11] we would get a linear communication complexity for this problem and using Groth [15] we would get a communication complexity of $O(\sqrt{D})$ group elements. Using stronger assumptions and a pairing-based argument by Groth [16] we could reduce this further to a communication complexity of $O(D^{\frac{1}{3}})$ group elements.

[1] Standard techniques can be used to make the argument fully zero-knowledge against malicious adversaries at a negligible cost of a few extra group elements in the common reference string as described in Section 2.2. There is therefore no loss of generality in considering just the special honest verifier zero-knowledge case.

[2] It is uncommon to have zero-knowledge arguments for large statements where an asymptotic communication complexity of $O(k \log k)$ suffices for a security level of 2^{-k}. Hash-trees and cut-and-choose techniques give a communication complexity of $\Omega(k^2)$ for instance and RSA-type assumptions require group elements to be $k^{3-o(1)}$ bits to guard against factorization with the general number field sieve.

Fujisaki and Okamoto [14] looked at the specific problem of polynomial evaluation in an RSA-based context but their zero-knowledge argument has linear complexity. The most efficient zero-knowledge argument for correct polynomial evaluation stems from Brands et al. [4], which has a communication complexity of $O(\sqrt{D})$ group elements and is based on the discrete logarithm assumption just like our argument.

Membership Proofs and Non-membership Proofs. Proofs for set membership and non-membership for a committed $u \in \mathcal{L}$ or $u \notin \mathcal{L}$ where $\mathcal{L} = \{\lambda_1, \ldots, \lambda_D\}$ have been studied in their own right. The most straightforward approach is to prove in a one by one manner the conjunction $\lambda_1 \neq u \wedge \ldots \wedge \lambda_D \neq u$ in the case of non-membership or the disjunction $\lambda_1 = u \vee \ldots \vee \lambda_D = u$ in the case of membership. In the context of revoking members of group signature schemes Bresson and Stern [5] proposed such a solution based on the strong RSA assumption. Peng and Bao [32] gave a general discrete logarithm based zero-knowledge arguments of non-membership with linear complexity. Brands et al. [4] improved the communication complexity to $O(\sqrt{D})$ group elements and later Peng [31] gave a solution for a non-membership proof with the same complexity using techniques similar to Brands et al. [4].

Accumulators [2, 9, 37, 29, 8, 38] provide another mechanism for giving membership proofs. An accumulator is a succinct aggregate of a set of values where it is possible to issue membership proofs for each accumulated value. A party in possession of such a membership proof, typically a few group elements, can then demonstrate that the value is included in the set. Non-membership accumulators have also been proposed [22, 39]. However, most accumulators rely on a trusted party to maintain the accumulator and if corrupt this trusted party can issue arbitrary membership proofs. Furthermore, accumulators rely on cryptographic assumptions that have been less studied than the discrete logarithm problem, for instance the strong RSA assumption or the pairing-based q-SDH assumption. These assumptions also mean that group elements have to be large and once this has been factored in the accumulator-based solutions end up having larger communication than our membership and non-membership proofs for groups over elliptic or hyper-elliptic curves. The construction of Camacho et al. [6] does not rely on a trusted party and only assumes the existence of hash functions; however proofs in their setting depend logarithmically on the number of accumulated elements.

In Song's non-membership proof [34] the prover publishes a constant number of elements and the verifier checks these elements against a blacklist by carrying out a few operations for each blacklist element; several systems along these lines have been proposed [1, 3, 27]. The operations consist either of exponentiations or pairings, so this scheme places a heavy computational burden on the verifier.

Camenisch et al. [7] gave a membership proof where the elements in the set are signed by a trusted third party. Now membership can be proven with a constant number of group elements by demonstrating that the value has been signed. Related ideas have recently been used by Libert et al. [23] in the context of revoking group signatures, where a trusted third party signs representatives of sets that cover the whitelist of non-revoked users and the user gives a zero-knowledge proof of belonging to this set [28].

All these solutions suffer from similar drawbacks that accumulator-based solutions have though. They require trust in a third party to be honest when blacklisting members or signing messages, and to get efficient proofs the signatures are built from strong assumptions such as the strong RSA assumption or pairing-based assumptions.

A different approach is taken by Nymble-like systems [36, 19, 18, 26], which also rely on a trusted third party. The user obtains a pseudonym, a "nymble", from the trusted third party which is only valid for a certain time frame with one server. The blacklist consists of nymbles by misbehaving users and in [36, 19] the server simply checks if the nymble of a connecting user is contained in the blacklist. To weaken the trust in the trusted third party Lofgren and Hopper [26] use accumulators together with the Nymble setup, while Henry and Goldberg [18] rely on the techniques of Brands et al. [4] for the user to give a zero-knowledge argument for the non-membership of the blacklist.

2 Preliminaries

We write $y = A(x; r)$ when the algorithm A on input x and randomness r, outputs y. We write $y \leftarrow A(x)$ for the process of picking randomness r at random and setting $y = A(x; r)$. We also write $y \leftarrow S$ for sampling y uniformly at random from a set S. We say a function $f : \mathbb{N} \to [0, 1]$ is negligible if $f(k) = O(k^{-c})$ for every constant $c > 0$. We say $1 - f$ is overwhelming if f is negligible. We will give a security parameter k written in unary as input to all parties in our protocols. Intuitively, the higher the security parameter the more secure the protocol.

2.1 The Pedersen Commitment Scheme

The Pedersen commitment scheme [30] works as follows. First the key generation algorithm $\mathcal{G}$ on input 1^k chooses a cyclic group $\mathbb{G}$ of k-bit prime order p and random generators g, h. The commitment key is $ck = (\mathbb{G}, p, g, h)$. To commit to $a \in \mathbb{Z}_p$ the committer picks randomness $r \in \mathbb{Z}_p$ and computes

$$\mathrm{com}_{ck}(a; r) = g^a h^r.$$

The Pedersen commitment scheme is computationally binding under the discrete logarithm assumption, i.e., a non-uniform probabilistic polynomial time adversary has negligible probability of finding two different openings of the same commitment. The Pedersen commitment scheme is perfectly hiding since the commitment is uniformly distributed in $\mathbb{G}$ no matter what the value a is.

The Pedersen commitment scheme is homomorphic. For all $a, b \in \mathbb{Z}_p$ and $r, s \in \mathbb{Z}_p$

$$\mathrm{com}_{ck}(a; r) \cdot \mathrm{com}_{ck}(b; s) = g^a h^r \cdot g^b h^s = g^{a+b} h^{r+s} = \mathrm{com}_{ck}(a + b; r + s).$$

We use the Pedersen commitment scheme because of its elegance and its security resting on the discrete logarithm assumption. However, our protocols could also work with other homomorphic commitment schemes and we will describe our arguments in a way such that it would be easy to plug in another homomorphic commitment scheme.

2.2 Zero-knowledge Arguments of Knowledge

In the arguments we consider a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ both of which are probabilistic polynomial time interactive algorithms. All our protocols will be 3-move public-coin arguments; first the prover sends a message, then the verifier responds with a random challenge, and finally the prover sends an answer to the challenge.

We assume the existence of a probabilistic polynomial time setup algorithm $\mathcal{G}$ that when given a security parameter k returns a common reference string σ. In this paper the common reference string will always be a public key ck for the Pedersen commitment scheme.

Let R be a polynomial time decidable ternary relation, we call w a witness for a statement x if $(\sigma, x, w) \in R$. We define the CRS-dependent language

$$L_\sigma = \{x \mid \exists w : (\sigma, x, w) \in R\}$$

as the set of statements x that have a witness w in the relation R.

The public transcript produced by $\mathcal{P}$ and $\mathcal{V}$ when interacting on inputs s and t is denoted by $tr \leftarrow \langle \mathcal{P}(s), \mathcal{V}(t) \rangle$. The transcript consists of the initial message from the prover, the random challenge from the verifier, the answer from the prover and the decision to accept or reject from the verifier. We write $\langle \mathcal{P}(s), \mathcal{V}(t) \rangle = b$ depending on whether the verifier rejects, $b = 0$, or accepts, $b = 1$.

Definition 1 (Argument of knowledge) *The triple* $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ *is called an* argument of knowledge *for the relation R if we have completeness and witness-extended emulation as defined below.*

Definition 2 (Perfect completeness) $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ *has perfect completeness if for all non-uniform polynomial time adversaries $\mathcal{A}$*

$$\Pr[\sigma \leftarrow \mathcal{G}(1^k); (x, w) \leftarrow \mathcal{A}(\sigma) : (\sigma, x, w) \notin R \text{ or } \langle \mathcal{P}(\sigma, x, w), \mathcal{V}(\sigma, x) \rangle = 1] = 1.$$

To define an argument of knowledge we follow Groth and Ishai [17] that borrowed the term witness-extended emulation from Lindell [25]. Informally, their definition says that given an adversary that produces an acceptable argument with some probability, there exist an emulator that produces a similar argument with the same probability and at the same time provides a witness w. We give a strong black-box definition where the emulator just has black-box access to a prover and verifier's interaction, which it can rewind and run again with fresh randomness for the verifier.

Definition 3 (Computational witness-extended emulation) $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ *has* witness-extended emulation *if for all deterministic polynomial time $\mathcal{P}^*$ there exists an expected polynomial time emulator $\mathcal{X}$ such that for all non-uniform polynomial time interactive adversaries $\mathcal{A}$*

$$\begin{aligned} &\Pr[\sigma \leftarrow \mathcal{G}(1^k); (x, s) \leftarrow \mathcal{A}(\sigma); tr \leftarrow \langle \mathcal{P}^*(\sigma, x, s), \mathcal{V}(\sigma, x) \rangle : \mathcal{A}(tr) = 1] \\ \approx\ &\Pr[\sigma \leftarrow \mathcal{G}(1^k); (x, s) \leftarrow \mathcal{A}(\sigma); (tr, w) \leftarrow \mathcal{X}^{\langle \mathcal{P}^*(\sigma, x, s), \mathcal{V}(\sigma, x) \rangle}(\sigma, x) : \\ &\quad \mathcal{A}(tr) = 1 \text{ and if } tr \text{ is accepting then } (\sigma, x, w) \in R], \end{aligned}$$

where the verifier picks fresh public coin challenges in each oracle call by $\mathcal{X}^{\langle \mathcal{P}^*(\sigma, x, s), \mathcal{V}(\sigma, x) \rangle}$.

In the definition, s can be interpreted as the state of $\mathcal{P}^*$, including the randomness. So, whenever $\mathcal{P}^*$ is able to make a convincing argument when in state s, the emulator can extract a witness. This is why we call it an argument of knowledge.

Definition 4 (Public coin) *An argument $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is called* public coin *if the verifier chooses his messages uniformly at random and independently of the messages sent by the prover, i.e., the challenges correspond to the verifier's randomness ρ.*

An argument is zero-knowledge if it does not leak information about the witness beyond what can be inferred from the truth of the statement. We will present arguments that have special honest verifier zero-knowledge in the sense that if the verifier's challenge is known in advance, then it is possible to simulate the entire argument without knowing the witness.

Definition 5 (Perfect special honest verifier zero-knowledge) *A public coin argument $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is called a* perfect special honest verifier zero knowledge (SHVZK) *argument for R if there exists a probabilistic polynomial time simulator $\mathcal{S}$ such that for all interactive non-uniform polynomial time adversaries $\mathcal{A}$ we have*

$$\Pr[\sigma \leftarrow \mathcal{G}(1^k); (x, w, \rho) \leftarrow \mathcal{A}(\sigma); tr \leftarrow \langle \mathcal{P}(\sigma, x, w), \mathcal{V}(\sigma, x; \rho)\rangle : (\sigma, x, w) \in R \textit{ and } \mathcal{A}(tr) = 1]$$
$$= \Pr[\sigma \leftarrow \mathcal{G}(1^k); (x, w, \rho) \leftarrow \mathcal{A}(\sigma); tr \leftarrow \mathcal{S}(\sigma, x, \rho) : (\sigma, x, w) \in R \textit{ and } \mathcal{A}(tr) = 1],$$

where ρ is the public coin randomness used by the verifier as the challenge.

Full zero-knowledge. In real life applications special honest verifier zero-knowledge may not suffice since a malicious verifier may give non-random challenges. However, it is easy to convert an SHVZK argument into a full zero-knowledge argument secure against *arbitrary* verifiers in the common reference string model using standard techniques. The conversion can be very efficient and only costs a small additive overhead, so we will in the paper without loss of generality just focus on building efficient SHVZK arguments.

One example of such a conversion that would work in our case is the following: The common reference string is set up with an additional group element y. The prover will now use an OR-proof [12] to show that she knows a witness for the statement being true or she knows the discrete logarithm of y. Since she does not know the discrete logarithm of y this is a convincing argument of knowledge. On the other hand, we can set the simulator up such that it does know the discrete logarithm of y and now it is easy to simulate proofs. This conversion yields an argument of knowledge with perfect zero-knowledge at the price of an extra group element in the common reference string.

3 Polynomial Evaluation Argument

Given a polynomial $P(U) = \sum_{i=0}^{D} a_i U^i$ and two commitments c_0, c_v, we will describe an argument of knowledge of openings of the commitments to values u and v such that $P(u) = v$. (The notation c_0 for the commitment to $u = u^{2^0}$ matches other commitments c_j to u^{2^j} that the prover will construct in the argument.)

By padding with zero-coefficients we can without loss of generality assume $D = 2^{d+1} - 1$. It is useful to write i in binary, i.e., $i = i_d \dots i_0$ where $i_j \in \{0, 1\}$. We can then rewrite the term U^i as $U^i = U^{\sum_{j=0}^d i_j 2^j} = \prod_{j=0}^d (U^{2^j})^{i_j}$. Substituting this in the polynomial we get

$$P(U) = \sum_{i=0}^{D} a_i U^i = \sum_{i_0,\dots,i_d=0}^{1} a_{i_d \dots i_0} \prod_{j=0}^{d} (U^{2^j})^{i_j}.$$

The prover will make commitments $c_1, \dots, c_d$ to $u^{2^1}, u^{2^2}, \dots, u^{2^d}$, use standard techniques to prove they are well-formed, and prove that when inserted into the rewritten polynomial we have $\sum_{i_0,\dots,i_d=0}^{1} a_{i_d \dots i_0} \prod_{j=0}^{d} (u^{2^j})^{i_j} = v$. Since $d = \lfloor \log D \rfloor$ the prover only makes a logarithmic number of commitments, which will help keep communication low.

To show the committed powers of u in $c_0, c_1, \dots, c_d$ evaluate to the commited v the prover picks random values $f_0, \dots, f_d \leftarrow \mathbb{Z}_p$ and defines a new polynomial

$$Q(X) = \sum_{i_0,\dots,i_d=0}^{1} a_{i_d \dots i_0} \prod_{j=0}^{d} (Xu^{2^j} + f_j)^{i_j} X^{1-i_j} = X^{d+1} P(u) + X^d \delta_d + \dots + X\delta_1 + \delta_0.$$

The idea behind this choice of $Q(X)$ is that for each i_j either an Xu^{2^j} factor is included or an X factor is included so the coefficient of X^{d+1} is $P(u)$. Each f_j on the other hand is not multiplied by X and will therefore only affect the lower degree coefficients $\delta_0, \dots, \delta_d$ of $Q(X)$.

The prover will now demonstrate that the coefficient of X^{d+1} in the secret $Q(X)$ is the same as v in a way that cancels out the $\delta_0, \dots, \delta_d$ coefficients. The prover sends the verifier commitments $c_{f_0}, \dots, c_{f_d}$ to $f_0, \dots, f_d$, and commitments $c_{\delta_0}, \dots, c_{\delta_d}$ to $\delta_0, \dots, \delta_d$. Afterwards, the verifier will pick a random challenge $x \leftarrow \mathbb{Z}_p$. The prover will now open suitable products of the commitments in a way such that the verifier can check that the committed values u, v satisfy $Q(x) = x^{d+1}v + x^d \delta_d + \dots + \delta_0$. More precisely, after receiving the challenge x the prover opens each product $c_j^x c_{f_j}$ to $\bar{f}_j = xu^{2^j} + f_j$. Furthermore, the prover opens $c_v^{x^{d+1}} \prod_{j=0}^{d} c_{\delta_j}^{x^j}$ to $\bar{\delta} = \sum_{i_0,\dots,i_d=0}^{1} a_{i_0 \dots i_d} \prod_{j=0}^{d} \bar{f}_j^{\,i_j} x^{1-i_j}$. Note that the verifier can calculate $\bar{\delta}$ himself and therefore only accepts the opening if

$$\sum_{i_0,\dots,i_d=0}^{1} a_{i_0 \dots i_d} \prod_{j=0}^{d} \bar{f}_j^{\,i_j} x^{1-i_j} = x^{d+1}v + x^d \delta_d + \dots + x\delta_1 + \delta_0.$$

This has negligible probability of being true unless $P(u) = v$.

Returning to the commitments $c_1, \dots, c_d$ to $u^{2^1}, \dots, u^{2^d}$ we said the prover could use standard techniques to show that they contain the correct powers of u. To do this the prover sends some commitments c_{fu_j} to $f_j u^{2^j}$ to the verifier and later opens the commitments $c_{u_{j+1}}^x c_{u_j}^{-\bar{f}_j} c_{fu_j}$ to

$$xu^{2^{j+1}} - (xu^{2^j} + f_j)u^{2^j} + f_j u^{2^j} = 0.$$

The full polynomial evaluation argument is given below.

Common reference string: $ck \leftarrow \mathcal{G}(1^k)$
Statement: $P(U) = \sum_{i=0}^{D} a_i U^i = \sum_{i_0,\ldots,i_d=0}^{1} a_{i_d\ldots i_0} \prod_{j=0}^{d} (U^{2^j})^{i_j} \in \mathbb{Z}_p[U]$ and $c_0, c_v \in \mathbb{G}$
Prover's witness: $u, v, r_0, t \in \mathbb{Z}_p$ such that $c_0 = \mathrm{com}_{ck}(u; r_0), c_v = \mathrm{com}_{ck}(v; t)$ and $P(u) = v$
Initial message: Compute

1. $c_1 = \mathrm{com}_{ck}(u^{2^1}; r_1), \ldots, c_d = \mathrm{com}_{ck}(u^{2^d}; r_d)$ where $r_1, \ldots, r_d \leftarrow \mathbb{Z}_p$
2. $c_{f_0} = \mathrm{com}_{ck}(f_0; s_0), \ldots, c_{f_d} = \mathrm{com}_{ck}(f_d; s_d)$ where $f_0, s_0, \ldots, f_d, s_d \leftarrow \mathbb{Z}_p$
3. $\delta_0, \ldots, \delta_d \in \mathbb{Z}_p$ such that

$$\sum_{i_0,\ldots,i_d=0}^{1} a_{i_d\ldots i_0} \prod_{j=0}^{d} (Xu^{2^j} + f_j)^{i_j} X^{1-i_j} = X^{d+1} v + \sum_{j=0}^{d} X^j \delta_j$$

4. $c_{\delta_0} = \mathrm{com}_{ck}(\delta_0; t_0), \ldots, c_{\delta_d} = \mathrm{com}_{ck}(\delta_d; t_d)$ where $t_0, \ldots, t_d \leftarrow \mathbb{Z}_p$
5. $c_{fu_0} = \mathrm{com}_{ck}(f_0 u^{2^0}; \xi_0), \ldots, c_{fu_{d-1}} = \mathrm{com}_{ck}(f_{d-1} u^{2^{d-1}}; \xi_{d-1})$ where $\xi_0, \ldots, \xi_{d-1} \leftarrow \mathbb{Z}_p$

Send: $c_1, \ldots, c_d, c_{f_0}, \ldots, c_{f_d}, c_{\delta_0}, \ldots, c_{\delta_d}, c_{fu_0}, \ldots, c_{fu_{d-1}}$
Challenge: $x \leftarrow \mathbb{Z}_p$
Answer: Compute for all j

$$\bar{f}_j = xu^{2^j} + f_j \qquad \bar{r}_j = xr_j + s_j \qquad \bar{t} = x^{d+1} t + \sum_{i=0}^{d} t_i x^i \qquad \bar{\xi}_j = xr_{j+1} - \bar{f}_j r_j + \xi_j$$

Send: $\bar{f}_0, \bar{r}_0, \ldots, \bar{f}_d, \bar{r}_d, \bar{t}, \bar{\xi}_0, \ldots, \bar{\xi}_{d-1}$
Verification: Accept if and only if for all j

$$c_j^x c_{f_j} = \mathrm{com}_{ck}(\bar{f}_j; \bar{r}_j) \qquad c_{j+1}^x c_j^{-\bar{f}_j} c_{fu_j} = \mathrm{com}_{ck}(0; \bar{\xi}_j)$$

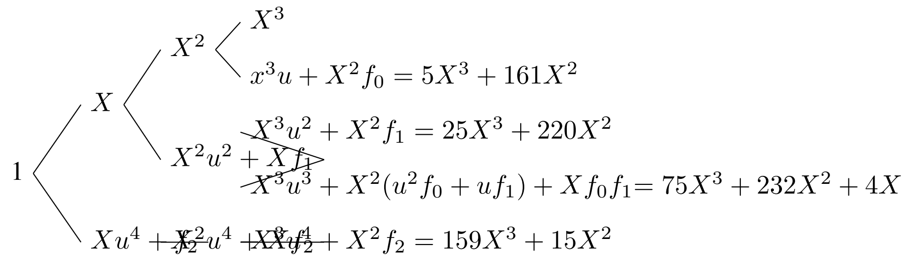
and

$$c_v^{x^{d+1}} \prod_{i=0}^{d} c_{\delta_i}^{x^i} = \mathrm{com}_{ck}\left(\sum_{i_0,\ldots,i_d=0}^{1} a_{i_d\ldots i_0} \prod_{j=0}^{d} \bar{f}_j^{\,i_j} x^{1-i_j}; \bar{t} \right)$$

Example: Let $\mathbb{G} = \langle g = 3 \rangle \subset \mathbb{Z}_{467}^*$, which has prime order $p = 233$ and let $h = 266$. The common reference string describing $(\mathbb{G}, p, g, h)$ is $ck = \{467, 233, 3, 266\}$. The statement consists of the polynomial $P(X) = 93X^4 + 3X^2 + 115X + 51 \in \mathbb{Z}_{233}[X]$ and commitments $c_{u_0} = 90, c_v = 68$. We have $d = \lfloor \log 4 \rfloor = 2$.

The prover knows values $u = 5, v = P(u) = 110 \in \mathbb{Z}_{233}$ and $r_0 = 201, t = 189 \in \mathbb{Z}_{233}$ such that $c_{u_0} = 90 \in \mathbb{G}$ and $c_v = 68 \in \mathbb{G}$. To prove knowledge of the witness the prover first picks $r_1 = 23, r_2 = 63$ at random from $\mathbb{Z}_{233}$ and computes commitments $c_1 = 387$ and $c_2 = 4$ to $u^{2^1} = 25$ and $u^{2^2} = 159$. The prover also picks $f_0 = 161, f_1 = 220, f_2 = 15, s_0 = 10, s_1 = 37, s_2 = 149$ randomly from $\mathbb{Z}_{233}$ and sets $c_{f_0} = 48, c_{f_1} = 4, c_{f_2} = 324$.

Next she computes $\delta_0, \delta_1, \delta_2$. She calculates the five products $\prod_{j=0}^{d}(Xu^{2^j} + f_j)^{i_j} X^{1-i_j}$ for $i = i_2 i_1 i_0 \in \{0,1,2,3,4\}$ using a binary tree.



1
X
X^2
X^3
$x^3u + X^2 f_0 = 5X^3 + 161X^2$
$X^2u^2 + Xf_1$
$X^3u^2 + X^2 f_1 = 25X^3 + 220X^2$
$X^3u^3 + X^2(u^2 f_0 + uf_1) + Xf_0 f_1 = 75X^3 + 232X^2 + 4X$
$Xu^4 + f_2$
$X^2u^4 + Xf_2$
$X^3u^4 + X^2 f_2 = 159X^3 + 15X^2$

The prover takes the a_i and multiplies them on the result of the binary tree, to get

$$\begin{aligned}
i = 0: &\quad a_0 \cdot X^3 = 51X^3 \\
i = 1: &\quad a_1 \cdot (5X^3 + 161X^2) = 109X^3 + 108X^2 \\
i = 2: &\quad a_2 \cdot (25X^3 + 220X^2) = 75X^3 + 194X^2 \\
i = 3: &\quad a_3 \cdot (75X^3 + 232X^2) = 0 \\
i = 4: &\quad a_4 \cdot (159X^3 + 15X^2) = 108X^3 + 230X^2
\end{aligned}$$

Last, to extract the values δ_i she adds for $i = 0,1,2$ the coefficients for each X^i to get $\delta_0 = 0, \delta_1 = 0, \delta_2 = 66 \mod 233$. Now the prover picks $t_0 = 33, t_1 = 201, t_2 = 205$ at random and commits to the δ_i's to get $c_{\delta_0} = 438, c_{\delta_1} = 329, c_{\delta_2} = 467$.

Finally, the prover calculates $f_0 u = 106, f_1 u^2 = 174$ and commits to these values. So, she picks $\xi_0 = 13, \xi_1 = 75$ and computes $c_{fu_0} = 352, c_{fu_1} = 141$. She sends all the commitments to the verifier.

The verifier returns a random challenge $x = 123 \in \mathbb{Z}_{233}$, and the prover calculates answers $\bar{f}_0 = 77, \bar{f}_1 = 33, \bar{f}_2 = 0, \bar{r}_0 = 35, \bar{r}_1 = 70, \bar{r}_2 = 209, \bar{t} = 189, \bar{\xi}_0 = 180, \bar{\xi}_1 = 75$, and sends all values to the verifier.

The verifier checks first if all commitments are in $\mathbb{G}$ and all answers valid numbers in $\mathbb{Z}_{233}$. Then he checks for $i = 0,1,2$ if $c_{u_i}^x c_{f_i} = \mathrm{com}_{ck}(\bar{f}_i; \bar{r}_i)$:

$$c_{u_0}^x c_{f_0} = 68 = \mathrm{com}_{ck}(\bar{f}_0; \bar{r}_0) \quad c_{u_1}^x c_{f_1} = 91 = \mathrm{com}_{ck}(\bar{f}_1; \bar{r}_1) \quad c_{u_2}^x c_{f_2} = 220 = \mathrm{com}_{ck}(\bar{f}_2; \bar{r}_2).$$

Next, he checks $c_{u_{i+1}}^x c_{u_i}^{-\bar{f}_i} c_{fu_i} = \mathrm{com}_{ck}(0, \bar{\xi}_i)$ for $i = 0, 1$, i.e.,

$$c_{u_1}^x c_{u_0}^{-\bar{f}_0} c_{fu_0} = 157 = \mathrm{com}_{ck}(0; \bar{\xi}_0) \quad c_{u_2}^x c_{u_1}^{-\bar{f}_1} c_{fu_1} = 250 = \mathrm{com}_{ck}(0; \bar{\xi}_1).$$

Then the verifier calculates $\bar{\delta} = \sum_{i_0,\ldots,i_d=0}^{1} a_{i_0\ldots i_d} \prod_{j=0}^{d} \bar{f}_j^{\,i_j} x^{1-i_j}$ in a binary tree fashion. The output leaves in the binary tree are $x^3 = 129, x^2\bar{f}_0 = 166, x^2\bar{f}_2 = 171, x\bar{f}_0\bar{f}_1 = 90, x^2\bar{f}_2 = 0$. He multiplies the values by the a_i's and adds the results together, to get $\bar{\delta} = a_0 129 + a_1 166 + a_2 171 + a_3 90 + a_4 0 = 86 \in \mathbb{Z}_{233}$. Finally, he checks the last verification equation $c_v^{x^3} c_{\delta_2}^{x^2} c_{\delta_1}^{x^1} c_{\delta_0}^{x^0} = 395 = \mathrm{com}_{ck}\left(\bar{\delta}; \bar{t}\right)$.

Efficiency. The communication in the polynomial evaluation argument for a degree $D = 2^{d+1} - 1$ polynomial is roughly $4d$ group elements and $3d$ field elements.

The prover uses $8d$ exponentiations to compute the commitments. She also has to calculate $\delta_0, \dots, \delta_d$ that are defined to satisfy $\sum_{i_0,\dots,i_d=0}^{1} a_{i_d \cdots i_0} \prod_{j=0}^{d} (Xu^{2^j} + f_j)^{i_j} X^{1-i_j} = X^{d+1}v + \sum_{j=0}^{d} X^j \delta_j$. The prover can calculate the D degree $d+1$ polynomials $\prod_{j=0}^{d}(Xu^{2^j} + f_j)^{i_j} X^{1-i_j}$ in a binary-tree fashion for all choices of $i_0 \dots, i_d \in \{0,1\}$ at a cost of dD multiplications in $\mathbb{Z}_p$. Multiplying with the $a_{i_d \dots i_0}$'s uses another dD multiplications. The total cost for the prover is therefore $8d$ exponentiations in $\mathbb{G}$ and $2dD$ multiplications in $\mathbb{Z}_p$.

The verifier can check the argument using $6d$ exponentiations in $\mathbb{G}$ since the exponent x is used twice in the verification equations. He also needs to compute the sum $\sum_{i_0,\dots,i_d=0}^{1} a_{i_d \cdots i_0} \prod_{j=0}^{d} \bar{f}_j^{\,i_j} x^{1-i_j}$, which can be done in a binary tree fashion for all choices of $i_0, \dots, i_d \in \{0,1\}$ using $2D$ multiplications in $\mathbb{Z}_p$.

We have ignored small constants in the calculations above and just focused on the dominant terms. Using multi-exponentiation techniques, randomized verification and other tricks it is possible to reduce the computation even further for the prover and verifier, so the estimates are quite conservative.

Theorem 6 *Assuming the discrete logarithm assumption holds the polynomial evaluation argument is a public coin perfect SHVZK argument of knowledge of openings of* c_0 *and* c_v *to* u *and* v *such that* $P(u) = v$.

Proof. Perfect completeness follows by careful inspection.

We will now argue perfect SHVZK. Given a challenge $x \in \mathbb{Z}_p$ the simulator picks $c_1, \dots, c_d, c_{\delta_1}, \dots, c_{\delta_d} \leftarrow \mathbb{G}$ and $\bar{f}_0, \bar{r}_0, \dots, \bar{f}_d, \bar{r}_d, \bar{t}, \bar{\xi}_0, \dots, \bar{\xi}_{d-1} \leftarrow \mathbb{Z}_p$ and for all j he sets $c_{f_j} = \mathrm{com}_{ck}(\bar{f}_j; \bar{r}_j) c_j^{-x} \quad c_{fu_j} = \mathrm{com}_{ck}(0; \bar{\xi}_j) c_{j+1}^{-x} c_j^{\bar{f}_j}$ and $c_{\delta_0} = \mathrm{com}_{ck}\left(\sum_{i_0,\dots,i_d=0}^{1} a_{i_d \cdots i_0} \prod_{j=0}^{d} \bar{f}_j^{\,i_j} x^{1-i_j}; \bar{t}\right) c_v^{-x^{d+1}} \prod_{i=1}^{d} c_{\delta_i}^{-x^i}$.

This is a perfect simulation. In a real argument $c_1, \dots, c_d, c_{\delta_1}, \dots, c_{\delta_d}$ are uniformly random perfectly hiding commitments as in the simulation. In a real argument $\bar{f}_0, \bar{r}_0, \dots, \bar{f}_d, \bar{r}_d, \bar{t}, \bar{\xi}_0, \dots, \bar{\xi}_{d-1} \in \mathbb{Z}_p$ are also uniformly random because of the random choice of $f_0, r_0, \dots, f_d, r_d, t_0, \xi_0, \dots, \xi_{d-1}$. Finally, both in the simulation and in the real argument given these choices the verification equations uniquely determine the values of $c_{f_0}, \dots, c_{f_d}, c_{\delta_0}$ and $c_{fu_0}, \dots, c_{fu_{d-1}}$. This means simulated and real arguments given a challenge x have identical probability distributions.

Finally, we will show that the argument has witness-extended emulation. The emulator $\mathcal{X}$ runs the argument with random challenge $x \leftarrow \mathbb{Z}_p$ and if the transcript tr is accepting it rewinds until it has $d+2$ accepting arguments. For a prover with ϵ chance of making a convincing argument we expect the emulator to rewind $\frac{d+2}{\epsilon}\epsilon = d+2$ times, so $\mathcal{X}$ runs in expected polynomial time.

There is negligible probability that the verifier will end up with two or more transcripts with the same challenge x, so we just need to be able to extract a witness when we have $d+2$ transcripts with different challenges. Given $\bar{f}_j^{(1)}, \bar{r}_j^{(1)}$ and $\bar{f}_j^{(2)}, \bar{r}_j^{(2)}$ in the first two answers to challenges x_1 and x_2 the emulator can take linear combinations of the verification equations to get openings of the commitments

c_j. More precisely, we have that the two answers satisfy $c_j^{x_1} c_{f_j} = \mathrm{com}_{ck}(\bar{f}_j^{(1)}; \bar{r}_j^{(1)})$ $c_j^{x_2} c_{f_j} = \mathrm{com}_{ck}(\bar{f}_j^{(2)}; \bar{r}_j^{(2)})$. Picking α_1, α_2 such that $\alpha_1 x_1 + \alpha_2 x_2 = 1$ and $\alpha_1 + \alpha_2 = 0$ gives us $c_j = c_j^{\alpha_1 x_1 + \alpha_2 x_2} c_{f_j}^{\alpha_1 + \alpha_2} = \mathrm{com}_{ck}(\alpha_1 \bar{f}_j^{(1)} + \alpha_2 \bar{f}_j^{(2)}; \alpha_1 \bar{r}_j^{(1)} + \alpha_2 \bar{r}_j^{(2)})$, which is an opening of c_j.

Other types of linear combinations of the verification equations give us openings of the other commitments c_{f_j}, c_{fu_j}, c_v and c_{δ_i} the prover sends in the initial message. In the case of c_{δ_i} we find the linear combination as follows: Let

$$M = \begin{pmatrix} 1 & x_1 & \dots & x_1^{d+1} \\ \vdots & & & \vdots \\ 1 & x_{d+2} & \dots & x_{d+2}^{d+1} \end{pmatrix}.$$

Since it is a Vandermonde matrix with different $x_1, \dots, x_{d+2}$ it is invertible. By taking linear combinations of the verification equations

$$c_v^{x^{d+1}} \prod_{i=0}^{d} c_{\delta_i}^{x^i} = \mathrm{com}_{ck}\left(\sum_{i_0,\dots,i_d=0}^{1} a_{i_d \dots i_0} \prod_{j=0}^{d} \bar{f}_j^{\,i_j} x^{1-i_j}; \bar{t} \right)$$

for different challenges $x_1, \dots, x_{d+2}$ we get that

$$\begin{pmatrix} \delta_0 & t_0 \\ \vdots & \vdots \\ \delta_d & t_d \\ v & t \end{pmatrix} = M^{-1} \begin{pmatrix} \sum_{i_0,\dots,i_d=0}^{1} a_{i_d \dots i_0} \prod_{j=0}^{d} (\bar{f}_j^{(1)})^{i_j} x_1^{1-i_j} & \bar{t}^{(1)} \\ \vdots & \vdots \\ \sum_{i_0,\dots,i_d=0}^{1} a_{i_d \dots i_0} \prod_{j=0}^{d} (\bar{f}_j^{(d+2)})^{i_j} x_{d+2}^{1-i_j} & \bar{t}^{(d+2)} \end{pmatrix}$$

which gives us openings of $c_{\delta_0}, \dots, c_{\delta_d}$ and c_v.

We now have openings to all the commitments. Because the commitments are binding, each answer must be computed as they are by an honest prover in the argument. Therefore, the verification equations $c_j^x c_{f_j} = \mathrm{com}_{ck}(\bar{f}_j, \bar{r}_j)$ give us $\bar{f}_j = x u_j + f_j$, where u_j is the extracted value in c_j and f_j is the extracted value in c_{f_j}.

The verification equations $c_{j+1}^x c_j^{-\bar{f}_j} c_{fu_j} = \mathrm{com}_{ck}(0; \bar{\xi}_j)$ give us that the committed values satisfy $x u_{j+1} - (x u_j + f_j) u_j + \phi_j = 0$ for $j = 0, \dots, d-1$ with ϕ_j being the value we extracted for c_{fu_j}. Since each of the polynomial equalities is of degree 1 in x and holds for $d+2$ different challenges x we see that $u_{j+1} = u_j u_j$. Since $u_0 = u$ this gives us $u_1 = u^{2^1}, u_2 = u^{2^2}, \dots, u_d = u^{2^d}$.

Turning to the verification equation

$$c_v^{x^{d+1}} \prod_{i=0}^{d} c_{\delta_i}^{x^i} = \mathrm{com}_{ck}\left(\sum_{i_0,\dots,i_d=0}^{1} a_{i_d \dots i_0} \prod_{j=0}^{d} \bar{f}_j^{\,i_j} x^{1-i_j}; \bar{t} \right)$$

we now have that this corresponds to the degree $d+1$ polynomial equation

$$X^{d+1} v + X^d \delta_d + \dots + X \delta_1 + \delta_0 = \sum_{i_0,\dots,i_d=0}^{1} a_{i_d \dots i_0} \prod_{j=0}^{d} (X u^{2^j} + f_j)^{i_j} X^{1-i_j}.$$

With $d+2$ different values $x_1, \dots, x_{d+2}$ satisfying the equation, we conclude the two polynomials are identical. Looking at the coefficient for X^{d+1} we conclude that the extracted openings of c_0 and c_v satisfy $P(u) = v$. $\square$

4 Membership and Non-membership Arguments

In this section we will construct membership and non-membership arguments for committed values being included in whitelists or excluded from blacklists. In both cases the whitelists or blacklists are given as a set $\mathcal{L} \subset \mathbb{Z}_p$, and the goal is to show that the committed value $u \in \mathcal{L}$ in the case of membership or $u \notin \mathcal{L}$ in the case of non-membership.

Let us first describe a non-membership argument for a committed value not belonging to a set $\mathcal{L} = \{\lambda_1, \dots, \lambda_D\}$ using ideas from Brands et al. [4]. We define a polynomial $P(X) = \prod_{i=1}^{D}(X - \lambda_i)$ with the elements in the set as roots. With this choice of polynomial we have $u \in \mathcal{L}$ if and only if $P(u) = 0$. The prover has a commitment c and will demonstrate that the committed value u does not belong to $\mathcal{L}$ by showing $P(u) \neq 0$.

The prover computes $v = P(u)$ and makes a commitment to v. She can now give an SHVZK argument that the new commitment contains $v = P(u)$ using the polynomial evaluation argument from Section 3. To prove non-membership she just needs to prove $v \neq 0$. To do this the prover commits to $w = v^{-1}$ and uses a multiplication argument to show $v \cdot w = 1$, which will convince the verifier that $v \neq 0$. The main cost in this argument is the polynomial evaluation argument; multiplication arguments are standard cryptographic tools [10] that only cost a couple of group elements in communication.

Common reference string: $ck \leftarrow \mathcal{G}(1^k)$
Statement: $\mathcal{L} = \{\lambda_1, \dots, \lambda_D\} \subset \mathbb{Z}_p, P(X) = \prod_{i=1}^{D}(X - \lambda_i) \in \mathbb{Z}_p[X], c \in \mathbb{G}$
Prover's witness: $u, r \in \mathbb{Z}_p$ such that $c = \mathrm{com}_{ck}(u; r)$ and $u \notin \mathcal{L}$
Argument: Pick $s, t \leftarrow \mathbb{Z}_p$, compute $v = P(u), w = v^{-1}$ and $c_v = \mathrm{com}_{ck}(v; s), c_w = \mathrm{com}_{ck}(w; t)$, and send c_v, c_w to the verifier. Engage in parallel in an SHVZK multiplication argument [10] to show $v \cdot w = 1$ and in the SHVZK polynomial evaluation argument from Section 3 to show $P(u) = v$.
Verification: The verifier accepts $u \notin \mathcal{L}$ if and only if $c_v, c_w \in \mathbb{G}$ and the two SHVZK arguments are valid.

Theorem 7 *If the discrete logarithm assumption holds, the above protocol is a public coin SHVZK argument of knowledge of an opening of c to $u \notin \mathcal{L}$.*

Proof. Perfect completeness follows from the perfect completeness of the two SHVZK arguments.

The SHVZK simulator picks $c_v, c_w \leftarrow \mathbb{G}$ at random and runs the SHVZK simulators for the two underlying SHVZK arguments. Since the commitment scheme is perfectly hiding and the underlying SHVZK arguments are perfect SHVZK this gives us perfect SHVZK.

The protocol has witness-extended emulation. The emulator $\mathcal{X}$ runs the witness-extended emulator for the two underlying SHVZK arguments to get openings u, v, w of the commitments satisfying $v \cdot w = 1$ and $P(u) = v$. The first equality tells us $v \neq 0$

and the second equality then tells us $P(u) \neq 0$. This means u is a not a root of the polynomial $P(X) = \prod_{i=1}^{D}(X - \lambda_i)$ and therefore $u \notin \mathcal{L}$. □

It is easy to modify the non-membership argument into a membership argument. If $u \in \mathcal{L}$ then $P(u) = 0$. We therefore get a membership argument by removing c_w and instead of a multiplication argument letting the prover give an SHVZK argument for c_v containing $v = 0$. Arguments of knowledge of an opening of a commitment to 0 are standard and only cost a couple of group elements in communication [33].

Efficiency. The coefficients of the polynomial $P(X) = \prod_{i}^{D}(X - \lambda_i)$ can be computed in a binary tree fashion with the linear functions $X - \lambda_i$ as leaves. Fast polynomial multiplication techniques that rely on the Fast Fourier Transform can if p is an FFT friendly prime multiply two degree n polynomials using $O(n \log n)$ multiplications in $\mathbb{Z}_p$. This means the prover and the verifier can compute the coefficients of $P(X)$ using $O(D \log^2 D)$ multiplications in $\mathbb{Z}_p$. If the list stays fixed, the computation of the polynomials coefficients is a one-time cost. Single element additions or deletions can be done using D multiplications. If multiple elements are added or deleted at the same time the per element cost can be reduced by using fast polynomial multiplication and division techniques.

Once the coefficients of $P(X)$ are given, the main cost of the membership and non-membership arguments are dominated by the underlying polynomial evaluation argument. For moderate D the computation is dominated by the logarithmic number of exponentiations involved in the polynomial evaluation argument. For large D the computational cost of computing the coefficients of the polynomial matters more as do the multiplications in the polynomial evaluation argument.

5 Comparison and Implementation

The first approaches to prove for committed $u, v \in \mathbb{Z}_p$ that $p(u) = v$ for a given polynomial $P(X)$ with order D split in two parts: first construct commitments $c_1, \ldots, c_D$ to values $u, u^2 \ldots, u^D$ and then use the homomorphic property of the commitment scheme to get $P(u)$ as a linear combination of $u, u^2, \ldots, u^D$. This requires D commitments and D multiplication arguments to show that the commitments $c_1, \ldots, c_D$ have been correctly constructed and contain the correct powers of u. The cost can reduced to $O(\sqrt{D})$ as suggested in Brands et al. [4] by splitting the polynomial in $\sqrt{D}$ polynomials of degree $\sqrt{D}$ each.

Our protocol also has a two part structure but only needs $\log D$ commitments $c_1, \ldots, c_d$ and $\log D$ multiplication arguments to prove they have been correctly formed and contain $u^2, u^4, u^8, u^{16}, \ldots, u^{2^d}$. By using a sophisticated combination of these values in combination with the homomorphic properties of the commitment scheme, we then get the desired argument for $v = p(u)$. This reduces our communication to $O(\log D)$ group elements.

Table 1 gives the asymptotic communication and computation costs of polynomial evaluation arguments based on the discrete logarithm assumption. The polynomial evaluation argument from Brands et al. [4] achieves the best complexity, so we will in the

SHVZK argument	Rounds	Time $\mathcal{P}$ Expos	Time $\mathcal{V}$ Expos	Size Elements
Fujisaki and Okamoto [14]	3	$O(D)$	$O(D)$	$O(D)\ \mathbb{G} + O(D)\ \mathbb{Z}_q$
Brands et al. [4]	3	$O(\sqrt{D})$	$O(\sqrt{D})$	$O(\sqrt{D})\ \mathbb{G} + O(\sqrt{D})\ \mathbb{Z}_q$
Groth [15]	7	$O(D)$	$O(\sqrt{D})$	$\sqrt{D}\ \mathbb{G} + \sqrt{D}\ \mathbb{Z}_q$
This paper	3	$O(\log D)$	$O(\log D)$	$O(\log D)\ \mathbb{G} + O(\log D)\ \mathbb{Z}_q$

Table 1. Comparison of our polynomial argument with former work

SHVZK argument	Rounds	Time $\mathcal{P}$ Expos	Time $\mathcal{P}$ Multip.	Time $\mathcal{V}$ Expos	Time $\mathcal{V}$ Multip.	Size Elements
[4]	3	$8\sqrt{D}$	$2D + 8\sqrt{D}$	$7\sqrt{D}$	$D + 4\sqrt{D}$	$4\sqrt{D}\ \mathbb{G} + 3\sqrt{D}\ \mathbb{Z}_q$
This paper	3	$8 \log D$	$2D \log D$	$7 \log D$	$2D$	$4 \log D\ \mathbb{G} + 3 \log D \mathbb{Z}_q$

Table 2. Detailed comparison of our blacklist argument with Brands et al. [4] argument

following give a more detailed theoretical and practical comparison. In Table 2 we give a more detailed theoretical analysis that also counts the number of multiplications.

Based on Table 2 we would expect our verifier to run faster than Brands et. al.'s as our asymptotic computation cost is much smaller and we expect our argument size to be much smaller. Just looking at the numbers of exponentiations needed by the prover can be a little deceptive though since in our polynomial evaluation argument we need $O(D \log D)$ multiplications in $\mathbb{Z}_p$ to compute the δ_j and for large D this cost becomes dominant. Our performance gain for the prover is therefore largest in the range, where D is large enough for $\log D$ to be significantly smaller than $\sqrt{D}$ yet not so large that the cost of $D \log D$ multiplications in $\mathbb{Z}_p$ becomes dominant.

We implemented our polynomial evaluation argument and Brands et al.'s argument in C++ with the NTL library to obtain experimental confirmation of our theoretical analysis and to get a real life comparison based on similar implementation techniques.

For the comparison of the polynomial evaluation arguments we have used a 256-bit subgroup modulo a 1536-bit prime, assumed that the polynomial $P(X)$ is pre-computed and obtained the running time for polynomials with degree between 10 and $500,000$ elements. The performance measurements have been obtained on a MacBook Pro with a 2.54 GHz Intel Core 2 Duo CPU, 4 GB RAM running Mac OS X 10.8.6; all code is single threaded and optimized code using the multi exponentiation techniques by [24]. The results can be found in Table 3.

As expected our verifier runs faster than Brands et al.'s verifier and we also see that our communication compares very favorably against Brands et al.'s communication. For moderate size D it is also the case that our prover is the most efficient, however, for very large D the cost to calculate the δ_js becomes dominant for our prover and here Brands et al.'s prover is faster. Other experiments we have conducted show that for larger security parameters the crossover happens for even larger D and for all reasonable degrees of the polynomial our argument is faster.

Elements in list D	Prover Brands et al.	Prover This paper	Verifier Brands et al.	Verifier This paper	Communication Brands et al.	Communication This paper
10	21 ms	13 ms	24 ms	17 ms	12 KB	8 KB
100	66 ms	24 ms	69 ms	30 ms	37 KB	15 KB
1000	227 ms	41 ms	234 ms	45 ms	128 KB	21 KB
10000	747 ms	182 ms	759 ms	81 ms	406 KB	29 KB
100000	2386 ms	1420ms	2402 ms	217 ms	1295 KB	35 KB
1000000	8650 ms	15512 ms	8165 ms	1315 ms	4161 KB	41 KB

Table 3. Our polynomial evaluation argument compared to Brands et al. [4]. All experiments used a 256-bit subgroup modulo a 1536-bit prime and a MacBook Pro, 2.54 CPU, 4 GB RAM.

The computation cost of the non-membership argument by Brands et al. is smaller than the cost of their polynomial argument. It still requires very large degree D also for the non-membership argument of Brands et al. to become better from a computational perspective though; for moderate size D we have a clear performance advantage.

References

1. G. Ateniese, D. Song, and G. Tsudik. Quasi-efficient revocation in group signatures. *Financial Cryptography*, LNCS vol. 2357:183–197, 2002.
2. J. Benaloh and M. de Mare. One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). *EUROCRYPT*, LNCS vol. 765:274–285, 1993.
3. D. Boneh and H. Shacham. Group signatures with verifier-local revocation. *ACM Conference on Computer and Communications Security*, pages 168–177, 2004.
4. S. Brands, L. Demuynck, and B. De Decker. A practical system for globally revoking the unlinkable pseudonyms of unknown users. *ACISP*, LNCS vol. 4586:400–415, 2007.
5. E. Bresson and J. Stern. Efficient revocation in group signatures. *Public Key Cryptography*, LNCS vol. 1992:190–206, 2001.
6. P. Camacho, Al. Hevia, M. Kiwi, and R. Opazo. Strong accumulators from collision-resistant hashing. *Int. J. Inf. Sec.*, 11(5):349–363, 2012.
7. J. Camenisch, R. Chaabouni, and S. Shelat. Efficient protocols for set membership and range proofs. *ASIACRYPT*, LNCS vol. 5350:234–252, 2008.
8. J. Camenisch, M. Kohlweiss, and C. Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. *Public Key Cryptography*, LNCS vol. 5443:481–500, 2009.
9. J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. *CRYPTO*, LNCS vol. 2442:61–76, 2002.
10. D. Chaum and T. P. Pedersen. Wallet databases with observers. In *CRYPTO*, volume LNCS vol. 740, pages 89–105, 1992.
11. R. Cramer and I. Damgård. Zero-knowledge proofs for finite field arithmetic; or: Can zero-knowledge be for free? *CRYPTO*, LNCS vol. 1462:424–441, 1998.
12. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. *CRYPTO*, LNCS vol. 839:174–187, 1994.
13. R. Dingledine. Tor and circumvention: Lessons learned - (abstract to go with invited talk). *CRYPTO*, LNCS vol. 6841:485–486, 2011.

14. E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. *CRYPTO*, LNCS vol. 1294:16–30, 1997.
15. J. Groth. Linear algebra with sub-linear zero-knowledge arguments. *CRYPTO*, LNCS vol. 5677:192–208, 2009.
16. J. Groth. Efficient zero-knowledge arguments from two-tiered homomorphic commitments. *ASIACRYPT*, LNCS vol. 7073:431–448, 2011.
17. J. Groth and Y. Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. *EUROCRYPT*, LNCS vol. 4965:379–396, 2008.
18. R. Henry and I. Goldberg. Extending nymble-like systems. *IEEE Symposium on Security and Privacy*, pages 523–537, 2011.
19. R. Henry, K. Henry, and I. Goldberg. Making a nymbler nymble using verbs. *Privacy Enhancing Technologies*, LNCS vol. 6205:111–129, 2010.
20. P. Johnson, A. Kapadia, P. Tsang, and S. Smith. Nymble: Anonymous ip-address blocking. *Privacy Enhancing Technologies*, LNCS vol. 4776:113–133, 2007.
21. J. Kilian. A note on efficient zero-knowledge proofs and arguments. In *STOC*, pages 723–732, 1992.
22. J. Li, N. Li, and R. Xue. Universal accumulators with efficient nonmembership proofs. *ACNS*, LNCS vol. 4521:253–269, 2007.
23. B. Libert, T. Peters, and M. Yung. Scalable group signatures with revocation. *EUROCRYPT*, LNCS vol. 7327:609–627, 2012.
24. C. Lim. Efficient multi-exponentiation and application to batch verification of digital signatures. `http://dasan.sejong.ac.kr/~chlim/pub/multiexp.ps`, 2000.
25. Y. Lindell. Parallel coin-tossing and constant-round secure two-party computation. *J. Cryptology*, 16(3):143–184, 2003.
26. P. Lofgren and N. Hopper. Bnymble: More anonymous blacklisting at almost no cost (a short paper). *Financial Cryptography*, LNCS vol. 7035:268–275, 2011.
27. T. Nakanishi and N. Funabiki. A short verifier-local revocation group signature scheme with backward unlinkability. *IWSEC*, LNCS vol. 4266:17–32, 2006.
28. D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. *CRYPTO*, LNCS vol. 2139:41–62, 2001.
29. L. Nguyen. Accumulators from bilinear pairings and applications. *CT-RSA*, LNCS vol. 3376:275–292, 2005.
30. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. *CRYPTO*, LNCS vol. 576:129–140, 1991.
31. K. Peng. A general, flexible and efficient proof of inclusion and exclusion. *CT-RSA*, LNCS vol. 6558:33–48, 2011.
32. K. Peng and F. Bao. Improving applicability, efficiency and security of non-membership proof. *International Symposium on Data, Privacy, and E-Commerce*, pages 39–44, 2010.
33. C. Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.
34. D. Song. Practical forward secure group signature schemes. *ACM Conference on Computer and Communications Security*, pages 225–234, 2001.
35. Tor. The tor project inc. `https://www.torproject.org/`.
36. P. Tsang, A. Kapadia, C. Cornelius, and S. Smith. Nymble: Blocking misbehaving users in anonymizing networks. *IEEE Trans. Dependable Sec. Comput.*, 8(2):256–269, 2011.
37. G. Tsudik and S. Xu. Accumulating composites and improved group signing. *ASIACRYPT*, LNCS vol. 2894:269–286, 2003.
38. P. Wang, H. Wang, and J. Pieprzyk. A new dynamic accumulator for batch updates. *ICICS*, LNCS vol. 4861:98–112, 2007.
39. K. Yu, T. Yuen, S. Chow, S. Yiu, and L. Hui. Pe(ar)2: Privacy-enhanced anonymous authentication with reputation and revocation. *ESORICS*, LNCS vol. 7459:679–696, 2012.