Arguments of Knowledge in hidden order groups

Steve Thakur

Abstract

In this paper, we study non-interactive arguments of knowledge (AoKs) in groups of hidden order. We provide protocols to aggregate the knowledge of the discrete logarithms between multiple elements and to prove certain relations between multiple discrete logarithms. In particular, we provide AoKs for disjointness of sets in cryptographic accumulators.

Recent work ([DGS20]) suggests that the hidden order groups need to be substantially larger in size that previously thought, in order to ensure 128-bit security. Thus, 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 exponents. We build on the techniques from [BBF19].

1 Introduction

To be written

1.1 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 usually mitigated by a 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 a candidate 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 adaptive root assumption.

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 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 $\mathrm{Jac}(C)(\mathbb{F}_p)$ is roughly of size p^3.

1.2 Cryptographic assumptions

Definition 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* $\mathcal{A}_1(l, \mathrm{state})$ *outputs* $w^{1/l} \in \mathbb{G}$.

Definition 1.2. *We say* $\mathbb{G}$ *satisfies the* ***strong-RSA*** *assumption if no PPT algorithm* $\mathcal{A}$ *is able to compute (except with negligible probability) the* l*-th root of a chosen element* $w \in \mathbb{G}$ *for some randomly chosen prime* l.

Definition 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* $n < 2^{\mathrm{poly}(\lambda)}$ *such that* $a^n = 1$.

Definition 1.4. *We say* $\mathbb{G}$ *satisfies the* ***fractional root assumption*** *if for a randomly generated element* $g \in \mathbb{G}$*, a PPT algorithm cannot output* $h \in \mathbb{G}$ *and* $d_1, d_2 \in \mathbb{Z}$ *such that*

$$g^{d_1} = h^{d_2} \;\wedge\; d_2 \nmid d_1$$

except with negligible probability.

The assumptions bear the following relations:

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

$$\{\text{Strong-RSA assumption}\} \implies \{\text{Fractional root assumption}\}.$$

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

1.3 Some background

Shamir's trick: Given elements $a_1, a_2, A \in \mathbb{G}$ and integers $d_1, d_2 \geq 1$ such that $a^{d_1} = a_2^{d_2} = A$, 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 of this algorithm is $\mathbf{O}(n \log(n))$.

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 allows us to compute elements $a_1, \cdots, a_n$ such that

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

in runtime $\mathbf{O}(\log(D) \log(\log(D)))$. Naively, this would take runtime $\mathbf{O}(\log^2(D))$.

1.4 Cryptographic Accumulators

A cryptographic accumulator [Bd94] is a primitive that produces a short binding commitment to a set 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: Merkle trees, pairing-based (aka bilinear) accumulators and accumulators based on groups of unknown order.

Let $\mathbb{G}$ be a group of hidden order and fix an element $g \in \mathbb{G}$. For a data set $\mathcal{D}$ represented by distinct λ-bit primes, the **accumulated digest** is given by

$$\mathbf{Acc}(\mathcal{D}) := g^{\prod_{d \in \mathcal{D}} d}.$$

For a data set $\mathcal{D}_{\prime} \subseteq \mathcal{D}$, the witness

$$w(\mathcal{D}_0) := g^{\prod_{d \notin \mathcal{D}_0}}$$

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

$$w(\mathcal{D}_0)^{\prod_{d \in \mathcal{D}_0}} \stackrel{?}{=} \mathbf{Acc}(\mathcal{D}).$$

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

Shamir's trick allows for aggregation of membership witnesses in accumulators based on hidden order groups. This is not possibe with Merkle trees. With bilinear accumulators, aggregation of membership witnesses has a linear runtime complexity, which is impractical for most use cases.

1.5 $\mathbb{Z}_{(\lambda)}$-integers and the equivalence relation $(\equiv_\lambda)$

This subsection can be skipped for the time being. The protocols have been modified so that they do not involve this equivalence relation.

Definition 1.5. *For elements $a, b \in \mathbb{G}$ and a rational $\alpha \in \mathbb{Q}$, we say $a^\alpha = \beta$ with respect to a Prover $\mathcal{P}$ if $\mathcal{P}$ verifiably possesses integers $d_1, d_2 \in \mathbb{Z}$ such that $\alpha = \frac{d_1}{d_2}$ and $a^{d_1} = b^{d_2}$.*

Note that if there exists an element $a \in \mathbb{G}$ and distinct rationals $\frac{d_1}{d_2}, \frac{d_3}{d_4}$ ($d_i \in \mathbb{Z}$) such that

$$a^{\frac{d_1}{d_2}} = a^{\frac{d_3}{d_4}},$$

then $a^{d_1 d_4 - d_2 d_3} = 1$ and $d_1 d_4 - d_2 d_3 \neq 0$. So the adaptive root assumption implies that a PPT algorithm cannot generate such a tuple (a, d_1, d_2, d_3, d_4) except with negligible probability. Furthermore, by Shamir's trick, the condition is equivalent to the Prover $\mathcal{P}$ being able to compute an element $a_0 \in \mathbb{G}$ and co-prime integers d_1, d_2 such that

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

Definition 1.6. *An integer is said to be λ-**smooth** is all of its prime divisors are $\leq 2^\lambda$. An integer is said to be λ-**rough** is all of its prime divisors are $> 2^\lambda$.*

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 1.7. *For a security parameter λ, we denote by $\mathbb{Z}_{(\lambda)}$ the integral domain obtained by localizing $\mathbb{Z}$ away from all primes $\geq 2^\lambda$.*

Thus,

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

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

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

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

Definition 1.8. *For $\mathbb{Z}_{(\lambda)}$-integers d_1, d_2 we say $d_1 =_{lam} d_2$ if $\frac{d_1}{d_2}$ is a unit in $\mathbb{Z}_{(\lambda)}$.*

This is clearly a homomorphic equivalence relation.

Definition 1.9. *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 $\mathbb{Z}_{(\lambda)}$.*

Let $d_1, \cdots, d_n$ be $\mathbb{Z}_{(\lambda)}$-integers and write $d_i = \widetilde{d}_i \frac{\alpha_i}{\beta_i}$ with $\widetilde{d}_i$ a λ-rough integer and α_i, β_i co-prime λ-smooth integers. Clearly, for each pair i, j, we have the equivalence

$$\gcd(\widetilde{d}_i, \widetilde{d}_j) = 1 \Longleftrightarrow d_i, d_j \text{ co-prime in } \mathbb{Z}_{(\lambda)}.$$

Definition 1.10. *For elements a, b in a hidden order group $\mathbb{G}$, we say*

$$a \equiv_\lambda b$$

with respect to a Prover $\mathcal{P}$ if $\mathcal{P}$ verifiably possesses relatively prime λ-smooth integers d_1, d_2 such that $a^{d_1} = b^{d_2}$.

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 1.1. *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 $e_1 := d_1 d_3 / d, d_2 d_4 / d$. 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, $\frac{d_1}{d_2} \in \mathbb{Z}_{(\lambda)}^\times$. 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,

$$a \equiv_\lambda b \Longleftrightarrow a^d \equiv_\lambda b^d.$$

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 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 $A = g^{\prod_{d \in \mathcal{D}}}$ the accumulated digest. For brevity, we write $D := \prod_{d \in \mathcal{D}} d$. Given a data set $\mathcal{D}_0$ disjoint with $\mathcal{D}$

and the product $D_0 := \prod_{d \in \mathcal{D}_0} d$, 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 $w^{D_0} A_1 = g$.
- A non-interactive proof for $\texttt{PoKE}[A,\ A_1]$.

Suppose, instead of $\texttt{PoKE}[A,\ A_1]$, 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
$$xD_0 + \frac{kD}{e} = 1$$
and hence,
$$exD_0 + kD = e.$$
Thus, $\mathbf{gcd}(D_0, D)$ 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.

1.6 Some preliminary lemmas

We will need the next two lemmas repeatedly in the subsequent protocols.

Lemma 1.2. *Let p be a prime and let $f(X)$ be a univariate degree n polynomial in $\mathbb{Z}[X]$ such that not all coefficients are divisible by p. For a randomly generated integer γ, the probability that $f(\gamma) \equiv 0 \pmod{p^{n\lambda}}$ is* $\mathrm{negl}(\lambda)$.

Proof. Let F be a splitting field of $f(X)$ 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, $x \in \bigcap_{i=1}^{g} \mathfrak{p}_i^{ek} = p^k\mathbb{Z}$.

For any two integers $x_1, x_2 \in \mathbb{Z}$, we have
$$x_1 - x_2 \in \mathfrak{p}_1^{e\lambda} \iff x_1 - x_2 \in \mathfrak{p}_1^{e\lambda} \cap \mathbb{Z} = p^\lambda\mathbb{Z}.$$
Hence, the set
$$S_\lambda := \{x + \mathfrak{p}_1^{e\lambda} :\ x \in \mathbb{Z}\} \subseteq \mathcal{O}_F/\mathfrak{p}^{e\lambda}$$
has cardinality p^λ. Now, for any integer γ,
$$f(\gamma) \equiv 0 \pmod{p^{n\lambda}} \iff f(\gamma) \equiv 0 \pmod{\mathfrak{p}_1^{en\lambda}} \implies \gamma \equiv \alpha_i \pmod{\mathfrak{p}_1^{e\lambda}} \text{ for at least one index } i.$$
Since γ is randomly generated, $\gamma \pmod{\mathfrak{p}_1^{e\lambda}}$ is randomly and uniformly distributed over the set S_λ. Hence,
$$\mathbf{Prob}\big(f(\gamma) \equiv 0 \pmod{p^{n\lambda}}\big) \leq \frac{n}{p^\lambda} = \mathrm{negl}(\lambda),$$
which completes the proof. □

Lemma 1.3. **1**. *For rationals $d_1, \cdots, d_n \in \mathbb{Q}$ and a randomly generated λ-bit integer γ, if*

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

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

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

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

then with overwhelming probability, $d_1, \cdots, d_n \in \mathbb{Z}$.

Proof. **1**. Let D be the least common denominator for $d_1, \cdots, d_n$ and write $d_i = \frac{c_i}{D}$ for $i = 1, \cdots, n$. Suppose, by way of contradiction that $(d_1, \cdots, d_n) \notin \mathbb{Z}^n_{(\lambda)}$. Then D is divisible by some prime $p > 2^\lambda$ and

$$\sum_{i=1}^{n} c_i \gamma^i \equiv 0 \pmod{p}.$$

Now, the polynomial $\sum\limits_{i=1}^{n} c_i X^i \in \mathbb{F}_p[X]$ has at most n distinct zeros in $\mathbb{F}_p$ and since γ is uniformly distributed modulo p, the probability of this occurring is $\mathrm{negl}(\lambda)$, a contradiction.

2. Let D be the least common denominator for $d_1, \cdots, d_n$ and write $d_i = \frac{c_i}{D}$ for $i = 1, \cdots, n$. Suppose, by way of contradiction that $(d_1^{n\lambda}, \cdots, d_n^{n\lambda}) \notin \mathbb{Z}^n$ and let p be a prime dividing D. Then

$$\sum_{i=1}^{n} c_i^{n\lambda} \gamma^i \equiv 0 \pmod{p^{n\lambda}}.$$

Now, the polynomial $f(X) := \sum\limits_{i=1}^{n} c_i^{n\lambda} X^i$ has degree n and by the preceding lemma,

$$\mathbf{Prob}\big(h(\gamma) \equiv 0 \pmod{p^{n\lambda}}\big) = \mathrm{negl}(\lambda).$$

Thus, with overwhelming probability, the rationals $d_i^{n\lambda}$ are integers, which in turn implies that the d_i are integers. □

In particular,

$$\mathbf{Prob}\Big((d_1, \cdots, d_n) \notin \mathbb{Z}^n_{(\lambda)} \;\Big|\; \sum_{i=1}^{n} d_i \gamma^i \in \mathbb{Z}\Big) = \mathrm{negl}(\lambda).$$

In a setting where the Verifier is not satisfied with the elements $d_1, \cdots, d_n$ being $\mathbb{Z}_{(\lambda)}$-integers and needs a probabilistic proof that they are, in fact, rational integers, the Prover could demonstrate that $\sum\limits_{i=1}^{n} d_i^{n\lambda} \gamma^i \in \mathbb{Z}$. The resulting trade-off is a higher computational burden for the Prover. Computing

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

entails

$$\mathbf{O}\big(\log(\sum_{i=1}^{n} d_i^{n\lambda} \gamma^i)\big) = \mathbf{O}\big(n \log(n) \lambda \max\{\log(d_i)\}\big)$$

exponentiations in $\mathbb{G}$. On the other hand, computing $g^{\sum_{i=1}^{n} d_i \gamma^i}$ entails

$$\mathbf{O}\big(\log(\sum_{i=1}^{n} d_i\gamma^i)\big) = \mathbf{O}\big(\log(n)\max\{\log(d_i)\}\big)$$

group exponentiations.

Given a randomly generated element $g \in \mathbb{G}$, if the Prover outputs an element $\widetilde{g} = g^{\sum_{i=1}^{n} d_i^{n\lambda}\gamma^i}$, then the fractional root assumption implies that $\sum_{i=1}^{n} d_i^{n\lambda}\gamma^i \in \mathbb{Z}$ except with negligible probability. The lemma 1.4 then implies that with overwhelming probability, $(d_1, \cdots, d_n) \in \mathbb{Z}^n$.

2 Arguments of Knowledge

We briefly review the protocol $\texttt{PoKE}$ from [BBF19].

Protocol 2.1. *Proof of Knowledge of the Exponent* ($\texttt{PoKE}$)

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

Step. 1. The Prover $\mathcal{P}$ computes $z := g^x$ and sends it to the Verifier $\mathcal{V}$.
2. The Fiat-Shamir heuristic generates a λ-bit prime l.
3. $\mathcal{P}$ computes the integers q, r such that

$$x = ql + r,\ \ r \in [l].$$

4. $\mathcal{P}$ computes $Q := u^q,\ Q' = g^q$ and sends (Q, Q', g^x, r) to $\mathcal{V}$.
5. $\mathcal{V}$ accepts if and only if

$$r \in [l],\ \ Q^l u^r w,\ \ Q'^l g^r = z.$$

□

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 be the case.

Clearly, the relation *Knowledge of the Exponent* 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 $\texttt{PoKE}[a, b]$.

In the following protocol, we show how a Prover could probabilistically demonstrate that two discrete logarithms are equal without revealing anything about the common discrete logarithm other than residues modulo a prime challenge. We provide an argument of knowledge for the following relation:

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

Protocol 2.2. *Proof of equality of discrete logarithms* ($\texttt{PoEqDLog}$) :

Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda),\ g \in \mathbb{G}$.
Inputs: $a_1, a_2, b_1, b_2 \in \mathbb{G}$.
Claim: The Prover possesses an integer d such that $a_1^d = b_1$ and $a_2^d = b_2$.

Step. 1. The Prover $\mathcal{P}$ sends $\widetilde{g} := g^d$ to the Verifier $\mathcal{V}$.
2. The Fiat-Shamir heuristic generates a λ-bit prime l.
3. $\mathcal{P}$ computes the integers q, r such that $d = ql + r$, $r \in [l]$ and the group elements

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

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

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

He accepts if and only if all equations hold. □

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

Proposition 2.3. *The protocol* $\texttt{EqDLog}[(a_1, b_1),\ (a_2, b_2)]$ *is an argument of knowledge for the relation* $\mathcal{R}_{\texttt{EqDLog}}$ *in the generic group model.*

Proof. Since the protocol `PoKE` is secure ([BBF19]), the validity of the equations

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

proves that $\mathcal{P}$ possesses the discrete logarithms between a_1, b_1 and a_2, b_2. Suppose, by way of contradiction, that these discrete logarithms are distinct and denote them by d_1, d_2 respectively. The adaptive root assumption implies that with overwhelming probability,

$$d_1 \equiv r \equiv d_2 \pmod{l}.$$

But since the λ-bit prime l is randomly generated, the integer $d_1 - d_2$ is randomly and uniformly distributed modulo l and hence,

$$\mathbf{Prob}\big(d_1 \equiv d_2 \pmod{l} \,\big|\, d_1 \neq d_2\big) = \frac{1}{l} = \mathrm{negl}(\lambda),$$

a contradiction. □

We now show how to generalize the last protocol for multiple discrete logarithms while keeping the communication complexity constant-sized and independent of the number of discrete logarithms. We call the following protocol the *Aggregated Equality of Discrete Logarithms* or `AggEqDLog` for short. We provide an argument of knowledge for the following relation:

$$\mathcal{R}_{\texttt{AggEqDLog}}[(a, b),\ (\mathcal{A},\ \mathcal{B})] = \left\{ \begin{array}{l} (\mathcal{A} = (a_1, \cdots, a_n),\ \mathcal{B} = (b_1, \cdots, b_n) \in \mathbb{G}^n); \\ d \in \mathbb{Z}) : \\ b_i = a_1^d \ \forall\ i \end{array} \right\}$$

Protocol 2.4. *Proof of Aggregated Equal Discrete Logarithms* (`PoAggEqDLog`) :
Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda),\ g \in \mathbb{G}$.
Inputs : $a, b \in \mathbb{G}$, $(a_1, \cdots, a_n) \in \mathbb{G}^n$, $(b_1, \cdots, b_n) \in \mathbb{G}^n$.
Claim: The Prover possesses an integer d such that $a^d = b$ and $a_i^d = b_i$ for $i = 1, \cdots, n$.

Step. 1. The Fiat-Shamir heuristic generates a λ-bit integer γ.
2. The Prover $\mathcal{P}$ computes the elements

$$\widetilde{a} := \prod_{i=1}^{n} a_i^{\gamma^i}, \ \ \widetilde{b} := \prod_{i=1}^{n} b_i^{\gamma^i} \quad \in \mathbb{G}.$$

3. $\mathcal{P}$ generates a non-interactive proof for $\mathtt{EqDLog}[(a,b),\ (\widetilde{a},\widetilde{b})]$ and sends it to the Verifier $\mathcal{V}$.
4. $\mathcal{V}$ independently computes the elements $\widetilde{a},\widetilde{b}$ and accepts if and only if the proof for $\mathtt{EqDLog}[(a,b),\ (\widetilde{a},\widetilde{b})]$ is valid. □

Thus, the proof consists of four $\mathbb{G}$-elements and one λ-bit integer. In particular, it is constant-sized and independent of the cardinalities $|\mathcal{A}|$, $|\mathcal{B}|$.

Proposition 2.5. *The protocol* $\mathtt{AggEqDLog}$ *is an argument of knowledge for the relation* $\mathcal{R}_{\mathtt{AggEqDLog}}$ *in the generic group model.*

Proof. (Sketch) With notations as in in the Protocol, $\mathcal{V}$ accepts if and only if $\mathcal{P}$ proves possession of an integer d such that

$$a^d = b, \ \ \widetilde{a}^d = \widetilde{b}$$

through the Protocol $\mathtt{PoEqDLog}[(a,b),\ (\widetilde{a},\widetilde{b})]$. Now, since the challenge γ is randomly generated, it follows that

$$\mathbf{Prob}\left((a_1^d, \cdots, a_n^d) \neq (b_1, \cdots, b_n) \;\middle|\; \widetilde{a}^d = \widetilde{b}\right) \in \mathrm{negl}(\lambda).$$

Since the Protocol $\mathtt{EqDLog}[(a,b),\ (\widetilde{a},\widetilde{b})]$ is secure under the strong-RSA and adaptive root assumptions, it follows that the Protocol $\mathtt{AggEqDLog}[(a_1,b_1),\ (a_2,b_2)]$ is also secure under these assumptions. □

We can also generalize the protocol $\mathtt{EqDLog}$ in another direction. 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

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

Protocol 2.6. *Proof of Polynomial equation between discrete logarithms* ($\mathtt{PoPolyDLog}$) :
Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda),\ g \in \mathbb{G}$.
Inputs: 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:
-$a_1^{d_1} = b_1,\ a_2^{d_2} = b_2$
-$f(d_1) = d_2$

Step. 1. The Prover $\mathcal{P}$ computes $\widetilde{g}_1, \widetilde{g}_2$ and sends them to the Verifier $\mathcal{V}$.
2. The Fiat-Shamir heuristic generates a λ-bit prime l.
3. $\mathcal{P}$ computes elements q_1, q_2, r_1, r_2 such that

$$d_1 = q_1 l + r_1,\ d_2 = q_1 l + r_1,\ \ r_1, r_2 \in [l].$$

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. The Verifer verifies that $r_1 \in [l]$. He independently computes $r_2 := f(r_1) \pmod{l}$.
6. $\mathcal{V}$ verifies the equations

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

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 2.7. *The protocol* `PoPolyDLog` *is an argument of knowledge in the generic group model.*

Proof. (Sketch) The Verifier independently computes $r_2 := f(r_1) \pmod{l}$. Hence, the equations

$$Q_1^l a_1^{r_1} \stackrel{?}{=} b_1 \ \bigwedge\ Q_2^l a_2^{r_2}$$

imply that with overwhelming probability, the Prover possesses rational d_1, d_2 such that

$$a^{d_1} = b_1\ ,\ a_2^{d_2} = b_2,\ ,\ d_2 \equiv f(d_1) \pmod{l}.$$

Furthermore, the equations

$$(g_1)^l g^{r_1} \stackrel{?}{=} g^{d_1} \ \bigwedge\ (g_2)^l g^{r_2} \stackrel{?}{=} g^{d_2}$$

imply that with overwhelming probability, $\widetilde{g}_1 = g^{d_1}\ ,\ \widetilde{g}_2 = g^{d_2}$. The fractional root assumption now implies that with overwhelming probability, $d_1, d_2 \in \mathbb{Z}$. □

In the next section, we will generalize this protocol to multivariate polynomial relations for multiple discrete logarithms.

2.1 Aggregating the knowledge of exponents

We call the following protocol the *Proof of Aggregated Knowledge of the Exponent 1* or `AggKE-1` for short. We provide an argument of knowledge for the relation:

$$\mathcal{R}_{\texttt{AggKE-1}}[a,\ \mathcal{B}] = \left\{ \begin{array}{l} (a \in \mathbb{G}, \mathcal{B} = (b_1, \cdots, b_n) \in \mathbb{G}^n); \\ (d_1, \cdots, d_n) \in \mathbb{Z}^n) : \\ b_i = a^{d_i}\ \forall\ i \end{array} \right\}$$

Protocol 2.8. *Proof of Aggregated knowledge of exponents* 1 (`PoAggKE-1`):
Parameters: $\mathbb{G} \stackrel{\$}{\leftarrow} \mathrm{GGen}(\lambda),\ g \in \mathbb{G}$.
Inputs: Elements $a \in \mathbb{G}$, $(b_1, \cdots, b_n) \in \mathbb{G}^n$ for some integer $n \geq 1$.
Claim: The Prover possesses integers $d_1, \cdots, d_n$ such that $a^{d_i} = b_i$ for $i = 1, \cdots, n$

Step. 1. The Fiat-Shamir heuristic generates λ-bit challenge γ.

2. The Prover $\mathcal{P}$ computes $\widetilde{g} := g^{\sum_{i=1}^{n} d_i^{n\lambda}\gamma^i}$ and sends it to the Verifier $\mathcal{V}$.
3. $\mathcal{P}$ computes

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

4. The Fiat-Shamir heuristic generates a λ-bit prime l.
5. $\mathcal{P}$ computes the integers $r_i := d_i \pmod{l}$ and the integers $q, r, \widetilde{q}, \widetilde{r}$ such that

$$\sum_{i=1}^{n} d_i\gamma^i = ql + r\ ,\ \sum_{i=1}^{n} d_i^{n\lambda}\gamma^i = \widetilde{q}l + \widetilde{r}, \quad r, \widetilde{r} \in [l]$$

and

$$Q := a^q\ ,\ \breve{g} := g^{\widetilde{q}}.$$

He sends $Q, \breve{g}, (r_1, \cdots, r_n)$ to the Verifier.
6. The Fiat-Shamir heuristic generates a λ-bit challenge γ_0.
7. $\mathcal{P}$ computes integers q_0, r_0 such that

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

He computes $Q_0 := a^{q_0}$ and sends it to $\mathcal{V}$.
8. $\mathcal{V}$ verifies that $(r_1, \cdots, r_n) \in [l]^n$ and independently computes

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

$$\widetilde{r} := \sum_{i=1}^{n} r_i^{n\lambda}\gamma^i \pmod{l}\ ,\ r := \sum_{i=1}^{n} r_i\gamma^i \pmod{l}\ ,\ r_0 := \sum_{i=1}^{n} r_i\gamma_0^i \pmod{l}.$$

9. V verifies the equations

$$Q^l a^r \stackrel{?}{=} b \bigwedge (Q_0)^l a^{r_0} \stackrel{?}{=} b_0 \bigwedge (\breve{g})^l g^{\widetilde{r}} \stackrel{?}{=} \widetilde{g}.$$

He accepts if and only if all equations hold. $\square$

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 set of remainders modulo l as one of its inputs. Hence, the remainders $r_i := d_i \pmod{l}$ must be honestly computed to succeed at the additional task of computing $Q_0 \in \mathbb{G}$ such that $(Q_0)^l a^{r_0} = b_0$.

Proposition 2.9. *The protocol* `PoAggKE-1` *is an argument of knowledge in the generic group model.*

Proof. (Sketch) The equation $Q_0^l a^{r_0} = b_0$ implies that $\mathcal{P}$ possesses rationals $d_1, \cdots, d_n$ such that $r_i \equiv d_i \pmod{l}$ and $a_i^{d_i} = b_i$. Furthermore, we have $\widetilde{g} = (\breve{g})^l g^{\widetilde{r}}$ and hence, the adaptive root assumption implies that with overwhelming probability,

$$\widetilde{g} = g^{(\sum_{i=1}^{n} d_i^{n\lambda}\gamma^i)+lk}$$

for some integer k. Since the λ-bit prime l is randomly generated, the discrete logarithm between $g, \widetilde{g}$ is randomly and uniformly distributed modulo l. Hence, the Schwartz-Zippel lemma implies that with overwhelming probability,

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

Now, the fractional root assumption implies that with overwhelming probability, $\sum_{i=1}^{n} d_i^{n\lambda}\gamma^i \in \mathbb{Z}$. From Lemma 1.3, it follows that with overwhelming probability, $(d_1, \cdots, d_n) \in \mathbb{Z}$. □

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

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

The soundness of the protocol hinges on the Schartz-Zippel lemma for multivariate polynomials, which we state here.

Lemma 2.10. (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*

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

Protocol 2.11. *Proof of multivariate polynomial relation between discrete logarithms* (`PoMultPolyDLog`) :

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

Inputs: Elements $a \in \mathbb{G}$, $(b_1, \cdots, b_n) \in \mathbb{G}^n$ for some integer $n \geq 1$; a public n-variate polynomial $f(X_1, \cdots, X_n) \in \mathbb{Z}[X_1, \cdots, X_n]$.

Claim: The Prover possesses integers $d_1, \cdots, d_n$ such that:

- $a^{d_i} = b_i$ for $i = 1, \cdots, n$.
- $f(d_1, \cdots, d_n) = 0$.

Step. 1. The Fiat-Shamir heuristic generates a λ-bit integer γ.

2. $\mathcal{P}$ computes $\widetilde{g} := g^{\sum_{i=1}^{n} d_i^{n\lambda}\gamma^i}$ and sends it to the Verifier $\mathcal{V}$.

3. The Fiat-Shamir heuristic generates a λ-bit prime l.

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

$$\sum_{i=1}^{n} d_i^{n\lambda}\gamma^i = \widetilde{q}l + \widetilde{r}\ ,\ \sum_{i=1}^{n} d_i\gamma^i = ql + r\ ,\quad \widetilde{r}, r \in [l].$$

5. $\mathcal{P}$ computes $Q := a^q$, $\breve{g} := g^{\widetilde{q}}$ and sends $Q, \breve{g}, (r_1, \cdots, r_n)$ to $\mathcal{V}$.

6. The Fiat-Shamir heuristic generates a λ-bit challenge γ_0.

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

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

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

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

$$\widetilde{r} := \sum_{i=1}^{n} r_i^{n\lambda} \gamma^i \pmod{l} \ , \ r := \sum_{i=1}^{n} r_i \gamma^i \pmod{l} \ , \ r_0 := \sum_{i=1}^{n} r_i \gamma_0^i \pmod{l}.$$

9. $\mathcal{V}$ computes

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

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

$$Q^l a^r \stackrel{?}{=} b \ \bigwedge \ (Q_0)^l a^{r_0} \stackrel{?}{=} b_0 \ \bigwedge \ (\breve{g})^l g^{\widetilde{r}} \stackrel{?}{=} \widetilde{g} \ \bigwedge \ f(r_1, \cdots, r_n) \stackrel{?}{\equiv} 0 \pmod{l}.$$

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

Thus, the proof consists of five $\mathbb{G}$-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 $\mathcal{P}_{\text{mal}}$ could forge a fake proof as follows.

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

$$\sum_{i=1}^{n} d_i^{n\lambda} \gamma^i \equiv \sum_{i=1}^{n} r_i^{n\lambda} \gamma^i \pmod{l} \ , \ f(r_1, \cdots, r_n) \equiv 0 \pmod{l}.$$

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

2. $\mathcal{P}_{\text{mal}}$ then sends $r_1, \cdots, r_n$ to the Verifier.
3. The Verifier is thus tricked into believing that $f(d_1, \cdots, d_n) = 0$, which might not 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-inertactive 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, we have

$$\textbf{Prob}\Big(\sum_{i=1}^{n} d_i \gamma_0^i \equiv \sum_{i=1}^{n} r_i \gamma_0^i \pmod{l} \ \Big| \ d_i \not\equiv r_i \pmod{l} \text{ for some } i \Big) = \text{negl}(\lambda).$$

Hence, 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

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

with a non-negligible probability.

An important special case 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 one of the subsequent protocols.

Proposition 2.12. *The protocol* `PoMultPolyDLog` *is an argument of knowledge in the generic group model.*

Proof. (Sketch) Since the equation $Q_0^l a^{r_0} = \prod_{i=1}^{n} b_i^{\gamma_0^i}$ holds, the adaptive root assumption implies that with overwhelming probability, the Prover possesses rationals $d_1, \cdots, d_n$ such that:
- $a^{d_i} = b_i$ for $i = 1, \cdots, n$ and
- $\sum_{i=1}^{n} d_i \gamma_0^i \equiv \sum_{i=1}^{n} r_i \gamma_0^i \pmod{l}$.

Since γ_0 is randomly generated after the Prover has committed $(r_1, \cdots, r_n)$, γ_0 is randomly and uniformly distributed modulo l. Hence, it follows that with overwhelming probability, $d_i \equiv r_i \pmod{l}$ for every index i. Now,

$$\textbf{Prob}\Big(f(d_1, \cdots, d_n) \equiv 0 \pmod{l} \;\Big|\; f(d_1, \cdots, d_n) \neq 0\Big) = \mathrm{negl}(\lambda).$$

Thus, with overwhelming probability, $f(d_1, \cdots, d_n) = 0$. Furthermore, the equation

$$\widetilde{g} = (\breve{g})^l g^{\widetilde{r}}$$

implies that $\widetilde{g} = g^{lk + \sum_{i=1}^{n} d_i^{n\lambda} \gamma^i}$ for some integer k. Since l is randomly generated *after* the Prover sends $\widetilde{g}$, it follows that with overwhelming probability, $\widetilde{g} = g^{\sum_{i=1}^{n} d_i^{n\lambda} \gamma^i}$. The fractional root assumption implies that with overwhelming probability, $\sum_{i=1}^{n} d_i^{n\lambda} \gamma^i \in \mathbb{Z}$ and hence, with overwhelming probability, $d_1, \cdots, d_n \in \mathbb{Z}$. □

We now discuss a relation that is a dual to the relation `AggKE-1`. We provide an argument of knowledge for the following relation:

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

Given elements $a_1, \cdots, a_n$ such that

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

where the integers d_i are known to the Prover, 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 protocol `PoAggKE-1`$[a,\ (a_1, \cdots, a_n)]$ and `PoKE`$[a,\ A]$ would demonstrate that the Prover possesses the discrete logarithms 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, a Prover needs to demonstrate that the rationals d/d_i $(i = 1, \cdots, n)$ are integers.

Unlike the protocol `PoAggKE-1`, the proof for `AggKE-2` is not constant-sized. Although the number of group elements is indeed constant, our proof contains n λ-bit integers arising from the remainders modulo the challenge.

Protocol 2.13. *Proof of Aggregated knowledge of exponents* 2 (`PoAggKE-2`):

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

Inputs: $(a_1, \cdots, a_n) \in \mathbb{G}^n$, $A \in \mathbb{G}$.

Claim: The Prover posseses integers $d_1, \cdots, d_n$ such that $a_i^{d_i} = A$.

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

$$D := \mathbf{lcm}(d_1, \cdots, d_n)\ ,\ \widehat{d_i} := D/d_i\ (i = 1, \cdots, n).$$

Using Shamir's trick, he computes an element $a \in \mathbb{G}$ such that $a^D = A$. He sends a to the Verifier $\mathcal{V}$ along with a non-interactive $\texttt{PoKE}[a,\ A]$.
2. The Fiat-Shamir heuristic generates a λ-bit integer γ.
3. $\mathcal{P}$ computes

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

and sends it to the Verifier $\mathcal{V}$.
4. $\mathcal{P}$ computes a non-interactive proof for $\texttt{AggKE-1}[a,\ \{a_1, \cdots, a_n\}]$ and sends it to $\mathcal{V}$.
5. The Fiat-Shamir heuristic generates a λ-bit prime l.
6. $\mathcal{P}$ computes $R := D \pmod{l}$ and $\breve{a} := a^{(D-R)/l}$. He sends $\breve{a}, R$ to $\mathcal{V}$.
7. $\mathcal{P}$ computes the integers $\widehat{r}_i := \widehat{d_i} \pmod{l}$ $(i = 1, \cdots, n)$ and the integer $\widehat{q}, \widehat{r}$ such that

$$\sum_{i=1}^{n} \widehat{d_i}\gamma^i = \widehat{q}l + \widehat{r}\ ,\ \widehat{r} \in [l].$$

He computes $\widehat{Q} := a^{\widehat{q}}$ and sends $\widehat{Q}, (\widehat{r}_1, \cdots, \widehat{r}_n)$ to $\mathcal{V}$.
8. $\mathcal{P}$ computes the integers $r_i := d_i \pmod{l}$ $(i = 1, \cdots, n)$ and the integers $q, r, \widetilde{q}, \widetilde{r}$ such that

$$\sum_{i=1}^{n} d_i\gamma^i = ql + r\ ,\ \sum_{i=1}^{n} d_i^{n\lambda}\gamma^i = \widetilde{q}l + \widetilde{r}\ , \quad r, \widetilde{r} \in [l]$$

He computes $Q := a^q$, $\breve{g} := g^{\widetilde{q}}$ and sends $Q, \breve{g}$ to $\mathcal{V}$.
9. The Fiat-Shamir heuristic generates a λ-bit challenge γ_0.
10. $\mathcal{P}$ computes the integers $\widehat{q}_0, \widehat{r}_0$ such that

$$\sum_{i=1}^{n} \widehat{d_i}\gamma^i = \widehat{q}_0 l + \widehat{r}_0\ ,\ \widehat{r}_0 \in [l].$$

He computes $\widehat{Q}_0 := a^{q_0}$ and sends Q_0 to $\mathcal{V}$.
11. $\mathcal{V}$ verifies that $(\widehat{r}_1, \cdots, \widehat{r}_n, R) \in [l]^{n+1}$ and independently computes $r_i \equiv \widehat{r}_i^{-1}R \pmod{l}$ $(i = 1, \cdots, n)$ and

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

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

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

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

We note that the additional challenge γ_0 is necessary for the security of this protocol. Note that the Prover commits the integer $\sum\limits_{i=1}^{n} d_i^{n\lambda}\gamma^i$ by computing $\widetilde{g} := g^{\sum\limits_{i=1}^{n} d_i^{n\lambda}\gamma^i}$ 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 $g^{\sum_{i=1}^{n} d_i^{n\lambda}\gamma^i}$
2. $\mathcal{P}_{\text{mal}}$ chooses integers $r_1, \cdots, r_n \in [l]$ such that

$$\sum_{i=1}^{n}(Dd_i^{-1})\gamma^i \equiv \sum_{i=1}^{n}(Dr_i^{-1})\gamma^i \pmod{l}\ ,\ \sum_{i=1}^{n} e_i\gamma^i \equiv \sum_{i=1}^{n} r_i\gamma^i \pmod{l},$$

but $d_i \not\equiv r_i \pmod{l}$ 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 $\sum_{i=1}^{n} d_i^{n\lambda}\gamma^i$ is an integer, which might not 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.

Now, in our protocol, γ_0 is randomly generated by the Fiat-Shamir heuristic *after* the Prover sends $(\widehat{r}_1, \cdots, \widehat{r}_n, R)$. Hence, we have

$$\mathbf{Prob}\Big(\sum_{i=1}^{n} d_i\gamma_0^i \equiv \sum_{i=1}^{n} r_i\gamma_0^i \pmod{l}\ \Big|\ d_i \not\equiv r_i \pmod{l} \text{ for some } i\Big) = \text{negl}(\lambda).$$

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

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

with non-negligible probability.

Proposition 2.14. *The protocol* `PoAggKE-2` *is an argument of knowledge in the generic group model.*

Proof. (Sketch) The subprotocol `PoAggKE-1`$[a,\ \{a_1, \cdots, a_n\}]$ demonstrates that with overwhelming probability, the Prover possesses integers $\widehat{d}_1, \cdots, \widehat{d}_n$ such that

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

Furthermore, since γ_0 is randomly generated and the equation

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

holds, the adaptive root assumption implies that with overwhelming probability,

$$\sum_{i=1}^{n} \widehat{r}_i\gamma_0^i \equiv \sum_{i=1}^{n} \widehat{d}_i\gamma_0^i \pmod{l}.$$

Hence, with overwhelming probability, $\widehat{r}_i \equiv \widehat{d}_i \pmod{l}$ (Schwartz-Zippel).

The equation $(\breve{a})^l a^R = A$ implies that with overwhelming probability, the Prover possesses a rational $D \equiv R \pmod{l}$ such that $a^D = A$. Thus, with overwhelming probability, the rationals $D\widehat{d}_1^{-1}, \cdots, D\widehat{d}_n^{-1}$ satisfy

$$a_i^{D\widehat{d_i}^{-1}} = A \text{ for every } i.$$

Now, the Verifier independently computes

$$r_i := R\widehat{r_i}^{-1} \equiv D\widehat{d_i}^{-1} \pmod{l}\ (i = 1, \cdots, n),\quad r := \sum_{i=1}^{n} r_i\gamma^i \equiv \sum_{i=1}^{n} D\widehat{d_i}^{-1} \pmod{l},$$

$$\widetilde{r} := \sum_{i=1}^{n} r_i^{n\lambda}\gamma^i \equiv \sum_{i=1}^{n} (D\widehat{d_i}^{-1})^{n\lambda}\gamma^i \pmod{l}.$$

Hence, the equation $(\breve{g})^l g^{\widetilde{r}} = \widetilde{g}$ implies that with overwhelming probability,

$$\widetilde{g} = g^{\sum_{i=1}^{n}(D\widehat{d_i}^{-1})^{n\lambda}\gamma^i} g^{kl}$$

for some integer k. Since the prime l is randomly generated, the Schwartz-Zippel lemma implies that with overwhelming probability,

$$\widetilde{g} = g^{\sum_{i=1}^{n}(D\widehat{d_i}^{-1})^{n\lambda}\gamma^i}.$$

The fractional root assumption implies that $\sum_{i=1}^{n}(D\widehat{d_i}^{-1})^{n\lambda}\gamma^i$ is a an integer and by lemma 1.3, it follows that with overwhelming probability, the rationals $D\widehat{d_i}^{-1}$ are integers. □

In what follows, we provide an argument of knowledge for the relation

$$\mathcal{R}_{\texttt{EqDLogPairs}}[(a_1, \mathcal{B}),\ (a_2, \mathcal{C})] = \left\{ \begin{array}{l} ((a_1, a_2) \in \mathbb{G}^2 \\ \mathcal{B} = (b_1, \cdots, b_n)\ ,\ \mathcal{C} = (c_1, \cdots, c_n)\ \in \mathbb{G}^n); \\ (d_1, \cdots, d_n) \in \mathbb{Z}^n) : \\ b_i = a_1^{d_i}\ ,\ c_i = a_2^{d_i}\ \forall\ i \end{array} \right\}$$

Protocol 2.15. *Proof of equalities of pairs of discrete logarithm* (`PoEqDLogPairs`) :

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

Inputs : $a_1, a_2 \in \mathbb{G}\ ,\ (b_1, \cdots, b_n),\ (c_1, \cdots, c_n)\ \in\ \mathbb{G}^n$.

Claim: The Prover possesses integers $d_1, \cdots, d_n$ such that $a_1^{d_i} = b_i,\ a_2^{d_i} = c_i$.

Step. 1. The Fiat-Shamir heuristic generates a λ-bit challenge γ.

2. The Prover $\mathcal{P}$ computes $\widetilde{g} := g^{\sum_{i=1}^{n} d_i^{n\lambda}\gamma^i}$ and sends it to the Verifier $\mathcal{V}$.

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

$$B := \prod_{i=1}^{n} b_i^{\gamma^i},\ \ C := \prod_{i=1}^{n} c_i^{\gamma^i}\ \in\ \mathbb{G}.$$

4. The Fiat-Shamir heuristic generates a λ-bit prime l.

5. $\mathcal{P}$ computes $r_i := d_i \pmod{l}$ and the integers $q, \widetilde{q}, r, \widetilde{r}$ such that

$$\sum_{i=1}^{n} d_i^{n\lambda}\gamma^i = \widetilde{q}l + \widetilde{r}\ ,\ \sum_{i=1}^{n} d_i\gamma^i = ql + r,\quad r, \widetilde{r} \in [l]$$

6. $\mathcal{P}$ computes $\breve{g} := g^{\widetilde{q}}$, $Q := a_1^q$ and sends $\breve{g}$, Q, $(r_1, \cdots, r_n)$ to $\mathcal{V}$.
7. $\mathcal{P}$ computes a non-interactive proof for $\texttt{EqDLog}[(a_1, B),\ (a_2, C)]$ and sends it to $\mathcal{V}$.
8. The Fiat-Shamir heuristic generates a λ-bit challenge γ_0.
9. $\mathcal{P}$ computes integers q_0, r_0 such that

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

He computes $Q_0 := a_1^{q_0}$ and sends Q_0 to $\mathcal{V}$.
10. $\mathcal{V}$ verifies that $(r_1, \cdots, r_n) \in [l]^n$ and independently computes

$$\widetilde{r} := \sum_{i=1}^{n} r_i^{n\lambda} \gamma^i \pmod{l}\ ,\ r := \sum_{i=1}^{n} r_i \gamma^i \pmod{l}\ ,\ r_0 := \sum_{i=1}^{n} r_i \gamma_0^i \pmod{l}.$$

11. $\mathcal{V}$ independently computes the elements

$$B := \prod_{i=1}^{n} b_i^{\gamma^i},\ \ B_0 := \prod_{i=1}^{n} b_i^{\gamma_0^i},\ \ C := \prod_{i=1}^{n} c_i^{\gamma^i}\ \in\ \mathbb{G}.$$

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

$$Q^l a_1^r \stackrel{?}{=} B \bigwedge (Q_0)^l a_1^{r_0} \stackrel{?}{=} B_0 \bigwedge (\breve{g})^l g^{\widetilde{r}} \stackrel{?}{=} \widetilde{g}.$$

He then accepts the validity of the claim if and only if the proof for $\texttt{EqDLog}[(a_1, B),\ (a_2, C)]$ is valid and all equations hold. □

2.2 Protocols for arguments of disjointness

Next, we describe a protocol whereby an honest Prover can show that the GCD of two discrete logarithms equals a third discrete logarithm without revealing any further information about them. One obvious application is proving disjointness of sets in accumulators instantiated with the group $\mathbb{G}$. 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 observation that

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

Protocol 2.16. *Proof of the greatest common divisor* ($\texttt{PoGCD}$):
Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda),\ g \in \mathbb{G}$.
Inputs: 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:
- $a_1^{d_1} = b_1$, $a_2^{d_2} = b_2$, $a_3^{d_3} = b_3$
- $\mathbf{gcd}(d_1, d_2) = d_2$

Step. 1. The Prover $\mathcal{P}$ computes $b_{1,2} := a_1^{d_2}$, $b_{1,3} := a_1^{d_3}$ and sends them to the Verifier $\mathcal{V}$.
2. He computes 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. $\mathcal{P}$ computes 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 algorithm `Bezout` to compute integers e_1, e_2 such that $e_1d_1 + e_2d_2 = d_3$.

5. $\mathcal{P}$ computes

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

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

6. $\mathcal{V}$ verifies all of the proofs he receives in addition to the equation $\widetilde{b_1}\widetilde{b}_{1,2} \stackrel{?}{=} b_{1,3}$. He accepts the validity of the claim if and only if all of these proofs are valid. □

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.

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

Protocol 2.17. *Proof of Relatively Prime Discrete Logarithms* (`PoRelPrimeDLog`):
Parameters: $\mathbb{G} \stackrel{\$}{\leftarrow} \mathrm{GGen}(\lambda),\ g \in \mathbb{G}$.
Inputs: Elements $a_1, a_2, b_1, b_2 \in \mathbb{G}$.
Claim: The Prover possesses integers $d_1,\ d_2$ such that:
- $a_1^{d_1} = b_1,\ a_2^{d_2} = b_2$
- $\mathbf{gcd}(d_1, d_2) = 1$

Step. 1. The Prover $\mathcal{P}$ computes $b_{1,2} := a_1^{d_2}$ and sends it to the Verifier $\mathcal{V}$.

2. $\mathcal{P}$ computes a non-interactive proof for $\texttt{EqDLog}[(a_2, b_2),\ (a_1, b_{1,2})]$ and sends it to $\mathcal{V}$.

3. $\mathcal{P}$ uses the algorithm `Bezout` to compute integers e_1, e_2 such that $e_1d_1 + e_2d_2 = 1$.

4. $\mathcal{P}$ computes

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

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

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

6. $\mathcal{V}$ verifies the equation $\widetilde{b_1}\widetilde{b}_{1,2} \stackrel{?}{=} a_1$ 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 2.18. *The Protocols* `PoGCD` *and* `PoRelPrimeDLog` *are arguments of knowledge in the generic group model.*

Proof. Since the relation `RelPrimeDLog` is a special case of the relation `GCD`, it suffices to show that the protocol `PoGCD` is correct and sound. Furthermore, since we showed that `PoEqDLog` is correct and sound, we may assume without loss of generality that - with notations as in the protocol `PoGCD` -

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

Now, the protocols $\texttt{PoKE}[b_3, b_1]$, $\texttt{PoKE}[b_3, b_2]$, $\texttt{PoKE}[a_1, b_3]$ imply that with overwhelming probability, the Prover $\mathcal{P}$ possesses integers d_1, d_2, d_3 such that

$$a_1^{d_1} = b_1\ ,\ a_1^{d_2} = b_2\ ,\ a_1^{d_3} = b_3\ ,\ \mathbf{gcd}(d_1, d_2) \equiv 0 \ (\mathrm{mod}\ d_3).$$

Furthermore, the Prover $\mathcal{P}$ sends elements $\widetilde{b}_1, \widetilde{b}_2 \in \mathbb{G}$ such that $\widetilde{b}_1\widetilde{b}_2 = b_3$ along with the non-interactive proofs for $\texttt{PoKE}[b_1, \widetilde{b}_1]$, $\texttt{PoKE}[b_2, \widetilde{b}_2]$. Hence, with overwhelming probability, the Prover possesses integers e_1, e_2 such that $e_1d_1 + e_2d_2 = d_3$. Hence, it follows that with overwhelming probability, $d_3 = \mathbf{gcd}(d_1, d_2)$. □

It is easy to see that the 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 $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$, we have

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

by setting

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

We now use the protocols AggKE-1 and AggKE-2 to generalize the protocol RelPrimeDLog 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,

$$A_i = g^{\prod\limits_{x \in \mathcal{D}_i} x}.$$

Suppose a Prover needs to demonstrate to a Verifier (with access to the accumulated digests) that the data sets $\mathcal{D}_i$ are pairwise disjoint, while keeping the communication complexity to a bare minimum. In particular, the Verifier should not need to access the data sets $\mathcal{D}_i$. A straightforward way would be to provide the $\binom{n}{2}$ proofs of pairwise disjointness. But this would entail $\mathbf{O}(n^2)$ group elements and $\mathbf{O}(n^2)$ λ-bit integers. Instead, we provide a protocol whereby the Prover can demonstrate the pairwise disjointness with a constant number of $\mathbb{G}$-elements and $2n$ 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 following relation:

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

The protocol rests on the following elementary lemma.

Lemma 2.19. *Let $d_1, \cdots, d_n$ be non-zero integers. Set*

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

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

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

the **RootFactor** algorithm allows us to compute elements a_i such that $a_i^{d_i} = A$ in runtime $\mathbf{O}(n\log(n))$. Thus, a Prover can compute the element $\widehat{A} := \prod_{i=1}^{n} a_i$ in runtime $\mathbf{O}(\log(D)\log(\log(D)))$ with the **RootFactor** algorithm followed by n group multiplications.

Protocol 2.20. *Proof of Aggregated Knowledge of Relatively Prime Discrete Logarithms* 1 (`PoAggRelPrimeDLog-1`) :

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

Inputs: 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 $i \neq j$.

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

$$D := \prod_{i=1}^{n} d_i \ \ , \ \ \widehat{D} := \sum_{i=1}^{n} \frac{D}{d_i}.$$

2. $\mathcal{P}$ computes $A := a^D,\ \widehat{A} := a^{\widehat{D}}$ (the latter using the **RootFactor** algorithm) and sends $A, \widehat{A}$ to the Verifier $\mathcal{V}$.
3. $\mathcal{P}$ computes a non-interactive proof for `AggKE-1`$[a,\ (a_1, \cdots, a_n)]$ and sends it to $\mathcal{V}$.
4. $\mathcal{P}$ computes a non-interactive proof for `MultPolyDLog`$[a,\ (a_1, \cdots, a_n, A),\ f]$ where

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

and sends the proof to V.

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

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

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

6. $\mathcal{P}$ computes a non-interactive proof for `RelPrimeDLog`$[(a, A),\ (a, \widehat{A})]$ and sends it to $\mathcal{V}$.
7. $\mathcal{V}$ verifies the four proofs and accepts the validity of the claim if and only if all proofs are valid. □

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

Proposition 2.21. *The protocol* `PoAggRelPrimeDLog-1` *is an argument of knowledge in the generic group model.*

Proof. (Sketch) The two `MultPolyDLog` proofs imply that with overwhelming probability, $\mathcal{P}$ possesses integers D, $\widehat{D}$, $d_1, \cdots, d_n$ such that

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

Furthermore, the proof for `RelPrimeDLog`$[(a, A),\ \ (a, \widehat{A})]$ implies that $\mathbf{gcd}(D, \sum_{i=1}^{n} D/d_i) = 1$. Hence, by the preceding lemma, the integers d_i are pairwise co-prime. □

Given elements $a_1, a_2 \in \mathbb{G}$ 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. Clearly, the latter part is equivalent to showing that the integers

$$d := \mathbf{lcm}(d_1, \cdots, d_m) \;, \; e := \mathbf{lcm}(e_1, \cdots, e_n)$$

are relatively prime. Our approach is to compute elements $B = a_1^d, \; C := a_2^e$. We then use the procols `AggPoKE-1` and `AggPoKE-2` to provide arguments of knowledge that d, e are divisible by $\{d_1, \cdots, d_n\}, \; \{e_, \cdots, e_n\}$ repectively. We then use the procol `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:

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

Protocol 2.22. *Proof of Aggregated Knowledge of Relatively Prime Discrete Logarithms 2* (`PoAggRelPrimeDLog-2`) :

Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda), \; g \in \mathbb{G}$.
Inputs: Elements $a_1, a_2 \in \mathbb{G}$, Elements $\mathcal{B} = (b_1, \cdots, b_m)$, $\mathcal{C} = (c_1, \cdots, c_n)$ of $\mathbb{G}^n$.
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.

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

$$d := \prod_{i=1}^{m} d_m \;\;, \;\; e := \prod_{j=1}^{n} e_n.$$

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. $\mathcal{P}$ computes non-interactive proofs for $\texttt{MultPolyDLog}[a_1, \; (b_1, \cdots, b_m, B), \; (\prod_{i=1}^{m} X_i) - X_{m+1}]$, $\texttt{MultPolyDLog}[a_2, \; (c_1, \cdots, c_n, C), \; (\prod_{j=1}^{n} X_j) - X_{n+1}]$ and sends them to $\mathcal{V}$.

4. $\mathcal{P}$ computes non-interactive proofs for $\texttt{AggKE-1}[a_1, \mathcal{B}]$ and $\texttt{AggKE-1}[a_2, \mathcal{C}]$ and sends them to the Verifer.

5. $\mathcal{P}$ computes 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 of these proofs are valid. $\square$

Remark: The proof consists of the element $B, C \in \mathbb{G}$, two `AggKE-1` proofs, two `MultPolyDLog` proofs and one `RelPrimeDLog` proof. Each of these proofs consists of a constant number of $\mathbb{G}$-elements and hence, the same is true regarding our proof for the Protocol `AggRelPrimeDLog-2`$[(a_1, \mathcal{B}),\ (a_2, \mathcal{C})]$. However, the `AggKE-1` and `PoMultPolyDLog` proofs result in $2(m+n)$ λ-bit integers in the proof. Hence, the size of the proof is $\mathbf{O}(m+n)$.

Proposition 2.23. *The Protocol* `AggRelPrimeDLog-2` *is an argument of knowledge in the generic group model.*

Proof. (Sketch) This boils down to showing that each of the subprotocols is correct and sound. □

An example: We discuss an example of an application of this last protocol. Consider two families

$$\mathcal{A}_1 = \{\mathbf{Acc}_{1,1}, \cdots, \mathbf{Acc}_{1,m}\},\ \ \mathcal{A}_2 = \{\mathbf{Acc}_{2,1}, \cdots, \mathbf{Acc}_{2,n}\}$$

of accumulators instantiated using the same group $\mathbb{G}$ of hidden order. As usual, each data element is represented by a λ-bit prime. Let g_1, g_2 be the genesis states for all accumulators in $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively.

As usual, each data element is represented by a distinct λ-bit prime. Let $\mathcal{D}_{1,1}$ ($\mathcal{D}_{2,j}$) denote the data set inserted into the accumulator $\mathbf{Acc}_{1,i}$ (respectively, $\mathbf{Acc}_{2,j}$) and write

$$\mathcal{D}_1 := \bigcup_{i=1}^{m} \mathcal{D}_{1,i}\ \ ,\ \ \mathcal{D}_2 = \bigcup_{j=1}^{n} \mathcal{D}_{2,j}.$$

Suppose a Verifier (with access to the accumulated digests) wants to verify that the unions are disjoint, i.e. $\mathcal{D}_1 \cap \mathcal{D}_2 = \emptyset$. An honest Prover could simply provide a non-interactive proof for the protocol `AggRelPrimeDLog-2`$[(g_1, \mathcal{D}_1),\ (g_2, \mathcal{D}_2)]$. In particular, the proof can be verified without access to the data sets $\mathcal{D}_1, \mathcal{D}_2$.

Consider a setting where we have data sets $\mathcal{D}_1, \cdots, \mathcal{D}_n$ in an accumulator. Let A denote the accumulated digest, w_i the witness for $\mathcal{D}_i$ and d_i the product of all elements of $\mathcal{D}_i$.

Suppose a Prover needs to demonstrate that the sets $\mathcal{D}_i$ are pairwise disjoint to a Verifier who has access to the witnesses $w_1, \cdots, w_n$ but not the data sets. A straightforward approach would be to provide a proof for `RelPrimeDLog`$[(w_i, A),\ (w_j, A)]$. But such a proof would contain $\mathbf{O}(n^2)$ $\mathbb{G}$-elements and $\mathbf{O}(n^2)$ λ-bit integers.

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

$$w^{\mathbf{lcm}(d_1, \cdots, d_n)} = A^k$$

for some integer k. More precisely,

$$k = \sum_{i=1}^{n} x_i \frac{\mathbf{lcm}(d_1, \cdots, d_n)}{d_i}.$$

Furthermore, the Prover can efficiently compute the integers

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

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

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

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

$$\mathbf{lcm}(d_1, \cdots, d_n) \equiv 0 \pmod{\prod_{i=1}^{n} d_i},$$

which forces equality. In what follows, we provide an argument of knowledge for the relation

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

Protocol 2.24. *Proof of Aggregated Knowledge of Relatively Prime Discrete Logarithms* 3 (`PoAggRelPrimeDLog-3`):

Parameters: $\mathbb{G} \overset{\$}{\leftarrow} \mathrm{GGen}(\lambda), \ g \in \mathbb{G}$.

Inputs: 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 $i \neq j$.

Step. 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 algorithm `Bezout` to compute integers $e, \widehat{e}$ such that $eD + \widehat{e}\widehat{D} = \mathbf{gcd}(D, \widehat{D}) = 1$.
4. $\mathcal{P}$ computes

$$A_0 := A^e \ , \ W := \big(\prod_{i=1}^{n} w_i\big)^{\widehat{e}}.$$

He sends A_0, W to $\mathcal{V}$ along with non-interactive proofs for $\mathtt{PoKE}[A,\ A_0]$ and $\mathtt{PoKE}[(\prod_{i=1}^{n} w_i),\ W]$.
5. The Fiat-Shamir heuristic generates a λ-bit challenge γ.
6. $\mathcal{P}$ computes $\widetilde{g} := g^{\sum_{i=1}^{n} d_i^{n\lambda}\gamma^i}$ and sends $\widetilde{g}$ to $\mathcal{V}$.
7. The Fiat-Shamir heuristic generates a λ-bit prime l.
8. $\mathcal{P}$ computes

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

and sends $\widehat{w}$ to $\mathcal{V}$.
9. $\mathcal{P}$ computes the integers

$$\widehat{r_i} := \widehat{d_i} \pmod{l} \ , \ r_i := d_i \pmod{l}$$

and sends $(r_1, \cdots, r_n)$ to $\mathcal{V}$.
10. $\mathcal{P}$ computes the integers $\widetilde{q}, \widehat{q}, \widetilde{r}, \widehat{r}$ such that

$$\sum_{i=1}^{n} d_i^{n\lambda}\gamma^i = \widetilde{q}l + \widetilde{r} \ , \ \sum_{i=1}^{n} \widehat{d_i}\gamma^i = \widehat{q}l + \widehat{r} \ , \ \ r, \widehat{r} \in [l].$$

11. $\mathcal{P}$ computes

$$Q := w^q \ , \ \widehat{Q} := w^{\widehat{q}} \ , \ \breve{g} := g^{\widetilde{q}}$$

and sends $Q, \widehat{Q}, \breve{g}$ to $\mathcal{V}$.
12. The Fiat-Shamir heuristic generates a λ-bit challenge γ_0.
13. $\mathcal{P}$ computes the integers $\widehat{q}_0, \widehat{r}_0$ such that

$$\sum_{i=1}^{n} \widehat{d_i}\gamma^i = \widehat{q}_0 l + \widehat{r}_0 \ , \ \widehat{r}_0 \in [l]$$

He computes $\widehat{Q}_0 := w^{\widehat{q}_0}$ and sends it to $\mathcal{V}$.
14. $\mathcal{V}$ verifies that $(r_1, \cdots, r_n) \in [l]^n$. He independently computes

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

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

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

$$(\widehat{Q})^l w^{\widehat{r}} \stackrel{?}{=} \prod_{i=1}^{n} w_i^{\gamma^i} \ \bigwedge \ (\widehat{Q}_0)^l w^{\widehat{r}_0} \stackrel{?}{=} \prod_{i=1}^{n} w_i^{\gamma_0^i} \ \bigwedge \ A_0 W \stackrel{?}{=} w \ \bigwedge \ (\breve{w})^l w^R \stackrel{?}{=} A \ \bigwedge \ (\breve{g})^l g^{\widetilde{r}} \stackrel{?}{=} \widetilde{g}.$$

16. $\mathcal{V}$ verifies the two `PoKE`s from Step 4. He accepts if and only if all equations hold and all proofs are valid. $\square$

Proposition 2.25. *The protocol* `PoAggRelPrimeDLog-3` *is an argument of knowledge in the generic group model.*

Proof. (Sketch) The equations verified in step 15 imply that with overwhelming probability, there exist integers $d_1, \cdots, d_n, D$ such that

$$D = \prod_{i=1}^{n} d_i \ , \ A = w^D \ , \ A = w_i^{d_i} \ \forall \ i.$$

Furthermore, since we have $A_0 W = w$ and the proofs for $\texttt{PoKE}[A,\ A_0]$ and $\texttt{PoKE}[(\prod_{i=1}^{n} w_i),\ W]$, it follows that, in particular, w is expressible as a product

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

Hence,

$$w^{\mathbf{lcm}(d_1, \cdots, d_n)} = A^k$$

for some integer k. Thus, $\mathbf{lcm}(d_1, \cdots, d_n)$ is divisible by the product $\prod_{i=1}^{n} d_i$. Hence, the integers d_i are pairwise co-prime. $\square$

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

$$\mathcal{B} = \{b_1, \cdots, b_m\}, \ \ \mathcal{C} = \{c_1, \cdots, c_n\} \in \mathbb{G}^n,$$

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:

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

An example: Consider the case where B, C 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 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 mn distinct proofs that $\mathbf{gcd}(d_i, e_j) = 1$ for every pair d_i, e_j. 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 `AggRelPrimeDLog-4`$[(\mathcal{W}, B), \ (\mathcal{U}, C)]$. The proof consists of a constant number of $\mathbb{G}$-elements and $(m+n)$ λ-bit integers.

Protocol 2.26. *Proof of Aggregated Knowledge of Relatively Prime Discrete Logarithms* 4 (`PoAggRelPrimeDLog-4`) :

Parameters: $\mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda), \ g \in \mathbb{G}$.
Inputs: Elements $B, C \in \mathbb{G}, \ \mathcal{B} = \{b_1, \cdots, b_m\}, \mathcal{C} = \{c_1, \cdots, c_n\}$ of $\mathbb{G}$
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.

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

$$d := \mathbf{lcm}(d_1, \cdots, d_m), \ \ e := \mathbf{lcm}(e_1, \cdots, e_n).$$

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

$$b^d = B, \ \ c^e = C$$

and sends b, c to the Verifier $\mathcal{V}$.
3. $\mathcal{P}$ computes non-interactive proofs for `AggKE-2`$[\mathcal{B}, B]$ and `AggKE-2`$[\mathcal{C}, C]$ and sends them to $\mathcal{V}$.
4. $\mathcal{P}$ computes non-interactive proofs for `AggKE-1`$[b, \mathcal{B}]$ and `AggKE-1`$[c, \mathcal{C}]$ and sends them to $\mathcal{V}$.
5. $\mathcal{P}$ computes a non-interactive proof for `RelPrime`$[(b, B), \ (c, C)]$ and sends the proof to $\mathcal{V}$.
6. $\mathcal{V}$ verifies all of these proofs and accepts the validity of the claim if and only if all proofs are valid. $\square$

Thus, the proof for `AggRelPrimeDLog-4` consists of a constant number of $\mathbb{G}$-elements and $m+n$ λ-bit integers arising from the proofs for `AggKE-2`$[\mathcal{B}, B]$ and `AggKE-2`$[\mathcal{C}, C]$.

Proposition 2.27. *The protocol* `PoAggRelPrimeDLog-4` *is an argument of knowledge in the generic group model.*

Proof. (Sketch) The proofs for `AggKE-2`$[\mathcal{B}, B]$ and `AggKE-1`$[b, \mathcal{B}]$ imply that, with overwhelming probability, there exist 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, the proofs for `AggKE-2`$[\mathcal{C}, C]$ and `AggKE-1`$[c, \mathcal{C}]$ imply that with overwhelming probability, there exist integers $e, e_1, \cdots, e_n, \widehat{e}_1, \cdots, \widehat{e}_n$ such that

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

Finally, the proof for `RelPrime`$[(b, B), \ (c, C)]$ implies with overwhelming probability, that $\mathbf{gcd}(d, e) = 1$. Hence, $\mathbf{gcd}(d_i, e_j) = 1 \ \forall \ i, j$.

$\square$

References

[BBF19] D. Boneh, B. Bünz, B. 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.

[BFS19] B. Bünz, B. Fisch, A. Szepieniec, *Transparent SNARKs from DARK Compilers*, Cryptology ePrint Archive, Report 2019/1229, 2019. https://eprint.iacr.org/2019/1229

[CFGKN20] M. Campanelli, D. Fiore, N. Greco, D. Kolonelos, L. Nizzardo *Vector Commitment Techniques and Applications to Verifiable Decentralized Storage*

[CSV20] W. Castryck, J, Sotakova, F. Vercauteren, *Breaking the decisional Diffie-Hellman problem for class group actions using genus theory*

[Can87] D. Cantor. *Computing in the Jacobian of a hyperelliptic curve. Mathematics of computation*, 48(177):95–101, 1987.

[Can94] D. Cantor. *On the analogue of the division polynomials for hyperelliptic curves*, Crelle's Journal, 447:91–146, 1994.

[DGS20] S. Dobson, S. Galbraith, B. Smith, *Trustless Groups of Unknown Order with Hyperelliptic Curves*, https://eprint.iacr.org/2020/196

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

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

[STY01] T. Sander, A. Ta-Shma, M. Yung, *Blind, auditable membership proofs*, In Yair Frankel, editor, FC 2000, volume 1962 of LNCS, pages 5371. Springer, Heidelberg, February 2001.

[Th19] S. Thakur, *Batching non-membership proofs with bilinear accumulators*, https://eprint.iacr.org/2019/1147

[Th20] S. Thakur, *Constructing hidden order groups using genus three Jacobians*, https://eprint.iacr.org/2020/348

[Wes19] B. Wesolowski, *Efficient verifiable delay functions.* In Yuval Ishai and Vincent Rijmen, editors, Advances in Cryptology – Eurocrypt 2019, pages 379–407, Cham, 2019. Springer International Publishing.

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

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

negl(λ): The set of functions negligible in λ.

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

PPT: Probabilistic Polynomial Time

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

$\mathcal{P}$: The Prover

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

$\mathcal{V}$: The Verifier

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

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.

1. `PoEqDLog`

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

2. `PoAggEqDLog`

$$\mathcal{R}_{\texttt{AggEqDLog}}[(a, b),\ (\mathcal{A},\ \mathcal{B})] = \left\{ \begin{array}{l} ((a, b) \in \mathbb{G}^2 \\ \mathcal{A} = (a_1, \cdots, a_n),\ \mathcal{B} = (b_1, \cdots, b_n) \in \mathbb{G}^n); \\ d \in \mathbb{Z}) : \\ b_i = a_1^d\ \forall\ i \end{array} \right\}$$

3. `PoPolyDLog`

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

4. `PoAggKE-1`

$$\mathcal{R}_{\texttt{AggKE-1}}[a,\ \mathcal{B}] = \left\{ \begin{array}{l} (a \in \mathbb{G},\ \mathcal{B} = (b_1, \cdots, b_n) \in \mathbb{G}^n); \\ (d_1, \cdots, d_n) \in \mathbb{Z}^n) : \\ b_i = a^{d_i}\ \forall\ i \end{array} \right\}$$

5. `PoAggKE-2`

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

6. `PoMultPolyDLog`

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

7. `PoEqDLogPairs`

$$\mathcal{R}_{\texttt{EqDLogPairs}}[(a_1, \mathcal{B}),\ (a_2, \mathcal{C})] = \left\{ \begin{array}{l} (a_1, a_2 \in \mathbb{G}; \\ (b_1, \cdots, b_n),\ (c_1, \cdots, c_n) \in \mathbb{G}^n); \\ (d_1, \cdots, d_n) \in \mathbb{Z}^n)\ : \\ b_i = a_1^{d_i}\ ,\ c_i = a_2^{d_i}\ \forall\ i \end{array} \right\}$$

8. `PoGCD`

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

9. `PoRelPrimeDLog` (special case of `PoGCD`)

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

10. PoAggRelPrimeDLog-1

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

11. PoAggRelPrimeDLog-2

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

12. PoAggRelPrimeDLog-3

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

13. PoAggRelPrimeDLog-4

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