Arguments of Knowledge via hidden order groups

Steve Thakur

Abstract

We study non-interactive arguments of knowledge (AoKs) for commitments in groups of hidden order. We provide protocols whereby a Prover can demonstrate certain properties of and relations between committed sets/multisets, with succinct proofs that are publicly verifiable against the constant-sized commitments. In particular, we provide AoKs for the disjointness of committed sets/multisets in cryptographic accumulators, with a view toward applications to verifiably outsourcing data storage and sharded stateless blockchains.

Recent work ([DGS20]) suggests that the hidden order groups need to be substantially larger in size that previously thought in order to ensure the desired security level. Thus, in order to keep the communication complexity between the Prover and the the Verifier to a minimum, we have designed the protocols so that the proofs entail a constant number of group elements, independent of the number of the committed sets/multisets rather than just independent of the sizes of these sets/multisets.

If the underlying group of hidden order is an appropriate imaginary quadratic class group or a genus three Jacobian, the argument systems are transparent. Furthermore, since all challenges are public coin, the protocols can be made non-interactive using the Fiat-Shamir heuristic. We build on the techniques from [BBF19] and [Wes18].

1 Introduction

A commitment scheme is a fundamental cryptographic primitive which is the digital analog of a sealed envelop. *Committing* to a message `m` is akin to putting `m` in the envelop. *Opening* the commitment is like opening the envelop and revealing the content within. Commitments are endowed with two basic properties. The *hiding* property entails that a commitment reveals no information about the underlying message. The *binding* property ensures that one cannot alter the message without altering the commitment. A cryptographic *accumulator* is a commitment to a set or a multiset. A Prover with access to the set/multiset can prove membership or non-membership of an element with a proof verifiable against the succinct commitment held by a Verifier.

Finite abelian groups of hidden order have seen a surging interest within cryptography in the last few years. The adaptive root and strong RSA assumptions in such groups yield a cryptographic accumulator which is universal and dynamic with batchable membership and non-membership proofs ([BBF19]). One of the best known verifiable delay functions is that constructed in ([Wes18]), which can be instantiated with any group of unknown order in which the adaptive root assumption holds. Such groups also form the basis for the transparent polynomial commitment constructed in ([BFS19]). This is a polynomial commitment with logarithmic sized proofs and verification time and yields the first known transparent ZK-SNARK.

In this paper, we explore non-interactive arguments of knowledge in groups of hidden order. We provide protocols whereby a Prover who stores data in the form of sets/multisets can prove relationships between these sets/multisets with communication complexity independent of the size of this data. These proofs can be publicly verified against the constant-sized commitments held

by the Verifier. Our primary goal is to provide protocols for proofs of disjointness of committed sets/multisets in cryptographic accumulators. The primary use cases for these AoKs are potential applications to sharded stateless blockchains and to verifiable outsourcing of data.

As was the case with previously studied arguments of knowledge in hidden order groups ([BBF19], [CFGNK20] etc.), the proofs consist of elements of the group $\mathbb{G}$ and λ-bit integers, where λ is the security parameter. Recent work ([DGS20]) suggests that the hidden order groups need to be substantially larger in size that previously thought, in order to ensure the desired security level. Furthermore, the two known candidates for transparent hidden order groups - imaginary quadratic class groups and genus three Jacobians - are not as well studied as RSA groups when it comes to potential attacks against the adaptive root and strong-RSA assumptions. Hence, it is conceivable that these groups might need to be even larger than presently believed. Bearing this in mind and with a view toward keeping the communication complexity between the Prover and the the Verifier to a minimum, we have designed the protocols so that the proofs consist of a constant number of group elements, independent of the number of committed sets/multisets involved, rather than just independent of the sizes of these sets/multisets.

1.1 Structure of the paper

In section 2, we go over some notations, background and a few lemmas we will need repeatedly in the subsequent sections. In section 3, we provide protocols for aggregating the knowledge of multiple discrete logarithms in hidden order groups. This is equivalent to AoKs for multiple committed sets/multisets, the proofs for which can be publicly verified against the succinct commitments.

In section 4, we provide protocols whereby a Prover holding the data can demonstrate the pairwise disjointness of multiple data sets/multisets with proofs publicly verifiable against the commitments. While a straightforward approach would entail proofs with $\mathbf{O}(n^2)$ group elements to prove pairwise disjointness of n committed sets/multisets, we provide protocols to do so with $\mathbf{O}(1)$ group elements and $n + \mathbf{O}(1)$ λ-bit integers.

In section 5, we discuss some applications such as verifiable outsourcing of data. The protocols allow a client node to outsource data sets/multisets to an untrusted server node who can verifiably demonstrate properties of and relations between these data sets/multisets with succinct proofs that can be publicly verified against the commitments held by the client node. The section also contains protocols whereby the server node can verifiably identify the committed multiset with the highest/lowest frequency of a batch of elements. We also briefly discuss how our protocols for arguments of disjointess might have applications to sharded stateless blockckhains instantiated with hidden order groups.

We have put a few other protocols such as those demonstrating relations between the underlying sets of committed multisets in the appendix. We also discuss a minor generalization of Wesolowski's proof of exponentiation in Appendix C. This helps reduce the Verifier's work in some of our protocols in Sections 3 and 4.

2 Preliminaries

We first state some definitions and notations used in this paper.

Notations: We denote the security parameter by λ and the set of all polynomial functions by $\texttt{poly}(\lambda)$. A function $\epsilon(\lambda)$ is said to be *negligible* - denoted $\epsilon(\lambda) \in \texttt{negl}(\lambda)$ - if it vanishes faster than the reciprocal of any polynomial. An algorithm $\mathcal{A}$ is said to be a probabilistic polynomial

time (PPT) algorithm if it is modeled as a Turing machine that runs in time $\texttt{poly}(\lambda)$. We denote by $y \leftarrow \mathcal{A}(x)$ the process of running $\mathcal{A}$ on input x and assigning the output to y. For a set S, $\#S$ or $|S|$ denote its cardinality and $x \xleftarrow{\$} S$ denotes selecting x uniformly at random over S. For a positive integer n, we write $[n] := \{0, 1, \cdots, n-1\}$. $\texttt{NextPrime}(n)$ denotes the smallest prime $\geq n$. For statements $\mathbf{A}$, $\mathbf{B}$ we say that $\mathbf{A}$ implies $\mathbf{B}$ with overwhelming probability (denoted by $\mathbf{A} \overset{\text{o.p.}}{\Longrightarrow} \mathbf{B}$) if

$$1 - \texttt{Pr}\big[\mathbf{B} \;\big|\; \mathbf{A}\big] = \texttt{negl}(\lambda).$$

$\texttt{H}_{\texttt{FS},\lambda}$ denotes the hashing algorithm used by the Fiat-Shamir-heuristic. It generates λ-bit primes.

2.1 Candidates for hidden order groups

At the moment, there are only three known families of finite abelian groups of unknown order. We briefly discuss them here.

1. **RSA groups:** For distinct 1536-bit primes p, q, define $N := pq$. The group $(\mathbb{Z}/N\mathbb{Z})^*$ has order $\phi(N) = (p-1)(q-1)$ which can only be computed by factorizing N. The strong-RSA assumption is believed to hold in the RS group. However, the group does contain the element $-1 \pmod N$ of a known order 2. For the adaptive root assumption to hold, the group has to be replaced by its quotient group $(\mathbb{Z}/N\mathbb{Z})^*/\{\pm 1\}$ of order $\frac{(p-1)(q-1)}{2}$.

The RSA groups suffer from the need for a trusted setup. In practice, this can be mitigated by a one-time secure multi-party computation. At the moment, a 3300-bit RSA modulus yields a security level of 128-bits.

2. **Class groups:** Computing the class group of a number field is a long-standing problem in algorithmic number theory. Hence, class groups are natural candidates for hidden order groups. At the moment, the only class groups that allow for efficient group operations are those of imaginary quadratic fields.

For a square-free integer $d > 0$, the field $\mathbb{Q}(\sqrt{-d})$ has a class group of size roughly $\sqrt{d}$. This group is believed to fulfill the strong-RSA assumption. Furthermore, if d is a prime $\equiv 3 \pmod 4$, the 2-torsion group is trivial, which eliminates the possibility of known elements of order 2. Such a group is believed to fulfill the strong-RSA, low order and $\{2\}$-fractional root assumptions unless the integer d lies within a certain thin set of integers.

A 6656-bit discriminant d yields a security level of 128-bits at the moment. Unlike RSA groups, class groups allow for a transparent (trustless) setup. The downside is that for the same level of security, the group operations are roughly 10 times slower than modular multiplication.

3. **Jacobians:** Recently, the group of $\mathbb{F}_p$-valued points of the Jacobian of a genus three hyperelliptic curve over a prime field $\mathbb{F}_p$ has been proposed as a candidate ([DGS20]). While this idea needs more scrutiny, it seems promising because of the transparent setup, the smaller key sizes and the fact that the group operations are 28 times faster than those in class groups for the same level of security.

For an irreducible polynomial $f(X) \in \mathbb{Z}[X]$ of degree 7 with Galois group $\mathbf{S}_7$ and a prime p such that $f(X) \pmod p$ is separable, the hyperelliptic curve $C : Y^2 = f(X)$ over $\mathbb{F}_p$ yields a Jacobian that is resistant to the known attacks. At the moment, such a genus three hyperelliptic Jacobian over a prime field $\mathbb{F}_p$ of bit-size 1100 allows for a security level of 128-bits. This group $\text{Jac}(C)(\mathbb{F}_p)$ is roughly of size p^3.

Unfortunately, the adaptive root assumption fails in the group $\mathrm{Jac}(C)(\mathbb{F}_p)$. However, the group obtained by replacing it by an appropriate quotient group appears to satisfy the adaptive root assumption and the weaker assumptions such as the fractional root and low order assumptions.

2.2 Argument Systems

An argument system for a relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$ is a triple of randomized polynomial time algorithms $(\mathtt{PGen}, \mathcal{P}, \mathcal{V})$, where $\mathtt{PGen}$ takes an (implicit) security parameter λ and outputs a common reference string (CRS) $\mathtt{pp}$. If the setup algorithm uses only public randomness we say that the setup is transparent and that the CRS is unstructured. The prover $\mathcal{P}$ takes as input a statement $x \in X$, a witness $w \in W$, and the CRS $\mathtt{pp}$. The verifier $\mathcal{V}$ takes as input $\mathtt{pp}$ and x and after interactions with $\mathcal{P}$ outputs 0 or 1. We denote the transcript between the prover and the verifier by $\langle \mathcal{V}(\mathtt{pp}, x), \mathcal{P}(\mathtt{pp}, x, w) \rangle$ and write $\mathcal{V}(\mathtt{pp}, x), \mathcal{P}(\mathtt{pp}, x, w) \rangle = 1$ to indicate that the verifier accepted the transcript. If $\mathcal{V}$ uses only public randomness we say that the protocol is *public coin*.

We now define soundness and knowledge extraction for our protocols. The adversary is modeled as two algorithms $\mathcal{A}_0$ and $\mathcal{A}_1$, where $\mathcal{A}_0$ outputs the instance $x \in \mathcal{X}$ after $\mathtt{PGen}$ is run, and $\mathcal{A}_1$ runs the interactive protocol with the verifier using a state output by $\mathcal{A}_0$. In a slight deviation from the soundness definition used in statistically sound proof systems, we do not universally quantify over the instance x (i.e. we do not require security to hold for all input instances x). This is due to the fact that in the computationally-sound setting the instance itself may encode a trapdoor of the common reference string, which can enable the adversary to fool a verifier. Requiring that an efficient adversary outputs the instance x prevents this. In our soundness definition the adversary $\mathcal{A}_1$ succeeds if he can make the verifier accept when no witness for x exists. For the stronger argument of knowledge definition we require that an extractor with access to $\mathcal{A}_1$'s internal state can extract a valid witness whenever $\mathcal{A}_1$ is convincing. We model this by enabling the extractor to rewind $\mathcal{A}_1$ and reinitialize the verifier's randomness.

Definition 2.1. *We say an argument system* $(\mathtt{PGen}, \mathcal{P}, \mathcal{V})$ *for a relation* $\mathcal{R}$ *is* ***complete*** *if for all* $(x, w) \in \mathcal{R}$,

$$\mathtt{Pr}\big[\langle \mathcal{V}(\mathtt{pp}, x)\ ,\ \mathcal{P}(\mathtt{pp}, w)\rangle\big) = 1\ :\ \mathtt{pp} \xleftarrow{\$} \mathtt{PGen}(\lambda)\big]\ =\ 1.$$

Definition 2.2. *We say an argument system* $(\mathtt{PGen}, \mathcal{P}, \mathcal{V})$ *is* ***sound*** *if* $\mathcal{P}$ *cannot forge a fake proof except with negligible probability.*

Definition 2.3. *We say a sound argument system is an* ***argument of knowledge*** *if for any polynomial time adversary* $\mathcal{A}$*, there exists an extractor* $\mathcal{E}$ *with access to* $\mathcal{A}$*'s internal state that can, with overwhelming probability, extract a valid witness whenever* $\mathcal{A}$ *is convincing.*

Definition 2.4. *An argument system is* ***non-interactive*** *if it consists of a single round of interaction between* $\mathcal{P}$ *and* $\mathcal{V}$*.*

The Fiat-Shamir heuristic ([FS87]) can be used to transform interactive public coin argument systems into non-interactive systems. Instead of the Verifier generating the challenges, this function is performed by a public hashing algorithm agreed upon in advance.

2.3 Cryptographic assumptions

The cryptographic protocols make extensive use of groups of unknown order, i.e., groups for which the order cannot be computed efficiently. Concretely, we require groups for which two hardness assumptions hold. The Strong RSA Assumption ([BP97]) roughly states that it is hard to take

arbitrary roots of random elements. The much newer Adaptive Root Assumption ([Wes19]) is the dual to the Strong RSA Assumption and states that it is hard to take random roots of arbitrary group elements. Both of these assumptions are believed to hold in generic groups of hidden order ([Wes18], [BBF19], [DGS20]).

Assumption 2.1. *We say that the **adaptive root** assumption holds for a group $\mathbb{G}$ if there is no efficient probabilistic polynomial time (PPT) adversary $(\mathcal{A}_0, \mathcal{A}_1)$ that succeeds in the following task. $\mathcal{A}_0$ outputs an element $w \in \mathbb{G}$ and some state. Then a random prime ℓ is chosen and $\mathcal{A}_1(\ell, \texttt{state})$ outputs $w^{1/\ell} \in \mathbb{G}$.*

Assumption 2.2. *For a set $\mathcal{S}$ of rational primes, we say $\mathbb{G}$ satisfies the $\mathcal{S}$-**strong RSA** assumption if given a random group element $g \in \mathbb{G}$ and a prime $\ell \notin \mathcal{S}$, no PPT algorithm $\mathcal{A}$ is able to compute (except with negligible probability) the ℓ-th root of a chosen element $w \in \mathbb{G}$. When $\mathcal{S} = \emptyset$, it is called the **strong RSA** assumption.*

Assumption 2.3. *We say $\mathbb{G}$ satisfies the **low order** assumption if no PPT algorithm can generate (except with negligible probability) an element $a \in \mathbb{G} \setminus \{1\}$ and an integer $n < 2^{\text{poly}(\lambda)}$ such that $a^n = 1 \in \mathbb{G}$.*

Assumption 2.4. *For a set $\mathcal{S}$ of rational primes, we say $\mathbb{G}$ satisfies the $\mathcal{S}$-**fractional root** assumption if for a randomly generated element $g \in \mathbb{G}$, no PPT algorithm can output $h \in \mathbb{G}$ and $d_1, d_2 \in \mathbb{Z}$ such that*

$$g^{d_1} = h^{d_2} \ \wedge \ \mathbf{gcd}(d_1, d_2) = 1 \ \wedge \ d_2 \text{ has a prime divisor outside } \mathcal{S}$$

*except with negligible probability. When $\mathcal{S} = \emptyset$, it is called the **fractional root** assumption.*

Clearly, if $\mathcal{S}_0 \subseteq \mathcal{S}$, the $\mathcal{S}_0$-fractional root assumption implies the $\mathcal{S}$-fractional root assumption. For instance, class groups of imaginary quadratic fields are believed to fulfill the $\{2\}$-fractional root assumption although they do not fulfill the (stronger) fractional root assumption. This is because there is a well-known algorithm to compute square roots in imaginary quadratic class groups ([BS96]). The assumptions bear the following relations:

$$\{\text{Adaptive root assumption}\} \implies \{\text{Low order assumption}\} ,$$

$$\{\text{Low order assumption}\} \wedge \{\mathcal{S}\text{-Strong-RSA assumption}\} \implies \{\mathcal{S}\text{-Fractional root assumption}\}.$$

We refer the reader to the appendix of [BBF19] for further details.

Definition 2.5. *For elements $a, b \in \mathbb{G}$ and a rational $\alpha \in \mathbb{Q}$, we say $a^\alpha = b$ with respect to a PPT algorithm $\mathcal{A}$ if $\mathcal{A}$ can generate integers $d_1, d_2 \in \mathbb{Z}$ such that:*
- $\alpha = d_1 {d_2}^{-1}$
- $a^{d_1} = b^{d_2}$
- $|d_1|, |d_2| < 2^{\texttt{poly}(\lambda)}$.

Note that if a PPT algorithm $\mathcal{A}$ generates an element $a \in \mathbb{G}$ and distinct rationals $d_1 {d_2}^{-1}$, $d_3 {d_4}^{-1}$, $(d_i \in \mathbb{Z})$ such that

$$a^{d_1 {d_2}^{-1}} = a^{d_3 {d_4}^{-1}} \in \mathbb{G},$$

then $a^{d_1 d_4 - d_2 d_3} = 1$ and $d_1 d_4 - d_2 d_3 \neq 0$. So the low order assumption implies that $\mathcal{A}$ cannot generate such a tuple $(a, d_1, d_2, d_3, d_4) \in \mathbb{G} \times \mathbb{Z}^4$, except with negligible probability. Furthermore, by Shamir's trick, $a^\alpha = b$ is equivalent to $\mathcal{A}$ being able to generate an element $a_0 \in \mathbb{G}$ and co-prime integers d_1, d_2 such that

$$\alpha = d_1 {d_2}^{-1} \ , \ a_0^{d_2} = a \ , \ a_0^{d_1} = b \, ,$$

2.3.1 Generic group models for hidden order groups

We will use the generic group model for groups of unknown order as defined by [DK02] and [BBF19]. The group is parametrized by two integer public parameters A, B. The order of the group is sampled uniformly from the interval $[A, B]$. The group $\mathbb{G}$ is defined by a random injective function $\sigma : \mathbb{Z}_{|\mathbb{G}|} \longrightarrow \{0,1\}^n$ for some $n >> \log_2(|\mathbb{G}|)$. A generic group algorithm $\mathcal{A}$ is a probabilistic algorithm. Let $\mathcal{L}$ be a list that is initialized with the encodings given to $\mathcal{A}$ as input. The algorithm can query two generic group oracles:

• $\mathcal{O}_1$ samples a random $r \in \mathbb{Z}_{\mathbb{G}}$ and returns $\sigma(r)$, which is appended to the list $\mathcal{L}$ of encodings.
• When $\mathcal{L}$ has size q, the second oracle $\mathcal{O}_2(i, j, \pm)$ takes two indices $i, j \in \{1, \cdots, q\}$ and a sign bit and returns $\sigma(x_i \pm x_j)$ which is appended to $\mathcal{L}$.

2.4 Multiset notations and operations

We first recall/introduce a few basic definitions and notations concerning multisets. For a multiset $\mathcal{M}$, we denote by $\texttt{Set}(\mathcal{M})$ the underlying set of $\mathcal{M}$. For any element x, we denote by $\text{mult}(\mathcal{M}, x)$ the multiplicity of x in $\mathcal{M}$. Thus, $\mathcal{M} = \{\text{mult}(\mathcal{M}, x) \times x : \ x \in \texttt{Set}(\mathcal{M})\}$. For brevity, we write

$$\Pi(\mathcal{M}) := \prod_{x \in \texttt{Set}(\mathcal{M})} x^{\text{mult}(\mathcal{M},x)}.$$

For two multisets $\mathcal{M}, \mathcal{N}$, we have the following operations:

- The sum $\mathcal{M} + \mathcal{N} := \{(\text{mult}(\mathcal{M}, x) + \text{mult}(\mathcal{N}, x)) \times x : \ x \in \texttt{Set}(\mathcal{M}) \cup \texttt{Set}(\mathcal{N})\}$
- The union $\mathcal{M} \cup \mathcal{N} := \{\max(\text{mult}(\mathcal{M}, x), \text{mult}(\mathcal{N}, x)) \times x : \ x \in \texttt{Set}(\mathcal{M}) \cup \texttt{Set}(\mathcal{N})\}$
- The intersection $\mathcal{M} \cap \mathcal{N} := \{\min(\text{mult}(\mathcal{M}, x), \ \text{mult}(\mathcal{N}, x)) \times x : \ x \in \texttt{Set}(\mathcal{M}) \cup \texttt{Set}(\mathcal{N})\}$
- The difference $\mathcal{M} \setminus \mathcal{N} := \{\min(\text{mult}(\mathcal{M}, x) - \text{mult}(\mathcal{N}, x), \ 0) \times x : \ x \in \texttt{Set}(\mathcal{M}) \cup \texttt{Set}(\mathcal{N})\}$.

The function $\Pi(\cdot)$ clearly has the following properties:

- $\Pi(\mathcal{M} + \mathcal{N}) = \Pi(\mathcal{M}) \cdot \Pi(\mathcal{N})$
- $\Pi(\mathcal{M} \cup \mathcal{N}) = \mathbf{lcm}(\Pi(\mathcal{M}), \Pi(\mathcal{M}))$
- $\Pi(\mathcal{M} \cap \mathcal{N}) = \mathbf{gcd}(\Pi(\mathcal{M}), \Pi(\mathcal{M}))$
- $\Pi(\mathcal{M} \setminus \mathcal{N}) = \Pi(\mathcal{M})/\Pi(\mathcal{M} \cap \mathcal{N})$

Multiset Commitments: For a multiset $\mathcal{M}$ represented by λ-bit primes and a hidden order group $\mathbb{G}$, a $\mathbb{G}$-*commitment to a multiset* $\mathcal{M}$ is a pair $(g, h) \in \mathbb{G}^2$ such that $g^{\Pi(\mathcal{M})} = h$. The hardness of the discrete logarithm problem implies that this commitment is *hiding* in the sense that no PPT algorithm can compute $\mathcal{M}$ from the pair $[g, h]$. The low order assumption implies that it is *binding* in the sense that no PPT algorithm can compute another multiset $\mathcal{M}'$ with the same commitment.

2.5 Cryptographic Accumulators

A cryptographic accumulator [Bd94] is a primitive that produces a short binding commitment to a set (or multiset) of elements together with short membership and/or non-membership proofs for any element in the set. These proofs can be publicly verified against the commitment. Broadly, there are three known types of accumulators at the moment:

- Merkle trees
- pairing-based (aka bilinear) accumulators
- accumulators based on groups of unknown order, which we study in this paper.

Let $\mathbb{G}$ be a group of hidden order and fix an element $g \in \mathbb{G}$. Let $\mathcal{M}$ be a multiset of λ-bit

primes. For each $x \in \mathcal{M}$, let $\text{mult}(\mathcal{M}, x)$ denote the multiplicity of x in $\mathcal{M}$. The *accumulated digest* or *accumulated state* of $\mathcal{M}$ is given by

$$\mathbf{Acc}(\mathcal{M}) := \texttt{Com}(g, \mathcal{M}) = g^{\Pi(\mathcal{M})} \in \mathbb{G},$$

where

$$\Pi(\mathcal{M}) := \prod_{x \in \texttt{Set}(\mathcal{M})} x^{\text{mult}(\mathcal{M}, x)}.$$

Let $\mathcal{M}_0$ be a multiset contained in $\mathcal{M}$, so that $\text{mult}(\mathcal{M}_0, x) \leq \text{mult}(\mathcal{M}, x)\ \forall\ x$. The element

$$w(\mathcal{M}_0) := g^{\prod_{x \in \texttt{Set}(\mathcal{M})} x^{\text{mult}(\mathcal{M} \setminus \mathcal{M}_0, x)}} \in \mathbb{G}$$

is called *the membership witness* of $\mathcal{M}_0$. Given this element, a Verifier can verify the membership of $\mathcal{M}_0$ in $\mathcal{M}$ by verifying the equation

$$w(\mathcal{M}_0)^{\Pi(\mathcal{M}_0)} \stackrel{?}{=} \mathbf{Acc}(\mathcal{D}).$$

When the multiset $\mathcal{M}_0$ is large, this verification can be sped up using Wesolowki's *Proof of Exponentiation* (`PoE`) protocol ([Wes18]).

Shamir's trick allows for aggregation of membership witnesses in accumulators based on hidden order groups. This is not possible with Merkle trees, which is the primary reason other families of accumulators have been explored as authentication data structures for stateless blockchains. With bilinear accumulators, aggregation of membership witnesses has a linear runtime complexity, which is impractical for most use cases. Thus, accumulators based on hidden order groups have a major advantage in this regard.

These accumulators also allow for non-membership proofs ([LLX07]). In [BBF19], the authors used a non-interactive argument of knowledge to compress batched non-membership proofs into constant-sized proofs, i.e. independent of the number of elements involved. This yields the first known Vector Commitment with constant-sized openings as well as constant-sized public parameters.

Hashing the data to primes: The security of cryptographic accumulators and vector commitments hinges on the assumption that for disjoint data sets $\mathcal{D}, \mathcal{E}$, the integers $\Pi(\mathcal{D}), \Pi(\mathcal{E})$ are relatively prime. The easiest way to ensure this is to map the data elements to distinct λ-bit primes. This is usually done by hashing the data to λ-bit integers and subjecting the output to a probabilistic primality test such as the Miller-Rabin test. The prime number theorem states that the number of primes less than n is $\mathbf{O}(\frac{n}{\log(n)})$ and hence, implies that the expected runtime for finding a prime is $\mathbf{O}(\lambda)$.

Dirichlet's theorem on primes in arithmetic progressions combined with the prime number theorem implies that for relatively prime integers k, r and an integer n, the number of primes less than n that are $\equiv r \pmod{k}$ is roughly $\frac{n}{\log(n)\phi(k)}$. Thus, we can modify the hashing algorithm so that for any element inserted into the accumulator, the prime reveals the position in which it was inserted. We proceed as follows.

1. Fix a prime p of size $\frac{\lambda}{2}$.

2. For a string inserted in position i, we map the string to the first prime of size λ which is $\equiv i \pmod{p}$. This (pseudo-)prime is obtained by subjecting the integers $\{pk + i :\ k \in \mathbb{Z}\}$ to the probabilistic Miller-Rabin test.

The number of such primes is roughly $\frac{2^\lambda}{\lambda(p-1)}$ and hence, the expected runtime is $\mathbf{O}(\lambda)$.

2.6 Aggregating and disaggregating membership witnesses

Shamir's trick: Given elements $a_1, a_2, A \in \mathbb{G}$ and integers d_1, d_2 such that $a_1^{d_1} = a_2^{d_2} = A$, Shamir's trick allows us to compute the $\mathbf{lcm}(d_1, d_2)$-th root of A as follows.

1. Compute integers e_1, e_2 such that $e_1 d_1 + e_2 d_2 = \mathbf{gcd}(d_1, d_2)$
2. Set $a_{1,2} := a_1^{e_2} a_2^{e_1}$.

Then

$$a_{1,2}^{d_1 d_2} = A^{d_2 e_2 + d_1 e_1} = A^{\mathbf{gcd}(d_1, d_2)}$$

and hence,

$$a_{1,2}^{\mathbf{lcm}(d_1, d_2)} = A.$$

More generally, given elements $a_1, \cdots, a_n$ such that

$$a_1^{d_1} = \cdots = a_n^{d_n} = A,$$

we can use Shamir's trick repeatedly to compute an element $a \in \mathbb{G}$ such that $a^{\mathbf{lcm}(d_1, \cdots, d_n)} = A$. The runtime for this algorithm is $\mathbf{O}\big(\log(D) \cdot \log(\log(D))\big)$ where $D := \prod_{i=1}^{n} d_i$.

The most important special case is when A is the accumulated digest $g^{\Pi(\mathcal{M})}$ for a set or multiset $\mathcal{M}$ and $w_1, \cdots, w_n$ are membership witnesses for sets/multisets $\mathcal{M}_1, \cdots, \mathcal{M}_n \subseteq \mathcal{M}$. Shamir's trick allows us to compute a membership witness for the union $\bigcup_{i=1}^{n} \mathcal{M}_i$.

The RootFactor Algorithm: Given elements $a, A \in \mathbb{G}$ and integers $d_1, \cdots, d_n$, D such that

$$D = \prod_{i=1}^{n} d_i \ , \ a^D = A,$$

the `RootFactor` algorithm ([BBF19], [STY01]) allows us to compute elements $a_1, \cdots, a_n \in \mathbb{G}$ such that

$$a_1^{d_1} = \cdots = a_n^{d_n} = A$$

in runtime $\mathbf{O}\big(\log(D) \cdot \log(\log(D))\big)$. Naïvely, this would take runtime $\mathbf{O}(\log^2(D))$, which would be impractical for many applications.

2.7 $\mathbb{Z}_{(\lambda)}$-integers

Localization at a set of primes: For a set $\mathcal{S}$ of rational primes, we denote by $\mathbb{Z}_{\mathcal{S}}$ the localization

$$\mathbb{Z}_{\mathcal{S}} := \Big\{ a \cdot \prod_{p \in \mathcal{S}} p^{e_p} \ : \ a \in \mathbb{Z},\ e_p \in \mathbb{Z},\ e_p = 0 \text{ for all but finitely many } p \Big\}$$

of $\mathbb{Z}$ at all primes in $\mathcal{S}$. This is a principal ideal domain whose prime ideals are those generated by rational primes outside the set $\mathcal{S}$. The group of units of $\mathbb{Z}_{\mathcal{S}}$ is given by

$$\mathbb{Z}_{\mathcal{S}}^{\times} := \Big\{ \pm \prod_{p \in \mathcal{S}} p^{e_p} \ : e_p \in \mathbb{Z},\ e_p = 0 \text{ for all but finitely many } p \Big\}.$$

For an element $\alpha \in \mathbb{Z}_{\mathcal{S}}$, we may uniquely write α as

$$\alpha = \alpha_1 {\alpha_2}^{-1} \text{ where } \alpha_1, \alpha_2 \in \mathbb{Z} \ , \ \alpha_2 > 0 \ , \ \mathbf{gcd}(\alpha_1, \alpha_2) = 1.$$

Similarly, for two elements, α, β, we write

$$\alpha = \alpha_1 {\alpha_2}^{-1} \ , \ \beta = \beta_1 {\beta_2}^{-1} \ \ , \ \ \alpha_1, \beta_1 \in \mathbb{Z} \ , \ \alpha_2, \beta_2 \in \mathbb{Z} \cap \mathbb{Z}_{\mathcal{S}}^{\times}$$

and define $\mathbf{gcd}_{\mathcal{S}}(\alpha, \beta) := \mathbf{gcd}(\alpha_1, \beta_1)$. Thus, $\mathbf{gcd}_{\mathcal{S}}(\alpha, \beta)$ is the unique non-negative integer that has no prime divisors in $\mathcal{S}$ and generates the ideal $\mathbb{Z}_{\mathcal{S}}\alpha + \mathbb{Z}_{\mathcal{S}}\beta \subseteq \mathbb{Z}_{\mathcal{S}}$.

Note: The localization $\mathbb{Z}_{\mathcal{S}}$ is not to be confused with the *non-archimedean completion* of the localization.

Definition 2.6. *An integer is said to be* λ***-smooth*** *is all of its prime divisors are* $\leq 2^{\lambda-1}$*. An integer is said to be* λ***-rough*** *is all of its prime divisors are* $> 2^{\lambda-1}$*. We say a set/multiset* $\mathcal{M}$ *of primes is* λ*-rough if the integer* $\Pi(\mathcal{M})$ *is* λ*-rough.*

Clearly, $\mathcal{M}$ being λ-rough is equivalent to each prime of $\mathcal{M}$ being larger than $2^{\lambda-1}$. The properties of λ-smoothness and λ-roughness are clearly preserved under products, greatest common divisors and least common multiples. Furthermore, any positive integer n is uniquely expressible as a product $n_{\lambda,s}n_{\lambda,r}$ of a λ-smooth integer $n_{\lambda,s} \geq 0$ and a λ-rough integer $n_{\lambda,r} \geq 0$.

Definition 2.7. *For a security parameter* λ*, we denote by* $\mathbb{Z}_{(\lambda)}$ *the integral domain obtained by localizing* $\mathbb{Z}$ *at all primes* $\leq 2^{\lambda-1}$*.*

Thus,

$$\mathbb{Z}_{(\lambda)} = \left\{\alpha\beta^{-1} : \ \alpha, \beta \in \mathbb{Z}, \ \mathbf{gcd}(\alpha, \beta) = 1, \ \beta \text{ is } \lambda\text{-smooth}\right\}.$$

Note that $\mathbb{Z}_{(\lambda)}$ inherits the structure of a principal ideal domain. The group of units of $\mathbb{Z}_{(\lambda)}$ is given by

$$\mathbb{Z}_{(\lambda)}^{\times} := \left\{\alpha\beta^{-1} : \ \alpha, \beta \in \mathbb{Z}, \ \mathbf{gcd}(\alpha, \beta) = 1, \ \alpha, \beta \text{ are } \lambda\text{-smooth}\right\}.$$

The prime ideals of $\mathbb{Z}_{(\lambda)}$ are the principal ideals generated by rational primes larger than $2^{\lambda-1}$.

2.8 Some preliminary lemmas

We will need the next two lemmas repeatedly in the subsequent protocols. We briefly explain the motivation for these lemmas here and provide further details in the next section. As before, for a set $\mathcal{S}$ of rational primes, we denote by $\mathbb{Z}_{\mathcal{S}}$ the localization

$$\mathbb{Z}_{\mathcal{S}} := \left\{a \cdot \prod_{p \in \mathcal{S}} p^{e_p} \ : \ a \in \mathbb{Z}, \ e_p \in \mathbb{Z}, \ e_p = 0 \text{ for all but finitely many } p\right\}$$

of $\mathbb{Z}$ at all primes in $\mathcal{S}$. This is a principal ideal domain and, in particular, is integrally closed.

Consider a setting where a Verifier possesses commitments $a_i = a^{d_i}$ to $\mathbb{Z}_{\mathcal{S}}$-integers d_i where $a \in \mathbb{G}$ is the common base and $\mathcal{S}$ is a set of rational primes. Suppose the Prover - who stores these integers - needs to demonstrate that the d_i are integers rather than merely rationals. A straightforward way to do this would be for the Prover to send the elements g^{d_i} along with proofs that the discrete logarithm between g, g^{d_i} is the same as that between a, a_i. The $\mathcal{S}$-fractional root assumption would then imply that the d_i are $\mathbb{Z}_{\mathcal{S}}$-integers.

But such a proof would entail $\mathbf{O}(n)$ elements of $\mathbb{G}$. Since the group elements are rather large, we would prefer to send a proof that contains a constant number of group elements. To this end, the Prover can instead demonstrate that for a randomly generated integer γ, the rational $\sum_{i=1}^{n} d_i^k \gamma^i$ is an element of $\mathbb{Z}_{\mathcal{S}}$ for some $k \geq n\lambda$. The next two lemmas show that this implies that all the d_i are elements of $\mathbb{Z}_{\mathcal{S}}$, except with negligible probability.

Lemma 2.1. *Let* p *be a prime and let* $f(X)$ *be a monic univariate degree* n *polynomial in* $\mathbb{Z}[X]$*. For a randomly generated integer* γ*, the probability that* $f(\gamma) \equiv 0 \pmod{p^{n\lambda}}$ *is negligible.*

Proof. Let F be a splitting field of $f(X)$, $\mathcal{O}_F$ its ring of integers and let

$$f(X) = \prod_{i=1}^{n}(X - \alpha_i)$$

be the factorization of $f(X)$ over F. Let $\mathfrak{p}_1, \cdots, \mathfrak{p}_g$ be the distinct primes of F lying over p. Since the extension $F/\mathbb{Q}$ is Galois, we have

$$p\mathcal{O}_F = \prod_{i=1}^{g}\mathfrak{p}_i^e = \bigcap_{i=1}^{g}\mathfrak{p}_i^e$$

where $e \geq 1$ is the ramification degree and the Galois group $\mathrm{Gal}(F/\mathbb{Q})$ acts transitively on the set $\{\mathfrak{p}_1, \cdots, \mathfrak{p}_g\}$.

We note that for any integer $k \geq 1$, $\mathfrak{p}_1^{ek} \cap \mathbb{Z} = p^k\mathbb{Z}$. The inclusion $p^k\mathbb{Z} \subseteq \mathfrak{p}_1^{ek} \cap \mathbb{Z}$ is obvious. For the reverse inclusion, let $x \in \mathfrak{p}_1^{ek} \cap \mathbb{Z}$. For any index i, there exists an automorphism $\sigma_i \in \mathrm{Gal}(F/\mathbb{Q})$ such that $\sigma_i(\mathfrak{p}_1) = \mathfrak{p}_i$. So $x = \sigma(x) \in \mathfrak{p}_i^e$. Hence, $x \in \bigcap_{i=1}^{g}\mathfrak{p}_i^{ek} = p^k\mathbb{Z}$. Thus, $\mathfrak{p}_1^{ek} \cap \mathbb{Z} = p^k\mathbb{Z}$.

For any two integers $x_1, x_2 \in \mathbb{Z}$, we have

$$x_1 - x_2 \in \mathfrak{p}_1^{e\lambda} \Longleftrightarrow x_1 - x_2 \in \mathfrak{p}_1^{e\lambda} \cap \mathbb{Z} = p^\lambda\mathbb{Z}.$$

Hence, the set $\mathbf{S}_\lambda := \{x + \mathfrak{p}_1^{e\lambda} : \ x \in \mathbb{Z}\} \subseteq \mathcal{O}_F/\mathfrak{p}_1^{e\lambda}$ has cardinality p^λ. Now, for any integer $\gamma \in \mathbb{Z}$,

$$\begin{aligned} f(\gamma) \equiv 0 \ (\mathrm{mod}\ p^{n\lambda}) &\Longleftrightarrow f(\gamma) \equiv 0 \ (\mathrm{mod}\ \mathfrak{p}_1^{en\lambda}) \quad (\text{since } \mathfrak{p}_1^{en\lambda} \cap \mathbb{Z} = p^{n\lambda}\mathbb{Z}) \\ &\Longrightarrow \gamma \equiv \alpha_i \ (\mathrm{mod}\ \mathfrak{p}_1^{e\lambda}) \text{ for at least one index } i \quad (\text{pigeonhole principle}), \end{aligned}$$

Since γ is randomly generated, $\gamma \ (\mathrm{mod}\ \mathfrak{p}_1^{e\lambda})$ is randomly and uniformly distributed over the set $\mathbf{S}_\lambda$. Hence,

$$\mathtt{Pr}\big[f(\gamma) \equiv 0 \ (\mathrm{mod}\ p^{n\lambda})\big] \leq \frac{n}{p^\lambda} = \mathtt{negl}(\lambda),$$

which completes the proof. □

Lemma 2.2. (i). *For rationals $d_1, \cdots, d_n \in \mathbb{Q}$ and a randomly generated λ-bit integer γ, if*

$$\sum_{i=1}^{n} d_i\gamma^i \ \in \ \mathbb{Z}_{(\lambda)},$$

with overwhelming probability, $(d_1, \cdots, d_n) \in \mathbb{Z}_{(\lambda)}^n$.

(ii). *Let k be any integer $\geq n\lambda$. Let $\mathcal{S}$ be a set of rational primes and let $\mathbb{Z}_\mathcal{S}$ be the localization of $\mathbb{Z}$ at all primes in S. For rationals $d_1, \cdots, d_n \in \mathbb{Q}$ and a randomly generated λ-bit integer γ, if*

$$\sum_{i=1}^{n} d_i^k\gamma^i \ \in \ \mathbb{Z}_\mathcal{S},$$

with overwhelming probability, $(d_1, \cdots, d_n) \in \mathbb{Z}_\mathcal{S}^n$.

Proof. (i). We show that if $(d_1, \cdots, d_n) \notin \mathbb{Z}_{(\lambda)}^n$, then with overwhelming probability, $\sum_{i=1}^{n} d_i^k\gamma^i \notin \mathbb{Z}_{(\lambda)}$ for a randomly generated λ-bit integer γ. Let D be the least common denominator for $d_1, \cdots, d_n$ and write

$$d_i = \frac{c_i}{D} \ , \ c_i \in \mathbb{Z} \quad (i = 1, \cdots, n).$$

Suppose, by way of contradiction that $(d_1, \cdots, d_n) \notin \mathbb{Z}^n_{(\lambda)}$. Then there exists a rational prime $p < 2^{\lambda-1}$ that divides D but not all of the integers $c_1, \cdots, c_n$. Now, the polynomial $\sum_{i=1}^n c_i X^i \pmod p \in \mathbb{F}_p[X]$ has at most n distinct zeros in $\mathbb{F}_p$ and since γ is randomly and uniformly distributed modulo p,

$$\mathtt{Pr}\big[f(\gamma) \equiv 0 \pmod p\big] = \mathtt{negl}(\lambda).$$

(ii). We show that if $(d_1, \cdots, d_n) \notin \mathbb{Z}^n_{\mathcal{S}}$, then with overwhelming probability, $\sum_{i=1}^n d_i^k \gamma^i \notin \mathbb{Z}_{\mathcal{S}}$ for a randomly generated λ-bit integer γ. Let D be the least common denominator for $d_1, \cdots, d_n$ and write

$$d_i = \frac{c_i}{D} \ \ (c_i \in \mathbb{Z}) \quad \text{for } i = 1, \cdots, n.$$

Suppose, by way of contradiction that $(d_1^k, \cdots, d_n^k) \notin \mathbb{Z}^n_{\mathcal{S}}$. Then there exists a rational prime $p \notin \mathcal{S}$ that divides D but not all of the integers $c_1, \cdots, c_n$. Let $I \subseteq \{1, \cdots, n\}$ be the non-empty subset of indices such that $c_i \notin p\mathbb{Z}$ or equivalently, $c_i^k \notin p^k\mathbb{Z}$. Then

$$\sum_{i \in I} c_i^k \gamma^i \equiv 0 \pmod{p^k}.$$

Now, the polynomial $f(X) := \sum_{i \in I} c_i^k X^i \in \mathbb{Z}[X]$ has degree $\leq n$ and none of its coefficients are divisible by p. By lemma 2.1,

$$\mathtt{Pr}\big[f(\gamma) \equiv 0 \pmod{p^k}\big] = \mathtt{negl}(\lambda).$$

Thus, with overwhelming probability, the rationals d_i^k lie in $\mathbb{Z}_{\mathcal{S}}$. Since the integral domain $\mathbb{Z}_{\mathcal{S}}$ is integrally closed, this implies that the rationals d_i lie in $\mathbb{Z}_{\mathcal{S}}$. □

In particular,

$$\mathtt{Pr}\Big[(d_1, \cdots, d_n) \notin \mathbb{Z}^n_{(\lambda)} \ \Big|\ \sum_{i=1}^n d_i \gamma^i \ \in \ \mathbb{Z}\Big] = \mathtt{negl}(\lambda).$$

Similarly,

$$\mathtt{Pr}\Big[(d_1, \cdots, d_n) \notin \mathbb{Z}^n \ \Big|\ \sum_{i=1}^n d_i^k \gamma^i \ \in \ \mathbb{Z}\Big] = \mathtt{negl}(\lambda) \quad \text{for any } k \geq n\lambda.$$

In a setting where we are dealing with accumulators and the accumulation of primes $< 2^{\lambda-1}$ is disallowed, it suffices for the Prover to show that the rational $\sum_{i=1}^n d_i \gamma^i$ is an element of $\mathbb{Z}_{\mathcal{S}}$ for most applications. This places a much lower computational burden on the Prover. While we haven't used this for the protocols in the paper, we have explored this a bit further in appendix D.

2.8.1 Representations of group elements

Lemma 2.3. (Element representation [Sho97]) *Let $\mathbb{G}$ be a generic group and $\mathcal{A}$ a generic algorithm that makes q_1 queries to $\mathcal{O}_1$ and q_2 queries to $\mathcal{O}_2$. Let $\{g_1, \cdots, g_k\}$ be the outputs of $\mathcal{O}_1$. There is an efficient algorithm Ext that given as input the transcript of $\mathcal{A}$'s interaction with the generic group oracles, produces for every element $u \in \mathbb{G}$ that $\mathcal{A}$ outputs, a tuple $(\alpha_1, \cdots, \alpha_m) \in \mathbb{Z}^m$ such that $u = \prod_{i=1}^m g_i^{\alpha_i}$ and $|\alpha_i| \leq 2^{q_2}$.*

`DLOG`: Following the terminology of [BBF19], the event `DLOG` refers to a generic PPT algorithm $\mathcal{A}$ generating random elements $g_1, \cdots g_n$ by making queries to the group oracle $\mathcal{O}_1$ and generating integers $x_1, \cdots, x_n$ such that

$$\prod_{i=1}^n g_i^{x_i} = 1 \ \in \ \mathbb{G}.$$

The adaptive root assumption implies that $\texttt{DLOG}$ occurs with at most negligible probability.

Lemma 2.4. *Let a, b be elements of a generic hidden order group $\mathbb{G}$. Let $\mathcal{A}$ be a generic PPT algorithm that succeeds at the following task:*
1. *The hashing algorithm $\mathtt{H}_{\mathtt{FS},\lambda}$ generates a λ-bit prime ℓ.*
2. *$\mathcal{A}$ generates an element $Q \in \mathbb{G}$ and an integer $r \in [\ell]$ such that $Q^\ell a^r = b$.*
Then there exists an extractor $\mathcal{E}$ that, if given as input the transcript of $\mathcal{A}$'s interaction with the group oracles, efficiently extracts integers d_1, d_2 bounded by $2^{\mathtt{poly}(\lambda)}$ such that

$$a^{d_1} = b^{d_2} \ \wedge \ d_1 \equiv r d_2 \pmod{\ell}.$$

Proof. By the element representation lemma of [Sho97], for every element $\mathcal{A}$ obtains in response to an $\mathcal{O}_2$ query, an extractor $\mathcal{E}$ succeeds with overwhelming probability at the task of expressing that element as a linear combination $\prod_{i=1}^{k} g_i^{x_i}$ where the group elements $\{g_1, \cdots, g_k\}$ are the responses to $\mathcal{A}$'s queries to the oracle $\mathcal{O}_1$ and the x_i are integers bounded by $2^{\mathtt{poly}(\lambda)}$. Let

$$a = \prod_{i=1}^{k} g_i^{\alpha_i} \ , \ b = \prod_{i=1}^{k} g_i^{\beta_i}, \qquad (\alpha_i, \beta_i \ \in \ \mathbb{Z} \ , \ |\alpha_i|, \ |\beta_i| < 2^{\mathtt{poly}(\lambda)}$$

Now, for a randomly generated λ-bit prime ℓ, $\mathcal{A}$ is able to output $(Q, r) \ \in \ \mathbb{G} \times [\ell]$ such that $Q^\ell a^r = b \in \mathbb{G}$. Hence, with probability $1 - \mathtt{Pr}[\mathtt{DLOG}]$,

$$\beta_1 {\alpha_1}^{-1} \equiv \cdots \equiv \beta_k {\alpha_k}^{-1} \equiv r \pmod{l}.$$

Since the λ-bit prime ℓ was randomly generated, this implies that with overwhelming probability,

$$\beta_1 {\alpha_1}^{-1} = \cdots = \beta_k {\alpha_k}^{-1}.$$

Thus, $a^{\beta_1} = b^{\alpha_1}$. □

Lemma 2.5. *Let $a, \ b_1, \cdots, b_n$ be elements of a generic hidden order group $\mathbb{G}$. Let $\mathcal{A}$ be a generic PPT algorithm that succeeds at the following task:*
1. *The hashing algorithm $\mathtt{H}_{\mathtt{FS},\lambda}$ generates a λ-bit prime ℓ.*
2. *$\mathcal{A}$ sends a tuple $\mathbf{r} = (r_1, \cdots, r_n) \ \in \ [\ell]^n$.*
3. *The hashing algorithm $\mathtt{H}_{\mathtt{FS},\lambda}$ generates a λ-bit challenge γ.*
4. *$\mathcal{A}$ generates an element $Q \in \mathbb{G}$ such that*

$$Q^\ell a^r = \prod_{i=1}^{n} b_i^{\gamma^i} \quad \textit{where} \quad r := \sum_{i=1}^{n} r_i \gamma^i \pmod{\ell}.$$

Then there exists an extractor $\mathcal{E}$ that, if given as input the transcript of $\mathcal{A}$'s interaction with the group oracles, extracts rationals $d_1, \cdots, d_n \in \mathbb{Q}$ such that the numerators/denominators of the d_i are bounded by $2^{\mathtt{poly}(\lambda)}$ and

$$a^{d_i} = b_i \ \ \wedge \ \ d_i \equiv r_i \pmod{\ell} \ \ \forall \, i.$$

Proof. As before, the extractor $\mathcal{E}$ extracts integers $\alpha_j \ (j = 1, \cdots, k)$, $\beta_{i,j} \ (i = 1, \cdots, n \ , \ j = 1, \cdots, k)$ such that

$$a = \prod_{j=1}^{k} g_j^{\alpha_j} \ \ , \ \ b_i = \prod_{j=1}^{k} g_j^{\beta_{i,j}}, \quad (\alpha_i, \ \beta_{i,j} \ \in \ \mathbb{Z} \ , \ |\alpha_i|, \ |\beta_{i,j}| < 2^{\mathtt{poly}(\lambda)})$$

where $\{g_1, \cdots, g_k\}$ are the $\mathbb{G}$-elements obtained from queries to the oracle $\mathcal{O}_1$. Now, in response

to a λ-bit challenge γ, $\mathcal{A}$ generates $Q \in \mathbb{G}$ such that $Q^\ell a^r = \prod_{i=1}^{n} b_i^{\gamma^i}$, where $r := \sum_{i=1}^{n} r_i \gamma^i \pmod{\ell}$. By lemma 2.4, $\mathcal{E}$ can generate a rational $\widetilde{d} = \widetilde{d_1}\widetilde{d_2}^{-1}$ ($\widetilde{d_1}, \widetilde{d_2} \in \mathbb{Z}$) with $\widetilde{d_1}, \widetilde{d_2}$ bounded by $2^{\mathtt{poly}(\lambda)}$ and

$$\prod_{j=1}^{k} g_j^{\widetilde{d}\alpha_j} = a^{\widetilde{d}} = \prod_{i=1}^{n} b_i^{\gamma^i} = \prod_{j=1}^{k} g_j^{\sum_{i=1}^{n} \beta_{i,j}\gamma^i}$$

Since the event DLOG occurs with at most negligible probability, it follows that

$$\widetilde{d} = \sum_{i=1}^{n} (\beta_{i,j} {\alpha_j}^{-1}) \gamma^i \quad \text{for } j = 1, \cdots, k.$$

Since γ is randomly generated, it follows that with overwhelming probability,

$$\beta_{i,1}\alpha_1^{-1} = \cdots = \beta_{i,k} {\alpha_n}^{-1} \quad \text{for } i = 1, \cdots, n.$$

Setting $d_i := \beta_{i,1}\alpha_1^{-1}$ yields $a^{d_i} = b_i \; \forall \; i$.

Now, if $\sum_{i=1}^{n} d_i \gamma^i \not\equiv r \pmod{\ell}$, the extractor $\mathcal{E}$ could efficiently extract an ℓ-th root of a, which would violate the adaptive root assumption. Hence, with overwhelming probability,

$$\sum_{i=1}^{n} d_i \gamma^i \equiv r \equiv \sum_{i=1}^{n} r_i \gamma^i \pmod{\ell}.$$

Since the λ-bit integer γ was randomly generated after the tuple $(r_1, \cdots, r_n)$ was sent, the Schwartz-Zippel lemma implies that with overwhelming probability, $d_i \equiv r_i \pmod{\ell} \; \forall \; i$. □

We will need the following basic fact repeatedly in the subsequent protocols to bridge the gap between sound argument systems and arguments of knowledge.

Fact 2.1. Chinese remainder theorem (CRT): *Let $\ell_1, \cdots, \ell_n$ be pairwise co-prime integers and let $r_1, \cdots, r_n$ be arbitrary integers. Then there is a unique integer x such that $0 \leq 2|x| < \prod_{i=1}^{n} \ell_n$ and $x \equiv r_1 \pmod{\ell_i} \; \forall \; i$. Furthermore, there is a an efficient algorithm for computing x.*

Consider an argument system where a Prover $\mathcal{P}$ claims knowledge of an integer d that satisfies certain properties and is bounded by $2^{\mathtt{poly}(\lambda)}$. Suppose the proof of knowledge sent by $\mathcal{P}$ contains the remainder $r := d \pmod{\ell}$ for a λ-bit prime ℓ randomly generated by the Verifier or by a Fiat-Shamir heuristic in a non-interactive system. An extractor $\mathcal{E}$ can query $\mathcal{P}$ for N accepting transcripts for a sufficiently large integer N. If $\ell_1, \cdots, \ell_N$ are the λ-bit primes and $r_i := d \pmod{\ell_i}$ are the resulting remainders in the accepting transcripts, $\mathcal{E}$ can use the Chinese remainder theorem to compute the unique integer d' such that

$$d' \equiv r_i \pmod{\ell_j} \; \forall \; i \;\; , \;\; 2|d'| < \prod_{i=1}^{N} \ell_i.$$

If N is large enough so that $2^{N\lambda} > |d|$, then with overwhelming probability, $d = d'$. Thus, *assuming the integer d exists*, $\mathcal{E}$ can extract it in expected polynomial runtime.

3 Arguments of Knowledge

In this section, we discuss succinct arguments of the knowledge of exponents in hidden order groups and of relations between these exponents. This is equivalent to the knowledge of relations

between the committed sets or multisets. The proofs can be publicly verified against the succinct commitments held by the Verifier.

3.1 Preliminaries

We briefly review the protocol PoKE (from [BBF19]) which we will need repeatedly in this paper.

Protocol 3.1. *Proof of Knowledge of the Exponent* (PoKE)

Parameters: $\mathbb{G} \xleftarrow{\$} \text{GGen}(\lambda),\ g \in \mathbb{G}$
Input: $u, w \in \mathbb{G}$.
Claim: The Prover posseses an integer x such that $u^x = w$.

1. The Prover $\mathcal{P}$ computes $z := g^x$ and sends z to the Verifier $\mathcal{V}$.
2. The hashing algorithm $\mathtt{H}_{\mathtt{FS},\lambda}$ generates a λ-bit prime ℓ.
3. $\mathcal{P}$ computes the integers q, r such that $x = q \cdot \ell + r,\ r \in [\ell]$.
4. $\mathcal{P}$ computes $Q := u^q,\ Q' = g^q$ and sends (Q, Q', r) to $\mathcal{V}$.
5. $\mathcal{V}$ accepts if and only if $r \in [\ell],\ \ Q^\ell u^r = w,\ \ Q'^\ell g^r = z$. □

The protocol PoKE is an argument of knowledge for the relation

$$\mathcal{R}_{\mathtt{PoKE}} := \left\{ \left((u, w) \in \mathbb{G});\ x \in \mathbb{Z} \right)\ :\ w = u^x \in \mathbb{G} \right\}.$$

The part where $\mathcal{P}$ computes g^x and sends it to $\mathcal{V}$ *before* receiving the challenge ℓ is necessary for the security of the protocol. Without this step, a malicious Prover could convince the Verifier that x is an integer, which might not necessarily be the case.

Clearly, the relation $\mathcal{R}_{\mathtt{PoKE}}$ is transitive in the sense that for elements $a_1, a_2, a_3 \in \mathbb{G}$, if a prover $\mathcal{P}$ possesses integers d_1, d_2 such that $a_1^{d_1} = a_2\ ,\ a_2^{d_2} = a_3$, then he possesses the integer $d_1 d_2$ which fulfills the equation $a_1^{d_1 d_2} = a_3$. Henceforth, we denote the proof of knowledge of the discrete logarithm between $a, b \in \mathbb{G}$ by $\mathtt{PoKE}[a,\ b]$.

The knowledge of exponents can easily be aggregated when they share a common base. Given elements

$$a \in \mathbb{G}\ ,\ (b_1, \cdots, b_n) \in \mathbb{G}^n,$$

and a radomly generate integer γ, if the Prover possesses a rational $\widetilde{d}$ such that $a^{\widetilde{d}} = \prod_{i=1}^n b_i^{\gamma^i}$, then the Prover, with overwhelming probability, possesses rationals d_i such that $a^{d_i} = b_i\ \forall\ i$. In the special case where $a = g$, a randomly generated element of the group $\mathbb{G}$, the fractional root assumption implies that the d_i must be integers. In the more general case, we use lemma 2.2 to show that the d_i are integers rather than merely rationals.

We now start out with a fairly simple protocol. We show how a Prover could probabilistically demonstrate that two discrete logarithms are equal, with a constant-sized proof. In other words, the protocol allows a Prover to show that two pairs $[a_1, b_1]$, $[a_2, b_2]$ of $\mathbb{G}$-elements are commitments to the same set/multiset. We provide an argument of knowledge for the following relation:

$$\mathcal{R}_{\mathtt{EqDLog}}[(a_1, b_1),\ (a_2, b_2)] = \left\{ \begin{array}{l} ((a_1, b_1),\ (a_2, b_2) \in \mathbb{G}^2 \\ d \in \mathbb{Z} : \\ (b_1, b_2) = (a_1^d, a_2^d) \end{array} \right\}$$

The protocol hinges on the observation that for two integers d_1, d_2, if we have $d_1 \equiv d_2 \pmod{\ell}$ for a randomly generated λ-bit prime ℓ, then with overwhelming probability, $d_1 = d_2$.

Protocol 3.2. *Proof of equality of discrete logarithms* (PoEqDLog) :
Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda),\ g \in \mathbb{G}$.
Input: $a_1, a_2, b_1, b_2 \in \mathbb{G}$.
Claim: The Prover possesses an integer d such that $a_1^d = b_1$ and $a_2^d = b_2$.

1. The Prover $\mathcal{P}$ sends $\widetilde{g} := g^d$ to the Verifier $\mathcal{V}$.
2. The hashing algorithm $\mathtt{H}_{\mathtt{FS},\lambda}$ generates a λ-bit prime ℓ.
3. $\mathcal{P}$ computes the integers q, r such that $d = q \cdot \ell + r$, $r \in [\ell]$ and the group elements

$$Q_1 := a_1^q,\ \ Q_2 := a_2^q,\ \ \breve{g} := g^q$$

and sends $(Q_1, Q_2, \breve{g}, r)$ to $\mathcal{V}$.
4. $\mathcal{V}$ verifies the equations

$$r \in [\ell],\ \ Q_1^\ell a_1^r \stackrel{?}{=} b_1,\ \ Q_2^\ell a_2^r \stackrel{?}{=} b_2,\ \ (\breve{g})^\ell g^r \stackrel{?}{=} \widetilde{g}$$

and accepts if and only if all equations hold. □

Thus, the proof consists of four $\mathbb{G}$-elements and one λ-bit integer. In a setting where the base of the commitment to a set/multiset needs to be changed, the Prover can verifiably send the new commitment using this protocol. The Verifier does not need to access the data to verify that the new commitment is the correct.

Proposition 3.3. *The protocol* $\mathtt{EqDLog}[(a_1, b_1),\ (a_2, b_2)]$ *is an argument of knowledge for the relation* $\mathcal{R}_{\mathtt{EqDLog}}$ *in the generic group model.*

Proof. An extractor $\mathcal{E}$ can simulate the extractors for the subprotocols $\mathtt{PoKE}[a_i,\ b_i]$ $(i = 1, 2)$ to extract integers d_1, d_2 such that $a_1^{d_1} = b_1$, $a_2^{d_2} = b_2$. The low order assumption implies that the pair (d_1, d_2) is unique except with negligible probability.

Now, for an accepting transcript, the equations $Q_i^l a_i^r = b_i$ imply that either $d_i \equiv r \pmod{\ell}$ or $\mathcal{E}$ can efficiently extract an ℓ-th root of a_i. Since the λ-bit prime was ℓ randomly generated, the latter event would violate the adaptive root assumption. So, with overwhelming probability, $d_1 \equiv r \equiv d_2 \pmod{\ell}$. Since the λ-bit prime was randomly generated, this implies that with overwhelming probability, $d_1 = d_2$. □

We can also generalize the protocol EqDLog as follows. For a public polynomial $f(X) \in \mathbb{Z}[X]$, an honest Prover can provide a constant-sized proof that he possesses integers d_1, d_2 such that

$$a_1^{d_1} = b_1\ ,\ a_2^{d_2} = b_2\ ,\ f(d_1) = d_2.$$

We provide an argument of knowledge for the relation

$$\mathcal{R}_{\mathtt{PolyDLog}}[(a_1, b_1),\ (a_2, b_2),\ f] = \left\{ \begin{array}{l} \big((a_1, b_1), (a_2, b_2) \in \mathbb{G}^2,\ f \in \mathbb{Z}[X]\big); \\ (d_1, d_2) \in \mathbb{Z}^2) : \\ b_1 = a_1^{d_1} \bigwedge b_1 = a_1^{d_1} \bigwedge d_2 = f(d_1) \end{array} \right\}$$

Protocol 3.4. *Proof of Polynomial relation between discrete logarithms* (PoPolyDLog) :
Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda),\ g \in \mathbb{G}$.
Input: Elements $a_1, b_1, a_2, b_2 \in \mathbb{G}$, a public polynomial $f(X) \in \mathbb{Z}[X]$.
Claim: The Prover possesses integers $d_1,\ d_2$ such that:
-$a_1^{d_1} = b_1,\ a_2^{d_2} = b_2$
-$f(d_1) = d_2$

1. The Prover $\mathcal{P}$ computes $\widetilde{g}_1 := g^{d_1}$, $\widetilde{g}_2 = g^{d_2} \in \mathbb{G}$ and sends them to the Verifier $\mathcal{V}$.
2. The hashing algorithm $\mathtt{H}_{\mathtt{FS},\lambda}$ generates a λ-bit prime ℓ.
3. $\mathcal{P}$ computes the integers q_1, q_2, r_1, r_2 such that

$$d_1 = q_1 \cdot \ell + r_1 \ , \ d_2 = q_2 \cdot \ell + r_2 \ , \ \ r_1, r_2 \in [\ell].$$

4. $\mathcal{P}$ computes the elements $Q_1 := a_1^{q_1}$, $Q_2 := a_2^{q_2}$, $g_1 := g^{q_1}$, $g_2 := g^{q_2} \in \mathbb{G}$ and sends them to $\mathcal{V}$ along with the integer r_1.
5. $\mathcal{V}$ verifies that $r_1 \in [\ell]$ and computes $r_2 := f(r_1) \pmod{\ell}$.
6. $\mathcal{V}$ verifies the equations

$$Q_1^{\ell} a_1^{r_1} \stackrel{?}{=} b_1 \ \bigwedge \ Q_2^{\ell} a_2^{r_2} \stackrel{?}{=} b_2 \ \bigwedge \ (g_1)^{\ell} g^{r_1} \stackrel{?}{=} \widetilde{g}_1 \ \bigwedge \ (g_2)^{\ell} g^{r_2} \stackrel{?}{=} \widetilde{g}_2$$

and accepts the validity of the claim if and only if all equations hold. □

Thus, the proof consists of six elements of $\mathbb{G}$ and one λ-bit integer.

Proposition 3.5. *The protocol* `PoPolyDLog` *is an argument of knowledge for the relation* $\mathcal{R}_{\mathtt{PolyDLog}}$ *in the generic group model.*

Proof. This is a special case of the protocol `PoMultPolyDLog`, which we provide a security proof for in the next section. □

If a Verifier holds commitments to two multisets $\mathcal{M}, \mathcal{N}$, a Prover with access to these multisets can use the protocol `PoPolyDLog` to show whether or not the underlying sets $\mathtt{Set}(\mathcal{M})$, $\mathtt{Set}(\mathcal{N})$ coincide or if one is contained in the other. We have described these protocols in the appendix. The proofs are constant-sized and are public verifiable against the commitments.

3.2 Aggregating the knowledge of multiple exponents

In this section, we discuss protocols for aggregating the proofs of knowledge of multiple exponents and proofs of certain relations between these exponents. This amounts to demonstrating relations between multiple sets/multisets through non-interactive proofs that can be publicly verified against the commitments to these sets/multisets. The proofs consist of elements of the hidden order $\mathbb{G}$ and λ-bit integers arising from the remainders of the exponents modulo the prime challenges. Using lemma 2.2 and a few more techniques, we have designed the protocols so that the number of $\mathbb{G}$-elements is constant and hence, independent of the number of exponents involved. The proof sizes are $\mathbf{O}(n)$ since they consist of $\mathbf{O}(n)$ λ-bit integers. However, in practice, this is a lot more efficient in terms of the communication complexity than proofs with $\mathbf{O}(n)$ group elements.

The first protocol in this section allows us demonstrate the knowledge of multiple integer exponents when they share a common base. We call this protocol the *Proof of Aggregated Knowledge of the Exponents* 1 or `PoAggKE-1` for short. We provide an argument of knowledge for the relation:

$$\mathcal{R}_{\mathtt{AggKE-1}}[a, \ \mathbf{a}] = \left\{ \begin{array}{l} (a \in \mathbb{G} \ , \ \mathbf{a} = (a_1, \cdots, a_n) \in \mathbb{G}^n); \\ (d_1, \cdots, d_n) \in \mathbb{Z}^n) : \\ a_i = a^{d_i} \ \forall \ i \end{array} \right\}$$

In addition to the cryptographic assumptions for generic groups, the security of the protocol hinges on the Schartz-Zippel lemma, which we state here.

Lemma 3.6. (Schwartz-Zippel) : *Let F be a field and let $f \in F[X_1, \cdots, X_n]$ be a polynomial. Let $r_1, \cdots, r_n$ be selected randomly and uniformly from a subset $S \subseteq F$. Then*

$$\texttt{Pr}\big[f(r_1, \cdots, r_n) = 0\big] \leq \frac{\deg(f)}{|S|}.$$

Protocol 3.7. *Proof of aggregated knowledge of exponents* 1 (`PoAggKE-1`):

Parameters: $\mathbb{G} \xleftarrow{\$} \text{GGen}(\lambda),\ g \in \mathbb{G}$.
Input: Elements $a \in \mathbb{G}$, $(b_1, \cdots, b_n) \in \mathbb{G}^n$ for some integer $n \geq 1$.
Claim: The Prover possesses $\mathbf{d} := (d_1, \cdots, d_n) \in \mathbb{Z}^n$ such that $a^{d_i} = b_i$ for $i = 1, \cdots, n$.

1. The hashing algorithm $\texttt{H}_{\texttt{FS},\lambda}$ generates a λ-bit challenge γ.
2. The Prover $\mathcal{P}$ computes

$$p := \texttt{NextPrime}(n\lambda)\ \ ,\ \ \widetilde{g} := g^{\sum_{i=1}^{n} d_i^p \gamma^i}$$

and sends $\widetilde{g}$ to the Verifier $\mathcal{V}$.

3. The hashing algorithm $\texttt{H}_{\texttt{FS},\lambda}$ generates a λ-bit prime $\ell \not\equiv 1 \pmod{p}$.
4. $\mathcal{P}$ computes the integers $r_i := d_i \pmod{\ell}$ and sends the tuple $(r_1, \cdots, r_n)$ to $\mathcal{V}$.
5. $\mathcal{P}$ computes the integers $\widetilde{q}$, $\widetilde{r}$ such that

$$\sum_{i=1}^{n} d_i^p \gamma^i = \widetilde{q} \cdot \ell + \widetilde{r}, \quad \widetilde{r} \in [\ell]$$

and sends $\breve{g} := g^{\widetilde{q}} \in \mathbb{G}$ to $\mathcal{V}$.

6. The hashing algorithm $\texttt{H}_{\texttt{FS},\lambda}$ generates a λ-bit prime γ_0.
7. $\mathcal{P}$ computes integers q_0, r_0 such that

$$\sum_{i=1}^{n} d_i \gamma_0^i = q_0 \cdot \ell + r_0\ ,\ r_0 \in [\ell]$$

and sends $Q_0 := a^{q_0} \in \mathbb{G}$ to $\mathcal{V}$.

8. $\mathcal{V}$ verifies that $(r_1, \cdots, r_n) \in [\ell]^n$ and computes

$$b := \prod_{i=1}^{n} b_i^{\gamma^i}\ ,\ b_0 := \prod_{i=1}^{n} b_i^{\gamma_0^i} \in \mathbb{G}\ \ ,\ \ \widetilde{r} := \sum_{i=1}^{n} r_i^p \gamma^i \pmod{\ell}\ ,\ r_0 := \sum_{i=1}^{n} r_i \gamma_0^i \pmod{\ell}.$$

9. $\mathcal{V}$ verifies the equations

$$(Q_0)^\ell a^{r_0} \stackrel{?}{=} b_0 \quad \wedge \quad (\breve{g})^\ell g^{\widetilde{r}} \stackrel{?}{=} \widetilde{g}.$$

and accepts the validity of the claim if and only if both equations hold. □

Thus, the proof consists of three $\mathbb{G}$-elements and n λ-bit integers. In particular, the number of $\mathbb{G}$-elements is constant-sized and independent of the number of exponents. For the security of the protocol, it is necessary that the challenge γ_0 is generated *after* the remainders $r_1, \cdots, r_n$ have been committed. In a non-interactive setting, this means the hashing algorithm that generates γ_0 takes the tuple $\mathbf{r} := (r_1, \cdots, r_n) \in [\ell]^n$ of remainders modulo ℓ as one of its inputs. Hence, the remainders $r_i := d_i \pmod{\ell}$ must be honestly computed by the Prover in order to succeed at the additional task of computing the element $Q_0 \in \mathbb{G}$ such that $(Q_0)^\ell a^{r_0} = b_0$.

In the special case where a is randomly generated by the oracle, the subprotocol where the Prover computes the element $\widetilde{g}$ and sends it to the Verifier is redundant. So the proof would be smaller and the Prover's computational burden would be substantially lower in this special case.

When $a \in \mathbb{G}$ is not a randomly generated element, the most expensive part of the proof generation is computing the element $\widetilde{g}$. The effective runtime for this can be mitigated by the Prover pre-computing the set

$$\{g^{2^i} : 1 \leq i \leq N\} \subseteq \mathbb{G}$$

for some appropriately large integer N.

The Verifier's work can be further reduced by the Prover sending succinct proofs of (multi-)exponentiation for the equations

$$b := \prod_{i=1}^{n} b_i^{\gamma^i} \ , \ b_0 := \prod_{i=1}^{n} b_i^{\gamma_0^i} \in \mathbb{G},$$

instead of the Verifier independently performing these computations. We have described this minor generalization of Wesolowski's protcol in Appendix C.

Theorem 3.8. *The protocol* $\texttt{PoAggKE} - \texttt{1}$ *is an argument of knowledge for the relation* $\mathcal{R}_{\texttt{AggKE}-\texttt{1}}$ *in the generic group model.*

Proof. The low order assumption implies that except with overwhelming probability, any PPT algorithm can generate at most a unique tuple $(x_1, \cdots, x_n) \in \mathbb{Q}^n$ such that $a^{x_i} = b_i$. We first show that the protocol is sound, i.e. if the Prover sends an accepting transcript, then with overwhelming probability, he possesses integers d_i bounded by $2^{\texttt{poly}(\lambda)}$ such that $a^{d_i} = b_i \ \forall \ i$. We then show how an extractor $\mathcal{E}$ could extract the integers d_i in expected polynomial runtime by applying the Chinese remainder theorem to a sufficiently large number of accepting transcripts.

Since the equation $Q_0^{\ell} a^{r_0} = b_0$ holds, lemma 2.5 implies that with overwhelming probability, $\mathcal{P}$ can generate rationals $d_1, \cdots, d_n$ with numerators/denominators bounded by $2^{\texttt{poly}(\lambda)}$ such that

$$a^{d_i} = b_i \ \ , \ \ \sum_{i=1}^{n} d_i \gamma_0^i \equiv \sum_{i=1}^{n} r_i \gamma_0^i \pmod{\ell},$$

where exponentiation to these rational powers is defined in Definition 2.5. Since the λ-bit challenge γ_0 is randomly generated after the tuple $(r_1, \cdots, r_n) \in [\ell^n]$ has been sent by the Prover, the Schwartz-Zippel lemma implies that with overwhelming probability, $d_i \equiv r_i \pmod{\ell} \ \forall \ i$.

Furthermore, since the λ-bit prime ℓ is randomly generated after the Prover sends $\widetilde{g}$, the $\widetilde{g} = (\breve{g})^{\ell} g^{\widetilde{r}}$ and lemma 2.4 imply that with overwhelming probability, $\mathcal{P}$ possesses a rational $\widetilde{d}$ such that

$$\widetilde{g} = g^{\widetilde{d}} \ \ , \ \ \widetilde{d} \equiv \widetilde{r} \equiv \sum_{i=1}^{n} r_i^p \gamma^i \equiv \sum_{i=1}^{n} d_i^p \gamma^i \pmod{\ell}.$$

The fractional root assumption then implies that with overwhelming probability, $\widetilde{d} \in \mathbb{Z}$. Since ℓ is randomly generated after $\widetilde{g}$ has been sent,

$$\widetilde{d} \equiv \sum_{i=1}^{n} d_i^p \gamma^i \pmod{\ell} \quad \stackrel{\text{o.p.}}{\Longrightarrow} \quad \widetilde{d} = \sum_{i=1}^{n} d_i^p \gamma^i.$$

Since $p > n\lambda$, lemma 2.2 now implies that with overwhelming probability, $d_i \in \mathbb{Z} \ \forall \ i$. This completes the proof of the soundness of the protocol.

Let N be the (polynomially bounded) number of queries from $\mathcal{A}$ to the group oracles $\mathcal{O}_1$, $\mathcal{O}_2$.

With overwhelming probability, $2^N \geq \max\{2|d_i| \ : \ 1 \leq i \leq n\}$. An extractor $\mathcal{E}$ can extract the tuple $(d_1, \cdots, d_n) \in \mathbb{Z}^n$ as follows.

$\mathcal{E}$ makes queries to $\mathcal{A}$, keeping the γ fixed and sampling the ℓ randomly and uniformly from the set of λ-bit primes. Let $\ell_1, \cdots, \ell_k$ denote the prime challenges in the first k accepting transcripts and let $\mathbf{r}_j \in [\ell_j]^n$ denote the tuple of remainders modulo ℓ_j in the j-th accepting transcript. $\mathcal{E}$ uses the Chinese remainder theorem to efficiently compute the unique tuple $\mathbf{e} = (e_1, \cdots, e_n) \in \mathbb{Z}^n$ such that

$$\mathbf{e} \equiv \mathbf{r}_j \pmod{\ell_j} \ \forall \ j \ \ , \ \ 2|e_i| < \prod_{j=1}^{k} \ell_j \ \forall \ i.$$

After receiving a new accepting transcript and updating the tuple $\mathbf{e}$, the extractor $\mathcal{E}$ checks whether $a^{e_i} = a_i$ for $i = 1, \cdots, n$. If so, he halts the process. Otherwise, he queries for another accepting transcript and continues to update $\mathbf{e}$ by applying the Chinese remainder theorem to the updated list of accepting transcripts.

When the number of accepting transcripts is $N + 1$, we have

$$\max\{2|d_i| \ : \ 1 \leq i \leq n\} < 2^{N+1} < \prod_{j=1}^{N+1} \ell_j$$

except with negligible probability. Thus, with overwhelming probability, $e_i = d_i \ \forall \ i$ at this point. Thus, $\mathcal{E}$ extracts the d_i in expected polynomial runtime. $\square$

In the next protocol, we generalize the protocol `PolyDLog` to multiple discrete logarithms. We provide an argument of knowledge for the relation:

$$\mathcal{R}_{\texttt{MultPolyDLog}}[a, \ (b_1, \cdots, b_n), \ (f_1, \cdots, f_k)] = \left\{ \begin{array}{l} (a \in \mathbb{G}, \ (b_1, \cdots, b_n) \in \mathbb{G}^n); \\ (f_1, \cdots, f_k) \in \mathbb{Z}[X_1, \cdots, X_n]^k; \\ (d_1, \cdots, d_n) \in \mathbb{Z}^n) \ : \\ b_i = a^{d_i} \ \forall \ i \ \bigwedge \\ f_j(d_1, \cdots, d_n) = 0 \ \forall \ j \end{array} \right\}$$

Protocol 3.9. *Proof of multivariate polynomial relations between discrete logarithms* (`PoMultPolyDLog`) :

Parameters: $\mathbb{G} \xleftarrow{\$} \text{GGen}(\lambda), \ g \in \mathbb{G}$.
Input: Elements $a \in \mathbb{G}$, $(b_1, \cdots, b_n) \in \mathbb{G}^n$ for some integer $n \geq 1$; public n-variate polynomials $f_1, \cdots, f_k \ \in \ \mathbb{Z}[X_1, \cdots, X_n]$.
Claim: The Prover possesses integers $d_1, \cdots, d_n$ such that:
- $a^{d_i} = b_i$ for $i = 1, \cdots, n$.
- $f_j(d_1, \cdots, d_n) = 0$ for $j = 1, \cdots, k$.

1. The hashing algorithm $\texttt{H}_{\texttt{FS},\lambda}$ generates a λ-bit integer γ.
2. $\mathcal{P}$ computes

$$p := \texttt{NextPrime}(n\lambda) \ \ , \ \ \widetilde{g} := g^{\sum_{i=1}^{n} d_i^p \gamma^i} \ \in \ \mathbb{G}$$

and sends $\widetilde{g}$ to the Verifier $\mathcal{V}$.

3. The hashing algorithm $\texttt{H}_{\texttt{FS},\lambda}$ generates a λ-bit prime $\ell \not\equiv 1 \pmod{p}$.
4. $\mathcal{P}$ computes the integers $r_i := d_i \pmod{\ell}$ $(i = 1, \cdots, n)$ and the integers $\widetilde{q}$, $\widetilde{r}$ such that

$$\sum_{i=1}^{n} d_i^p \gamma^i = \widetilde{q} \cdot \ell + \widetilde{r} \ , \ \widetilde{r} \in [\ell]$$

and sends $(r_1, \cdots, r_n) \in [\ell]^n$ to V.

5. $\mathcal{P}$ computes $\breve{g} := g^{\widetilde{q}} \in \mathbb{G}$ and sends $\breve{g}$ to $\mathcal{V}$.

6. The hashing algorithm $\mathtt{H}_{\mathtt{FS},\lambda}$ generates a λ-bit prime γ_0.

7. $\mathcal{P}$ computes the integers q_0, r_0 such that

$$\sum_{i=1}^{n} d_i \gamma_0^i = q_0 \cdot \ell + r_0 \ , \ r_0 \in [\ell].$$

He computes $Q_0 := a^{q_0} \in \mathbb{G}$ and sends Q_0 to $\mathcal{V}$.

8. $\mathcal{V}$ verifies that $(r_1, \cdots, r_n) \in [\ell]^n$ and computes

$$b := \prod_{i=1}^{n} b_i^{\gamma^i} \ , \ b_0 := \prod_{i=1}^{n} b_i^{\gamma_0^i} \ \in \ \mathbb{G} \ , \ \widetilde{r} := \sum_{i=1}^{n} r_i^p \gamma^i \ (\mathrm{mod}\ \ell) \ , \ r_0 := \sum_{i=1}^{n} r_i \gamma_0^i \ (\mathrm{mod}\ \ell).$$

9. $\mathcal{V}$ verifies the equations

$$(Q_0)^\ell a^{r_0} \stackrel{?}{=} b_0 \ \bigwedge \ (\breve{g})^\ell g^{\widetilde{r}} \stackrel{?}{=} \widetilde{g} \ \bigwedge \ \Big(\bigwedge_{j=1}^{k} f_j(r_1, \cdots, r_n) \stackrel{?}{\equiv} 0 \ (\mathrm{mod}\ \ell) \Big)$$

and accepts the validity of the claim if and only if all equations hold. □

Thus, the proof consists of four $\mathbb{G}$-elements and n λ-bit integers. We note that the additional challenge γ_0 is necessary for the security of the protocol. A malicious Prover $\mathcal{P}_{\mathrm{mal}}$ could forge a fake proof as follows.

1. $\mathcal{P}_{\mathrm{mal}}$ computes integers $r_1, \cdots, r_n \in [\ell]$ such that

$$\sum_{i=1}^{n} d_i^p \gamma^i \equiv \sum_{i=1}^{n} r_i^p \gamma^i \ (\mathrm{mod}\ \ell) \ , \ \bigwedge_{j=1}^{k} f_j(r_1, \cdots, r_n) \equiv 0 \ (\mathrm{mod}\ \ell)$$

but $d_i \not\equiv r_i \ (\mathrm{mod}\ \ell)$ for some or all indices i. The malicious Prover can succeed in this task with non-negligible probability.

2. $\mathcal{P}_{\mathrm{mal}}$ then sends $(r_1, \cdots, r_n)$ to the Verifier.

3. The Verifier is thus tricked into believing that $f_j(d_1, \cdots, d_n) = 0$, which might not necessarily be the case.

Now, in our protocol, γ_0 is randomly generated by the Fiat-Shamir heuristic *after* the Prover sends $(r_1, \cdots, r_n)$. In a non-interactive setting, this means the hashing algorithm that generates the challenge γ_0 takes the λ-bit integers $(r_1, \cdots, r_n)$ as one of its inputs. Hence,

$$\mathtt{Pr}\Big[\sum_{i=1}^{n} d_i \gamma_0^i \equiv \sum_{i=1}^{n} r_i \gamma_0^i \ (\mathrm{mod}\ \ell) \ \Big| \ d_i \not\equiv r_i \ (\mathrm{mod}\ \ell) \text{ for some } i \Big] = \mathtt{negl}(\lambda).$$

So the elements $(r_1, \cdots, r_n)$ must be honestly computed in order to succeed at the additional task of computing the element $\widehat{Q}_0$ such that

$$(\widehat{Q}_0)^\ell a^{\widehat{r}_0} = \prod_{i=1}^{n} a_i^{\gamma_0^i}$$

with non-negligible probability. A special case that we will need repeatedly in the subsequent protocols is where f is the $(n+1)$-variate polynomial

$$f(X_1, \cdots, X_n, X_{n+1}) := \big(\prod_{i=1}^{n} X_i\big) - X_{n+1}.$$

We will need this case for some of the subsequent protocols for demonstrating pairwise disjointness of committed data sets/multisets.

As was the case with `AggKE-1`, in the special case where $a = g$, the subprotocol where the Prover computes the element $\widetilde{g}$ and sends it to the Verifier is redundant. So the proof would be smaller and the Prover's computational burden would be substantially lower in this special case.

Proposition 3.10. *The protocol* `PoMultPolyDLog` *is an argument of knowledge for the relation* $\mathcal{R}_{\texttt{MultPolyDLog}}$ *in the generic group model.*

Proof. The low order assumption implies that except with negligible probability, any PPT algorithm can output at most a unique tuple $(d_1, \cdots, d_n) \in \mathbb{Q}^n$ such that $a^{d_i} = b_i$. We first show that the protocol is sound, i.e. in case of an accepting transcript, the Prover possesses integers d_i such that

$$a^{d_i} = b_i \; \forall \; i \;\; \bigwedge \;\; f_j(d_1, \cdots, d_n) = 0 \; \forall \; j$$

except with negligible probability. We then show that an extractor $\mathcal{E}$ can efficiently extract the integers d_i from sufficiently many accepting transcripts in expected polynomial runtime, using the Chinese remainder theorem.

Since the equation $Q_0^{\ell} a^{r_0} = \prod_{i=1}^{n} b_i^{\gamma_0^i}$ holds, lemma 2.5 implies that with overwhelming probability, the Prover can generate rationals $d_1, \cdots, d_n$ with numerators and denominators bounded by $2^{\texttt{poly}(\lambda)}$ such that

$$a^{d_i} = b_i \text{ for } i = 1, \cdots, n \;\; \bigwedge \;\; \sum_{i=1}^{n} d_i \gamma_0^i \equiv \sum_{i=1}^{n} r_i \gamma_0^i \pmod{\ell}.$$

Since the challenge γ_0 is randomly generated after the Prover has sent $(r_1, \cdots, r_n)$, γ_0 is randomly and uniformly distributed modulo ℓ. Hence, the Schwartz-Zippel lemma implies that with overwhelming probability, $d_i \equiv r_i \pmod{\ell}$ for every index i. Now, since the λ-bit prime ℓ is randomly generated,

$$\texttt{Pr}\Big[f_j(d_1, \cdots, d_n) \equiv 0 \pmod{\ell} \;\Big|\; f_j(d_1, \cdots, d_n) \neq 0 \Big] = \texttt{negl}(\lambda).$$

Hence, a union bound over the f_j show that if $f_j(d_1, \cdots, d_n) \equiv 0 \pmod{\ell} \; \forall \; j$, then $f_j(d_1, \cdots, d_n) = 0 \; \forall \; j$ except with negligible probability.

It remains to argue that the rationals d_i are integers. In the protocol, the Verifier independently computes

$$\widetilde{r} := \sum_{i=1}^{n} r_i^p \gamma^i \;\in\; [\ell]$$

So, the equation $\widetilde{g} = (\breve{g})^{\ell} g^{\widetilde{r}}$ and lemma 2.4 imply that the Prover can generate a rational $\widetilde{d}$ with its numerator and denominator bounded by $2^{\texttt{poly}(\lambda)}$ and such that

$$\widetilde{g} = g^{\widetilde{d}} \;\; , \;\; \widetilde{d} \equiv \widetilde{r} \equiv \sum_{i=1}^{n} r_i^p \gamma^i \equiv \sum_{i=1}^{n} d_i^p \gamma^i \pmod{\ell}.$$

If $\widetilde{d} \notin \mathbb{Z}$, the tuple $(g, \widetilde{g}, \widetilde{d}) \in \mathbb{G}^2 \times (\mathbb{Q} \setminus \mathbb{Z})$ would violate the fractional root assumption. So $\widetilde{d}$

is an integer, except with negligible probability. Since the λ-bit prime ℓ is randomly generated after $\widetilde{g}$ has been sent by the Prover, the congruence

$$\widetilde{d} \equiv \sum_{i=1}^{n} d_i^p \gamma^i \pmod{\ell}$$

implies that with overwhelming probability, $\widetilde{d} = \sum_{i=1}^{n} d_i^p \gamma^i$. Hence, $\sum_{i=1}^{n} d_i^p \gamma^i \in \mathbb{Z}$ and lemma 2.2 then implies that $d_i \in \mathbb{Z} \; \forall \; i$,except with negligible probability.

Let N be the (polynomially bounded) number of queries from $\mathcal{A}$ to the group oracles $\mathcal{O}_1$, $\mathcal{O}_2$. With overwhelming probability, $2^N \geq \max\{2|d_i| \; : \; 1 \leq i \leq n\}$. An extractor $\mathcal{E}$ can extract the tuple $(d_1, \cdots, d_n) \in \mathbb{Z}^n$ as follows.

$\mathcal{E}$ makes queries to $\mathcal{A}$, keeping the γ fixed and sampling the ℓ randomly and uniformly from the set of λ-bit primes. Let $\ell_1, \cdots, \ell_k$ denote the prime challenges in the first k accepting transcripts and let $\mathbf{r}_j \in [\ell_j]^n$ denote the tuple of remainders modulo ℓ_j in the j-th accepting transcript. $\mathcal{E}$ uses the Chinese remainder theorem to efficiently compute the unique tuple $\mathbf{e} = (e_1, \cdots, e_n) \in \mathbb{Z}^n$ such that

$$\mathbf{e} \equiv \mathbf{r}_j \pmod{\ell_j} \; \forall \; j \;\; , \;\; 2|e_i| < \prod_{j=1}^{k} \ell_j \; \forall \; i.$$

After receiving a new accepting transcript and updating the tuple $\mathbf{e}$, the extractor $\mathcal{E}$ checks whether $a^{e_i} = a_i$ for $i = 1, \cdots, n$. If so, he halts the process. Otherwise, he queries for another accepting transcript and continues to update $\mathbf{e}$ by applying the Chinese remainder theorem to the updated set of accepting transcripts.

When the number of accepting transcripts is $N + 1$, we have

$$\max\{2|d_i| \; : \; 1 \leq i \leq n\} < 2^{N+1} < \prod_{j=1}^{N+1} \ell_j$$

except with negligible probability. Thus, with overwhelming probability, $e_i = d_i \; \forall \; i$ at this point. Thus, $\mathcal{E}$ extracts the d_i in expected polynomial runtime. $\square$

We now discuss a relation (and its argument of knowledge) that is a dual to the relation `AggKE-1`. Instead of the multiple exponents having a common base, we consider the case where they exponentiate to the same power. We provide an argument of knowledge for the following relation:

$$\mathcal{R}_{\texttt{AggKE-2}}[(a_1, \cdots, a_n), \; A] = \left\{ \begin{array}{l} ((a_1, \cdots, a_n) \in \mathbb{G}^n, \; A \in \mathbb{G}) \\ (d_1, \cdots, d_n) \in \mathbb{Z}^n) : \\ A = a_i^{d_i} \; \forall \; i \end{array} \right\}$$

The most important special case is when $A \in \mathbb{G}$ is the accumulated digest of a data set and the a_i are membership witnesses of subsets. In this case, the protocol allows a Prover to show that he possesses certain data sets inserted in to the accumulator, the membership witnesses for which are held by the Verifier.

Given elements $a_1, \cdots, a_n, A \in \mathbb{G}$ such that

$$A = a_1^{d_1} = \cdots = a_n^{d_n}$$

where the integers d_i are known to him, the Prover can efficiently compute $D := \mathbf{lcm}(d_1, \cdots, d_n)$ and using Shamir's trick, an element $a \in \mathbb{G}$ such that $a^D = A$ in runtime $\mathbf{O}(n \log(n))$. Now,

the protocols $\texttt{PoAggKE-1}[a,\ (a_1,\cdots,a_n)]$ and $\texttt{PoKE}[a,\ A]$, when combined, would demonstrate that the Prover possesses the integers $\widehat{d}_i$, D such that $a^{\widehat{d}_i} = a_i$, $a^D = A$ and hence, can generate the discrete logarithms $D\widehat{d}^{-1}$ between a_i and A for every i. However, these protocols do not prove that these discrete logarithms are, in fact, integers. To that end, the Prover needs to demonstrate that the rationals $D\widehat{d}_i^{-1}$ $(i = 1,\cdots,n)$ are integers. This can be addressed by the Prover verifiably sending the element

$$\widetilde{g} := g^{\sum_{i=1}^{n} (D\widehat{d}^{-1})^{n\lambda}\gamma^i} \in \mathbb{G}$$

for some randomly generated λ-bit integer γ. lemma 2.2 and the fractional root assumption then imply that with overwhelming probability, the dd_i^{-1} are integers.

Protocol 3.11. *Proof of aggregated knowledge of exponents* 2 ($\texttt{PoAggKE-2}$):

Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda),\ g \in \mathbb{G}$
Input: $(a_1,\cdots,a_n) \in \mathbb{G}^n$, $A \in \mathbb{G}$.
Claim: The Prover posseses integers $d_1,\cdots,d_n$ such that $a_i^{d_i} = A$.

1. The Prover $\mathcal{P}$ computes the integers

$$D := \mathbf{lcm}(d_1,\cdots,d_n)\ \ ,\ \ \widehat{d}_i := D \cdot d_i^{-1}\ (i = 1,\cdots,n).$$

Using Shamir's trick, he computes an element $a \in \mathbb{G}$ such that $a^D = A$ and sends a to the Verifier $\mathcal{V}$ along with a non-interactive $\texttt{PoKE}[a,\ A]$.

2. The hashing algorithm $\mathtt{H}_{\mathtt{FS},\lambda}$ generates a λ-bit integer γ.

3. $\mathcal{P}$ computes

$$p := \texttt{NextPrime}(n\lambda)\ \ ,\ \ \widetilde{g} := g^{\sum_{i=1}^{n} d_i^p\gamma^i}\ \in\ \mathbb{G}$$

and sends it to the Verifier $\mathcal{V}$.

4. $\mathcal{P}$ generates a non-interactive proof for $\texttt{AggKE-1}[a,\ (a_1,\cdots,a_n)]$ and sends it to $\mathcal{V}$.

5. The hashing algorithm $\mathtt{H}_{\mathtt{FS},\lambda}$ generates a λ-bit prime $\ell \not\equiv 1 \pmod{p}$.

6. $\mathcal{P}$ computes $R := D \pmod{\ell}$, $\breve{a} := a^{(D-R)/\ell}$ and sends $(\breve{a}, R) \in \mathbb{G} \times [\ell]$ to $\mathcal{V}$.

7. $\mathcal{P}$ computes the integers $\widehat{r}_i := \widehat{d}_i \pmod{\ell}$ $(i = 1,\cdots,n)$ and sends $(\widehat{r}_1,\cdots,\widehat{r}_n)\ \in\ [\ell]^n$ to $\mathcal{V}$.

8. $\mathcal{P}$ computes the integers $r_i := d_i \pmod{\ell}$ $(i = 1,\cdots,n)$ and the integers $\widetilde{q}$, $\widetilde{r}$ such that

$$\sum_{i=1}^{n} d_i^p\gamma^i = \widetilde{q}\cdot\ell + \widetilde{r}\ ,\quad \widetilde{r} \in [\ell]$$

and sends $\breve{g} := g^{\widetilde{q}}\ \in\ \mathbb{G}$ to $\mathcal{V}$.

9. The hashing algorithm $\mathtt{H}_{\mathtt{FS},\lambda}$ generates a λ-bit prime γ_0.

10. $\mathcal{P}$ computes the integers $\widehat{q}_0$, $\widehat{r}_0$ such that

$$\sum_{i=1}^{n} \widehat{d}_i\gamma^i = \widehat{q}_0\cdot\ell + \widehat{r}_0\ \ ,\ \ \widehat{r}_0 \in [\ell]$$

and sends $\widehat{Q}_0 := a^{\widehat{q}_0} \in \mathbb{G}$ to $\mathcal{V}$.

11. $\mathcal{V}$ verifies that $(\widehat{r}_1,\cdots,\widehat{r}_n, R) \in [\ell]^{n+1}$ and computes

$$r_i \equiv \widehat{r}_i^{-1} R \pmod{\ell} \ \ (i = 1, \cdots, n) \ \ , \ \ \widetilde{r} := \sum_{i=1}^{n} r_i^p \gamma^i \pmod{\ell} \ , \ \widehat{r}_0 := \sum_{i=1}^{n} \widehat{r}_i \gamma_0^i \pmod{\ell}.$$

12. $\mathcal{V}$ verifies the equations

$$(\breve{a})^\ell a^R \stackrel{?}{=} A \ \bigwedge \ (\widehat{Q}_0)^\ell a^{\widehat{r}_0} \stackrel{?}{=} \prod_{i=1}^{n} a_i^{\gamma_0^i} \ \bigwedge \ (\breve{g})^\ell g^{\widetilde{r}} \stackrel{?}{=} \widetilde{g}.$$

He accepts the validity of the claim if and only if all three equations hold and the proofs for $\texttt{PoKE}[a, \ A]$, $\texttt{AggKE-1}[a, \ (a_1, \cdots, a_n)]$ are valid. □

Thus, the proof consists of a constant number of $\mathbb{G}$-elements and $2n + \mathbf{O}(1)$ λ-bit integers. We note that the additional challenge γ_0 is necessary for the security of this protocol. The Prover commits the integer $\sum_{i=1}^{n} d_i^p \gamma^i$ by computing $\widetilde{g} := g^{\sum_{i=1}^{n} d_i^p \gamma^i}$ and sending it to the Verifier *before* the challenge ℓ is generated by the Fiat-Shamir heuristic. However, a malicious Prover $\mathcal{P}_{\text{mal}}$ could forge a fake proof as follows:

1. $\mathcal{P}_{\text{mal}}$ chooses integers $e_1, \cdots, e_n$ and sends $g^{\sum_{i=1}^{n} e_i \gamma^i}$ to the Verifier instead of $g^{\sum_{i=1}^{n} d_i^p \gamma^i}$
2. $\mathcal{P}_{\text{mal}}$ chooses integers $r_1, \cdots, r_n \in [\ell]$ such that

$$\sum_{i=1}^{n} (D d_i^{-1}) \gamma^i \equiv \sum_{i=1}^{n} (D r_i^{-1}) \gamma^i \pmod{\ell} \ \ , \ \ \sum_{i=1}^{n} e_i \gamma^i \equiv \sum_{i=1}^{n} r_i \gamma^i \pmod{\ell},$$

but $d_i \not\equiv r_i \pmod{\ell}$ for some or all indices i. The Prover $\mathcal{P}_{\text{mal}}$ can do so with non-negligible probability.

3. Thus, the Verifier is tricked into believing that $\sum_{i=1}^{n} d_i^p \gamma^i$ is an integer, which might not necessarily be the case. In fact, even if the Fiat-Shamir heuristic outputs the additional challenge γ_0 before the remainders $(r_1, \cdots, r_n, R)$ are committed, $\mathcal{P}_{\text{mal}}$ can forge a fake proof with non-negligible probability.

To address this, γ_0 is randomly generated by the Fiat-Shamir heuristic *after* the Prover sends the tuple $(\widehat{r}_1, \cdots, \widehat{r}_n, R) \in [\ell]^{n+1}$. Hence, we have

$$\texttt{Pr}\Big[\sum_{i=1}^{n} d_i \gamma_0^i \equiv \sum_{i=1}^{n} r_i \gamma_0^i \pmod{\ell} \ \Big| \ d_i \not\equiv r_i \pmod{\ell} \text{ for some } i \Big] = \texttt{negl}(\lambda).$$

Hence, the elements $(r_1, \cdots, r_n)$ must be honestly computed in order to succeed at the additional challenge of computing the element $\widehat{Q}_0$ such that

$$\widehat{Q}_0^\ell a^{\widehat{r}_0} = \prod_{i=1}^{n} a_i^{\gamma_0^i}$$

with non-negligible probability.

The most expensive part of the proof generation is computing the element $\widetilde{g}$. The effective runtime of this part could be reduced if the Prover pre-computes the set

$$\{g^{2^i} : 1 \leq i \leq N\}$$

for an appropriately large integer N. The Verifier's work can be reduced by the Prover sending a succinct proof of (multi-)exponentiation for the computation $\prod_{i=1}^{n} a_i^{\gamma_0^i}$, instead of the Verifier

independently performing this computation.

Theorem 3.12. *The protocol* PoAggKE-2 *is an argument of knowledge for the relation* $\mathcal{R}_{\texttt{AggKE}-2}$ *in the generic group model.*

Proof. The low order assumption implies that except with negligible probability, any PPT algorithm can output at most a unique tuple $(d_1, \cdots, d_n) \in \mathbb{Q}^n$ such that $a_i^{d_i} = A$. We first show that the protocol is sound, i.e. in case of an accepting transcript, the Prover possesses integers d_i such that $a_i^{d_i} = A \ \forall \ i$ except with negligible probability. We then show that an extractor $\mathcal{E}$ can efficiently extract the integers d_i from sufficiently many accepting transcripts using the Chinese remainder theorem, in expected polynomial runtime.

The subprotocol $\texttt{PoAggKE-1}[a, (a_1, \cdots, a_n)]$ demonstrates that with overwhelming probability, the Prover can generate integers $\widehat{d}_1, \cdots, \widehat{d}_n$ bounded by $2^{\texttt{poly}(\lambda)}$ such that

$$a_i = a^{\widehat{d}_i} \ (i = 1. \cdots, n).$$

The low order assumption implies that the tuple $(\widehat{d}_1, \cdots, \widehat{d}_n) \in \mathbb{Q}^n$ is unique except with negligible probability. Furthermore, since the equation

$$\widehat{Q}_0^{\ell} a^{\widehat{r}_0} = \prod_{i=1}^{n} a_i^{\gamma_0^i} = a^{\sum_{i=1}^{n} \widehat{d}_i \gamma_0^i} \ \in \ \mathbb{G}$$

holds, it follows that either

$$\sum_{i=1}^{n} \widehat{r}_i \gamma_0^i \equiv \sum_{i=1}^{n} \widehat{d}_i \gamma_0^i \ (\text{mod } \ell)$$

or the Prover can efficiently extract an ℓ-th root of $a \in \mathbb{G}$. Since the λ-bit prime is randomly generated after $a \in \mathbb{G}$ has been sent by the Prover, the latter event would violate the adaptive root assumption. So, with overwhelming probability, $\sum_{i=1}^{n} \widehat{r}_i \gamma_0^i \equiv \sum_{i=1}^{n} \widehat{d}_i \gamma_0^i \ (\text{mod } \ell)$. Now, since the λ-bit challenge γ_0 is randomly generated after the Prover sends the tuple $(\widehat{r}_1, \cdots, \widehat{r}_n, R) \ \in \ [\ell]^{n+1}$, the Schwartz-Zippel lemma implies that with overwhelming probability, $\widehat{r}_i \equiv \widehat{d}_i \ (\text{mod } \ell) \ \forall \ i$, which completes the proof of soundness.

The equation $(\breve{a})^{\ell} a^R = A$ and lemma 2.4 imply that with overwhelming probability, the Prover possesses a rational $D \equiv R \ (\text{mod } \ell)$ such that $a^D = A$. Thus, with overwhelming probability, the rationals $D\widehat{d}_1^{\ -1}, \cdots, D\widehat{d}_n^{-1}$ satisfy

$$D\widehat{d}_i^{\ -1} \equiv R\widehat{r}_i^{-1} \ (\text{mod } \ell) \text{ for every } i.$$

Now, $a_i^D = a^{D\widehat{d}_i} = A^{\widehat{d}_i}$ for every i. In the protocol, the Verifier independently computes the λ-bit integers

$$r_i := R\widehat{r}_i^{-1} \equiv D\widehat{d}_i^{-1} \ (\text{mod } \ell) \quad (i = 1, \cdots, n) \ , \ \widetilde{r} := \sum_{i=1}^{n} r_i^p \gamma^i \equiv \sum_{i=1}^{n} (D\widehat{d}_i^{-1})^p \gamma^i \ (\text{mod } \ell).$$

Hence, the equation $(\breve{\widetilde{g}})^{\ell} g^{\widetilde{r}} = \widetilde{g}$ and lemma 2.5 imply that with overwhelming probability, the Prover possesses a rational $\widetilde{d}$ such that $\widetilde{g} = g^{\widetilde{d}}$. If $\widetilde{d} \not\equiv \widetilde{r} \ (\text{mod } \ell)$, the Prover could efficiently extract an ℓ-th root of $a \in \mathbb{G}$, thus violating the adaptive root assumption. Hence, with overwhelming probability,

$$\widetilde{d} \equiv \widetilde{r} \equiv \sum_{i=1}^{n} r_i^p \gamma^i \equiv \sum_{i=1}^{n} \left(D\widehat{d}_i^{\ -1}\right)^p \gamma^i \ (\text{mod } \ell).$$

Since the λ-bit prime ℓ is randomly generated after the element $\widetilde{g} \in \mathbb{G}$ has been sent by the Prover, it follows that with overwhelming probability,

$$\widetilde{d} = \sum_{i=1}^{n} \left(D\widehat{d}_i^{-1}\right)^p \gamma^i.$$

Now, if $\widetilde{d}$ were not an integer, the tuple $(g, \widetilde{g}, \widetilde{d})$ would violate the fractional root assumption. Thus, with overwhelming probability, $\widetilde{d}$ is an integer and since $p > n\lambda$, lemma 2.2 then implies that with overwhelming probability, the rationals $D\widehat{d}_i^{-1}$ are integers.

Now, an extractor $\mathcal{E}$ can simulate the extractors for $\texttt{PoAggKE-1}[a,\ (a_1, \cdots, a_n)]$ and $\texttt{PoKE}[a,\ A]$ to extract integers $\widehat{e}_1, \cdots, \widehat{e}_n,\ E$ such that $a^{\widehat{e}_i} = a_i\ (i = 1, \cdots, n)$ and $a^E = A$ in expected polynomial runtime. Setting $e_i := E \cdot \widehat{e}_i^{-1}$ yields $a_i^{e_i} = A$. Since we showed that the protocol is sound, the low order assumption implies that with overwhelming probability, the e_i are integers. □

4 Protocols for arguments of disjointness

The goal of this section is to provide protocols for demonstrating disjointness of multiple data sets/multisets. The proofs can be publicly verified against the succinct commitments to these multisets. To that end, we first describe a protocol whereby an honest Prover can show that the GCD of two discrete logarithms equals a third discrete logarithm while keeping the communication complexity constant. One obvious application is proving disjointness of sets/multisets in accumulators instantiated with hidden order groups. We formulate an argument of knowledge for the relation

$$\mathcal{R}_{\texttt{GCD}}[(a_1, b_1),\ (a_2, b_2),\ (a_3, b_3)] = \{((a_i, b_i \in \mathbb{G});\ d_i \in \mathbb{Z})\ :\ b_i = a_i^{d_i},\ \mathbf{gcd}(d_1, d_2) = d_3\}.$$

We construct a protocol that has communication complexity independent of the elements a_i, b_i. The protocol rests on the basic fact that

$$d_3 = \mathbf{gcd}(d_1, d_2) \iff (d_1 \equiv d_2 \equiv 0 \pmod{d_3}) \bigwedge \left(\exists\, (x_1, x_2) \in \mathbb{Z}^2 :\ d_3 = x_1 d_1 + x_2 d_2\right).$$

Protocol 4.1. *Proof of the greatest common divisor* (PoGCD):
Parameters: $\mathbb{G} \xleftarrow{\$} \text{GGen}(\lambda),\ g \in \mathbb{G}$.
Input: Elements $a_1, a_2, a_3, b_1, b_2, b_3 \in \mathbb{G}$.
Claim: The Prover possesses integers $d_1,\ d_2,\ d_3$ such that:
- $a_1^{d_1} = b_1,\ a_2^{d_2} = b_2,\ a_3^{d_3} = b_3$
- $\mathbf{gcd}(d_1, d_2) = d_3$

1. The Prover $\mathcal{P}$ computes $b_{1,2} := a_1^{d_2}\ ,\ b_{1,3} := a_1^{d_3}\ \in\ \mathbb{G}$ and sends them to the Verifier $\mathcal{V}$. 2. $\mathcal{P}$ generates non-interactive proofs for $\texttt{EqDLog}[(a_2, b_2),\ (a_1, b_{1,2})]$, $\texttt{EqDLog}[(a_3, b_3),\ (a_1, b_{1,3})]$ and sends them to $\mathcal{V}$.
3. $\mathcal{P}$ generates non-interactive proofs for $\texttt{PoKE}[b_{1,3},\ b_1]$ and $\texttt{PoKE}[b_{1,3},\ b_{1,2}]$ and sends them to $\mathcal{V}$.
4. $\mathcal{P}$ uses the Eulidean algorithm to compute integers e_1, e_2 such that

$$e_1 d_1 + e_2 d_2 = d_3\ \ ,\ \ |e_1| < |d_2|\ ,\ |e_2| < |d_1|.$$

5. $\mathcal{P}$ computes

$$\widetilde{b}_1 := b_1^{e_1}\ ,\ \widetilde{b}_{1,2} := b_{1,2}^{e_2}\ \in\ \mathbb{G}$$

and sends them to $\mathcal{V}$ along with non-interactive proofs for $\texttt{PoKE}[b_1,\ \widetilde{b}_1]$ and $\texttt{PoKE}[b_{1,2},\ \widetilde{b}_{1,2}]$.

6. $\mathcal{V}$ verifies all of the proofs he receives in addition to the equation $\widetilde{b}_1 \cdot \widetilde{b}_{1,2} \stackrel{?}{=} b_{1,3}$. He accepts the validity of the claim if and only if all of these proofs are valid. □

Theorem 4.2. *The Protocol* PoGCD *is an argument of knowledge for the relation* $\mathcal{R}_{\texttt{GCD}}$ *in the generic group model.*

Proof. Since we showed that PoEqDLog is an argument of knowledge for the relation $\mathcal{R}_{\texttt{EqDLog}}$, we may assume without loss of generality that - with notations as in the protocol PoGCD -

$$a_1 = a_2 = a_3 \ , \ b_{1,2} = b_2 \ , \ b_{1,3} = b_3 \ , \ \widetilde{b}_2 = \widetilde{b}_{1,2}.$$

An extractor $\mathcal{E}$ can simulate the extractors for the subprotocols $\texttt{PoKE}[a_1, \ b_i]$ $(i = 1,2,3)$, $\texttt{PoKE}[b_i, \ \widetilde{b}_i]$ $(i = 1,2)$ to extract integers $d_1, d_2, d_3, \ e_1, e_2$ such that

$$a_1^{d_i} = b_i \in \mathbb{G} \ \ (i = 1,2,3) \ \ , \ \ b_i^{e_i} = \widetilde{b}_i \in \mathbb{G} \ \ (i = 1,2).$$

Since the PoKE extractors runs in expected polynomial runtime, the same holds for $\mathcal{E}$. Now, the equation $\widetilde{b}_1 \cdot \widetilde{b}_2 = b_3 \in \mathbb{G}$ and the low order assumption imply that with overwhelming probability, $d_3 = d_1 e_1 + d_2 e_2$ and hence, d_3 is divisible $\mathbf{gcd}(d_1, d_2)$. On the other hand, the subprotocols $\texttt{PoKE}[b_3, \ b_1]$, $\texttt{PoKE}[b_3, \ b_2]$ and the low order assumption imply that with overwhelming probability, d_3 divides both d_1 and d_2 and hence, divides $\mathbf{gcd}(d_1, d_2)$. This foces the equality $\mathbf{gcd}(d_1, d_2) = d_3$. □

An important special case is where $\mathbf{gcd}(d_1, d_2) = 1$. In this case, Step 3 is redundant and hence, the proof size is smaller. We call this special case the Protocol for *Relatively Prime Discrete Logarithms* or RelPrimeDLog for short:

$$\mathcal{R}_{\texttt{RelPrimeDLog}}[(a_1, b_1), \ (a_2, b_2)] = \big\{((a_i, b_i \in \mathbb{G}); \ d_i \in \mathbb{Z}) \ : \ b_i = a_i^{d_i}, \ \mathbf{gcd}(d_1, d_2) = 1\big\}.$$

Protocol 4.3. *Proof of Relatively Prime Discrete Logarithms* (PoRelPrimeDLog):

Parameters: $\mathbb{G} \stackrel{\$}{\leftarrow} \mathrm{GGen}(\lambda), \ g \in \mathbb{G}$.
Input: Elements $a_1, a_2, b_1, b_2 \in \mathbb{G}$.
Claim: The Prover possesses integers d_1, d_2 such that:
- $a_1^{d_1} = b_1$, $a_2^{d_2} = b_2$
- $\mathbf{gcd}(d_1, d_2) = 1$

1. The Prover $\mathcal{P}$ computes $b_{1,2} := a_1^{d_2}$ and sends it to the Verifier $\mathcal{V}$ along with a non-interactive proof for $\texttt{EqDLog}[(a_2, b_2), \ (a_1, b_{1,2})]$.
2. $\mathcal{P}$ uses the Euclidean algorithm to compute integers e_1, e_2 such that $e_1 d_1 + e_2 d_2 = 1$.
3. $\mathcal{P}$ computes

$$\widetilde{b}_1 := b_1^{e_1} \ \ , \ \ \widetilde{b}_{1,2} := b_{1,2}^{e_2} \ \in \ \mathbb{G}$$

and sends them to $\mathcal{V}$ along with non-interactive proofs for $\texttt{PoKE}[b_1, \ \widetilde{b}_1]$ and $\texttt{PoKE}[b_{1,2}, \ \widetilde{b}_{1,2}]$.
4. $\mathcal{V}$ verifies the equation $\widetilde{b}_1 \cdot \widetilde{b}_{1,2} \stackrel{?}{=} a_1 \in \mathbb{G}$ and the proofs for $\texttt{EqDLog}[(a_2, b_2), \ (a_1, b_{1,2})]$, $\texttt{PoKE}[b_1, \ \widetilde{b}_1]$ and $\texttt{PoKE}[b_{1,2}, \ \widetilde{b}_{1,2}]$. He accepts the validity of the claim if and only if all of these proofs are valid. □

Proposition 4.4. *The Protocol* PoRelPrimeDLog *is an argument of knowledge for the relation* $\mathcal{R}_{\texttt{RelPrimeDLog}}$ *in the generic group model.*

Proof. This is a special case of Proposition 4.2. □

It is easy to see that the protocol PoGCD may be combined with the protcol PoMultPolyDLog to provide an argument of knowledge for the relation

$$\mathcal{R}_{\texttt{LCM}}[(a_1,b_1),\ (a_2,b_2),\ (a_3,b_3)] = \{((a_i,b_i\in\mathbb{G});\ d_i\in\mathbb{Z})\ :\ b_i = a_i^{d_i},\ \mathbf{lcm}(d_1,d_2)=d_3\}.$$

This argument of knowledge can demonstrate that for data sets/multisets $\mathcal{D}_1,\mathcal{D}_2,\mathcal{D}_3$, we have

$$\mathcal{D}_3 = \mathcal{D}_1\cup\mathcal{D}_2$$

by setting

$$d_i = \prod_{d\in\mathcal{D}_i} x\ \ (i=1,2,3).$$

4.1 Protocols for aggregated arguments of disjointness

We now use the protocols PoAggKE-1 and PoAggKE-2 and a few more techniques to generalize the protocol PoRelPrimeDLog to multiple discrete logarithms. Consider a setting where we have n accumulators $\mathbf{Acc}_1,\cdots,\mathbf{Acc}_n$ instantiated in the same group $\mathbb{G}$ and with the common genesis state $g\in\mathbb{G}$. Let $\mathcal{D}_i$ denote the data inserted into $\mathbf{Acc}_i$ and let A_i denote the accumulated digest of $\mathbf{Acc}_i$. Thus,

$$A_i = \texttt{Com}(g,\mathcal{D}_i) = g^{\Pi(\mathcal{D}_i)}.$$

Suppose a Prover needs to demonstrate to a Verifier (with access to the accumulated digests) that the data sets/multisets $\mathcal{D}_i$ are pairwise disjoint, while keeping the communication complexity to a bare minimum. In particular, the Verifier should not need to access the data sets/multisets $\mathcal{D}_i$ which might be too large for the storage capacity of the verifying node. A straightforward way would be to provide the $\binom{n}{2}$ proofs of pairwise disjointness using the protocol PoRelPrimeDLog. But this would entail $\mathbf{O}(n^2)$ group elements and $\mathbf{O}(n^2)$ λ-bit integers, which we would like to avoid. Instead, we provide a protocol whereby the Prover can demonstrate the pairwise disjointness with a constant number of $\mathbb{G}$-elements and $2n+\mathbf{O}(1)$ λ-bit integers.

We call the next protocol the *Aggregated Knowledge of Relatively Prime Exponents* 1 or AggRelPrimeDLog-1 for short. We provide an argument of knowledge for the relation:

$$\mathcal{R}_{\texttt{AggRelPrimeDLog-1}}[a,\ \mathbf{a}] = \left\{ \begin{array}{l} (a\in\mathbb{G},\ \mathbf{a}:=(a_1,\cdots,a_n)\in\mathbb{G}^n); \\ (d_1,\cdots,d_n)\in\mathbb{Z}^n): \\ a_i = a^{d_i}\ \forall\ i\ \ ,\ \ \mathbf{gcd}(d_i,d_j)=1)\ \forall\ i\neq j \end{array} \right\}$$

The protocol rests on the following elementary lemma.

Lemma 4.5. *Let $d_1,\cdots,d_n$ be non-zero integers. Set*

$$D := \prod_{i=1}^{n} d_i\ ,\ \widehat{d_i} := D{d_i}^{-1}\ (i=1,\cdots,n)\ ,\ \widehat{D} := \sum_{i=1}^{n}\widehat{d_i}.$$

Then

$$\mathbf{gcd}(d_i,d_j)=1\ \forall\ i\neq j\ \iff\ \mathbf{gcd}(D,\widehat{D})=1.$$

Proof. First, suppose there exists a pair i,j such that $\mathbf{gcd}(d_i,d_j)>1$. Then $\mathbf{gcd}(d_i,d_j)$ divides $\widehat{d_k}$ for every index k and in particular, $\mathbf{gcd}(d_i,d_j)$ divides $\widehat{D}$. Hence, $\mathbf{gcd}(D,\widehat{D})$ is divisible by $\mathbf{gcd}(d_i,d_j)$.

Conversely, suppose $\mathbf{gcd}(d_i,d_j)=1\ \forall\ i\neq j$. Then for every index i, $\widehat{D}\equiv\widehat{d_i}\pmod{d_i}$ and hence, $\mathbf{gcd}(\widehat{D},d_i)=\mathbf{gcd}(\widehat{d_i},d_i)=1$. Thus, $\mathbf{gcd}(D,\widehat{D})=1$. □

Recall that given integers $d_1, \cdots, d_n$ and elements $a, A \in \mathbb{G}$ such that

$$a^D = a^{\prod_{i=1}^{n} d_i} = A,$$

the `RootFactor` algorithm allows us to compute elements a_i such that $a_i^{d_i} = A$ in runtime $\mathbf{O}\big(\log(D) \cdot \log(\log(D))\big)$, whereas the naïve approach would take runtime $\mathbf{O}(\log^2(D))$. Thus, a Prover can compute the element

$$\widehat{A} := \prod_{i=1}^{n} a_i \ \in \ \mathbb{G}$$

in runtime $\mathbf{O}\big(\log(D) \cdot \log(\log(D))\big)$ with the `RootFactor` algorithm followed by n group multiplications.

Protocol 4.6. *Proof of Aggregated Knowledge of Relatively Prime Discrete Logarithms* 1 (`PoAggRelPrimeDLog-1`) :
Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda), \ g \in \mathbb{G}$.
Input: Element $a \in \mathbb{G}$, $(a_1, \cdots, a_n) \in \mathbb{G}^n$.
Claim: The Prover possesses integers $d_1, \cdots, d_n$ such that:
- $a^{d_i} = a_i$ for $i = 1, \cdots, n$.
- $\mathbf{gcd}(d_i, d_j) = 1$ for every pair $i \neq j$.

1. The Prover $\mathcal{P}$ computes the integers $D := \prod_{i=1}^{n} d_i \ , \ \widehat{D} := \sum_{i=1}^{n} (D \cdot d_i^{-1})$.

2. $\mathcal{P}$ computes $A := a^D, \ \widehat{A} := a^{\widehat{D}} \ \in \ \mathbb{G}$ (the latter using the `RootFactor` algorithm) and sends $A, \ \widehat{A}$ to the Verifier $\mathcal{V}$.

3. $\mathcal{P}$ generates a non-interactive proof for `MultPolyDLog`$[a, \ (a_1, \cdots, a_n, A, \widehat{A}), \ (f, \widehat{f})]$ where

$$f(X_1, \cdots, X_{n+2}) := \big(\prod_{i=1}^{n} X_i\big) - X_{n+1} \ , \ \widehat{f}(X_1, \cdots, X_{n+2}) := \big(\sum_{i=1}^{n} \prod_{\substack{1 \leq j \leq n \\ j \neq i}} X_j\big) - X_{n+2}$$

and sends the proof to $\mathcal{V}$.

4. $\mathcal{P}$ generates a non-interactive proof for `RelPrimeDLog`$[(a, A), \ (a, \widehat{A})]$ and sends it to $\mathcal{V}$.

5. $\mathcal{V}$ verifies the three proofs and accepts the validity of the claim if and only if all proofs are valid. □

Recall that the protocol `PoMultPolyDLog`$[a, \ (a_1, \cdots, a_n, A, \widehat{A}), \ (f, \widehat{f})]$ contains `PoAggKE-1`$[a, \ (a_1, \cdots, a_n)]$ as a subprotocol. So the protocol `PoAggRelPrimeDLog-1` demonstrates that there exist n integers d_i such that $a^{d_i} = a_i$ and the product $\prod_{i=1}^{n} d_i$ is relative prime with the integer given by the $(n-1)$-th elementary symmetric function

$$\sum_{i=1}^{n} \prod_{\substack{1 \leq j \leq n \\ j \neq i}} d_j.$$

By lemma 4.5, this is equivalent to the integers d_i being pairwise co-prime. Thus, the proof consists of a constant number of $\mathbb{G}$-elements and $2n + \mathbf{O}(1)$ λ-bit integers.

In the special case where a is an element randomly generated by the oracle, the fractional root assumption implies that it is infeasible to compute any roots of a. Hence, the subprotocol of `PoAggKE-1` or `PoMultPolyDLog` where the Prover computes the element

$$\widetilde{g} := g^{\sum_{i=1}^{n} d_i^p \gamma^i} \in \mathbb{G}$$

and sends it to the Verifier is redundant. So the proof would be a bit smaller and the Prover's computational burden would be substantially lower in this special case.

Theorem 4.7. *The protocol* `PoAggRelPrimeDLog−1` *is an argument of knowledge for the relation* $\mathcal{R}_{\texttt{AggRelPrimeDLog}-1}$ *in the generic group model.*

Proof. An extractor $\mathcal{E}$ can, with overwhelming probability, simulate the extractor for the subprotocol `PoMultPolyDLog` to extract integers D, $\widehat{D}$, $d_1, \cdots, d_n$ such that

$$D = \prod_{i=1}^{n} d_i \ , \ \widehat{D} = \sum_{i=1}^{n} \frac{D}{d_i} \ , \ a^{d_i} = a_i \ \forall \ i \ , \ a^D = A \ , \ a^{\widehat{D}} = \widehat{A}$$

in expected polynomial runtime. Furthermore, the subprotocol `PoRelPrimeDLog`$[(a, A),\ (a, \widehat{A})]$ implies that with overwhelming probability, $\mathbf{gcd}(D, \widehat{D}) = 1$. Hence, by lemma 4.5, the integers d_i are pairwise co-prime. □

Given elements $a_1, a_2 \in \mathbb{G}$, $(b_1, \cdots, b_m) \in \mathbb{G}^m$, $(c_1, \cdots, c_m) \in \mathbb{G}^n$ and equations

$$a_1^{d_1} = b_1, \cdots, a_1^{d_m} = b_m \ \ , \ \ a_2^{e_1} = c_1, \cdots, a_2^{e_n} = c_n,$$

a Prover may provide a proof that he possesses the integers $d_1, \cdots, d_m,\ e_1, \cdots, e_n$ and that every pair d_i, e_j is relatively prime. Our primary use case is demonstrating the disjointness of two families of accumulators. Clearly, the latter part is equivalent to the the integers $d := \prod_{i=1}^{m} d_i$, $e := \prod_{j=1}^{n} e_j$ being relatively prime. Our approach is to first compute the elements $B = a_1^d,\ C := a_2^e$. We then use the protocol `RelPrimeDLog` to show that $\mathbf{gcd}(d, e) = 1$.

We call the next protocol the *Aggregated Knowledge of Relatively Prime Exponents* 2 or `AggRelPrimeDLog-2` for short. We provide an argument of knowledge for the following relation:

$$\mathcal{R}_{\texttt{AggRelPrimeDLog-2}}[a_1, a_2,\ \mathbf{b}, \mathbf{c}] = \left\{ \begin{array}{l} ((a_1, a_2) \in \mathbb{G}^2, \\ \mathbf{b} := (b_1, \cdots, b_m) \in \mathbb{G}^m \ , \ \mathbf{c} := (c_1, \cdots, c_n) \in \mathbb{G}^n); \\ ((d_1, \cdots, d_m) \in \mathbb{Z}^m,\ (e_1, \cdots, e_n) \in \mathbb{Z}^n) \ : \\ (b_i = a_1^{d_i} \ \bigwedge \ c_j = a_2^{e_j} \ \bigwedge \ \mathbf{gcd}(d_i, e_j) = 1) \ \forall \ i, j \end{array} \right\}$$

Protocol 4.8. *Proof of Aggregated Knowledge of Relatively Prime Discrete Logarithms* 2 (`PoAggRelPrimeDLog-2`) :

Parameters: $\mathbb{G} \xleftarrow{\$} \text{GGen}(\lambda),\ g \in \mathbb{G}$.
Input: Elements $a_1, a_2 \in \mathbb{G}$; Elements $\mathbf{b} = (b_1, \cdots, b_m) \in \mathbb{G}^m,\ \mathbf{c} = (c_1, \cdots, c_n) \in \mathbb{G}^n$.
Claim: The Prover possesses integers $d_1, \cdots, d_m,\ e_1, \cdots, e_n$ such that:
- $a_1^{d_i} = b_i$ for $i = 1, \cdots, m$.
- $a_2^{e_j} = c_j$ for $j = 1, \cdots, n$.
- $\mathbf{gcd}(d_i, e_j) = 1$ for every pair i, j.

1. The Prover $\mathcal{P}$ computes the integers $d := \prod_{i=1}^{m} d_m \ \ , \ \ e := \prod_{j=1}^{n} e_n$.
2. $\mathcal{P}$ computes $B := a_1^d,\ \ C := a_2^e \ \in \mathbb{G}$ and sends B, C to the Verifier $\mathcal{V}$.
3. $\mathcal{P}$ generates a non-interactive proof for `MultPolyDLog`$[a_1,\ (b_1, \cdots, b_m, B),\ f_1]$ where

$$f_1(X_1, \cdots, X_{m+1}) := \big(\prod_{i=1}^{m} X_i\big) - X_{m+1}$$

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

4. $\mathcal{P}$ generates a non-interactive proof for $\texttt{MultPolyDLog}[a_2,\ (c_1,\cdots,c_n,C),\ f_2]$ where

$$f_2(X_1,\cdots,X_{n+1}) := \big(\prod_{j=1}^{n} X_j\big) - X_{n+1}$$

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

5. $\mathcal{P}$ generates a non-interactive proof for $\texttt{RelPrimeDLog}[(a_1,B),\ (a_2,C)]$ and sends it to $\mathcal{V}$.

6. $\mathcal{V}$ accepts the validity of the claim if and only if all three proofs are valid. □

Thus, the proof consists of a constant number of $\mathbb{G}$-elements and $2(m+n)+\mathbf{O}(1)$ λ-bit integers. We now prove the security of the protocol.

Proposition 4.9. *The Protocol* `AggRelPrimeDLog-2` *is an argument of knowledge for the relation* $\mathcal{R}_{\texttt{AggRelPrimeDLog-2}}$ *in the generic group model.*

Proof. An extractor $\mathcal{E}$ can simulate the extractors for the subprotocols

$$\texttt{PoMultPolyDLog}[a_1,\ (b_1,\cdots,b_m,B),\ f_1]\ \bigwedge\ \texttt{PoMultPolyDLog}[a_2,\ (c_1,\cdots,c_n,C),\ f_2]$$

to extract integers $d_1,\cdots,d_m,\ e_1,\cdots,e_n$ such that

$$a_1^{d_i} = b_i\ ,\ a_2^{e_j} = c_j\ ,\ a_1^{\prod_{i=1}^{m} d_i} = B\ ,\ a_2^{\prod_{j=1}^{n} e_j} = C\ ,\ \ 1\le i\le m\ ,\ 1\le j\le n.$$

Furthermore, the subprotocol $\texttt{PoRelPrimeDLog}[(a_1,B),\ (a_2,C)]$ and the low order assumption imply that with overwhelming probability, $\mathbf{gcd}(\prod_{i=1}^{m} d_i\ ,\ \prod_{j=1}^{n} e_j) = 1$. This, in turn, implies that $\mathbf{gcd}(d_i,e_j)=1$ for every pair i,j. □

4.2 Protocols for disjointness of sets/multisets in a single accumulator

We now discuss a dual to the protocol `PoAggRelPrimeDLog-1`. Consider a setting where we have data sets/multisets $\mathcal{D}_1,\cdots,\mathcal{D}_n$ inserted into an accumulator. Let A denote the accumulated digest, w_i the witness for $\mathcal{D}_i$ and $d_i := \Pi(\mathcal{D}_i)$. Suppose a Prover needs to demonstrate that the multisets $\mathcal{D}_i$ are pairwise disjoint to a Verifier who has access to the witnesses $w_1,\cdots,w_n$ but not the data multisets. A straightforward approach would be to provide a proof for the relation $\texttt{RelPrimeDLog}[(w_i,A),\ (w_j,A)]$ for each pair i,j. But such a proof would entail $\mathbf{O}(n^2)$ $\mathbb{G}$-elements and $\mathbf{O}(n^2)$ λ-bit integers, which is impractical for larger values of n.

Instead, we provide a protocol whereby the proof consists of a constant number of $\mathbb{G}$-elements and n λ-bit integers. The protocol rests on two simple observations. First, note that for integers $d_1,\cdots,d_n$,

$$\mathbf{gcd}(d_i,d_j)=1\ \forall\ i\neq j\ \iff\ \prod_{i=1}^{n} d_i = \mathbf{lcm}(d_1,\cdots,d_n),$$

as can be easily proved by induction. Secondly, if an element $w\in\mathbb{G}$ can be expressed in the form

$$w=\prod_{i=1}^{n} w_i^{x_i},\ \ (x_1,\cdots,x_n)\in\mathbb{Z}^n,$$

then

$$w^{\mathbf{lcm}(d_1,\cdots,d_n)} = A^k \quad \text{where } k := \sum_{i=1}^{n} x_i \frac{\mathbf{lcm}(d_1,\cdots,d_n)}{d_i}.$$

Furthermore, the Prover can efficiently compute the integers

$$d := \prod_{i=1}^{n} d_i = \mathbf{lcm}(d_1, \cdots, d_n) \ , \ \widehat{d}_i := \prod_{\substack{1 \leq j \leq n \\ j \neq i}} d_j \quad (i = 1, \cdots, n) \ \ , \ \ \widehat{d} := \sum_{i=1}^{n} \widehat{d}_i.$$

Now, d is relatively prime to $\widehat{d}$ by lemma 4.5. Hence, the Prover can efficiently compute integers e, $\widehat{e}$ such that

$$de + \widehat{d}\widehat{e} = 1 \ , \ A^e \cdot (\prod_{i=1}^{n} w_i)^{\widehat{e}} = w.$$

In particular, since $\prod_{i=1}^{n} w_i$ is publicly computable, the Prover can demonstrate - with constant communication complexity - that w is expressible as a product $\prod_{i=1}^{n} w_i^{x_i}$ where the x_i are integers known to him. If the Prover can also demonstrate that

$$w^{\prod_{i=1}^{n} d_i} = A,$$

(with a subprotocol virtually identical to `MultPolyDLog`), then this implies that $\mathbf{lcm}(d_1, \cdots, d_n)$ divides the product $\prod_{i=1}^{n} d_i$, which forces equality between these two integers. In what follows, we provide an argument of knowledge for the relation

$$\mathcal{R}_{\texttt{AggRelPrimeDLog-3}}[(w_1, \cdots, w_n), \ A] = \left\{ \begin{array}{l} (A \in \mathbb{G}, \ (w_1, \cdots, w_n) \ \in \mathbb{G}^n); \\ ((d_1, \cdots, d_n) \in \mathbb{Z}^n) \ : \\ w_i^{d_i} = A \ \forall \ i \ \bigwedge \\ \mathbf{gcd}(d_i, d_j) = 1 \ \forall \ i, j : i \neq j \end{array} \right\}$$

Protocol 4.10. *Proof of Aggregated Knowledge of Relatively Prime Discrete Logarithms* 3 (`PoAggRelPrimeDLog-3`):

Parameters: $\mathbb{G} \stackrel{\$}{\leftarrow} \mathrm{GGen}(\lambda), \ g \in \mathbb{G}$.
Input: Elements $(w_1, \cdots, w_n) \in \mathbb{G}^n \ , \ A \in \mathbb{G}$
Claim: The Prover possesses integers $d_1, \cdots, d_n$ such that:
- $w_i^{d_i} = A$ for $i = 1, \cdots, n$.
- $\mathbf{gcd}(d_i, d_j) = 1$ for every pair $i \neq j$.

1. The Prover $\mathcal{P}$ computes the integers

$$D := \prod_{i=1}^{n} d_i \ \ , \ \ \widehat{d}_i = \prod_{\substack{1 \leq j \leq n \\ j \neq i}} d_j \ (i = 1, \cdots, n) \ \ , \ \ \widehat{D} := \sum_{i=1}^{n} \widehat{d}_i.$$

2. Using Shamir's trick, $\mathcal{P}$ computes an element $w \in \mathbb{G}$ such that $w^D = A$ and sends w to the Verifier $\mathcal{V}$.

3. $\mathcal{P}$ uses the Euclidean algorithm to compute integers e, $\widehat{e}$ such that

$$eD + \widehat{e}\widehat{D} = \mathbf{gcd}(D, \widehat{D}) = 1 \ \ , \ \ |e| < |\widehat{D}|, \ |\widehat{e}| < |D|.$$

4. $\mathcal{P}$ computes the elements

$$\widetilde{A} := A^e \ , \ W := \big(\prod_{i=1}^{n} w_i\big)^{\widehat{e}} \ \in \ \mathbb{G}$$

and sends $\widetilde{A}, W$ to $\mathcal{V}$ along with non-interactive proofs for $\texttt{PoKE}[A,\ \widetilde{A}]$ and $\texttt{PoKE}[(\prod_{i=1}^{n} w_i),\ W]$.

5. The hashing algorithm $\texttt{H}_{\texttt{FS},\lambda}$ generates a λ-bit prime γ.

6. $\mathcal{P}$ computes

$$p := \texttt{NextPrime}(n\lambda)\ \ ,\ \ \widetilde{g} := g^{\sum_{i=1}^{n} d_i^p \gamma^i}\ \in\ \mathbb{G}$$

and sends $\widetilde{g}$ to $\mathcal{V}$.

7. The hashing algorithm $\texttt{H}_{\texttt{FS},\lambda}$ generates a λ-bit prime $\ell \not\equiv 1 \pmod{p}$.

8. $\mathcal{P}$ computes

$$R := D \pmod{\ell}\ ,\ \breve{w} := w^{(D-R)/\ell}$$

and sends $\widehat{w}\ \in\ \mathbb{G}$ to $\mathcal{V}$.

9. $\mathcal{P}$ computes the integers

$$\widehat{r}_i := \widehat{d}_i \pmod{\ell}\ ,\ r_i := d_i \pmod{\ell}$$

and sends the tuple $(r_1, \cdots, r_n)\ \in\ [\ell]^n$ to $\mathcal{V}$.

10. $\mathcal{P}$ computes the integers $\widetilde{q}$, $\widetilde{r}$ such that

$$\sum_{i=1}^{n} d_i^p \gamma^i = \widetilde{q} \cdot \ell + \widetilde{r}\ \ ,\quad \widetilde{r}\ \in\ [\ell]$$

and sends $\breve{g} := g^{\widetilde{q}}\ \in\ \mathbb{G}$ to $\mathcal{V}$.

11. The hashing algorithm $\texttt{H}_{\texttt{FS},\lambda}$ generates a λ-bit prime γ_0.

12. $\mathcal{P}$ computes the integers $\widehat{q}_0$, $\widehat{r}_0$ such that

$$\sum_{i=1}^{n} \widehat{d}_i \gamma^i = \widehat{q}_0 \cdot \ell + \widehat{r}_0\ \ ,\ \ \widehat{r}_0 \in [\ell]$$

and sends $\widehat{Q}_0 := w^{\widehat{q}_0}\ \in\ \mathbb{G}$ to $\mathcal{V}$.

13. $\mathcal{V}$ verifies that $(r_1, \cdots, r_n) \in [\ell]^n$ and computes the λ-bit integers

$$R := \prod_{i=1}^{n} r_i \pmod{\ell}\ \ ,\ \ \widehat{r}_i = R r_i^{-1} \pmod{\ell}\ \ (i = 1, \cdots, n),$$

$$\widetilde{r} := \sum_{i=1}^{n} r_i^p \gamma^i \pmod{\ell}\ \ ,\ \ \widehat{r}_0 := \sum_{i=1}^{n} \widehat{r}_i \gamma_0^i \pmod{\ell}.$$

14. $\mathcal{V}$ computes the elements $\prod_{i=1}^{n} w_i$, $\prod_{i=1}^{n} w_i^{\gamma^i}$, $\prod_{i=1}^{n} w_i^{\gamma_0^i}\ \in\ \mathbb{G}$.

15. $\mathcal{V}$ verifies the equations

$$(\widehat{Q}_0)^\ell w^{\widehat{r}_0} \stackrel{?}{=} \prod_{i=1}^{n} w_i^{\gamma_0^i}\ \bigwedge\ \widetilde{A} \cdot W \stackrel{?}{=} w\ \bigwedge\ (\breve{w})^\ell w^R \stackrel{?}{=} A\ \bigwedge\ (\breve{g})^\ell g^{\widetilde{r}} \stackrel{?}{=} \widetilde{g}$$

and the two PoKEs from Step 4. He accepts if and only if all four equations hold and the two PoKEs are valid. □

The proof consists of $\mathbf{O}(1)$ $\mathbb{G}$-elements and $n + \mathbf{O}(1)$ λ-bit integers. The most expensive part of the proof generation is computing the element $\widetilde{g}$. The effective runtime can be reduced if the Prover pre-computes the set

$$\{g^{2^i}\ :\ 1 \leq i \leq N\}$$

for an appropriately large integer N. If n is large, the Verifier's work can be reduced by the Prover sending succinct proofs of multi-exponentiation for the computations $\prod_{i=1}^{n} w_i^{\gamma^i}$, $\prod_{i=1}^{n} w_i^{\gamma_0^i}$ instead of the Verifier computing them independently.

Theorem 4.11. *The protocol* $\texttt{PoAggRelPrimeDLog}-3$ *is an argument of knowledge for the relation* $\mathcal{R}_{\texttt{AggRelPrimeDLog}-3}$ *in the generic group model.*

Proof. The low order assumption implies that except with negligible probability, any PPT algorithm can generate at most a unique tuple $(d_1, \cdots, d_n) \in \mathbb{Q}^n$ such that $w_i^{d_i} = A$. We first show that the protocol is sound, i.e. in case of an accepting transcript, the Prover can generate integers d_i bounded by $2^{\texttt{poly}(\lambda)}$ such that

$$w_i^{d_i} = A \; \forall \; i \;\; \wedge \;\; \mathbf{gcd}(d_i, d_j) = 1 \;\; \forall \;\; i, j \;\; ,$$

except with negligible probability. We then show that an extractor $\mathcal{E}$ can efficiently extract the integers d_i from sufficiently many accepting transcripts using the Chinese remainder theorem, in expected polynomial runtime.

Since the λ-bit challenge γ_0 is generated after the Prover sends the tuple $(r_1, \cdots, r_n) \in [\ell]^n$, the equation

$$(\widehat{Q}_0)^{\ell} w^{\widehat{r}_0} = \prod_{i=1}^{n} w_i^{\gamma_0^i},$$

and lemma 2.5 imply that with overwhelming probability, the Prover possesses rationals $\widehat{d}_1, \cdots, \widehat{d}_n$ such that $w^{\widehat{d}_i} = w_i \in \mathbb{G}$, $\widehat{d}_i \equiv \widehat{r}_i \pmod{\ell}$ for every i. Furthermore, the equation $(\breve{w})^{\ell} w^R = A$ implies that with overwhelming probability, the Prover possesses a rational D such that $D \equiv R \pmod{\ell}$, $w^D = A$. Now,

$$D \equiv R \equiv \prod_{i=1}^{n} r_i \equiv \prod_{i=1}^{n} d_i \pmod{\ell}$$

and since the λ-bit prime ℓ is randomly generated after the Prover sends w, it follows that with overwhelming probability, $D = \prod_{i=1}^{n} D\widehat{d}_i^{-1}$. Setting $d_i := D\widehat{d}_i^{-1}$ yields $w_i^{d_i} = A$, $D = \prod_{i=1}^{n} d_i$.

The equation $(\breve{g})^{\ell} \widetilde{g}^{\widetilde{r}} = \widetilde{g}$ and lemma 2.4 imply that with overwhelming probability, the Prover possesses a rational $\widetilde{d}$ such that

$$\widetilde{g} = g^{\widetilde{d}} \;\; , \;\; \widetilde{d} \equiv \widetilde{r} \equiv \sum_{i=1}^{n} r_i^p \gamma^i \equiv \sum_{i=1}^{n} d_i^p \gamma^i \pmod{\ell}.$$

Since the λ-bit prime ℓ is randomly generated after $\widetilde{g}$ has been sent by the Prover, it follows that with overwhelming probability,

$$\widetilde{d} = \sum_{i=1}^{n} d_i^p \gamma^i.$$

The fractional root assumption implies that $\widetilde{d}$ is an integer except with negligible probability. So $\sum_{i=1}^{n} d_i^p \gamma^i \in \mathbb{Z}$ and lemma 2.2 then implies that with overwhelming probability, the d_i are all integers.

Furthermore, since the equation $\widetilde{A} \cdot W = w \in \mathbb{G}$ and the proofs for $\texttt{PoKE}[A, \; \widetilde{A}]$ and $\texttt{PoKE}[(\prod_{i=1}^{n} w_i), \; W]$ are valid, it follows that, in particular, w is expressible as a product

$$w = \prod_{i=1}^{n} w_i^{x_i} \quad , \quad (x_1, \cdots, x_n) \in \mathbb{Z}^n \;\; , \;\; |x_i| < 2^{\texttt{poly}(\lambda)}.$$

for some tuple $(x_1, \cdots, x_n)$ known to the Prover. Hence,

$$w^{\mathbf{lcm}(d_1,\cdots,d_n)} = A^k = w^{k\prod_{i=1}^{n} d_i} \in \mathbb{G}$$

for some integer k known to the Prover. Now, the low order assumption implies that with overwhelming probability, $\mathbf{lcm}(d_1, \cdots, d_n)$ is divisible by the product $\prod_{i=1}^{n} d_i$, which forces equality between these two integers. Hence, the integers d_i are pairwise co-prime. This completes the proof of the soundness of the protocol.

Let N be the (polynomially bounded) number of queries from $\mathcal{A}$ to the group oracles $\mathcal{O}_1$, $\mathcal{O}_2$. With overwhelming probability, $2^N \geq \max\{2|d_i| \ : \ 1 \leq i \leq n\}$. An extractor $\mathcal{E}$ can extract the tuple $(d_1, \cdots, d_n) \in \mathbb{Z}^n$ as follows.

$\mathcal{E}$ makes queries to $\mathcal{A}$, keeping the γ fixed and sampling the ℓ randomly and uniformly from the set of λ-bit primes. Let $\ell_1, \cdots, \ell_k$ denote the prime challenges in the first k accepting transcripts and let $\mathbf{r}_j \in [\ell_j]^n$ denote the tuple of remainders modulo ℓ_j in the j-th accepting transcript. $\mathcal{E}$ uses the Chinese remainder theorem to efficiently compute the unique tuple $\mathbf{e} = (e_1, \cdots, e_n) \in \mathbb{Z}^n$ such that

$$\mathbf{e} \equiv \mathbf{r}_j \pmod{\ell_j} \ \forall \ j \ \ , \ \ 2|e_i| < \prod_{j=1}^{k} \ell_j \ \forall \ i.$$

After receiving a new accepting transcript and updating the tuple $\mathbf{e}$, the extractor $\mathcal{E}$ checks whether $a_i^{e_i} = A$ for $i = 1, \cdots, n$. If so, he halts the process. Otherwise, he queries for another accepting transcript and continues to update $\mathbf{e}$ by applying the Chinese remainder theorem to the updated list of accepting transcripts.

When the number of accepting transcripts is $N + 1$, we have

$$\max\{2|d_i| \ : \ 1 \leq i \leq n\} < 2^{N+1} < \prod_{j=1}^{N+1} \ell_j$$

except with negligible probability. Thus, with overwhelming probability, $e_i = d_i \ \forall \ i$ at this point. Thus, $\mathcal{E}$ extracts the d_i in expected polynomial runtime. $\square$

Next, we discuss a dual to the Protocol `AggRelPrimeDLog-2`. Given elements $B, C \in \mathbb{G}$ and subsets

$$\mathbf{b} = \{b_1, \cdots, b_m\} \in \mathbb{G}^m \ \ , \ \ \mathbf{c} = \{c_1, \cdots, c_n\} \in \mathbb{G}^n,$$

an honest Prover may provide a proof that he possesses integers $\{d_1, \cdots, d_m\}$, $\{e_1, \cdots, e_n\}$ such that $b_i^{d_i} = B$, $c_j^{e_j} = C$ and every pair d_i, e_j is relatively prime. We call this relation the *Aggregated Relatively Prime Discrete Logarithms* 4 or `AggRelPrimeDLog-4` for short. We provide an argument of knowledge for the following relation:

$$\mathcal{R}_{\texttt{AggRelPrimeDLog-4}}[\mathbf{b}, \mathbf{c}, B, C] = \left\{ \begin{array}{l} ((B, C) \in \mathbb{G}^2, \\ \mathbf{b} = (b_1, \cdots, b_m) \in \mathbb{G}^m \ , \ \mathbf{c} = (c_1, \cdots, c_n) \in \mathbb{G}^n); \\ ((d_1, \cdots, d_m) \in \mathbb{Z}^m, \ (e_1, \cdots, e_n) \in \mathbb{Z}^n) \ : \\ (B = b_i^{d_i}, \ C = c_j^{e_j} \ \bigwedge \ \mathbf{gcd}(d_i, e_j) = 1) \ \forall \ i, j \end{array} \right\}$$

An example: Consider the case where $B, C \in \mathbb{G}$ are accumulated digests for accumulators $\mathbf{Acc}_1$ and $\mathbf{Acc}_2$ respectively. Let $\mathcal{D}_1, \cdots, \mathcal{D}_m$ and $\mathcal{E}_1, \cdots, \mathcal{E}_m$ be data sets/multisets inserted into the two accumulators. Let w_i, u_j denote the membership witnesses for $\mathcal{D}_i$, $\mathcal{E}_j$ and let d_i, e_j denote the products of elements of $\mathcal{D}_i$, $\mathcal{E}_j$ respectively($1 \leq i \leq m$, $1 \leq j \leq n$). Then

$$w_i^{d_i} = B \ \ , \ \ u_j^{e_j} = C.$$

Suppose a Prover needs to prove the disjointness of the unions

$$\mathcal{D} := \bigcup_{i=1}^{m} \mathcal{D}_i \ , \ \mathcal{E} := \bigcup_{j=1}^{n} \mathcal{E}_j$$

to a Verifier with access to the witnesses $\mathcal{W} := \{w_1, \cdots, w_m\}$, $\mathcal{U} := \{u_1, \cdots, u_n\}$.

A straightforward approach would be to provide $m \cdot n$ distinct proofs that $\mathbf{gcd}(d_i, e_j) = 1$ for every pair d_i, e_j using the protocol `PoRelPrimeDLog`. But such a proof would entail $\mathbf{O}(mn)$ elements of $\mathbb{G}$ in addition to $\mathbf{O}(mn)$ λ-bit integers. Instead, the Prover could simply send a non-interactive proof for the relation `AggRelPrimeDLog-4`$[(\mathcal{W}, B),\ (\mathcal{U}, C)]$. The proof consists of a constant number of $\mathbb{G}$-elements and $2(m+n) + \mathbf{O}(1)$ λ-bit integers. The protocol hinges on the simple observation that the following are equivalent:

- $\mathbf{gcd}(d_i, e_j) = 1$ for every pair i, j.
- $\mathbf{gcd}\big(\mathbf{lcm}(d_1, \cdots, d_m),\ \mathbf{lcm}(e_1, \cdots, e_n)\big) = 1$

Protocol 4.12. *Proof of Aggregated Knowledge of Relatively Prime Discrete Logarithms* 4 (`PoAggRelPrimeDLog-4`)
Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda),\ g \in \mathbb{G}$.
Input: Elements $B, C \in \mathbb{G},\ \mathbf{b} = (b_1, \cdots, b_m) \in \mathbb{G}^m,\ \ \mathbf{c} = (c_1, \cdots, c_n) \in \mathbb{G}^n$
Claim: The Prover possesses integers $d_1, \cdots, d_m,\ e_1, \cdots, e_n$ such that:
- $b_i^{d_i} = B$ for $i = 1, \cdots, m$.
- $c_j^{e_j} = C$ for $j = 1, \cdots, n$.
- $\mathbf{gcd}(d_i, e_j) = 1$ for every pair i, j.

1. Using Shamir's trick, $\mathcal{P}$ computes elements $b, c \in \mathbb{G}$ such that

$$b^{\mathbf{lcm}(d_1, \cdots, d_m)} = B \ , \ c^{\mathbf{lcm}(e_1, \cdots, e_n)} = C$$

and sends b, c to the Verifier $\mathcal{V}$.

2. $\mathcal{P}$ generates non-interactive proofs for `AggKE-2`$[\mathbf{b},\ B]$ and `AggKE-2`$[\mathbf{c},\ C]$ and sends the proofs to $\mathcal{V}$.

3. $\mathcal{P}$ generates non-interactive proofs for `AggKE-1`$[b,\ \mathbf{b}]$ and `AggKE-1`$[c,\ \mathbf{c}]$ and sends the proofs to $\mathcal{V}$.

4. $\mathcal{P}$ generates a non-interactive proof for `RelPrime`$[(b, B),\ (c, C)]$ and sends the proof to $\mathcal{V}$.

5. $\mathcal{V}$ verifies all of these proofs and accepts the validity of the claim if and only if all proofs are valid. □

Thus, the proof for `AggRelPrimeDLog-4` consists of a constant number of $\mathbb{G}$-elements and $2(m+n) + \mathbf{O}(1)$ λ-bit integers.

Proposition 4.13. *The protocol* `AggRelPrimeDLog-4` *is an argument of knowledge for the relation* $\mathcal{R}_{\texttt{AggRelPrimeDLog-4}}$ *in the generic group model.*

Proof. An extractor $\mathcal{E}$ can simulate the extractors for `AggKE-2`$[\mathbf{b},\ B]$ and `AggKE-1`$[b,\ \mathbf{b}]$ to extract integers $d, d_1, \cdots, d_m, \widehat{d}_1, \cdots, \widehat{d}_m$ such that

$$b^d = B \ , \ b^{\widehat{d}_i} = b_i \ , \ b_i^{d_i} = B \ \forall \ i.$$

Similarly, $\mathcal{E}$ can simulate the extractors for `AggKE-2`$[\mathbf{c},\ C]$ and `AggKE-1`$[c,\ \mathbf{c}]$ to extract integers $e, e_1, \cdots, e_n, \widehat{e}_1, \cdots, \widehat{e}_n$ such that

$$c^e = C \ , \ c^{\widehat{e}_j} = c_j \ , \ c_j^{e_j} = C \ \forall \ j.$$

The low order assumption implies that with overwhelming probability,

$$d_i \cdot \widehat{d_i} = d \ \forall \ i \ \ , \ \ e_j \cdot \widehat{e}_j = e \ \forall \ j$$

and in particular, d, e are divisible by $\mathbf{lcm}(d_1, \cdots, d_m)$, $\mathbf{lcm}(e_1, \cdots, e_n)$ respectively. Lastly, the proof for $\texttt{RelPrimeDLog}[(b, B), \ (c, C)]$ implies that with overwhelming probability, $\mathbf{gcd}(d, e) = 1$. This implies that the integers $\mathbf{lcm}(d_1, \cdots, d_m)$, $\mathbf{lcm}(e_1, \cdots, e_n)$ are relatively prime and hence with overwhelming probability, $\mathbf{gcd}(d_i, e_j) = 1$ for each pair i, j. □

5 Applications

In this section, we discuss two potential applications of the techniques developed in this paper: verifiable outsourcing of data and sharded stateless blockchains.

5.1 Verifiably outsourcing storage

The protocols we have developed so far allow us to build a mechanism whereby a client can verifiably outsource data multisets to a server node. The client stores constant-sized commitments to these multisets and can query the server for information regarding these data sets. The server node, in turn, submits this information with proofs that can be publicly verified against the constant-sized commitments stored by the client.

As before, $\mathbb{G}$ is a group of hidden order in which we assume the adaptive root and strong-RSA assumptions to hold. The (fixed) element $g \in \mathbb{G}$ is a randomly generated element of $\mathbb{G}$. For a multiset $\mathcal{M}$, the commitment to $\mathcal{M}$ is given by the element

$$\texttt{Com}(g, \mathcal{M}) := g^{\Pi(\mathcal{M})} \in \mathbb{G}$$

where $\Pi(\mathcal{M})$ is the product of all elements of $\mathcal{M}$, with the appropriate multiplicities.

Proof of storage: To ask the server $\mathcal{P}$ to prove that he is storing the data multiset $\mathcal{M}$, the client $\mathcal{V}$ can generate a random element $g_1 \in \mathbb{G}$ and ask the server to provide the element

$$\texttt{Com}(g_1, \mathcal{M}) := g_1^{\Pi(\mathcal{M})} \in \mathbb{G}$$

along with a non-interactive proof for $\texttt{EqDLog}[(g, \texttt{Com}(g, \mathcal{M})), \ (g_1, \texttt{Com}(g_1, \mathcal{M}))]$. The communication complexity is constant and in particular, is independent of the size of $\mathcal{M}$.

If the client needs the proof of storage for multiple data multisets $\mathcal{M}_1, \cdots, \mathcal{M}_n$, they may proceed as follows:

1. The hashing algorithm $\texttt{H}_{\texttt{FS},\lambda}$ generates a λ-bit prime γ and a group element g_1.
2. The Prover $\mathcal{P}$ computes

$$h := \prod_{i=1}^{n} \texttt{Com}(g, \mathcal{M}_i)^{\gamma^i} \ \ , \ \ h_1 := g_1^{\sum_{i=1}^{n} \Pi(\mathcal{M}_i)\gamma^i}$$

and sends h_1 to the Verifier $\mathcal{V}$ along with a non-interactive proof for $\texttt{EqDLog}[(g, h), \ (g_1, h_1)]$.

3. $\mathcal{V}$ independently computes h and accepts if and only if the $\texttt{EqDLog}$ proof is valid.

Thus, the proof is constant-sized irrespective of the number of data sets/multisets or their sizes.

Updates: When the data multiset is to be updated, the client sends the changes to the server. The server stores these changes and sends back the updated commitment to the multiset, along

with a non-interactive proof of exponentiation (PoE) so that the client can efficiently verify that the new commitment is the correct one.

Multiset sums: For multisets $\mathcal{M}_1, \cdots, \mathcal{M}_n$, the server node can verifiably send the client the commitment $A_{\sum}$ for the sum

$$\sum_{i=1}^{n} \mathcal{M}_i := \Big\{ \big(\sum_{i=1}^{n} \text{mult}(\mathcal{M}_i, x) \big) \times x : \ x \in \bigcup_{i=1}^{n} \texttt{Set}(\mathcal{M}_i) \Big\}$$

using the Protocol

$$\texttt{PoMultPolyDLog}[g, \ (A_1, \cdots, A_n, A_{\sum}), \ \big(\prod_{i=1}^{n} X_i \big) - X_{n+1}].$$

The proof consists of a constant number of $\mathbb{G}$-elements and $2n$ λ-bit integers.

Multiset differences: For multisets $\mathcal{M}, \mathcal{N}$, the difference $\mathcal{M} \setminus \mathcal{N}$ has commitment

$$\texttt{Com}(g, \mathcal{M} \setminus \mathcal{N}) = g^{\frac{\Pi(\mathcal{M})}{\Pi(\mathcal{M} \cap \mathcal{N})}}.$$

So the Prover can combine the Protocols PoGCD and PoMultPolyDLog to verifiably send the commitment to $\mathcal{M} \setminus \mathcal{N}$.

Disjointness: The server node can verifiably demonstrate that the multisets $\mathcal{M}_1, \cdots, \mathcal{M}_n$ are pairwise disjoint using the protocol AggRelPrimeDLog-1$[g, \ (A_1, \cdots, A_n)]$. Similarly, for any subset $I \subseteq \{1, \cdots, n\}$ of indices, the server node can use the protocol AggRelPrimeDLog-2 to verifiably show that the multisets

$$\widetilde{\mathcal{M}}_1 = \bigcup_{i \in I} \mathcal{M}_i \ , \ \widetilde{\mathcal{M}}_2 = \bigcup_{j \in \{1, \cdots, n\} \setminus I} \mathcal{M}_j$$

are disjoint. In both cases, the proofs, consist of $\mathbf{O}(1)$ group elements and $2n + \mathbf{O}(1)$ λ-bit integers.

Underlying sets: Given commitments to two multisets $\mathcal{M}$, $\mathcal{N}$, if the client needs to know whether the underlying sets coincide or if one is contained in the other, the server node can demonstrate this using the protocol PoConSets or PoNonConSets. The proof is constant-sized in each case.

5.1.1 Multiset intersections

Consider a setting where a client $\mathcal{V}$ who stores commitments $A_i := \texttt{Com}(g, \mathcal{M}_i)$ for data multisets $\mathcal{M}_1, \cdots, \mathcal{M}_n$ needs a commitment $\texttt{Com}(g, \mathcal{M}_\cap)$ to the intersection

$$\mathcal{M}_\cap := \bigcap_{i=1}^{n} \mathcal{M}_i.$$

In keeping with the rest of this paper, we would like to design a protocol that allows the Prover to do so while keeping the communication complexity to a minimum. Note that

$$d = \textbf{gcd}(d_1, \cdots, d_n) \iff (d | d_i \ \forall \ i) \ \bigwedge \ \exists \, (e_1, \cdots, e_n) \in \mathbb{Z}^n : \ \sum_{i=1}^{n} e_i d_i = d.$$

Furthermore,

$$d = \mathbf{gcd}(d_1, \cdots, d_n) \iff \left(d_1 d^{-1}, \cdots, d_n d^{-1}\right) \in \mathbb{Z}^n \bigwedge \mathbf{gcd}\left(d_1 d^{-1}, \cdots, d_n d^{-1}\right) = 1.$$

Hence, by changing the base from g to $a := g^d$, we can reduce this to the case where the GCD of the n integers is 1. So, the Prover can verifiably send the commitment for $\mathcal{M}_\cap$ as follows:

Protocol 5.1. *Protocol for the intersection of multisets.*

Parameters: $\mathbb{G} \xleftarrow{\$} \text{GGen}(\lambda),\ g \in \mathbb{G}$.

Input: Commitments $A_i = \texttt{Com}(g, \mathcal{M}_i) := g^{\Pi(\mathcal{M}_i)}$ for multisets $\mathcal{M}_i$; an element $A_\cap \in \mathbb{G}$

Claim: $A_\cap = \texttt{Com}(g, \bigcap_{i=1}^{n} \mathcal{M}_i) := g^{\Pi(\bigcap_{i=1}^{n} \mathcal{M}_i)}$.

1. The Prover $\mathcal{P}$ computes the integers $d_i := \frac{\Pi(\mathcal{M}_i)}{\Pi(\bigcap_{i=1}^{n} \mathcal{M}_i)} \quad (i = 1, \cdots, n)$.
2. $\mathcal{P}$ generates a non-interactive proof for $\texttt{AggKE-1}[A_\cap,\ (A_1, \cdots, A_n)]$ and sends it to the Verifier.
3. $\mathcal{P}$ uses the Euclidean algorithm to compute integers $e_1, \cdots, e_n$ such that $\sum_{i=1}^{n} e_i d_i = 1$.
4. $\mathcal{P}$ computes the elements $\breve{A}_i := a^{e_i}\ (i = 1, \cdots, n)$ and sends them to $\mathcal{V}$.
5. $\mathcal{P}$ generates a non-interactive proof for $\texttt{MultPolyDLog}[A_\cap,\ (A_1, \cdots, A_n, \breve{A}_1, \cdots, \breve{A}_n),\ f]$ where
$$f(X_1, \cdots, X_{2n}) := \sum_{i=1}^{n} X_i X_{n+i} - 1$$
and sends the proof to $\mathcal{V}$.
6. $\mathcal{V}$ verifies the proofs and accepts if and only if they are all valid. □

This proof entails $n + \mathbf{O}(1)$ group elements and $\mathbf{O}(n)$ λ-bit integers. As before, we would like to keep the number of group elements constant. To this end, the Prover can sample λ-bit integers γ until he finds one such that $\mathbf{gcd}(\sum_{i=1}^{n} d_i \gamma^i\ ,\ d_1) = 1$. He can then send a non-interactive proof for
$$\texttt{RelPrimeDLog}[(A_\cap, \prod_{i=1}^{n} A_i^{\gamma^i}),\ (A_\cap, A_1)].$$

When the elements of the multisets $\mathcal{M}_i$ are all λ-bit primes, the integers $\Pi(\mathcal{M}_i)$ are λ-rough and hence, finding an appropriate γ takes runtime $\mathbf{O}(1)$. This is because for an arbitrary γ and any λ-bit prime p, the Schwartz-Zippel lemma implies that
$$\texttt{Pr}\Big[\sum_{i=1}^{n} d_i \gamma^i \equiv 0 \pmod{p}\Big] = \texttt{negl}(\lambda).$$

As before, we say a multiset $\mathcal{M}$ is λ-**rough** if the integer $\Pi(\mathcal{M})$ is λ-rough or equivalently, if all elements of $\mathcal{M}$ are primes $> 2^{\lambda-1}$.

Protocol 5.2. *Protocol for the intersection of λ-rough multisets.*

Parameters: $\mathbb{G} \xleftarrow{\$} \text{GGen}(\lambda),\ g \in \mathbb{G}$.

Input: Commitments $A_i = \texttt{Com}(g, \mathcal{M}_i) := g^{\Pi(\mathcal{M}_i)}$ for λ-rough multisets $\mathcal{M}_i$ whose elements are λ-bit primes; an element $A_\cap \in \mathbb{G}$

Claim: $A_\cap = \texttt{Com}(g, \bigcap_{i=1}^{n} \mathcal{M}_i) := g^{\Pi(\bigcap_{i=1}^{n} \mathcal{M}_i)}$.

1. The Prover $\mathcal{P}$ computes the integers $d_i := \frac{\Pi(\mathcal{M}_i)}{\Pi(\bigcap_{i=1}^{n} \mathcal{M}_i)} \quad (i = 1, \cdots, n)$.

2. $\mathcal{P}$ generates non-interactive proofs for $\texttt{AggKE-1}[A_\cap,\ (A_1, \cdots, A_n)]$, $\texttt{PoKE}[g,\ A_\cap]$ and sends them to the Verifier $\mathcal{V}$.

3. $\mathcal{P}$ samples λ-bit primes γ until he finds one such that $\mathbf{gcd}(\prod_{i=1}^{n} d_i, \sum_{i=1}^{n} d_i\gamma^i) = 1$ and sends the integer γ to $\mathcal{V}$.

4. $\mathcal{P}$ computes $\widetilde{A} := g^{(\prod_{i=1}^{n} d_i)} \in \mathbb{G}$ and sends $\widetilde{A}$ to $\mathcal{V}$ along with a non-interactive proof for $\texttt{MultPolyDLog}[A_\cap,\ (A_1, \cdots, A_n, \widetilde{A}),\ (\prod_{i=1}^{n} X_i) - X_{n+1}]$.

5. $\mathcal{P}$ generates a non-interactive proof for the relation $\texttt{RelPrimeDLog}[(A_\cap, \widetilde{A}),\ (A_\cap, \prod_{i=1}^{n} A_i^{\gamma^i})]$ and sends it to $\mathcal{V}$.

6. $\mathcal{V}$ verifies all the proofs he receives and accepts if and only if they are all valid. □

Thus, the proof consists of a constant number of $\mathbb{G}$-elements and $2n + \mathbf{O}(1)$ λ-bit integers.

5.1.2 Multiset unions

The techniques in the last protocol also allow a server node to verifiably send over a commitment for the union of multisets $\mathcal{M}_1, \cdots, \mathcal{M}_n$. The proof that this commitment is valid can be publicly verified against the commitments $\texttt{Com}(g, \mathcal{M}_i)$ $(i = 1, \cdots, n)$ which the client stores. The basic idea here is as follows.

Lemma 5.3. *For integers $d_1, \cdots, d_n$, set $\widehat{d_j} := \prod_{\substack{1 \le i \le n \\ i \ne j}} d_i \ \ (j = 1, \cdots, n)$. Then we have*

$$\mathbf{lcm}(d_1, \cdots, d_n) \cdot \mathbf{gcd}(\widehat{d_1}, \cdots, \widehat{d_n}) = \prod_{i=1}^{n} d_i.$$

We omit the proof since it is straightforward. Thus, the following are equivalent:
- $\mathbf{lcm}(d_1, \cdots, d_n) = d$
- $d_i \big| d \ \forall\ i$ and there exist integers $\widehat{e}_1, \cdots, \widehat{e}_n$ such that $\prod_{i=1}^{n} d_i = d \cdot \sum_{i=1}^{n} \widehat{e}_i \widehat{d_i}$.
- The rationals $dd_i{}^{-1}$ are integers and $\mathbf{gcd}(dd_1{}^{-1}, \cdots, dd_n{}^{-1}) = 1$.

Protocol 5.4. *Protocol for the union of multisets.*

Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda),\ g \in \mathbb{G}$.
Input: Commitments $A_i = \texttt{Com}(g, \mathcal{M}_i) := g^{\Pi(\mathcal{M}_i)}$; an element $A_\cup \in \mathbb{G}$
Claim: $A_\cup = \texttt{Com}(g, \bigcup_{i=1}^{n} \mathcal{M}_i) := g^{\Pi(\bigcup_{i=1}^{n} \mathcal{M}_i)}$.

1. The Prover $\mathcal{P}$ computes the integers $d := \Pi\Big(\bigcup_{i=1}^{n} \mathcal{M}_i\Big)$, $\breve{d}_i := \frac{d}{\Pi(\mathcal{M}_i)}$ $(i = 1, \cdots, n)$.
2. $\mathcal{P}$ generates a non-interactive proof for the relation $\texttt{AggKE-2}[(A_1, \cdots, A_n),\ A_\cup]$ and sends it to the Verifier $\mathcal{V}$.
3. $\mathcal{P}$ uses the Euclidean algorithm to compute integers $\breve{e}_1, \cdots, \breve{e}_n$ such that $\sum_{i=1}^{n} \breve{e}_i \breve{d}_i = 1$.
4. $\mathcal{P}$ computes the elements $\breve{A}_i := g^{\breve{e}_i}$ and sends them to $\mathcal{V}$.
5. $\mathcal{P}$ generates a non-interactive proof for $\texttt{MultPolyDLog}[g,\ (A_1, \cdots, A_n, \breve{A}_1, \cdots, \breve{A}_n, A_\cup),\ \breve{f}]$ where

$$\breve{f}(X_1, \cdots, X_{2n+1}) := X_{2n+1}\Big(\sum_{i=1}^{n} X_{n+i}\big(\prod_{\substack{1\le j\le n\\ j\neq i}} X_j\big)\Big) - \prod_{i=1}^{n} X_i$$

and sends the proof to $\mathcal{V}$.
6. $\mathcal{V}$ verifies all of the proofs he receives and accepts if and only if they are valid. □

The proof entails $n + \mathbf{O}(1)$ group elements and $2n + \mathbf{O}(1)$ λ-bit integers. As was the case with multiset intersections, when the multisets are λ-rough, the protocol can be modified so that the number of group elements is constant.

Protocol 5.5. *Protocol for the union of λ-rough multisets.*
Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda)$, $g \in \mathbb{G}$.
Input: Commitments $A_i = \mathtt{Com}(g, \mathcal{M}_i) := g^{\Pi(\mathcal{M}_i)}$ for λ-rough multisets $\mathcal{M}_i$; an element $A_\cup \in \mathbb{G}$
Claim: $A_\cup = \mathtt{Com}(g, \bigcup_{i=1}^{n} \mathcal{M}_i) := g^{\Pi(\bigcup_{i=1}^{n} \mathcal{M}_i)}$.

1. The Prover $\mathcal{P}$ computes the integers $d := \Pi(\bigcup_{i=1}^{n} \mathcal{M}_i)$, $\breve{d}_i := \frac{d}{\Pi(\mathcal{M}_i)}$ $(i = 1, \cdots, n)$.
2. $\mathcal{P}$ generates a non-interactive proof for the relation $\mathtt{AggKE\text{-}2}[(A_1, \cdots, A_n),\ A_\cup]$ and sends it to the Verifier $\mathcal{V}$.
3. $\mathcal{P}$ samples λ-bit integers γ until he finds one such that $\mathbf{gcd}(d, \sum_{i=1}^{n} \breve{d}_i\gamma^i) = 1$ and sends γ to $\mathcal{V}$.
4. $\mathcal{P}$ computes

$$\breve{A} := g^{\sum_{i=1}^{n} \breve{d}_i\gamma^i} \in \mathbb{G}$$

and sends $\breve{A}$ to $\mathcal{V}$ along with a non-interactive proof for $\mathtt{MultPolyDLog}[g,\ (A_1, \cdots, A_n, A_\cup, \breve{A}), \breve{f}]$ where

$$\breve{f}(X_1, \cdots X_{n+2}) := X_{n+1}\Big(\sum_{i=1}^{n} \gamma^i\big(\prod_{\substack{1\le j\le n\\ j\neq i}} X_j\big)\Big) - X_{n+2}\prod_{i=1}^{n} X_i.$$

5. $\mathcal{P}$ generates a non-interactive proof for $\mathtt{RelPrimeDLog}[(g, A_\cup),\ (g, \prod_{i=1}^{n} A_i^{\gamma^i})]$ and sends it to $\mathcal{V}$.
6. $\mathcal{V}$ verifies all the proofs he receives and accepts if and only if they are all valid. □

The proof consists of a constant number of $\mathbb{G}$-elements and $2n+\mathbf{O}(1)$ λ-bit integers. A server node storing the data multisets $\mathcal{M}_i$ can use this protocol to verifiably send the succinct commitment for the union $\bigcup_{i=1}^{n} \mathcal{M}_i$. The validity of this commitment can be verified against the commitments to the $\mathcal{M}_i$ held by the client.

Thus, to summarize, the protocols in this paper allow the server node to verifiably send the client succinct commitments to unions, intersections, sums (and combinations thereof) of multisets for which the client holds succinct commitments.

5.1.3 Frequencies of elements

Consider a setting where a client node needs to keep track of the occurrences of a certain keyword or certain blocks of keywords in the files that he stores for a client node. If the client node suffers from a low storage capacity or weak computational power, he would prefer to outsource the files to an untrusted server node. As before, he stores succinct commitments to the data

sets/multisets derived from hashing the files. This allows the server node storing the data to send publicly verifiable proofs about the occurrences of batches of elements in the data sets/multisets.

Let $\mathcal{M}_1, \cdots, \mathcal{M}_n$ be data multisets and let

$$A_i := \mathtt{Com}(g, \mathcal{M}_i) = g^{\Pi(\mathcal{M}_i)} \ \ (i = 1, \cdots, n)$$

be the commitments with the same base $g \in \mathbb{G}$. Suppose the server node storing the data for the client needs to identify the data multiset with the highest frequency of a data set $\mathcal{D}$. The protocols we have developed so far allows him to do so with a proof that the client can verify against the commitments $A_1, \cdots, A_n$.

The protocol hinges on the basic fact that for integers d, d_1, d_2, the following are equivalent:
1. For every prime p dividing d, $\mathrm{val}_p(d_1) > \mathrm{val}_p(d_2)$, where $\mathrm{val}_p(x)$ is the largest integer k such that p^k divides x.
2. There exists an integer e such that de divides d_1 and $\mathbf{gcd}(de, d_2) = e$.

Protocol 5.6. *Protocol for the frequency of elements* 1.

Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda)$, $g \in \mathbb{G}$.
Input: Commitments $A_i := g^{\Pi(\mathcal{M}_i)}$ for multisets $\mathcal{M}_i$; a data set $\mathcal{D}$.
Claim: Each element of $\mathcal{D}$ occurs with a higher frequency in $\mathcal{M}_1$ than in $\mathcal{M}_i \ \forall\ i \geq 2$.

1. The Prover $\mathcal{P}$ computes

$$\widehat{A}_1 := g^{\Pi\left(\bigcup_{i=2}^{n} \mathcal{M}_i\right)}$$

and sends it to the verifier $\mathcal{V}$ along with a non-interactive proof for $\mathtt{AggKE\text{-}2}[(A_2, \cdots, A_n), \widehat{A}_1]$.

2. The Prover computes the group elements

$$B_1 := g^{\prod_{x \in \mathcal{D}} x^{\mathrm{mult}\left(\bigcup_{i=2}^{n} \mathcal{M}_{i,x}\right)}}, \ B_2 := B_1^{\Pi(\mathcal{D})} \ \in \ \mathbb{G}.$$

3. $\mathcal{P}$ sends B_1, B_2 to $\mathcal{V}$ along with a non-interactive $\mathtt{PoE}$ for the equation $B_2 = B_1^{\Pi(\mathcal{D})}$.

4. $\mathcal{P}$ generates non-interactive proofs for $\mathtt{PoKE}[B_2,\ A_1]$, $\mathtt{PoGCD}[(g, \widehat{A}_1),\ (g, B_2),\ (g, B_1)]$ and sends them to $\mathcal{V}$.

5. $\mathcal{V}$ verifies the three proofs and accepts if and only if all of them are valid. □

Proposition 5.7. *The Protocol for the frequency of elements* 1 *is secure in the generic group model.*

Proof. Write $d := \Pi(\mathcal{D})$ for brevity. The subprotocols

$$\mathtt{PoGCD}[(g, \widehat{A}_1),\ (g, B_2),\ (g, B_1)] \ \ \wedge \ \ \mathtt{PoE}[B_1,\ d,\ B_2]$$

imply that with overwhelming probability, the Prover possesses integers $\widehat{m}_1$, e such that

$$g^{\widehat{m}_1} = \widehat{A}_1 \ , \ g^e = B_1 \ , \ g^{ed} = B_2 \ , \ \ \mathbf{gcd}(\widehat{m}_1, ed) = e.$$

The subprotocol $\mathtt{PoKE}[B_2,\ A_1]$ implies that with overwhelming probability, ed divides $\Pi(\mathcal{M}_1)$.

Furthermore, the subprotocol $\mathtt{PoAggKE\text{-}2}[(A_2, \cdots, A_n),\ \widehat{A}_1]$ implies that with overwhelming probability, the integer $\mathbf{lcm}\big(\Pi(\mathcal{M}_2), \cdots, \Pi(\mathcal{M}_n)\big)$ divides $\widehat{m}_1$. Hence, for any prime p dividing d and any index $i \geq 2$,

$$\mathrm{val}_p(\Pi(\mathcal{M}_i)) \leq \mathrm{val}_p(\widehat{m}_1) < \mathrm{val}_p(ed).$$

On the other hand, ed divides d_1 and hence, $\mathrm{val}_p(ed) \leq \mathrm{val}_p(d_1)$.

Thus, for any prime $p \in \mathcal{D}$, $\mathrm{val}_p(\Pi(\mathcal{M}_1)) > \max\left\{\mathrm{val}_p(\Pi(\mathcal{M}_2)), \cdots, \mathrm{val}_p(\Pi(\mathcal{M}_n))\right\}$, which completes the proof. □

Similarly, the following protocol allows the server to prove that every element of a certain data set $\mathcal{D}$ occurs with a *lower* frequency in $\mathcal{M}_1$ than in any of the multisets $\mathcal{M}_2, \cdots, \mathcal{M}_n$.

Protocol 5.8. *Protocol for the frequency of elements* 2*.*
Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda),\ g \in \mathbb{G}$.
Input: Commitments $A_i := g^{\Pi(\mathcal{M}_i)}$ for multisets $\mathcal{M}_i$; a data set $\mathcal{D}$.
Claim: Each element of $\mathcal{D}$ occurs with a lower frequency in $\mathcal{M}_1$ than in $\mathcal{M}_i\ \forall\ i \geq 2$.
1. The Prover $\mathcal{P}$ computes

$$B_1 := g^{\prod\limits_{x \in \mathcal{D}} x^{\mathrm{mult}(\mathcal{M}_1, x)}},\ B_2 = B_1^{\Pi(\mathcal{D})}.$$

2. $\mathcal{P}$ sends B_1, B_2 to $\mathcal{V}$ along with a non-interactive `PoE` for the equation $B_2 = B_1^{\Pi(\mathcal{D})}$.
3. $\mathcal{P}$ generates non-interactive proofs for `AggKE-1`$[B_2,\ (A_2, \cdots, A_n)]$ and `PoGCD`$[(g, A_1),\ (g, B_2),\ (g, B_1)]$ and sends them to $\mathcal{V}$.
4. $\mathcal{V}$ verifies the three proofs and accepts if and only if they are all valid. □

We omit the security proof since it is virtually identical to the last one. The next two protocols are duals to the last two. They allow the server node to verifiably identify the multiset with the highest/lowest occurrence of a data set in a setting where the client stores the membership witnesses for the multisets with respect to a single accumulated digest.

Protocol 5.9. *Protocol for the frequency of elements* 3*.*
Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda),\ g \in \mathbb{G}$.
Input: Membership witnesses w_i for multisets $\mathcal{M}_i$ with respect to an accumulated digest A; a data set $\mathcal{D}$.
Claim: Each element of $\mathcal{D}$ occurs with a higher frequency in $\mathcal{M}_1$ than in $\mathcal{M}_i\ \forall\ i \geq 2$.
1. The Prover $\mathcal{P}$ uses Shamir's trick to compute an element $\breve{w}_1 \in \mathbb{G}$ such that

$$\breve{w}_1^{\Pi(\bigcup\limits_{i=2}^{n} \mathcal{M}_i)} = A$$

and sends it to the Verifier $\mathcal{V}$.
2. $\mathcal{P}$ computes the elements

$$\widetilde{w}_1 := w_1^{\prod\limits_{x \in \mathcal{D}} x^{\mathrm{mult}(\bigcup\limits_{i=2}^{n} \mathcal{M}_i, x)}},\quad \widetilde{w}_1^{\Pi(\mathcal{D})} \in \mathbb{G}$$

and sends them to $\mathcal{V}$ along with a non-interactive `PoE`$[\widetilde{w}_1,\ \Pi(\mathcal{D}),\ \widetilde{w}_1^{\Pi(\mathcal{D})}]$.
3. $\mathcal{P}$ generates a non-interactive proof for `AggKE-1`$[\breve{w}_1,\ (w_2, \cdots, w_n)]$ and sends it to $\mathcal{V}$.
4. $\mathcal{P}$ generates non-interactive proofs for `PoGCD`$[(\breve{w}_1, A),\ (w_1, \widetilde{w}_1^{\Pi(\mathcal{D})}),\ (w_1, \widetilde{w}_1)]$ and `PoKE`$[\widetilde{w}_1^{\Pi(\mathcal{D})},\ A]$ and sends them to $\mathcal{V}$.
5. $\mathcal{V}$ verifies the three proofs and accepts if and only if they are all valid. □

Proposition 5.10. *The Protocol for the frequency of elements* 3 *is secure in the generic group model.*

Proof. Set $d := \Pi(\mathcal{D})$. The subprotocols

$$\texttt{PoGCD}[(\breve{w}_1, A),\ (w_1, \widetilde{w}_1^{\Pi(\mathcal{D})}),\ (w_1, \widetilde{w}_1)] \ \bigwedge\ \texttt{PoE}[\widetilde{w}_1,\ \Pi(\mathcal{D}),\ \widetilde{w}_1^{\Pi(\mathcal{D})}]$$

imply that with overwhelming probability, the Prover possesses integers $e, \widehat{m}_1$ such that

$$\widehat{w}_1^{\widehat{m}_1} = A\ ,\ w_1^e = \widetilde{w}_1\ ,\ \mathbf{gcd}(ed, \widehat{m}_1) = e.$$

The subprotocol $\texttt{PoAggKE-1}[\breve{w}_1,\ (w_2, \cdots, w_n)]$ implies that with overwhelming probability, $\widehat{m}_1$ is divisible by $\mathbf{lcm}(\Pi(\mathcal{M}_2), \cdots, \Pi(\mathcal{M}_n))$. So, for any prime p dividing d and any index $i \geq 2$, we have

$$\text{val}_p(\Pi(\mathcal{M}_i)) \leq \text{val}_p(\widehat{m}_1) < \text{val}_p(ed).$$

On the other hand, the subprotocol $\texttt{PoKE}[\widetilde{w}_1^{\Pi(\mathcal{D})},\ A]$ implies that with overwhelming probability, ed divides $\Pi(\mathcal{M}_1)$. Hence, $\text{val}_p(ed) \leq \text{val}_p(\Pi(\mathcal{M}_1))$.

Thus, for any prime $p \in \mathcal{D}$, $\text{val}_p(\Pi(\mathcal{M}_1)) > \max(\text{val}_p(\mathcal{M}_2), \cdots, \text{val}_p(\Pi(\mathcal{M}_n)))$, which completes the proof. □

Protocol 5.11. *Protocol for the frequency of elements* 4.

Parameters: $\mathbb{G} \xleftarrow{\$} \text{GGen}(\lambda),\ g \in \mathbb{G}$.

Input: Membership witnesses w_i for multisets $\mathcal{M}_i$ with respect to an accumulated digest A; a data set $\mathcal{D}$.

Claim: Each element of $\mathcal{D}$ occurs with a lower frequency in $\mathcal{M}_1$ than in $\mathcal{M}_i\ \forall\ i \geq 2$.

1. The Prover $\mathcal{P}$ computes an element $\widehat{w}_1 \in \mathbb{G}$ such that

$$\widehat{w}_1^{\Pi(\bigcap_{i=2}^{n} \mathcal{M}_i)} := A$$

and sends $\widehat{w}_1$ to the Verifier $\mathcal{V}$ along with a non-interactive proof for $\texttt{AggKE-2}[(w_2, \cdots, w_n),\ \widehat{w}_1]$.

2. $\mathcal{P}$ computes

$$\widehat{A} := \widehat{w}_1^{\prod_{x \in \mathcal{D}} x^{\text{mult}(\mathcal{M}_1, x)}},\ \widehat{A}^{\Pi(\mathcal{D})}\ \in \mathbb{G}$$

and sends them to $\mathcal{V}$ along with a non-interactive $\texttt{PoE}$ for the exponentiation $\widehat{A}^{\Pi(\mathcal{D})}$.

3. $\mathcal{P}$ generates non-interactive proofs for $\texttt{PoKE}[\widehat{w}_1^{\Pi(\mathcal{D})},\ A]$ and $\texttt{PoGCD}[(w_1, A),\ (w_1, \widehat{A}^{\Pi(\mathcal{D})}),\ (w_1, \widehat{A})]$ and sends them to $\mathcal{V}$.

4. $\mathcal{V}$ verifies the three proofs and accepts if and only if they are all valid. □

We omit the security proof since it is virtually identical to the last one. In all four cases of the protocols for frequency, the proof consists of $\mathbf{O}(1)$ $\mathbb{G}$-elements and $n + \mathbf{O}(1)$ λ-bit integers.

5.1.4 Updates

We briefly discuss a few ways to handle updates and the tradeoffs between them. As before, let $\mathcal{M}_1, \cdots, \mathcal{M}_n$ be multisets a client $\mathcal{V}$ outsources to a server node $\mathcal{P}$. The client stores succinct commitments

$$\texttt{Com}(g, \mathcal{M}_i) = g^{\Pi(\mathcal{M}_i)},$$

where g is a randomly generated element of $\mathbb{G}$. The multisets are dynamic and hence, the commitments need to be updated in response to changes.

Inserts: When the multiset $\mathcal{M}_1$ changes to $\mathcal{M}_1 + \mathcal{M}_1'$, the server changes the commitment from $[g,\ g^{\Pi(\mathcal{M}_1)}]$ to $[g,\ g^{\Pi(\mathcal{M}_1+\mathcal{M}_1')}]$. He sends $g^{\Pi(\mathcal{M}_1+\mathcal{M}_1')}$ to the client along with a non-interactive `PoE` for the equation

$$(g^{\Pi(\mathcal{M}_1)})^{\Pi(\mathcal{M}_1')} = g^{\Pi(\mathcal{M}_1+\mathcal{M}_1')}.$$

Deletes: Let $\mathcal{M}_1'$ be a multiset contained in $\mathcal{M}_1$ and suppose $\mathcal{M}_1'$ is to be deleted from $\mathcal{M}_1$. Broadly there are three ways of handling deletes, each with some tradeoffs. We discuss them here.

1. The server node changes the commitment from $[g,\ g^{\Pi(\mathcal{M}_1)}]$ to $[g,\ g^{\Pi(\mathcal{M}_1\setminus\mathcal{M}_1')}]$. He sends $g^{\Pi(\mathcal{M}_1\setminus\mathcal{M}_1')}$ to the client along with a non-interactive `PoE` for the equation

$$(g^{\Pi(\mathcal{M}_1\setminus\mathcal{M}_1')})^{\Pi(\mathcal{M}_1')} = g^{\Pi(\mathcal{M}_1)}.$$

While this is probably the simplest way of handling deletions, the computational burden is linear in the size of $\mathcal{M}_1 \setminus \mathcal{M}_1'$.

2. The server node changes the commitment from $[g,\ g^{\Pi(\mathcal{M}_1)}]$ to $[g^{\Pi(\mathcal{M}_1')},\ g^{\Pi(\mathcal{M}_1)}]$. He sends $g_{\text{new}} := g^{\Pi(\mathcal{M}_1')}$ to the client along with a non-interactive `PoE` for the equation

$$g^{\Pi(\mathcal{M}_1')} = g_{\text{new}}.$$

The advantage is that the runtime complexity is $\mathbf{O}(\#\mathcal{M}_1')$. The downside is that the client needs to keep track of the different bases for the commitments as opposed to a single base g. This increases the communication complexity and the Prover's work when the client asks for an argument of knowledge for any relation between the multisets $\mathcal{M}_1, \cdots, \mathcal{M}_n$.

3. The server node changes the commitment for $\mathcal{M}_1$ from $[g,\ g^{\Pi(\mathcal{M}_1)}]$ to $[g^{\Pi(\mathcal{M}_1')},\ g^{\Pi(\mathcal{M}_1)}]$. Furthermore, he updates the commitments for $\mathcal{M}_i$ $(i = 2, \cdots, n)$ from $[g,\ g^{\Pi(\mathcal{M}_i)}]$ to $[g^{\Pi(\mathcal{M}_1')},\ g^{\Pi(\mathcal{M}_i+\mathcal{M}_1')}]$. He sends

$$g_{\text{new}} := g^{\Pi(\mathcal{M}_1')}\ ,\ g^{\Pi(\mathcal{M}_i+\mathcal{M}_1')}\ (i = 2, \cdots, n)$$

to the client along with the relevant non-interactive `PoE`s.

This preserves a common base for the n commitments. The runtime complexity is $\mathbf{O}(n \cdot \#\mathcal{M}_1')$, but the n exponentiations are completely parallelizable.

5.2 Sharded stateless blockchains

We briefly discuss the concept of a stateless blockchain and the need for it. Currently, in every existing blockchain, every full node in the system needs to store the entire state of the blockchain in order to validate incoming transactions. This has already become cumbersome as the size of the state grows. To this end, [Tod16] suggested the concept of a *stateless* blockchain. In this proposed model, every node stores the data relevant to itself and the accumulated digest. The miners no longer need to store the state since the concerns of state storage and consensus are decoupled.

Recently, the idea of introducing shards has been gaining ground within the blockchain ecosystem. While a sharded blockchain would theoretically have a higher throughput, it would be less secure since each individual shard would have fewer validators than the entire blockchain. To prevent collusion, an idea that has been floated is to periodically switch the validators assigned to the shards. Such a model makes stateless validation highly desirable since a stateful model would make it cumbersome for a validator to download the data for a new shard he gets assigned to.

While this stateless model would drastically alleviate the problem of state bloat, the big tradeoff would be that in such a model, a node sending over transactions must also send over proofs attesting to the validity of these transactions. At the moment, the authentication data structure used by blockchains is that of a Merkle tree and the membership proofs are what we call Merkle branches/paths. Despite the several advantages that Merkle trees provide, a drawback is that the membership proofs cannot be batched or aggregated. Thus, a stateless model that continues to use Merkle trees as the accumulator would suffer from a bandwidth bottleneck. To address this problem, [BBF19] etc. have proposed using a cryptographic accumulator with batchable/aggregable membership and non-membership proofs for a UTXO model. An accounts based model such as Ethereum could use a Vector Commitment with similar properties. In this regard, accumulators hinging on groups of unknown order have an important advantage over bilinear accumulators in that they are dynamic (constant runtime updates), have constant-sized public parameters and are transparent if the underlying hidden order group is a class group or a genus three Jacobian.

Consider the setting of a stateless sharded blockchain that, instead of a Merkle tree, hinges on a cryptographic accumulator instantiated with a hidden order group $\mathbb{G}$ ([BBF19]). Let g be a randomly selected element of $\mathbb{G}$ and $\mathbf{S}_1, \cdots, \mathbf{S}_n$ the distinct shards. Let $\mathcal{D}_i$ denote the data in shard $\mathbf{S}_i$ and $\mathcal{D} := \bigcup_{i=1}^{n} \mathcal{D}_i$. Then the accumulated digest (the analog of the Merkle root hash) of $\mathbf{S}_i$ is given by

$$A_i := \mathtt{Com}(g, \mathcal{D}_i) = g^{\Pi(\mathcal{D}_i)} \ \in \ \mathbb{G}.$$

The accumulated digest of the blockchain is given by

$$A := \mathtt{Com}(g, \mathcal{D}) = g^{\Pi(\mathcal{D})} \ \in \ \mathbb{G}.$$

In order to demonstrate that the data sets in distinct shards are pairwise disjoint, a Prover (such as a miner or an untrusted server node) can provide a non-interactive proof for the relation $\mathcal{R}_{\mathtt{AggRelPrimeDLog\text{-}1}}[a, \ (A_1, \cdots, A_n)]$. The proof consists of $\mathbf{O}(1)$ $\mathbb{G}$-elements and $2n + \mathbf{O}(1)$ λ-bit integers.

Disjointness verifiable against membership witnesses: Now consider the setting of a single shard. Let $\mathcal{V}_1, \cdots, \mathcal{V}_n$ be verifiers (such as light nodes) on the network. Let $\mathcal{E}_i$ denote the data set corresponding to $\mathcal{V}_i$. Suppose the verifiers need to verify that the data sets $\mathcal{E}_i$ are pairwise disjoint, but do not have access to the data sets outside their shards. A Prover $\mathcal{P}$ (such as a miner or an untrusted server node) can prove this pairwise disjointness as follows.

1. $\mathcal{V}_i$ $(i = 1, \cdots, n)$ broadcasts the membership witness w_i for $\mathcal{D}_i$ to the other $n-1$ nodes $\mathcal{V}_j$ $(j \neq i)$.
2. $\mathcal{V}_i$ sends the data $\mathcal{D}_i$ to the prover $\mathcal{P}$.
3. $\mathcal{P}$ computes

$$d_i := \prod_{d \in \mathcal{D}_i} d \ \ (i = 1, \cdots, n)$$

and generates a non-interactive proof for the protocol $\mathtt{PoAggRelPrimeDLog\text{-}3}[(w_1, \cdots, w_n), \ A]$, which he then broadcasts to the Verifiers.

The proof consists of $\mathbf{O}(1)$ $\mathbb{G}$-elements and $2n+\mathbf{O}(1)$ λ-bit integers. In particular, the Verifiers do not need access to the data sets $\mathcal{D}_i$ to verify this proof of disjointness.

Pre-computation: Although the argument systems in this paper have certain advantages such as transparency and succinctness, an important drawback is that the Prover's computational burden is rather high. This is primarily because exponentiations in hidden order groups are

expensive. Furthermore, these exponentiations are conjectured to be almost purely sequential, which means there is no way to make them faster, aside from better hardware. However, the effective runtime in several cases can be reduced using pre-computations on the Prover's part. In most of the protocols, the most expensive part is the computation

$$\widetilde{g} := g^{\sum_{i=1}^{n} d_i^{n\lambda}\gamma^i}$$

where the d_i are usually the products $\Pi(\mathcal{M}_i)$ of all elements of a data set/multiset and γ is a λ-bit integer randomly generated by the Fiat-Shamir heuristic. We use this step in quite a few of the protocols in this paper with the sole purpose of demonstrating that the d_i are integers rather than merely rationals. The Prover can reduce the effective runtime of this computation by pre-computing and storing the set $\{g^{2^i} \ : \ 1 \leq i \leq N\}$ for an appropriately large integer N. In certain settings such as an accumulator, this part can be circumvented by disallowing accumulation of primes $< 2^{\lambda-1}$.

6 Conclusion

We hope that at least some of the techniques will find more applications than what we have discussed in this paper. Several open questions remain, the foremost of which is whether the computational burden of the Prover can be alleviated. Furthermore, most of our proofs consist of a constant number of group elements and $\mathbf{O}(n)$ λ-bit integers, where n is the number of comitted sets/multisets involved. It would be desirable to compress the proof sizes further so that the proofs are genuinely constant-sized, independent of n.

The protocols involving disjointness of the committed sets/multisets inherently rely on the Prover having access to the entirety of the data. It would be interesting to see if we can modify the protocols so that they are more amenable to proof generation in a distributed setting where multiple Provers can only access some of the data.

A closely related line of research is to further explore class groups and Jacobians as candidates for transparent hidden order groups. The adaptive root assumption in these groups and the weaker assumptions such as low order and fractional root need further scrutiny.

Acknowledgements: The author thanks Benedikt Bünz and Dimitris Kolonelos for helpful feedback on previous drafts.

References

[BBF19] Dan Boneh, Benedikt Bünz, Ben Fisch, *Batching Techniques for Accumulators with Applications to IOPs and Stateless Blockchains.* In Alexandra Boldyreva and Daniele Micciancio, editors, Advances in Cryptology – Crypto 2019, pages 561–586, Cham, 2019. Springer International Publishing.

[BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz and Ben Fisch, *Verifiable delay functions.* In Hovav Shacham and Alexandra Boldyreva, editors, Crypto 2018, Part I, volume 10991 of LNCS

[BBF18] Dan Boneh, Benedikt Bünz, and Ben Fisch, *A survey of two verifiable delay functions.* Cryptology ePrint Archive, Report 2018/712, 2018. https://eprint.iacr.org/2018/712

[BFS19] Benedikt Bünz, Ben Fisch, Alan Szepieniec, *Transparent SNARKs from DARK Compilers*, Cryptology ePrint Archive, Report 2019/1229, 2019.

[BCM05] Endre Bangerter, Jan Camenisch, and Ueli Maurer. *Efficient proofs of knowledge of discrete logarithms and representations in groups with hidden order.* In Serge Vaudenay, editor, PKC 2005, volume 3386 of LNCS, Springer, Heidelberg, January 2005.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, Nicholas Spooner. *Interactive oracle proofs.* In Martin Hirt and Adam D. Smith, editors, TCC 2016-B, Part II, volume 9986 of LNCS, pages 31-60. Springer, Heidelberg, October/November 2016.

[BD94] Josh Cohen Benaloh and Michael de Mare. *One-way accumulators: A decentralized alternative to digital sinatures* (extended abstract). In Tor Helleseth, editor, Eurocrypt'93, volume 765 of LNCS, pages 274-285. Springer, Heidelberg, May 1994.

[BH01] Johannes Buchmann and Safuat Hamdy. *A survey on IQ cryptography*, In Public-Key Cryptography and Computational Number Theory.

[BKSW20] Karim Belabas, Thorsten Kleinjung, Antonio Sanso, Benjamin Wesolowski, *A note on the low order assumption in class group of an imaginary quadratic number fields*, Preprint

[BP97] Niko Bari and Birgit Pfitzmann. *Collision-free accumulators and fail-stop signature schemes without trees.* In Walter Fumy, editor, Eurocrypt'97, volume 1233 of LNCS, pages 480-494. Springer, Heidelberg, May 1997.

[BS96] Wieb Bosma and Peter Stevenhagen. *On the computation of quadratic 2-class groups* In Journal de Theorie des Nombres, 1996.

[CF13] Dario Catalano and Dario Fiore. *Vector commitments and their applications*, In Kaoru Kurosawa and Goichiro Hanaoka, editors, PKC 2013, volume 7778 of LNCS, pages 55-72. Springer, Heidelberg, February/March 2013.

[CFGKN20] Matteo Campanelli, Dario Fiore, Nicola Greco, Dimitris Kolonelos, Luca Nizzardo, *Vector Commitment Techniques and Applications to Verifiable Decentralized Storage*

[Can87] David Cantor. *Computing in the Jacobian of a hyperelliptic curve. Mathematics of computation*, 1987.

[Can94] David Cantor. *On the analogue of the division polynomials for hyperelliptic curves*, Crelle's Journal, 1994.

[DGS20] Samuel Dobson, Steven Galbraith, Benjamin Smith, *Trustless Groups of Unknown Order with Hyperelliptic Curves*

[DK02] Ivan Damgard and Maciej Koprowski. *Generic lower bounds for root extraction and signature schemes in general groups.* In Lars R. Knudsen, editor, Eurocrypt 2002, volume 2332 of LNCS, pages 256-271. Springer, Heidelberg, April / May 2002.

[Fis18] Ben Fisch. *Tight Proofs of Space and Replication.* In Y. Ishai and V. Rijmen, editors, Eurocrypt 2019, Part II, volume 11477 of LNCS, pages 324-348. Springer, Heidelberg, May 2019.

[FS87] Amos Fiat and Adi Shamir. *How to prove yourself: Practical solutions to identification and signature problems.*CRYPTO'86, volume 263 of LNCS, pages 186–194. Springer, Heidelberg, August 1987

[HM00] Safuat Hamdy and Bodo Moller. *Security of cryptosystems based on class groups of imaginary quadratic orders.* ASIACRYPT 2000, volume 1976 of LNCS, pages 234-247. Springer, Heidelberg, December 2000.

[KPZ17] N. Katz, C. Papamanthou, Y. Zhang, *An Expressive (Zero-Knowledge) Set Accumulator*, 2017 IEEE European Symposium on Security and Privacy

[Lip12] Helger Lipmaa. *Secure accumulators from euclidean rings without trusted setup.* ACNS 12, volume 7341 of LNCS, pages 224240. Springer, Heidelberg, June 2012.

[LLX07] Jiangtao Li, Ninghui Li, and Rui Xue. *Universal accumulators with efficient nonmembership proofs* ACNS 07, volume 4521 of LNCS, pages 253-269. Springer, Heidelberg, June 2007.

[LM18] Russell W.F. Lai and Giulio Malavolta, *Optimal succinct arguments via hidden order groups.* Cryptology ePrint Archive, Report 2018/705, 2018.

[Mic94] Silvio Micali. CS proofs (extended abstracts). In 35th FOCS, pages 436-453. IEEE Computer Society Press, November 1994.

[Ngu05] L. Nguyen. *Accumulators from bilinear maps and applications.* CT- RSA, 2005.

[Sha83] Adi Shamir. *On the generation of cryptographically strong pseudorandom sequences.* ACM Transactions on Computer Systems (TOCS), 1983 .

[Sho97] Victor Shoup, *Lower bounds for discrete logarithms and related problems.* Eurocrypt'97, volume 1233 of LNCS, pages 256266. Springer, Heidelberg, May 1997.

[Sut07] Andrew Sutherland, *Order Computations in Generic Groups*, MIT Thesis, 2007

[STY01] Tomas Sander, Amnon Ta-Shma, Moti Yung, *Blind, auditable membership proofs*, FC 2000, volume 1962 of LNCS, pages 5371. Springer, Heidelberg, February 2001.

[Th20] Steve Thakur, *Constructing hidden order groups using genus three Jacobians*, Preprint

[Tod16] Peter Todd. *Making UTXO Set Growth Irrelevant With Low-Latency Delayed TXO Commitments.* https://petertodd.org/2016/delayed-txo-commitments, May 2016.

[Wes19] Benjamin Wesolowski, *Efficient verifiable delay functions.* Advances in Cryptology – Eurocrypt 2019, pages 379–407, Cham, 2019. Springer International Publishing.

Steve Thakur
Axoni Research Group
New York City, NY
Email: stevethakur01@gmail.com

A List of symbols/abbreviations

$\mathbb{G}$: a group of hidden order in which we assume the adaptive root and strong-RSA assumptions to hold.

λ: The security parameter

$\texttt{negl}(\lambda)$: The set of functions negligible in λ.

$[n]$: The set of integers $\{0, 1, \cdots, n-1\}$

$\texttt{NextPrime}(n)$: the smallest prime $\geq n$

PPT: Probabilistic Polynomial Time

$a \equiv_\lambda b$: The equivalence of $a, b \in \mathbb{G}$ with respect to the relation $\equiv_\lambda$

$\mathbb{Z}_{\mathcal{S}}$: The localization of $\mathbb{Z}$ at a set $\mathcal{S}$ of rational primes

$\mathbb{Z}_{\mathcal{S}}^{\times}$: The group of units of $\mathbb{Z}_{\mathcal{S}}$

$\mathbf{gcd}_S(a, b)$: The GCD of elements $a, b \in \mathbb{Z}_{\mathcal{S}}$ in the principal ideal domain $\mathbb{Z}_{\mathcal{S}}$

$\mathbb{Z}_{(\lambda)}$: The localization of $\mathbb{Z}$ at the set of all rational primes $\leq 2^{\lambda-1}$.

$\mathcal{P}$: The Prover

$\mathcal{P}_{\text{mal}}$: A malicious Prover

$\mathcal{V}$: The Verifier

$\texttt{H}_{\texttt{FS},\lambda}$: The hashing algorithm used by the Fiat-Shamir heuristic

$\stackrel{\text{o.p.}}{\Longrightarrow}$: Implies with overwhelming probability

$\texttt{Set}(\mathcal{M})$: The underlying set of a multiset $\mathcal{M}$

$\text{mult}(\mathcal{M}, x)$: The multiplicity of an element x in a multiset $\mathcal{M}$ of λ-bit primes

$\Pi(\mathcal{M})$: The product $\prod_{x \in \texttt{Set}(\mathcal{M})} x^{\text{mult}(\mathcal{M},x)}$ of all elements of a multiset $\mathcal{M}$

$\text{val}_p(n)$: The largest integer k such that p^k divides n

PoE: Proof of Exponentiation ([Wes18], [BBF19])

PoKE: Proof of Knowledge of the Exponent ([BBF19])

$\mathcal{O}_1$, $\mathcal{O}_2$: The two oracles in a generic hidden order group

DLOG: The unlikely event that a PPT algorithm generate integers x_i such that $\prod_{i=1}^{k} g_i^{x_i} = 1$, where the g_i are the responses to the queries to the oracle $\mathcal{O}_1$.

B List of Protocols:

The following is a list of the protocols in this paper and the relations that the protocols are arguments of knowledge for, in the generic group model. In each of the protocols, we may replace $\mathbb{Z}$ by the localization $\mathbb{Z}_{\mathcal{S}}$ at any set $\mathcal{S}$ of rational primes.

1. PoEqDLog (*Proof of equality of discrete logarithms*)

$$\mathcal{R}_{\texttt{EqDLog}}[(a_1, b_1),\ (a_2, b_2)] = \left\{ \begin{array}{l} ((a_1, b_1),\ (a_2, b_2) \in \mathbb{G}^2 \\ d \in \mathbb{Z}) : \\ (b_1, b_2) = (a_1^d, a_2^d) \end{array} \right\}$$

2. PoPolyDLog (*Proof of polynomial relation between (two) discrete logarithms*)

$$\mathcal{R}_{\texttt{PolyDLog}}[(a_1,b_1),\ (a_2,b_2),\ f] = \left\{ \begin{array}{l} ((a_1,b_1),\ (a_2,b_2) \in \mathbb{G}^2,\ f \in \mathbb{Z}[X]); \\ (d_1,d_2) \in \mathbb{Z}^2 : \\ b_1 = a_1^{d_1} \ \bigwedge \ b_1 = a_1^{d_1} \ \bigwedge \ d_2 = f(d_1) \end{array} \right\}$$

3. PoAggKE-1 (*Proof of aggregated knowledge of exponents*-1)

$$\mathcal{R}_{\texttt{AggKE-1}}[a,\ (a_1,\cdots,a_n)] = \left\{ \begin{array}{l} (a \in \mathbb{G},\ (a_1,\cdots,a_n) \in \mathbb{G}^n); \\ (d_1,\cdots,d_n) \in \mathbb{Z}^n) : \\ a_i = a^{d_i}\ \forall\ i \end{array} \right\}$$

4. PoAggKE-2 (*Proof of aggregated knowledge of exponents*-2)

$$\mathcal{R}_{\texttt{AggKE-2}}[(a_1,\cdots,a_n),\ A] = \left\{ \begin{array}{l} ((a_1,\cdots,a_n) \in \mathbb{G}^n,\ A \in \mathbb{G}) \\ (d_1,\cdots,d_n) \in \mathbb{Z}^n) : \\ A = a_i^{d_i}\ \forall\ i \end{array} \right\}$$

5. PoMultPolyDLog (*Proof of multivariate polynomial relations between discrete logarithms*)

$$\mathcal{R}_{\texttt{MultPolyDLog}}[a,\ (b_1,\cdots,b_n),\ (f_1,\cdots,f_k)] = \left\{ \begin{array}{l} (a \in \mathbb{G},\ (b_1,\cdots,b_n) \in \mathbb{G}^n); \\ (f_1,\cdots,f_k) \in \mathbb{Z}[X_1,\cdots,X_n]^k; \\ (d_1,\cdots,d_n) \in \mathbb{Z}^n)\ : \\ b_i = a^{d_i}\ \forall\ i\ \bigwedge \\ f_j(d_1,\cdots,d_n) = 0\ \forall\ j \end{array} \right\}$$

6. PoGCD (*Proof of GCD*)

$$\mathcal{R}_{\texttt{GCD}}[(a_1,b_1),\ (a_2,b_2),(a_3,b_3)] = \{((a_i,b_i \in \mathbb{G});\ d_i \in \mathbb{Z})\ :\ b_i = a_i^{d_i},\ \mathbf{gcd}(d_1,d_2) = d_3\}$$

7. PoRelPrimeDLog (*Proof of relatively prime discrete logarithms*; special case of PoGCD)

$$\mathcal{R}_{\texttt{RelPrimeDLog}}[(a_1,b_1),\ (a_2,b_2)] = \{((a_i,b_i \in \mathbb{G});\ d_i \in \mathbb{Z})\ :\ b_i = a_i^{d_i},\ \mathbf{gcd}(d_1,d_2) = 1\}.$$

8. PoAggRelPrimeDLog-1 (*Aggregated proof of relatively prime discrete logarithms*-1)

$$\mathcal{R}_{\texttt{AggRelPrimeDLog-1}}[a,\mathbf{a}] = \left\{ \begin{array}{l} \big(a \in \mathbb{G},\ \mathbf{a} := (a_1,\cdots,a_n) \in \mathbb{G}^n); \\ (d_1,\cdots,d_n) \in \mathbb{Z}^n\big) : \\ a_i = a^{d_i}\ \forall\ i\ \ ,\ \ \mathbf{gcd}(d_i,d_j) = 1)\ \forall\ i \neq j \end{array} \right\}$$

9. PoAggRelPrimeDLog-2 (*Aggregated proof of relatively prime discrete logarithms*-2)

$$\mathcal{R}_{\texttt{AggRelPrimeDLog-2}}[a_1,a_2,\mathbf{b},\mathbf{c}] = \left\{ \begin{array}{l} \big((a_1,a_2) \in \mathbb{G}^2, \\ \mathbf{b} := (b_1,\cdots,b_m) \in \mathbb{G}^m,\ \mathbf{c} := (c_1,\cdots,c_n) \in \mathbb{G}^n); \\ ((d_1,\cdots,d_m) \in \mathbb{Z}^m,\ (e_1,\cdots,e_n) \in \mathbb{Z}^n\big)\ : \\ (b_i = a_1^{d_i}\ \ \bigwedge\ \ c_j = a_2^{e_j}\ \bigwedge\ \mathbf{gcd}(d_i,e_j) = 1)\ \forall\ i,j \end{array} \right\}$$

10. PoAggRelPrimeDLog-3 (*Aggregated proof of relatively prime discrete logarithms*-3)

$$\mathcal{R}_{\texttt{AggRelPrimeDLog-3}}[(w_1,\cdots,w_n),\ A] = \left\{ \begin{array}{l} \big(A \in \mathbb{G},\ (w_1,\cdots,w_n)\ \in \mathbb{G}^n); \\ ((d_1,\cdots,d_n) \in \mathbb{Z}^n\big)\ : \\ w_i^{d_i} = A\ \forall\ i\ \bigwedge \\ \mathbf{gcd}(d_i,d_j) = 1\ \forall\ i,j : i \neq j \end{array} \right\}$$

11. `PoAggRelPrimeDLog-4` (*Aggregated proof of relatively prime discrete logarithms*-4)

$$\mathcal{R}_{\texttt{AggRelPrimeDLog-4}}[\mathbf{b}, \mathbf{c}, B, C] = \left\{ \begin{array}{l} \big((B, C) \in \mathbb{G}^2, \\ \mathbf{b} = (b_1, \cdots, b_m) \in \mathbb{G}^m,\ \mathbf{c} = (c_1, \cdots, c_n) \in \mathbb{G}^n); \\ ((d_1, \cdots, d_m) \in \mathbb{Z}^m,\ (e_1, \cdots, e_n) \in \mathbb{Z}^n\big)\ : \\ (B = b_i^{d_i},\ C = c_j^{e_j}\ \bigwedge\ \mathbf{gcd}(d_i, e_j) = 1)\ \forall\ i, j \end{array} \right\}$$

12. *Protocol for the intersection of multisets*

$$\mathcal{R}_{\cap}[a,\ (A_1, \cdots, A_n),\ A_{\cap}] = \left\{ \begin{array}{l} (a \in \mathbb{G}; \\ (A_1, \cdots, A_n) \in \mathbb{G}^n\ ,\ A_{\cap} \in \mathbb{G}; \\ ((d_1, \cdots, d_n, d_{\cap}) \in \mathbb{Z}^{n+1}) : \\ A_i = g^{d_i}\ \forall\ i\ ,\ A_{\cap} = g^{d_{\cap}} \\ \mathbf{gcd}(d_1, \cdots, d_n) = d_{\cap} \end{array} \right\}$$

13. *Protocol for the union of multisets*

$$\mathcal{R}_{\cup}[a,\ (A_1, \cdots, A_n),\ A_{\cup}] = \left\{ \begin{array}{l} (a \in \mathbb{G}; \\ (A_1, \cdots, A_n) \in \mathbb{G}^n)\ ,\ ,A_{\cup} \in \mathbb{G}; \\ ((d_1, \cdots, d_n, d_{\cup}) \in \mathbb{Z}^{n+1}) : \\ A_i = g^{d_i}\ \forall\ i\ ,\ A_{\cup} = g^{d_{\cup}} \\ \mathbf{lcm}(d_1, \cdots, d_n) = d_{\cup} \end{array} \right\}$$

14. *Proof of the containment/non-containment of the underlying sets*

$$\mathcal{R}_{\texttt{ConSets}}[(a_1, A_1),\ (a_2, A_2)] = \left\{ \begin{array}{l} (a_1, a_2, A_1, A_2\ \in\ \mathbb{G}; \\ (d_1, d_2, N) \in \mathbb{Z}^3) : \\ a_1^{d_1} = A_1\ \wedge\ a_2^{d_2} = A_2 \\ \wedge\ d_2^N \equiv 0\ (\mathrm{mod}\ d_1) \end{array} \right\}$$

$$\mathcal{R}_{\texttt{NonConSets}}[(a_1, A_1),\ (a_2, A_2)] = \left\{ \begin{array}{l} (a_1, a_2, A_1, A_2\ \in\ \mathbb{G}; \\ (d_1, d_2, p) \in \mathbb{Z}^3) : \\ a_1^{d_1} = A_1\ \wedge\ a_2^{d_2} = A_2\ \wedge \\ p \big| d_2\ \wedge\ p \big| d_1\ \wedge\ p \neq \pm 1 \end{array} \right\}$$

15. *Protocol for the highest/lowest frequency of elements*

If the Verifier has access to the commitments $\texttt{Com}(g, \mathcal{M}_i) = g^{\Pi(\mathcal{M}_i)}$ to multisets $\mathcal{M}_1, \cdots, \mathcal{M}_n$ and a data set $\mathcal{D}$, the Prover can show that every element of $\mathcal{D}$ occurs with a higher/lower frequency in $\mathcal{M}_1$ than in any $\mathcal{M}_i$ for $i \geq 2$.

Similarly, given a single accumulated digest A and witnesses w_i for multisets $\mathcal{M}_i$ such that $w_i^{\Pi(\mathcal{M}_i)} = A$, the Prover can show that every element of $\mathcal{D}$ occurs with a higher/lower frequency in $\mathcal{M}_1$ than in any $\mathcal{M}_i$ for $i \geq 2$.

C Proof of multi-exponentiation

As before, let $\mathbb{G}$ be a generic hidden order group for which we assume the adaptive root and strong-RSA assumptions to hold. Consider an equation $\prod_{i=1}^{n} a_i^{e_i} = b$, where the $e_i \in \mathbb{Z}$ are public integers and $a_1, \cdots, a_n, b$ are elements of $\mathbb{G}$. Wesolowski's `PoE` protocol ([Wes18]) can be slightly generalized so as to reduce the Verifier's burden of computation for this equation.

Protocol C.1. *Proof of multi-exponentiation*
Parameters: $\mathbb{G} \xleftarrow{\$} \text{GGen}(\lambda)$, $g \in \mathbb{G}$
Input: $(a_1, \cdots, a_n) \in \mathbb{G}^n$; $b \in \mathbb{G}$; $(e_1, \cdots, e_n) \in \mathbb{Z}^n$
Claim: $\prod_{i=1}^{n} a_i^{e_i} = b$.

1. The hashing algorithm $\mathtt{H}_{\mathtt{FS},\lambda}$ generates a λ-bit prime ℓ.
2. The Prover $\mathcal{P}$ computes integers q_i, r_i such that $e_i = q_i \cdot \ell + r_i \ \ , \ \ r_i \in [\ell]$.
3. $\mathcal{P}$ computes $Q := \prod_{i=1}^{n} a_i^{q_i} \ \in \ \mathbb{G}$ and sends Q to the Verifier $\mathcal{V}$.
4. $\mathcal{V}$ computes $r_i := e_i \pmod{\ell}$ and verifies the equation $Q^\ell \prod_{i=1}^{n} a_i^{r_i} \stackrel{?}{=} b$ and accepts if and only if the equation holds. □

Thus, the proof consists of a single element $\mathbb{G}$-element Q. The Verifier's work is thus reduced to $n+1$ parallelizable λ-bit exponentiations in $\mathbb{G}$.

Soundness of the protocol: Suppose the Prover $\mathcal{P}$ sends an element $Q \in \mathbb{G}$ such that

$$Q^\ell \prod_{i=1}^{n} a_i^{r_i} = b \ \ , \ \ r_i := e_i \pmod{\ell}$$

for a randomly generated λ-bit prime ℓ. Let $\{g_1, \cdots, g_k\} \in \mathbb{G}$ be the responses to the queries from $\mathcal{A}$ to the oracle $\mathcal{O}_1$. With overwhelming probability, $\mathcal{A}$ is able to extract integers $\alpha_{i,j}$ $(1 \leq i \leq n \ , \ 1 \leq j \leq k)$, $\beta_1, \ldots, \beta_k$ bounded by $2^{\mathtt{poly}(\lambda)}$ such that

$$b = \prod_{j=1}^{k} g_j^{\beta_j} \ , \ a_i = \prod_{j=1}^{k} g_j^{\alpha_{i,j}}.$$

Since the equation $Q^\ell \prod_{i=1}^{n} a_i^{r_i} = b$ holds, the adaptive root assumption implies that with overwhelming probability, $\beta_j \equiv \sum_{i=1}^{n} e_i \alpha_{i,j} \pmod{\ell} \ \ \forall \ j$. Since the λ-bit prime ℓ was randomly generated, it follows that with overwhelming probability, $\beta_j = \sum_{i=1}^{n} e_i \alpha_{i,j} \ \ \forall \ j$ and hence, $b = \prod_{i=1}^{n} a_i^{e_i}$. □

D $\mathbb{Z}_{(\lambda)}$-integers and the equivalence relation $(\equiv_\lambda)$

Definition D.1. *For $\mathbb{Z}_{(\lambda)}$-integers d_1, d_2 we say $d_1 \equiv_\lambda d_2$ if $d_1 {d_2}^{-1}$ is a unit in $\mathbb{Z}_{(\lambda)}$.*

This is clearly a homomorphic equivalence relation.

Definition D.2. *For $\mathbb{Z}_{(\lambda)}$-integers d_1, d_2, we denote by $\mathbf{gcd}_\lambda(d_1, d_2)$ the largest λ-rough integer that divides both d_1 and d_2 in the principal ideal domain $\mathbb{Z}_{(\lambda)}$. Similarly, we denote by $\mathbf{lcm}_\lambda(d_1, d_2)$ the smallest λ-rough integer divisible by d_1 and d_2 in $\mathbb{Z}_{(\lambda)}$.*

Definition D.3. *For elements a, b in a hidden order group $\mathbb{G}$, we say $a \equiv_\lambda b$ with respect to a PPT algorithm $\mathcal{A}$ if $\mathcal{A}$ can verifiably generate relatively prime λ-smooth integers d_1, d_2 such that $a^{d_1} = b^{d_2} \in \mathbb{G}$ and $|d_1|, |d_2| < 2^{\mathtt{poly}(\lambda)}$.*

Because of Shamir's trick, this is equivalent to $\mathcal{P}$ being able to generate an element $a_0 \in \mathbb{G}$ and relatively prime λ-smooth integers d_1, d_2 such that $a_0^{d_2} = a,\ a_0^{d_1} = b$. It is easy to see that this an equivalence relation.

Proposition D.1. *The relation $(\equiv_\lambda)$ is an equivalence relation.*

Proof. Since the reflexivity and the symmetry are obvious, it suffices to show that the relation is transitive.
(Transitivity): Suppose $a \equiv_\lambda b$ and $b \equiv_\lambda c$ for elements $a, b, c \in \mathbb{G}$. Then $\mathcal{P}$ possesses λ-smooth integers d_1, d_2, d_3, d_4 such that

$$a^{d_1} = b^{d_2}\ ,\ b^{d_3} = c^{d_4}\ ,\ \mathbf{gcd}(d_1, d_2) = \mathbf{gcd}(d_3, d_4) = 1.$$

Now,

$$a^{d_1 d_3} = b^{d_2 d_3} = c^{d_2 d_4}$$

and clearly, the integers $d_1 d_3, d_2 d_4$ are λ-smooth. Set $d := \mathbf{gcd}(d_1 d_3, d_2 d_4)$ and $e_1 := d_1 d_3/d,\ e_2 := d_2 d_4/d$. Then e_1, e_2 are co-prime and λ-smooth and

$$a^{e_1} = c^{e_2}.$$

Thus, $a \equiv_\lambda c$. □

For elements $a, b \in \mathbb{G}$ the following are equivalent:

1. $a^d \equiv_\lambda b$ for some integer d.
2. $a^d \equiv_\lambda b$ for some λ-rough integer d.
3. $b = a^{d_1}$ for some $\mathbb{Z}_{(\lambda)}$-integer d_1.

Furthermore, if a PPT algorithm is able to output an element $a \in \mathbb{G}$ and integers d_1, d_2 such that $a^{d_1} \equiv_\lambda a^{d_2}$, then with overwhelming probability, $d_1 {d_2}^{-1} \in \mathbb{Z}_{(\lambda)}^\times$. In particular, no PPT algorithm can output an element $a \in \mathbb{G}$ and distinct λ-rough integers d_1, d_2 such that $a^{d_1} \equiv_\lambda a^{d_2}$.

We note, however, that the relation $(\equiv_\lambda)$ is not homomorphic, meaning that $a_1 \equiv_\lambda a_2, b_1 \equiv_\lambda b_2$ does not imply $a_1 a_2 \equiv_\lambda b_1 b_2$. But the relation is *partly* homomorphic in the sense that for any integer d,

$$a \equiv_\lambda b \iff a^d \equiv_\lambda b^d.$$

Non-membership proofs in accumulators: The best-known application of the knowledge of exponent protocol is constant-sized batched non-membership proofs in accumulators ([BBF19]). We briefly discuss the implications of replacing equality of $\mathbb{G}$-elements with the equivalence relation $\equiv_\lambda$ in this regard.

Let $g \in \mathbb{G}$ denote the genesis state of the accumulator, $\mathcal{D}$ the inserted data set and $A = g^{\Pi(\mathcal{D})}$ the accumulated digest. Given a data set $\mathcal{D}_0$ disjoint with $\mathcal{D}$, the Prover demonstrates non-membership for all elements of $\mathcal{D}_0$ by sending the following to the Verifier:
- Elements $w, A_1 \in \mathbb{G}$ such that $w^{\Pi(\mathcal{D}_0)} A_1 = g$.
- A non-interactive proof for $\mathtt{PoKE}[A,\ A_1]$.

Suppose, instead of $\mathtt{PoKE}[A,\ A_1]$, the Prover proves the weaker statement that he possesses an integer k such that $A^k \equiv_\lambda A_1$. By definition, there exist an integer k and a λ-smooth integer e such that $\mathbf{gcd}(k, e) = 1$ and $A^k = A_1^e$. Write $w = g^x$. Then

$$x \cdot \Pi(\mathcal{D}_0) + \frac{k \cdot \Pi(\mathcal{D})}{e} = 1$$

and hence,

$$e \cdot x\Pi(\mathcal{D}_0) + k \cdot \Pi(\mathcal{D}) = e.$$

Thus, $\mathbf{gcd}(\Pi(\mathcal{D}_0), \Pi(\mathcal{D}))$ divides e which is a λ-smooth integer. Since each element of $\mathcal{D}$ is a λ-bit prime, it follows that $\mathcal{D} \cap \mathcal{D}_0 = \emptyset$, despite $\frac{k}{e}$ possibly not being an integer. Thus, the equivalence relation $\equiv_\lambda$ is compatible with the nonmembership protocol of [BBF19].

E Underlying sets of committed multisets

Let a_1, a_2 be elements of $\mathbb{G}$. Let $\mathcal{M}, \mathcal{N}$ be multisets of rational primes. Let

$$A_1 := \mathtt{Com}(g, \mathcal{M}) = a_1^{\Pi(\mathcal{M})}\ ,\ A_2 := \mathtt{Com}(g, \mathcal{N}) = a_2^{\Pi(\mathcal{N})}\ \in\ \mathbb{G}$$

be the commitments to $\mathcal{M}$, $\mathcal{N}$ with bases $a_1, a_2 \in \mathbb{G}$.

Clearly, the relation $\mathcal{N} \subseteq \mathcal{M}$ can be demonstrated by the protocol $\mathtt{PoKE}[A_2, A_1]$. We now show that the protocol `PolyDLog` allows a Prover to succinctly demonstrate the following relations between the underlying sets of $\mathcal{M}$, $\mathcal{N}$, the proofs for which can be publicly verified against the commitments to $\mathcal{M}$ and $\mathcal{N}$.

1. $\mathtt{Set}(\mathcal{M}) \subseteq \mathtt{Set}(\mathcal{N})$.
2. $\mathtt{Set}(\mathcal{M}) \nsubseteq \mathtt{Set}(\mathcal{N})$.
3. $\mathtt{Set}(\mathcal{M}) = \mathtt{Set}(\mathcal{N})$

Before we describe the protocols, we note a few basic facts. Clearly, we have

$$\mathtt{Set}(\mathcal{M}) = \mathtt{Set}(\mathcal{N}) \Longleftrightarrow \mathtt{Set}(\mathcal{M}) \subseteq \mathtt{Set}(\mathcal{N}) \bigwedge \mathtt{Set}(\mathcal{N}) \subseteq \mathtt{Set}(\mathcal{M}).$$

Furthermore, with notations as before, we have

$$\mathtt{Set}(\mathcal{M}) \subseteq \mathtt{Set}(\mathcal{N}) \Longleftrightarrow \exists\ N \in \mathbb{Z}\ :\ \Pi(\mathcal{M})^N \equiv 0\ (\mathrm{mod}\ \Pi(\mathcal{N})).$$

Likewise, to show that $\mathtt{Set}(\mathcal{M}) \nsubseteq \mathtt{Set}(\mathcal{N})$, it suffices to show that there exists an integer p such that

$$p \notin \{-1, 1\}\ \bigwedge\ \Pi(\mathcal{M}) \equiv 0\ (\mathrm{mod}\ p)\ \bigwedge\ \mathbf{gcd}(\Pi(\mathcal{N}), p) = 1.$$

Protocol E.1. *Protocol for the containment of underlying sets* (`PoConSets`).

Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda),\ g \in \mathbb{G}$.

Input: Elements $a_1, a_2 \in \mathbb{G}$; commitments $A_1 := \mathtt{Com}(a_1, \mathcal{M}) = a_1^{\Pi(\mathcal{M})}\ ,\ A_2 := \mathtt{Com}(a_2, \mathcal{N}) = a_2^{\Pi(\mathcal{N})}$ to multisets $\mathcal{M}$, $\mathcal{N}$.

Claim: $\mathtt{Set}(\mathcal{N}) \subseteq \mathtt{Set}(\mathcal{M})$.

1. The Prover $\mathcal{P}$ computes $N := \max\{\mathrm{mult}(\mathcal{N}, x) : x \in \mathcal{N}\}$ and

$$A_3 := a_1^{\Pi(\mathcal{M})^N}.$$

He sends A_3 and N to the Verifier $\mathcal{V}$.

2. $\mathcal{P}$ generates a non-interactive proof for $\mathtt{PoPolyDLog}[(a_1, A_1),\ (a_2, A_3),\ X^N]$ and sends it to $\mathcal{V}$.

3. $\mathcal{P}$ generates a non-interactive proof for $\mathtt{PoKE}[A_2, A_3]$ and sends it to $\mathcal{V}$.

4. $\mathcal{V}$ verifies the two proofs and accepts if and only if both are valid. □

Protocol E.2. *Protocol for the non-containment of underlying sets* (`PoNonConSets`).

Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda),\ g \in \mathbb{G}$.

Input: Elements $a_1, a_2 \in \mathbb{G}$; commitments $A_1 := \mathtt{Com}(a_1, \mathcal{M}) = a_1^{\Pi(\mathcal{M})}$, $A_2 := \mathtt{Com}(a_2, \mathcal{N}) = a_2^{\Pi(\mathcal{N})}$ to multisets $\mathcal{M}$, $\mathcal{N}$.

Claim: $\mathtt{Set}(\mathcal{M}) \nsubseteq \mathtt{Set}(\mathcal{N})$.

1. The Prover chooses an integer p such that $p \in \mathtt{Set}(\mathcal{M}) \setminus \mathtt{Set}(\mathcal{N})$. and computes $b_1 := a_1^p$. He sends b_1 to the Verifier $\mathcal{V}$ along with a non-interactive proof for $\mathtt{PoKE}[b_1, A_1]$.

2. $\mathcal{P}$ generates a non-interactive proof for $\mathtt{RelPrimeDLog}[(a_1, b_1),\ (a_2, A_2)]$ and sends it to $\mathcal{V}$.

3. $\mathcal{V}$ verifies that $b_1 \notin \{a_1, a_1^{-1}\}$ and the proofs for $\mathtt{PoKE}[b_1, A_1]$, $\mathtt{RelPrimeDLog}[(a_1, b_1),\ (a_2, A_2)]$. He accepts if and only if both proofs are valid. □

In both cases, the proofs consists of a constant number of $\mathbb{G}$-elements and λ-bit integers. In particular, the proof size is independent of the sizes of $\mathcal{M}$ and $\mathcal{N}$.