ZK Arguments of Knowledge via hidden order groups

Steve Thakur

Abstract

We study non-interactive zero- knowledge (HVZK) arguments of knowledge 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 zero-knowledge proofs that are publicly verifiable against the constant-sized commitments.

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. This is a follow-up to our recent work ([Th20]).

1 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

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

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

1.1 Argument Systems

An argument system for a relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$ is a triple of randomized polynomial time algorithms $(\texttt{PGen}, \mathcal{P}, \mathcal{V})$, where $\texttt{PGen}$ takes an (implicit) security parameter λ and outputs a common reference string (CRS) $\texttt{pp}$. 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 $\texttt{pp}$. The verifier $\mathcal{V}$ takes as input $\texttt{pp}$ and x and after interactions with $\mathcal{P}$ outputs 0 or 1. We denote the transcript between the prover and the verifier by $\langle \mathcal{V}(\texttt{pp}, x), \mathcal{P}(\texttt{pp}, x, w)\rangle$ and write $\mathcal{V}(\texttt{pp}, x), \mathcal{P}(\texttt{pp}, x, w)\rangle = 1$ 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 $\texttt{PGen}$ 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 1.1. *We say an argument system* $(\texttt{PGen}, \mathcal{P}, \mathcal{V})$ *for a relation* $\mathcal{R}$ *is **complete** if for all* $(x, w) \in \mathcal{R}$,

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

Definition 1.2. *We say an argument system* $(\texttt{PGen}, \mathcal{P}, \mathcal{V})$ *is **sound** if* $\mathcal{P}$ *cannot forge a fake proof except with negligible probability.*

Definition 1.3. *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 1.4. *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.

1.2 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 1.1. *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 1.2. *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 1.3. *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 1.4. *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*

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

*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 1.5. *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

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

1.2.1 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 $\mathcal{L}$.

1.3 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

$\mathrm{mult}(\mathcal{M}, x)$ the multiplicity of x in $\mathcal{M}$. Thus, $\mathcal{M} = \{\mathrm{mult}(\mathcal{M}, x) \times x : \ x \in \mathtt{Set}(\mathcal{M})\}$. For brevity, we write

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

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

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

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.

1.4 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 $\mathrm{mult}(\mathcal{M}, x)$ denote the multiplicity of x in $\mathcal{M}$. The *accumulated digest* or *accumulated state* of $\mathcal{M}$ is given by

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

where

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

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

$$w(\mathcal{M}_0) := g^{\prod\limits_{x \in \mathtt{Set}(\mathcal{M})} x^{\mathrm{mult}(\mathcal{M} \setminus \mathcal{M}_0, x)}} \ \in \ \mathbb{G}$$

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

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

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 HVZK Proofs

We first briefly review the protocol `ZKPoKE` from [BBF19], which we will need repeeatedly in the subsequent protocols.

Protocol 2.1. *Zero knowledge proof of the knowledge of the Exponent* (`ZKPoKE`):
Parameters: $\mathbb{G} \xleftarrow{\$} \text{GGen}(\lambda)$, $g, h \in \mathbb{G}$ such that $\langle g \rangle = \langle h \rangle$.
Inputs: Elements $u, w \in \mathbb{G}$, $B > 2^{2\lambda}|\mathbb{G}|$
Claim: The Prover possesses an integer x such that $u^x = w$.

1. The Prover chooses random $k, \rho_x, \rho_k \in [-B, B]$ and sends (z, A_g, A_u) to the Verifier, where $z := g^x h^{\rho_x}$, $A_g := g^k h^{\rho_k}$, $A_u := u^k$.

2. The Verifier generates a random λ-bit integer c and a random λ-bit prime ℓ and sends them to the Prover.

3. The Prover computes the integers q_x, q_ρ, r_x, r_ρ such that

$$k + c \cdot \rho_k = q_x \cdot \ell + r_x \ , \ \rho_k + c \cdot \rho_x = q_\rho \cdot \ell + r_\rho$$

and sends $Q_g := g^{q_x} h^{q_\rho}$, $Q_u := u^{q_x}$ and r_x, r_ρ to the Verifier.

4. The Verifier accepts if and only if $r_x, r_\rho \in [\ell]$ and the equations $Q_g^\ell \cdot g^{r_x} h^{r_\rho} = A_g z^c$, $Q_u^\ell \cdot u^{r_x} = A_u w^c$ hold. □

The protocol ZKPoKE is an HVZK protocol PoKE is an argument of knowledge for the relation

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

Clearly, the relation $\mathcal{R}_{\texttt{PoKE}}$ 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 zero-knowledge proof of knowledge of the discrete logarithm between $a, b \in \mathbb{G}$ by $\texttt{ZKPoKE}[a,\ b]$.

In particular, if a, b are commitments

$$a = \texttt{Com}(g, \mathcal{M}) := g^{\Pi(\mathcal{M})} \ \ , \ \ a = \texttt{Com}(g, \mathcal{N}) := g^{\Pi(\mathcal{N})}$$

to sets/multisets $\mathcal{M}$, $\mathcal{N}$ with a common base $x \in \mathbb{G}$, the protocol demonstrates that $\mathcal{M} \subseteq \mathcal{N}$ without revealing anything about $\mathcal{M}$ or $\mathcal{N}$. The protocol can also be adapted to a setting where the commitments to $\mathcal{M}$, $\mathcal{N}$ are made perfectly hiding using a Pedersen commitment.

Suppose $a = g^d h^r$, $b = g^{\widetilde{d}} h^{\widetilde{r}}$ where the integers r, $\widetilde{r}$ are randomly selected. To demonstrate that d divides $\widetilde{d}$, the Prover and the Verifier may proceed as follows:

Protocol 2.2. *Zero knowledge proof of the knowledge of the Exponent for Pedersen commitments* :
Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda)$, $g, h \in \mathbb{G}$ such that $\langle g \rangle = \langle h \rangle$.
Inputs: Elements $a, b \in \mathbb{G}$
Claim: The Prover possesses integers d_1, d_2, r_1, r_2 such that

$$a = g^{d_1} h^{r_1} \ \wedge \ b = g^{d_2} h^{r_2} \ \wedge \ d_2 \equiv 0 \pmod{d_1}.$$

1. 1. The Prover $\mathcal{P}$ computes the integer $e := \widetilde{d} \cdot d^{-1}$ and sends $\widetilde{a} := a^e \in \mathbb{G}$ to the Verifier $\mathcal{V}$ along with a non-interactive proof for $\texttt{ZKPoKE}[a,\ \widetilde{a}]$.

2. $\mathcal{P}$ computes $\widetilde{h} := \widetilde{a}^{-1} \cdot b \in \mathbb{G}$ and sends a non-interactive proof for $\texttt{ZKPoKE}[h,\ \widetilde{h}]$ to $\mathcal{V}$.

3. $\mathcal{V}$ accepts if and only if both ZKPoKEs are valid. □

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. The protocol is useful whenever we need to compare two discrete logarithms with different bases. We provide an HVZK argument of knowledge for the following relation:

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

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 2.3. *Zero knowledge proof of equality of discrete logarithms* (`ZKPoEqDLog`):
Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda)$, $g, h \in \mathbb{G}$ such that $\langle g \rangle = \langle h \rangle$.
Inputs: Elements $a_1, a_2, b_1, b_2 \in \mathbb{G}$, $B > 2^{2\lambda}|\mathbb{G}|$
Claim: The Prover possesses an integer d such that $a_1^d = b_1$, $a_2^d = b_2$.

1. The Prover $\mathcal{P}$ chooses random integers $k, \rho_k, \rho_d \in [-B, B]$ and sends the group elements

$$\widetilde{g} := g^d h^{\rho_d} \ , \ A_g := g^k h^{\rho_k} \ , \ \widehat{a}_1 := a_1^k \ , \ \widehat{a}_2 := a_2^k \ \in \ \mathbb{G}$$

to the Verifier $\mathcal{V}$.

2. The hashing algorithm $\mathtt{H}_{\mathtt{FS},\lambda}$ generates λ-bit primes γ, ℓ.
3. $\mathcal{P}$ computes the integers q_d, r_d, q_ρ, r_ρ such that

$$d\gamma + k = q_d \cdot \ell + r_d \ \ , \ \ \rho_d \cdot \ell + \rho_k = q_\rho \cdot \ell + r_\rho \ \ , \ \ r_d, r_\rho \in [\ell]$$

and sends r_d, r_ρ to $\mathcal{V}$.

4. $\mathcal{P}$ computes

$$Q_g := g^{q_d} h^{q_\rho} \ , \ Q_1 := a_1^{q_d} \ , \ Q_2 := a_2^{q_d} \ \in \ \mathbb{G}$$

and sends Q_g, Q_1, Q_2 to $\mathcal{V}$.

5. $\mathcal{V}$ verifies that $r_d, r_\rho \in [\ell]$ and the equations

$$Q_g^\ell g^{r_d} h^{r_\rho} \stackrel{?}{=} \widetilde{g}^\gamma A_g \ \bigwedge \ Q_1^\ell a_1^{r_d} \stackrel{?}{=} \widehat{a}_1 b_1^\gamma \ \bigwedge \ Q_2^\ell a_2^{r_d} \stackrel{?}{=} \widehat{a}_2 b_2^\gamma.$$

He accepts if and only if all three equations hold. □

We provide an HVZK argument of knowledge for the relation

$$\mathcal{R}_{\mathtt{Prod}}[(a_1, b_1),\ (a_2, b_2),\ (a_3, b_3)] = \left\{ \begin{array}{l} \big((a_1, b_1), (a_2, b_2), (a_3, b_3) \in \mathbb{G}^2); \\ (d_1, d_2, d_3) \in \mathbb{Z}^3) : \\ b_i = a_i^{d_i} \ (i = 1, 2, 3) \ \bigwedge \ d_1 d_2 = d_3 \end{array} \right\}$$

Protocol 2.4. *Zero knowledge proof of product of discrete logarithms* (`ZKPoProd`):
Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda)$, $g, h \in \mathbb{G}$ such that $\langle g \rangle = \langle h \rangle$.
Inputs: Elements a, $b_1, b_2, b_3 \in \mathbb{G}$, $B > 2^{2\lambda}|\mathbb{G}|$
Claim: The Prover possesses integers d_i $(i = 1, 2, 3)$ such that $a_i^{d_i} = b_i$ and $d_1 d_2 = d_3$.

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 $\mathcal{V}$ along with non-interactive proofs for $\mathtt{ZKPoEqDLog}[(a_1, b_2),\ (a_2, b_{1,2})]$ and $\mathtt{ZKPoEqDLog}[(a_1, b_3),\ (a_3, b_{1,3})]$.

2. $\mathcal{P}$ generates a non-interactive proof for $\mathtt{ZKPoEqDLog}[(a, b_1),\ (b_{1,2}, b_{1,3})]$ and sends it to $\mathcal{V}$.
3. $\mathcal{V}$ accepts if an only if all three proofs are valid. □

We can also generalize the protocol ZKPoEqDLog as follows. For a public polynomial $f(X) \in \mathbb{Z}[X]$, an honest Prover can provide a constant-sized ZK 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 HVZK argument of knowledge for the relation

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

The non-ZK argument of knowledge for this relation ([TH20B]) is fairly simple. The Prover sends the remainders $r_i := d_i \pmod{\ell}$ for a randomly chosen λ-bit prime challenge ℓ. He verifiably shows that $r_i \equiv d_i \pmod{\ell}$ by sending the ℓ-th roots of $b_i a_i^{-r_i}$. The Verifier verifies the exponentiations and the congruence $r_2 \equiv f(r_1) \pmod{\ell}$.

The zero-knowledge variant is a bit more subtle since it requires a blinding factor. The Prover can no longer send over the remainders $r_i := d_i \pmod{\ell}$ since such a protocol is inherently *not* zero-knowledge. However, the protocol can be be made zero-knowledge with a modification that hinges on the observation for integers d_1, d_2, that the following are equivalent with overwhelming probability.

1. $d_2 = f(d_1)$
2. $f(d_1) \equiv d_2 \pmod{\gamma}$ for a randomly generated λ-bit prime γ.
3. $d_2 \equiv f(d + k\gamma) \pmod{k\gamma}$ for an integer k chosen by the Prover and a randomly generated λ-bit prime γ.

Protocol 2.5. *Zero knowledge proof of polynomial relation between discrete logarithms* (ZKPoPolyDLog):
Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda)$, $g, h \in \mathbb{G}$ such that $\langle g \rangle = \langle h \rangle$.
Inputs: Elements $a_1, a_2, b_1, b_2, \in \mathbb{G}$; a public univariate polynomial $f(X) \in \mathbb{Z}[X]$; $B > 2^{2\lambda}|\mathbb{G}|$
Claim: The Prover possesses integers d_i $(i = 1, 2)$ such that $a_i^{d_i} = b_i$ and $f(d_1) = d_2$.

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 $\texttt{ZKPoEqDLog}[(a_1, b_{1,2}),\ (a_2, b_2)]$.
2. $\mathcal{P}$ chooses a random $k \in [-B, B]$ and sends $A_1 := a_1^k$ to $\mathcal{V}$.
3. The hashing algorithm $\mathtt{H}_{\mathtt{FS},\lambda}$ generates a λ-bit prime γ.
4. $\mathcal{P}$ sends $A_2 := a_1^{f(d_1 + k\gamma)} \in \mathbb{G}$ to $\mathcal{V}$.
5. The hashing algorithm $\mathtt{H}_{\mathtt{FS},\lambda}$ generates a λ-bit prime ℓ.
6. $\mathcal{P}$ computes the integers q_1, r_1, q_2, r_2 such that

$$d_1 + k\gamma = q_1 \cdot \ell + r_1 \ , \ f(d_1 + k\gamma) = q_2 \cdot \ell + r_2 \quad , \quad r_1, r_2 \in [\ell]$$

and sends r_1, $Q_1 := a_1^{q_1}$, $Q_2 := a_1^{q_2}$ to $\mathcal{V}$.
7. $\mathcal{P}$ generates a non-interactive proof for $\texttt{ZKPoKE}[A_1^{\gamma},\ A_2 \cdot b_{1,2}^{-1}]$ and sends it to $\mathcal{V}$.
8. $\mathcal{P}$ chooses random $\rho_d, \rho_k \in [-B, B]$ and sends $\widetilde{g} := g^d h^{\rho_d}$, $A_g := g^k h^{\rho_k} \in \mathbb{G}$ to $\mathcal{V}$.
9. The hashing algorithm $\mathtt{H}_{\mathtt{FS},\lambda}$ generates λ-bit primes c, ℓ_0.
10. $\mathcal{P}$ computes the integers q_d, r_d, q_ρ, r_ρ such that

$$c \cdot d_1 + k = q_d \cdot \ell_0 + r_d \ , \ c \cdot \rho_d + \rho_k = q_\rho \cdot \ell_0 + r_\rho \quad , \quad r_\rho, r_d \in [\ell_0]$$

and sends r_d, r_ρ, $Q_g := g^{q_d} h^{q_\rho}$, $Q := a_1^{q_d}$ to $\mathcal{V}$.

11. $\mathcal{V}$ verifies that $r_d, r_\rho \in [\ell_0]$, $r_1 \in [\ell]$ and computes $r_2 := f(r_1) \pmod{\ell}$.
12. $\mathcal{V}$ verifies the equations

$$Q_1^{\ell} a_1^{r_1} \stackrel{?}{=} b_1 \cdot A_1^{\gamma} \bigwedge Q_2^{\ell} a_2^{r_2} \stackrel{?}{=} A_2 \bigwedge Q_g^{\ell_0} g^{r_d} h^{r_\rho} \stackrel{?}{=} A_1 \cdot \widetilde{g}^{c} \bigwedge Q^{\ell_0} a_1^{r_d} \stackrel{?}{=} A_1 b_1^{c}.$$

He accepts if and only if all four equations hold and the proof for $\mathtt{ZKPoKE}[A_1^{\gamma},\ A_2 \cdot b_{1,2}^{-1}]$ is valid. □

For instance, the special case of protocol ZKPoPolyDLog where $f(X) = X^n$ for some integer n can be used to show that for a tuple $(a_1, a_2, b_1, b_2) \in \mathbb{G}^4$, there exists a multiset $\mathcal{M}$ such that

$$a_1^{\Pi(\mathcal{M})} = b_1 \ , \ a_2^{\Pi(n \cdot \mathcal{M})} = b_2$$

without revealing anything about $\mathcal{M}$, where $n \cdot \mathcal{M} := \left\{ \left(n \cdot \mathrm{mult}(x, \mathcal{M})\right) \times x : \ x \in \mathtt{Set}(\mathcal{M}) \right\}$.

Proposition 2.6. *The protocol ZKPoPolyDLog is an HVZK argument of knowledge for the relation* $\mathcal{R}_{\mathtt{PolyDLog}}$.

Proof. □

2.1 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

$$\mathtt{Set}(\mathcal{M}) \subseteq \mathtt{Set}(\mathcal{N}) \Longleftrightarrow \exists\, N \in \mathbb{Z} \ : \ \Pi(\mathcal{M})^N \equiv 0 \pmod{\Pi(\mathcal{N})}.$$

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

$$p \notin \{-1, 1\} \bigwedge \Pi(\mathcal{M}) \equiv 0 \pmod{p} \bigwedge \mathbf{gcd}(\Pi(\mathcal{N}), p) = 1.$$

Protocol 2.7. *ZK Proof of containment of underlying sets* (ZKPoConSets).

Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda)$, $g, h \in \mathbb{G}$ such that $\langle g \rangle = \langle h \rangle$.

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 $N := \max\{\text{mult}(\mathcal{N}, x) : x \in \mathcal{N}\}$ and

$$A_3 := a_1^{\Pi(\mathcal{M})^N} \in \mathbb{G}.$$

He sends A_3 and N to the Verifier $\mathcal{V}$.

2. $\mathcal{P}$ generates a non-interactive proof for $\texttt{ZKPoPolyDLog}[(a_1, A_1),\ (a_2, A_3),\ X^N]$ and sends it to $\mathcal{V}$.

3. $\mathcal{P}$ generates a non-interactive proof for $\texttt{ZKPoKE}[A_2, A_3]$ and sends it to $\mathcal{V}$.

4. $\mathcal{V}$ verifies the two proofs and accepts if and only if both are valid. □

Protocol 2.8. *ZK Protocol for the non-containment of underlying sets* (ZKPoNonConSets).
Parameters: $\mathbb{G} \xleftarrow{\$} \text{GGen}(\lambda),\ g \in \mathbb{G}$.
Input: Elements $a_1, a_2 \in \mathbb{G}$; commitments $A_1 := \texttt{Com}(a_1, \mathcal{M}) = a_1^{\Pi(\mathcal{M})}$, $A_2 := \texttt{Com}(a_2, \mathcal{N}) = a_2^{\Pi(\mathcal{N})}$ to multisets $\mathcal{M}$, $\mathcal{N}$.
Claim: $\texttt{Set}(\mathcal{M}) \nsubseteq \texttt{Set}(\mathcal{N})$.

1. The Prover chooses an integer $p \in \texttt{Set}(\mathcal{M}) \setminus \texttt{Set}(\mathcal{N})$. and computes $b_1 := a_1^p$. He sends b_1 to the Verifier $\mathcal{V}$ along with a non-interactive proof for $\texttt{ZKPoKE}[b_1, A_1]$.

2. $\mathcal{P}$ generates a non-interactive proof for $\texttt{ZKPoRelPrimeDLog}[(a_1, b_1),\ (a_2, A_2)]$ and sends it to $\mathcal{V}$.

3. $\mathcal{V}$ checks that $b_1 \notin \{a_1, a_1^{-1}\}$ and verifies the proofs for $\texttt{ZKPoRelPrimeDLog}[(a_1, b_1),\ (a_2, A_2)]$ and $\texttt{ZKPoKE}[b_1, A_1]$. He accepts if and only if both proofs are valid. □

3 HVZK arguments for set operations

The goal of this section is to provide a protocol 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 HVZK 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 3.1. *Zero knowledge proof of the greatest common divisor* (ZKPoGCD):
Parameters: $\mathbb{G} \xleftarrow{\$} \text{GGen}(\lambda)$, $g, h \in \mathbb{G}$ such that $\langle g \rangle = \langle h \rangle$.
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 $\texttt{ZKPoEqDLog}[(a_2, b_2),\ (a_1, b_{1,2})]$, $\texttt{ZKPoEqDLog}[(a_3, b_3),\ (a_1, b_{1,3})]$ and sends them to $\mathcal{V}$.

3. $\mathcal{P}$ generates non-interactive proofs for $\texttt{ZKPoKE}[b_{1,3},\ b_1]$ and $\texttt{ZKPoKE}[b_{1,3},\ b_{1,2}]$ and sends them to $\mathcal{V}$.
3. $\mathcal{P}$ uses the Euclidean 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 $\texttt{ZKPoKE}[b_1,\ \widetilde{b}_1]$ and $\texttt{ZKPoKE}[b_{1,2},\ \widetilde{b}_{1,2}]$.
6. $\mathcal{V}$ verifies all of the proofs he receives in addition to the equation

$$\widetilde{b}_1 \cdot \widetilde{b}_{1,2} \stackrel{?}{=} b_{1,3} \ \in \ \mathbb{G}.$$

He accepts the validity of the claim if and only if all of these proofs are valid. □

Theorem 3.2. *The Protocol* $\texttt{ZKPoGCD}$ *is an HVZK argument of knowledge for the relation* $\mathcal{R}_{\texttt{GCD}}$ *in the generic group model.*

Proof. □

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 $\texttt{RelPrimeDLog}$ 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 3.3. *ZK Proof of Relatively Prime Discrete Logarithms* ($\texttt{ZKPoRelPrimeDLog}$):
Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda)$, $g, h \in \mathbb{G}$ such that $\langle g \rangle = \langle h \rangle$.
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 $\texttt{ZKPoEqDLog}[(a_2, b_2),\ (a_1, b_{1,2})]$.
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 $\texttt{ZKPoKE}[b_1,\ \widetilde{b}_1]$ and $\texttt{ZKPoKE}[b_{1,2},\ \widetilde{b}_{1,2}]$.
4. $\mathcal{V}$ verifies the equation $\widetilde{b}_1 \cdot \widetilde{b}_{1,2} \stackrel{?}{=} a_1 \in \mathbb{G}$ and the proofs for $\texttt{ZKPoEqDLog}[(a_2, b_2),\ (a_1, b_{1,2})]$, $\texttt{ZKPoKE}[b_1,\ \widetilde{b}_1]$ and $\texttt{ZKPoKE}[b_{1,2},\ \widetilde{b}_{1,2}]$. He accepts the validity of the claim if and only if all of these proofs are valid. □

Proposition 3.4. *The Protocol* $\texttt{ZKPoRelPrimeDLog}$ *is a HVZK argument for the relation* $\mathcal{R}_{\texttt{RelPrimeDLog}}$ *in the generic group model.*

Proof. This is a special case of Proposition 3.2. □

The protocol ZKPoGCD can also be adapted for Pedersen commitments to sets or multisets. Suppose $a_i := g^{d_i} h^{r_i} \in \mathbb{G}$ $(i = 1, 2, 3)$ where the d_i are integers such that $\mathbf{gcd}(d_1, d_2) = d_3$ and the r_i are randomly selected integers. The Prover can demonstrate that $\mathbf{gcd}(d_1, d_2) = d_3$ as follows, without revealing anything about the integer d_i.

Protocol 3.5. *Zero knowledge proof of the greatest common divisor for Pedersen commitments* (ZKPoGCD):
Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda)$, $g, h \in \mathbb{G}$ such that $\langle g \rangle = \langle h \rangle$.
Input: Elements $a_1, a_2, a_3 \in \mathbb{G}$.
Claim: The Prover possesses integers d_i, r_i $(i = 1, 2, 3)$ such that:
- $a_i = g^{d_i} h^{r_i}$
- $\mathbf{gcd}(d_1, d_2) = d_3$

1. $\mathcal{P}$ computes the integers $d_{1,3} := d_1 \cdot d_3^{-1}$, $d_{2,3} := d_2 \cdot d_3^{-1}$ and sends $a_{1,3} := a_3^{d_{1,3}}$, $a_{1,3} := a_3^{d_{1,3}} \in \mathbb{G}$ to $\mathcal{V}$ along with non-interactive proofs for ZKPoKE$[a_3,\ a_{1,3}]$, ZKPoKE$[a_3,\ a_{2,3}]$.

2. $\mathcal{P}$ computes
$$h_{1,3} := h^{r_1 - d_{1,3} r_3} \ ,\ h_{2,3} := h^{r_2 - d_{2,3} r_3} \ \in \ \mathbb{G}$$
and sends non-interactive proofs for ZKPoKE$[h,\ h_{1,3}]$, ZKPoKE$[h,\ h_{2,3}]$ to $\mathcal{V}$.

3. $\mathcal{P}$ computes integers e_1, e_2 such that $e_1 d_1 + e_2 d_2 = d_3$.

4. $\mathcal{P}$ computes
$$\widetilde{a}_1 := a_1^{e_1} \ ,\ \widetilde{a}_2 := a_2^{e_2} \ \in \ \mathbb{G}$$
and sends them to $\mathcal{V}$ along with non-interactive proofs for ZKPoKE$[a_1, \widetilde{a}_1]$, ZKPoKE$[a_2, \widetilde{a}_2]$.

5. $\mathcal{P}$ computes $\widetilde{h} := h^{e_1 r_1 + e_2 r_2 - r_3} \in \mathbb{G}$ and sends it to $\mathcal{V}$ along with a non-interactive proof for ZKPoKE$[h,\ \widetilde{h}]$.

5. $\mathcal{V}$ verifies the five ZKPoKEs he receives in addition to the equations $\widetilde{a}_1 \cdot \widetilde{a}_2 \stackrel{?}{=} a_3$. He accepts if and only if the equaiton holds and all five ZKPoKEs are valid. □

It is easy to see that the protocols ZKPoGCD may be combined with the protcol ZKPoProd to provide an HVZK argument of knowledge for the relation

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

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

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

Protocol 3.6. *Zero knowledge proof of the least common multiple* (ZKPoLCM):
Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda)$, $g, h \in \mathbb{G}$ such that $\langle g \rangle = \langle h \rangle$.
Input: Elements $a, b_1, b_2, b_3 \in \mathbb{G}$.
Claim: The Prover possesses integers d_1, d_2, d_3 such that:
- $a^{d_1} = b_1$, $a^{d_2} = b_2$, $a^{d_3} = b_3$
- $\mathbf{lcm}(d_1, d_2) = d_3$

1. The Prover $\mathcal{P}$ computes $b_{\texttt{gcd}} := a^{\mathbf{gcd}(d_1,d_2)} \in \mathbb{G}$ and sends $b_{\texttt{gcd}}$ to the Verifier $\mathcal{V}$ along with a non-interactive proof for $\texttt{ZKPoGCD}[(a,b_1),\ (a,b_2),\ (a,b_{\texttt{gcd}})]$.

2. $\mathcal{P}$ computes $b_{\texttt{prod}} := a^{d_1 d_2} \in \mathbb{G}$ and sends it to $\mathcal{V}$ along with a non-interactive proof for $\texttt{ZKPoProd}[(a,b_1),(a,b_2),(a,b_{\texttt{prod}})]$.

3. $\mathcal{P}$ generates a non-interactive proof for $\texttt{ZKPoProd}[(a,b_{\texttt{gcd}}),\ (a,b_3),\ (a,b_{\texttt{prod}})]$ and sends it to $\mathcal{V}$.

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

The HVZK arguments of knowledge for the GCD and the product can be combined to get an HVZK argument of knowledge for the difference of two sets or multisets.

Protocol 3.7. *ZK Protocol for multiset differences* (ZKPoDiff)
Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda)$, $g, h \in \mathbb{G}$ such that $\langle g \rangle = \langle h \rangle$.
Input: Elements $a, b_1, b_2, b_3 \in \mathbb{G}$.
Claim: The Prover possesses integers d_1, d_2, d_3 such that:
- $a^{d_1} = b_1$, $a^{d_2} = b_2$, $a^{d_3} = b_3$
- $d_1 = d_3 \cdot \mathbf{gcd}(d_1, d_2)$

1. The Prover $\mathcal{P}$ computes $b_{\texttt{gcd}} := a^{\mathbf{gcd}(d_1,d_2)} \in \mathbb{G}$ and sends to the Verifier $\mathcal{V}$ along with a non-interactive proof for $\texttt{ZKPoGCD}[(a,b_1),\ (a,b_2),\ (a,b_{\texttt{gcd}})]$.

2. $\mathcal{P}$ generates a non-interactive proof for $\texttt{ZKPoProd}[(a,b_{\texttt{gcd}}),\ (a,b_3),\ (a,b_1)]$ and sends it to $\mathcal{V}$.

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

4 HVZK for sets for hiding commitments

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.

[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

[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.

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

[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.

[Th20A] Steve Thakur, *Constructing hidden order groups using genus three Jacobians*, Preprint

[Th20B] Steve Thakur, *Arguments of Knowledge in hidden order groups*, Preprint

[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 Protocols:

The following is a list of the protocols in this paper and the relations that the protocols are HVZK 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. `ZKPoEqDLog` (*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. `ZKPoPolyDLog` (*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. `ZKPoProd` (*Proof of Product*)

$$\mathcal{R}_{\texttt{Prod}}[(a_1,b_1),\ (a_2,b_2),(a_3,b_3)] = \{((a_i,b_i \in \mathbb{G});\ d_i \in \mathbb{Z})\ :\ b_i = a_i^{d_i},\ d_1d_2 = d_3\}$$

4. `ZKPoGCD` (*Proof of GCD*)

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

5. `ZKPoRelPrimeDLog` (*Proof of relatively prime discrete logarithms*; special case of `PoGCD`)

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

6. `ZKPoLCM` (*Proof of LCM*)

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

7. `ZKPoRelPrimeDLog` (*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. *Zero-knowledge proof of the containment/non-containment of the underlying sets*

$$\mathcal{R}_{\texttt{ConSets}}[(a_1,A_1),\ (a_2,A_2)] = \left\{ \begin{array}{l} (a_1,a_2,A_1,A_2\ \in\ \mathbb{G}; \\ (d_1,d_2,N) \in \mathbb{Z}^3) : \\ a_1^{d_1} = A_1\ \wedge\ a_2^{d_2} = A_2 \\ \wedge\ d_2^N \equiv 0 \pmod{d_1} \end{array} \right\}$$

$$\mathcal{R}_{\texttt{NonConSets}}[(a_1,A_1),\ (a_2,A_2)] = \left\{ \begin{array}{l} (a_1,a_2,A_1,A_2\ \in\ \mathbb{G}; \\ (d_1,d_2,p) \in \mathbb{Z}^3) : \\ a_1^{d_1} = A_1\ \wedge\ a_2^{d_2} = A_2\ \wedge \\ p\big|d_2\ \wedge\ p\big|d_1\ \wedge\ p \neq \pm 1 \end{array} \right\}$$

B Pedersen Commitments

Protocol B.1. *Zero knowledge proof of polynomial relation between discrete logarithms for Pedersen commitments*(`ZKPoPolyDLog`):

Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda)$, $g, h \in \mathbb{G}$ such that $\langle g \rangle = \langle h \rangle$.
Inputs: Elements $a, b \in \mathbb{G}$; a public univariate polynomial $\mathbf{f}(X) \in \mathbb{Z}[X]$; $B > 2^{2\lambda}|\mathbb{G}|$
Claim: The Prover possesses integers d, e, $\widehat{e}$ $(i = 1, 2)$ such that $a = g^d h^e$, $b = g^{\mathbf{f}(d)} h^{\widehat{e}}$.

1. The Prover $\mathcal{P}$ chooses random $k, e_1 \in [-B, B]$ and sends $A_1 := g^k h^{e_1} \in \mathbb{G}$ to the Verifier $\mathcal{V}$.
2. The hashing algorithm $\mathtt{H}_{\mathtt{FS},\lambda}$ generates a λ-bit prime γ.
3. $\mathcal{P}$ chooses a random $e_2 \in [-B, B]$ and sends $A_2 := g^{\mathbf{f}(d_1 + k\gamma)} \cdot h^{e_2} \in \mathbb{G}$ to $\mathcal{V}$.
4. $\mathcal{P}$ computes
$$\widetilde{e} := e_2 - \widehat{e} \pmod{e_1 \cdot \gamma} \ , \quad \widetilde{h} := h^{\widehat{e}} \in \mathbb{G}$$
and sends $\widetilde{h}$ to $\mathcal{V}$ along with a non-interactive proof for $\mathtt{ZKPoKE}[h,\ \widetilde{h}]$ and $\mathtt{ZKPoKE}[A_1^{\gamma},\ A_2 \cdot b^{-1} \cdot \widetilde{h}^{-1}]$.
5. The hashing algorithm $\mathtt{H}_{\mathtt{FS},\lambda}$ generates a λ-bit prime ℓ.
6. $\mathcal{P}$ computes the integers q_1, r_1, q_2, r_2, q'_1, s_1, q'_2, s_2 such that $r_1, r_2, s_1, s_2 \in [\ell]$ and
$$d + k\gamma = q_1 \cdot \ell + r_1 \ , \ \mathbf{f}(d + k\gamma) = q_2 \cdot \ell + r_2 \ , \ \ e_1 = q'_1 \cdot \ell + s_1 \ , \ \ e_2 = q'_2 \cdot \ell + s_2$$
and sends $r_1, s_1, s_2 \in [\ell]^3$, $Q_1 := g^{q_1} h^{q'_1}$, $Q_2 := g^{q_2} h^{q'_2} \in \mathbb{G}$ to $\mathcal{V}$.
7. $\mathcal{V}$ verifies that $r_1, s_1, s_2 \in [\ell]^3$ and computes $r_r := \mathbf{f}(r_1) \pmod{\ell}$.
8. $\mathcal{V}$ verifies the two $\mathtt{ZKPoKE}$s he receives and verifies the equations
$$Q_1^{\ell} \cdot g^{r_1} \cdot h^{s_1} \stackrel{?}{=} a \cdot A_1^{\gamma} \ \ \bigwedge \ \ Q_2^{\ell} \cdot g^{r_2} \cdot h^{s_2} \stackrel{?}{=} A_2.$$

He accepts if and only if all equations hol and the two $\mathtt{ZKPoKE}$s are valid. $\square$