Approximate Lower Bound Arguments

Pyrros Chaidos[1,4], Aggelos Kiayias[2,5], Leonid Reyzin[3], and Anatoliy Zinovyev[3](✉)

[1] National and Kapodistrian University of Athens, Athens, Greece
[2] University of Edinburgh, Edinburgh, UK
[3] Boston University, Boston, USA
tolik@bu.edu
[4] IOG, Athens, Greece
[5] IOG, Edinburgh, UK

Abstract. Suppose a prover, in possession of a large body of valuable evidence, wants to quickly convince a verifier by presenting only a small portion of the evidence.

We define an Approximate Lower Bound Argument, or ALBA, which allows the prover to do just that: to succinctly prove knowledge of a large number of elements satisfying a predicate (or, more generally, elements of a sufficient total weight when a predicate is generalized to a weight function). The argument is approximate because there is a small gap between what the prover actually knows and what the verifier is convinced the prover knows. This gap enables very efficient schemes.

We present noninteractive constructions of ALBA in the random oracle and Uniform Random String models and show that our proof sizes are nearly optimal. We also show how our constructions can be made particularly communication-efficient when the evidence is distributed among multiple provers working together, which is of practical importance when ALBA is applied to a decentralized setting.

We demonstrate two very different applications of ALBAs: for large-scale decentralized signatures and for achieving universal composability in general-purpose succinct proof systems (SNARKs).

1 Introduction

Suppose a prover is in possession of a large body of valuable evidence that is individually verifiable. The evidence is so voluminous that presenting and verifying all of it is very expensive. Instead, the prover wants to convince a verifier by presenting only a small portion of the evidence.

More formally, let R be a predicate. We explore succinct arguments of knowledge for a prover who knows a set S_p of values that satisfy R such that $|S_\mathrm{p}| \geq n_\mathrm{p}$ and wants to convince a verifier that $|S_\mathrm{p}| > n_\mathrm{f}$, where n_f is somewhat smaller than n_p. Because $n_\mathrm{f} < n_\mathrm{p}$, the verifier obtains a lower bound approximation to the actual cardinality of S_p; hence we call this primitive an *Approximate Lower Bound Argument* or ALBA.

L. Reyzin—Work done while visiting the Blockchain Technology Lab at the University of Edinburgh.

M. Joye and G. Leander (Eds.): EUROCRYPT 2024, LNCS 14654, pp. 55–84, 2024.
https://doi.org/10.1007/978-3-031-58737-5_3

This problem has a long history. In 1983, in order to prove that BPP $\subseteq$ RP$^{\text{NP}}$, Sipser and Gács [Sip83, Section V, Corollary to Theorem 6] showed a simple two-round interactive protocol for proving a lower bound on the size of the set S of accepting random strings. Their construction is based on hash collisions: the verifier chooses some number of universal hash functions $h_1, \ldots, h_m$ [CW79] and the prover shows s, s' such that $s \neq s'$ and $h_i(s) = h_i(s')$ for some $i \in \{1, \ldots, m\}$. If S is small (of size at most n_f), then such hash collisions are very unlikely to exist, and if S is big (of size at least n_p), then they must exist by the pigeonhole principle. In 1986, Goldwasser and Sipser [GS86, Section 4.1] used a slightly different approach, based on the existence of inverses rather than collisions, for proving that public coins suffice for interactive proofs (we provide more details in the full version of our paper [CKRZ23]). To the best of our knowledge, the term "approximate lower bound" in the context of proof systems appears first in Babai's work [Bab85, Section 5.2].

In designing ALBAs, we will aim to minimize communication and computational complexity; these metrics improve as the "gap" n_p/n_f increases. The proof size and verifier time in classical techniques above are far from optimal. While this does not affect the classical applications of ALBAs (such as in proving that any IP language can be decided by an Arthur-Merlin protocol, where the gap can be a large constant and the prover has exponential time), as we will see it does become a pressing concern in modern applications of ALBAs.

1.1 Our Setting

The prover and verifier have access to a predicate R and the prover needs to show some elements of S_p to the verifier so it is convinced that the prover possesses more than n_f elements that pass R. The goal is to find some property that is unlikely to hold for small sets S_f of size n_f, likely to hold for large sets S_p of size n_p, and can be shown with just a few elements.

Generalization to Weighted Sets. We generalize a predicate R that determines validity of set elements, and consider instead a weight function W that takes a set element and outputs its nonnegative integer weight. In that context we wish to explore succinct arguments of knowledge that convince a verifier that the prover knows a set S that satisfies a lower bound $\sum_{s \in S} W(s) > n_\text{f}$. When W is $\{0, 1\}$-valued, we are in the setting of a predicate, and we call this case "unweighted."

We emphasize that R or W are used in a black-box way in our protocols. Thus, our protocols can be used in settings when these functions do not have a known specification—for example, they may be evaluated by human judges who weigh evidence or via some complex MPC protocol that uses secret inputs.

Setup and Interaction Models. Our main focus is on building ALBA protocols that are succinct Non-Interactive Random Oracle Proofs of Knowledge or NIROPK (see Sect. 2 for the definition). If the prover is successful in convincing the verifier, then the knowledge extractor can obtain a set of total weight exceeding n_f by simply observing the random oracle queries; in other words, the protocol is straight-line extractable in the nonprogrammable random oracle model. Our security is information-theoretic as long as the predicate R (or the weight function W) is independent of the random oracle; by the standard technique of adding a commitment to R (or W) to every random oracle query, we obtain computational security even if this function is adaptively chosen to depend on the oracle.

We also show simple modifications of our protocols that replace random oracles with pseudorandom functions (PRFs). By simply publishing the PRF seed as a shared random string, we obtain a non-interactive proof of knowledge in the Uniform Random String (URS) model, in which extractor works by reprogramming the URS. Alternatively, we can obtain a two-round public coin proof of knowledge by having the verifier send the PRF seed (we would then use rewinding for extraction). Protocols in these two models are non-adaptively secure—i.e., they require that the predicate R is independent of the URS or the verifier's first message.

Decentralized Setting. The set S_p may be distributed among many parties. For instance, in a blockchain setting it could be the case that multiple contributing peers hold signatures on a block of transactions and they wish to collectively advance a protocol which approves that block. To capture such settings, we introduce decentralized ALBAs: in such a scheme, the provers diffuse messages via a peer to peer network, and an aggregator (who may be one of the provers themselves) collects the messages and produces the proof. Note that not all provers may decide to transmit a message. In addition to the complexity considerations of regular ALBAs, in the decentralized setting we also wish to minimize the total communication complexity in the prover interaction phase as well as the computational complexity of the aggregator.

1.2 Our Results

Our goal is to design protocols that give the prover a short, carefully chosen, sequence of elements from S_p. We show how to do this with near optimal efficiency.

Let λ be the parameter that controls soundness and completeness: the honest prover (who possesses a set of weight n_p) will fail with probability $2^{-\lambda}$ and the dishonest prover (who possesses a set of weight at most n_f) will succeed with, say, also probability $2^{-\lambda}$. Let u be the length of the sequence the prover sends.

The unweighted case. We first show an unweighted ALBA in which the prover sends only

$$u = \frac{\lambda + \log \lambda}{\log \frac{n_\mathrm{p}}{n_\mathrm{f}}} \tag{1}$$

elements of S_p. Moreover, we show that this number is essentially tight, by proving that at least

$$u = \frac{\lambda}{\log \frac{n_\mathrm{p}}{n_\mathrm{f}}}$$

elements of S_p are necessary. (Note that all formulas in this section omit small additive constants for simplicity; the exact formulas are given in subsequent sections.)

Such a protocol is relatively easy to build in the random oracle model if one disregards the running time of the prover: just ask the prover to brute force a sequence of u elements of S_{p} on which the random oracle gives a sufficiently rare output. Calibrate the probability ε of this output so that $\varepsilon \cdot n_{\mathrm{f}}^u \leq 2^{-\lambda}$ for soundness, but $(1-\varepsilon)^{n_{\mathrm{p}}^u} \leq 2^{-\lambda}$ for completeness. A bit of algebra shows that $u = \frac{\lambda + \log \lambda}{\log \frac{n_{\mathrm{p}}}{n_{\mathrm{f}}}}$ suffices to satisfy both soundness and completeness constraints, so the proof is short.[1] However, in this scheme, the prover has to do an exhaustive search of $1/\varepsilon$ sequences of length u, and thus the running time is exponential.

It follows that the main technical challenge is in finding a scheme that maintains the short proof while allowing the prover to find one quickly. In other words, the prover needs to be able to find a sequence of u elements with some special rare property (that is likely to occur among n_{p} elements but not among n_{f} elements), without looking through all sequences. We do so in Sect. 3 by demonstrating the *Telescope* construction.

Its core idea is to find a sequence of values that itself and also all its prefixes satisfy a suitable condition determined by a hash function (and modeled as a random oracle). This prefix invariant property enables the prover to sieve through the possible sequences efficiently expanding gradually the candidate sequence as in an unfolding telescope. We augment this basic technique further via parallel self composition to match the proof length of the exhaustive search scheme. The resulting prover time (as measured in the number of random oracle queries) is dropped from exponential to $O(n_{\mathrm{p}} \cdot \lambda^2)$. We then show how to drop further the prover complexity to $O(n_{\mathrm{p}} + \lambda^2)$ by prehashing all elements and expressing the prefix invariant property as a hash collision. We also establish that our constructions are essentially optimal in terms of proof size by proving a lower bound in the number of elements than must be communicated by any ALBA scheme that satisfies the extractability requirements of Definition 4.

Weights and Decentralized Provers. In the case where all elements have an integer weight, the straightforward way to design a weighted scheme is to give each set element a multiplicity equal to its weight and apply the algorithms we described above. However, the prover's running time becomes linear in the input's total weight n_{p} which could be in the order of 2^{64} (number of coins in popular cryptocurrencies). A way to solve this problem is to select (with the help of the random oracle) a reasonably-sized subset of the resulting multiset by sampling, for each weighted element, a binomial distribution in accordance with its weight. Given this precomputation, we can then proceed with the Telescope construction as above and with only a (poly)logarithmic penalty due to the weights. We detail this technique in Sect. 5.

Turning our attention to the decentralized setting we present two constructions. In the first one, each party performs a private random-oracle-based coin flip to decide whether to share her value. The aggregator produces a proof by

[1] Let $\varepsilon = 2^{-\lambda} n_{\mathrm{f}}^{-u}$ to satisfy soundness. Then $(1-\varepsilon)^{n_{\mathrm{p}}^u} < \exp(-2^{-\lambda} n_{\mathrm{f}}^{-u})^{n_{\mathrm{p}}^u} = \exp(-2^{-\lambda}(n_{\mathrm{p}}/n_{\mathrm{f}})^u)$ is needed for completeness, so it suffices to have $\exp(-2^{-\lambda}(n_{\mathrm{p}}/n_{\mathrm{f}})^u) \leq 2^{-\lambda}$, i.e., $2^{-\lambda}(n_{\mathrm{p}}/n_{\mathrm{f}})^u \cdot \log e \geq \lambda$, i.e. $(n_{\mathrm{p}}/n_{\mathrm{f}})^u \geq 2^{\lambda} \cdot \lambda / \log e$. Taking logarithm gives the desired result.

concatenating a number of the resulting values equal to a set threshold. In the second construction, we combine the above idea with the Telescope construction letting the aggregator do a bit more work; this results in essentially optimal proof size with total communication complexity $O(\lambda^3)$, or proof size an additive term $\sqrt{\lambda}$ larger than optimal and total communication complexity $O(\lambda^2)$.

1.3 Applications

Beyond the classical applications of ALBAs in complexity theory described earlier [CW79, Sip83, Bab85, GS86], there are further applications of the primitive in cryptography.

Weighted Multisignatures and Compact Certificates. In a multisignature scheme, a signature is accepted if sufficiently many parties have signed the message (depending on the flavor, the signature may reveal with certainty, fully hide, or reveal partially who the signers are). In consensus protocols and blockchain applications, schemes that accommodate large numbers of parties have been put to use in the context of certifying the state of the ledger. In a "proof-of-stake" setting, each party is assigned a weight (corresponding to its stake), and the verifier needs to be assured that parties with sufficient stake have signed a message.

Most existing approaches to building large-scale multisignatures exploit properties of particular signatures or algebraic structures. For example, the recent results of Das et al. and Garg et al. [GJM+23, DCX+23] are based on bilinear pairings and require a structured setup.

In contrast, our work relies *only* on random oracles, making it compatible with any complexity assumption used for the underlying signature scheme, including ones that are post-quantum secure. Expectedly, the black box nature of our construction with respect to the underlying signature results in longer proofs (they can be shortened using succinct proof systems, as we discuss in Sect. 1.4).

In more detail, in order to apply an ALBA scheme to the problem of multisignatures, we treat individual signatures as set elements. The underlying signature scheme needs to be *unique*: it should be impossible (or computationally infeasible) to come up with two different signatures for the same message and public key. Otherwise, it is easy to come up with a set of sufficient total weight by producing multiple signatures for just a few keys[2]. Alternatively, if the knowledge extractor is allowed to rewind (need not be straight-line), one can use an arbitrary (not necessarily unique) signature scheme as follows: treat the public keys as set elements and for every selected public key in the ALBA proof, add its signature. Using an ALBA with decentralized provers is particularly suited to the blockchain setting as signatures will be collected from all participants.

A closely related approach is compact certificates by Micali et al. [MRV+21] who also treat the underlying signature scheme as a black box. In more detail, their construction collects all individual signatures in a Merkle tree, and selects

[2] The verifier could check that all public keys are distinct, but since the proof contains just a small subset of the signatures, a malicious prover could try many signatures, or "grind," until it finds a proof that satisfies this check.

a subset of signatures to reveal via lottery (that can be instantiated via the Fiat-Shamir transform [BR93]). Compared to compact certificates, our Telescope scheme obviates the need for the Merkle tree and hence shaves off a multiplicative logarithmic factor in the certificate length. It is also not susceptible to grinding while in compact certificates the adversary can try different signatures to include in the Merkle tree, and unlike compact certificates that rely inherently on the random oracle, our scheme can be instantiated in the CRS/URS model. Finally, our decentralized prover constructions drastically reduce communication. On the other hand, compact certificates cleverly tie the lottery to public keys rather than signatures and support an arbitrary signature scheme (not necessarily unique) while still providing straight-line knowledge extraction.

Reducing communication complexity was also the focus of Chaidos and Kiayias in Mithril, a weighted threshold multisignature, [CK21], that also uses unique signatures and random-oracle-based selection. In our terminology, Mithril applies a decentralized ALBA scheme to unique signatures (possibly followed by compactification via succinct proof systems, as discussed in Sect. 1.4). In comparison to Mithril, our decentralized prover construction achieves significantly smaller proof sizes (when comparing with the simple concatenation version of [CK21]) at the cost of higher communication. In Sect. 4.1 we present a simple lottery that is asymptotically similar to Mithril with concatenation proofs, and offer a comparison in Sect. 8.

Straight-Line Witness Extraction for SNARKs. Ganesh et al. [GKO+23] addressed the problem of universal composability [Can00] for witness-succinct non-interactive arguments of knowledge. Universal composability requires the ability to extract the witness without rewinding the prover. However, since the proof is witness-succinct (i.e., shorter than the witness), the extractor must look elsewhere to obtain the witness. Building on the ideas of Pass [Pas03] and Fischlin [Fis05], Ganesh et al. proposed the following approach: the prover represents the witness as a polynomial of some degree d, uses a polynomial commitment scheme to commit to it, and then makes multiple random oracle queries on evaluations of this polynomial (together with proofs that the evaluations are correct with respect to the commitment) until it obtains some rare output of the random oracle (much like the Bitcoin proof of work). The prover repeats this process many times, and includes in the proof only the queries that result in the rare outputs. The verifier can be assured that the prover made more than d queries with high probability, because otherwise it would not be able to obtain sufficiently many rare outputs. Thus, the knowledge extractor can reconstruct the witness via polynomial interpolation by simply observing the prover's random oracle queries.

We observe that this approach really involves the prover trying to convince the verifier that the size of the set of random oracle queries is greater than d. This approach is just an ALBA protocol, but not a particularly efficient one. Applying our scheme instead of the one custom-built by Ganesh et al. results in less work for the prover. To get a proof of size $u \leq \lambda$, the protocol of Ganesh et al. requires the prover to compute $d \cdot u \cdot 2^{\lambda/u}$ polynomial evaluations and decommitment proofs,[3] whereas our Telescope construction from Sect. 3 requires only $d \cdot \lambda^{1/u} \cdot 2^{\lambda/u}$ of those.[4] Thus, our approach speeds up this part of prover's work by a factor of about u (which is close to the security parameter λ).

1.4 Relation to General-Purpose Witness-Succinct Proofs

In cases where the weight function can be realized by a program, one can use general-purpose witness-succinct proofs to tackle the construction of ALBA schemes via utilizing SNARKs [Gro16, GWC19].

These general purpose tools, however, are quite expensive, especially for the prover. First, the proving time can become impractical when the number of set elements in the witness is large. Second, given that the weight function W must be encoded as a circuit, the proving cost also depends heavily on the complexity of W. Moreover, W cannot always be specified as a circuit, but is evaluated by a more complex process—via a secure multi-party computation protocol or a human judge weighing the strength of the evidence.

On the other hand, these tools can give very short, even constant-size, proofs. To get the best of both worlds—prover efficiency and constant-size proofs—one can combine an ALBA proof with a witness-succinct proof of knowledge of the ALBA proof. This is indeed the approach proposed by Chaidos and Kiayias [CK21]: it first reduces witness size n_{f} to u by using very fast random-oracle-based techniques, and then has the prover prove u (instead of n_{f}) weight computations. We can also apply this technique to our constructions, something that can result in a constant size proof with a computationally efficient prover. And given that our constructions can work in the CRS model, one can avoid heuristically instantiating the random oracle inside a circuit.

2 Definitions

Below we present a definition of ALBA inspired by the non-interactive random oracle proof of knowledge (NIROPK) [BCS16] with straight-line extraction. To introduce arbitrary weights, we use a weight oracle $W : \{0,1\}^* \to \mathbb{N} \cup \{0\}$ and denote for a set S, $W(S) = \sum_{s \in S} W(s)$.

Definition 1. *The triple* (Prove, Verify, Extract) *is a* $(\lambda_{sec}, \lambda_{rel}, n_p, n_f)$*-NIROPK ALBA scheme if and only if*

- $\mathsf{Prove}^{H,W}$ *is a probabilistic program that has access to the random oracle* H *and a weight oracle* W*;*
- $\mathsf{Verify}^{H,W}$ *is a program that has access to the random oracle* H *and a weight oracle* W*;*

[3] This value follows from the formula $\lambda = r(b - \log d)$ in the "Succinctness" paragraph of [GKO+23, Section 3.1]. Note that r is u in our notation, and the expected number of random oracle queries by the prover is $r \cdot 2^b$. Solving the formula for b, we get $2^b = d2^{\lambda/r}$.

[4] This value is obtained by setting $n_{\mathrm{f}} = d$ and solving (1) for n_{p}.

– $\mathsf{Extract}^{H,W,\mathcal{A}}$ *is a probabilistic program that has access to the random oracle* H*, a weight oracle* W *and an adversary program* $\mathcal{A}$*;*
– *completeness: for all weight oracles* W *and all* S_p *such that* $W(S_p) \geq n_p$*,* $\Pr[\mathsf{Verify}^{H,W}(\mathsf{Prove}^{H,W}(S_p)) = 1] \geq 1 - 2^{-\lambda_{rel}}$*;*
– *proof of knowledge: consider the following experiment ExtractExp*$(\mathcal{A}^{H,W}, W)$*:*

```
$S_f \leftarrow \mathsf{Extract}^{H,W,\mathcal{A}}()$;
**output** 1 iff $W(S_f) > n_f$;
```

we require that for all weight oracles W *and all probabilistic oracle access programs* $\mathcal{A}^{H,W}$*,*

$$\Pr[\mathit{ExtractExp}(\mathcal{A}, W) = 1] \geq \Pr\left[\mathsf{Verify}^{H,W}\left(\mathcal{A}^{H,W}()\right) = 1\right] - 2^{-\lambda_{sec}};$$

moreover, $\mathsf{Extract}^{H,W,\mathcal{A}}()$ *is only allowed to run* $\mathcal{A}^{H,W}$ *once with the real* H *and* W *and only observes the transcript with its oracles (straight-line extraction property),* $\mathsf{Extract}$ *runs in time polynomial in the size of this transcript.*

As presented, this definition is non-adaptive; i.e., it does not allow W to depend on H; adaptivity can be added if it is possible to commit to W; see Sect. 6 for further discussion.

The above formulation of ALBAs captures the setting where a prover has the entire set S_p in its possession. We will also be interested in ALBAs where the prover is *decentralized*—by this we refer to a setting where a number of prover entities, each one possessing an element $s \in S_\mathrm{p}$ wish to act in coordination towards convincing the verifier. We now define a decentralized ALBA.

Definition 2. *The quadruple* $(\mathsf{Prove}, \mathsf{Aggregate}, \mathsf{Verify}, \mathsf{Extract})$ *is a* $(\lambda_{sec}, \lambda_{rel}, n_p, n_f)$*-decentralized NIROPK ALBA scheme if and only if*

– $\mathsf{Prove}^{H,W}$ *is a probabilistic program that has access to the random oracle* H *and a weight oracle* W*;*
– $\mathsf{Aggregate}^{H,W}$ *is a probabilistic program that has access to the random oracle* H *and a weight oracle* W*;*
– $\mathsf{Verify}^{H,W}$ *is a program that has access to the random oracle* H *and a weight oracle* W*;*
– $\mathsf{Extract}^{H,W,\mathcal{A}}$ *is a probabilistic program that has access to the random oracle* H*, a weight oracle* W *and an adversary program* $\mathcal{A}$*;*
– *completeness: consider the following experiment CompExp*(S_p, W)*:*

```
$S := \emptyset$;
**for** $s \in S_p$ **do**
    $m \leftarrow \mathsf{Prove}^{H,W}(s)$;
    **if** $m \neq \varepsilon$ **then**                    ▷ if m is not empty string
        $S := S \cup \{m\}$;
$\pi \leftarrow \mathsf{Aggregate}^{H,W}(S)$;
$r \leftarrow \mathsf{Verify}^{H,W}(\pi)$;
**return** $r$;
```

we require that for all weight oracles W and all S_p such that $W(S_p) \geq n_p$, $\Pr[\mathit{CompExp}(S_p, W) = 1] \geq 1 - 2^{-\lambda_{rel}}$*;*

- *proof of knowledge: consider the following experiment ExtractExp*$(\mathcal{A}^{H,W}, W)$*:*

```
$S_f \leftarrow \mathsf{Extract}^{H,W,\mathcal{A}}()$;
**output** 1 iff $W(S_f) > n_f$;
```

  we require that for all weight oracles W and all probabilistic oracle access programs $\mathcal{A}^{H,W}$,

$$\Pr[\mathit{ExtractExp}(\mathcal{A}, W) = 1] \geq \Pr\left[\mathsf{Verify}^{H,W}\left(\mathcal{A}^{H,W}()\right) = 1\right] - 2^{-\lambda_{sec}};$$

  moreover, $\mathsf{Extract}^{H,W,\mathcal{A}}()$ *is only allowed to run $\mathcal{A}^{H,W}$ once with the real H and W and only observes the transcript with its oracles (straight-line extraction property),* Extract *runs in time polynomial in the size of this transcript.*

In this model, we would like to minimize not only the proof size, but also the amount of communication characterized by the size of S in CompExp. Note that the above definition can be extended to multiple rounds of communication, but this is not something we explore in this work—all our decentralized constructions are "1-round."

Finally, we present a proof of knowledge ALBA definition in the CRS model. Unlike for NIROPK, the knowledge extractor here is allowed to rewind the adversary $\mathcal{A}$ and is given it as regular input. Note that the definition crucially requires the CRS to be independent of W; see Sect. 7 for further discussion.

Definition 3. (Prove, Verify, Extract, GenCRS) *is a* $(\lambda_{sec}, \lambda_{rel}, n_p, n_f)$*-CRS proof of knowledge ALBA scheme if and only if*

- Prove^W *is a probabilistic program;*
- Verify^W *is a program having access to a weight oracle W;*
- $\mathsf{Extract}^W$ *is a probabilistic program having access to a weight oracle W;*
- GenCRS *is a probabilistic program;*
- *completeness: consider the following experiment CompExp*(W, S_p)*:*

```
$crs \leftarrow \mathsf{GenCRS}()$;
$\pi \leftarrow \mathsf{Prove}(crs, S_p)$;
$r \leftarrow \mathsf{Verify}^W(crs, \pi)$;
return $r$;
```

  we require that for all weight oracles W and all S_p such that $W(S_p) \geq n_p$, $\Pr[\mathit{CompExp}(W, S_p) = 1] \geq 1 - 2^{-\lambda_{rel}}$*;*
- *proof of knowledge: consider the following experiment SoundExp*$(\mathcal{A}^W, W)$*:*

```
$crs \leftarrow \mathsf{GenCRS}()$;
$\pi \leftarrow \mathcal{A}^W(crs)$;
$r \leftarrow \mathsf{Verify}^W(crs, \pi)$;
return $r$;
```

we require that for all weight oracles W and all probabilistic oracle access programs $\mathcal{A}^W$, if $\mathcal{A}$ runs in time T and $\varepsilon = \Pr[SoundExp(\mathcal{A}^W, W) = 1] - 2^{-\lambda_{sec}} > 0$, then $S_f \leftarrow \mathsf{Extract}^W(\mathcal{A})$ runs in expected time poly$(T, 1/\varepsilon)$ and $\Pr\left[W(S_f) > n_f\right] = 1$.

3 Telescope ALBA

In this section we present two ALBA schemes in sequence. We start with a less efficient but simpler construction to illustrate the main idea. We then proceed to optimize the scheme's efficiency.

For both constructions, we will assume we have three random oracles H_0, H_1, and H_2 having particular output distributions. We explain how to implement these using a single random oracle which outputs binary strings in the full version [CKRZ23] of this work. Further, we initially restrict weights to be either 0 or 1, and generalize to integers in Sect. 5. Finally, we postpone showing the proof of knowledge property and instead consider a simpler notion of soundness: given n_f elements fixed in advance, what is the probability that a valid proof exists containing only those elements? Sect. 6 will then show how a knowledge extractor can be constructed.

3.1 Basic Construction

The main idea is as follows. Let d, u and q be parameters. The prover first considers all pairs consisting of an integer in $[d]$ and one of the elements of S_p and selects each of the $n_\mathrm{p}d$ pairs with probability $1/n_\mathrm{p}$. In expectation he will have d pairs selected. Now these pairs are treated as single units and they are paired with each element of S_p, resulting in triples that are selected again with probability $1/n_\mathrm{p}$. This process is repeated u times ending with, in expectation, d tuples consisting of one integer in $[d]$ and u set elements. Now, each of the tuples is selected with probability q and any selected tuple will be a valid proof.

More formally, let H_1 and H_2 be random functions returning 1 with probability $1/n_\mathrm{p}$ and q respectively, and returning 0 otherwise. Any tuple $(t, s_1, ..., s_u)$ such that

- $1 \leq t \leq d$;
- for all $1 \leq i \leq u$, $H_1(t, s_1, ..., s_i) = 1$;
- $H_2(t, s_1, ..., s_u) = 1$;

is a valid proof (see Sect. 3.3 how to implement H_1 efficiently). Define the program Verify accordingly.

Intuitively, this works because the honest prover maintains d tuples in expectation at each stage, while the malicious prover's tuples decrease $n_\mathrm{p}/n_\mathrm{f}$ times with each stage. However, to implement and analyze the prover algorithm, it will be convenient to represent all tuples $(t, s_1, ..., s_i)$, where $1 \leq t \leq d$, $0 \leq i \leq u$ and $s_1, ..., s_i \in S_\mathrm{p}$, as d trees of height u with $\{(1), ..., (d)\}$ being the roots of the trees and $\{(t, s_1, ..., s_u)\}_{1 \le t \le d, s_1, ..., s_u \in S_\mathrm{p}}$ being the leaves. To implement Prove, simply run depth first search (DFS) to find a "valid" path from a root to a leaf.

We will now analyze soundness of this construction. As mentioned above, the soundness error is defined to be the probability that a valid proof exists containing only elements from a fixed set S_f of size n_f.

Theorem 1. *Let*

$$u \ge \frac{\lambda_{sec} + \log(qd)}{\log \frac{n_p}{n_f}}.$$

Then soundness error is $\le 2^{-\lambda_{sec}}$.

Proof. We analyze soundness error, denoted by S, using simple union bound.

$$S \le \left(\frac{1}{n_\mathrm{p}}\right)^u \cdot q \cdot d \cdot n_\mathrm{f}^u = \left(\frac{n_\mathrm{f}}{n_\mathrm{p}}\right)^u \cdot qd.$$

Then

$$-\log S \ge -\left(u \log \frac{n_\mathrm{f}}{n_\mathrm{p}} + \log(qd)\right) = u \log \frac{n_\mathrm{p}}{n_\mathrm{f}} - \log(qd) \ge \lambda_\mathrm{sec}.$$

We now analyze completeness.

Theorem 2. *Let*

$$d \ge \frac{2u\lambda_{rel}}{\log e}; q = \frac{2\lambda_{rel}}{d \log e}.$$

Then completeness error is $\le 2^{-\lambda_{rel}}$ *and the probability that there exists a valid proof with a particular integer* t *is at least* $q - (u+1) \cdot \frac{q^2}{2}$.

Proof. Completeness can be described using the following recursive formula. For $0 \le k \le u$, let $f(k)$ be the probability that when fixing a prefix of an integer in $[d]$ and $u - k$ elements $t, s_1, ..., s_{u-k}$, there is no suffix of honest player's elements that works, meaning there is no $s_{u-k+1}, ..., s_u \in S_\mathrm{p}$ such that for all $u - k + 1 \le i \le u$, $H_1(t, s_1, ..., s_i) = 1$, and $H_2(t, s_1, ..., s_u) = 1$. Then one can see that

- $f(0) = 1 - q$;
- for $0 \le k < u$, $f(k+1) = \big((1 - \frac{1}{n_\mathrm{p}}) + \frac{1}{n_\mathrm{p}} \cdot f(k)\big)^{n_\mathrm{p}}$;
- the probability that there does not exist a valid proof with a particular integer t is $f(u)$;
- the probability that the algorithm fails in the honest case is $\big(f(u)\big)^d$.

This recursive formula can be approximated:

$$f(k+1) = \left(1 + \frac{1}{n_\mathrm{p}}\big(f(k) - 1\big)\right)^{n_\mathrm{p}} \le \left(e^{\frac{1}{n_\mathrm{p}}(f(k)-1)}\right)^{n_\mathrm{p}} = e^{f(k)-1}. \qquad (2)$$

It is convenient to look at the negative logarithm of this expression; we will prove by induction that $-\ln f(k) \geq q - k \cdot \frac{q^2}{2}$.
Basic case: $-\ln f(0) = -\ln(1-q) \geq -\ln(e^{-q}) = q$.
Inductive step: by Eq. 2,

$$-\ln f(k+1) \geq 1 - f(k) \geq 1 - e^{-\left(q-k\cdot\frac{q^2}{2}\right)} [\geq]$$

Using the values for d and q, one can see that $k \cdot \frac{q^2}{2} \leq u \cdot \frac{q^2}{2} \leq q$, then

$$[\geq] 1 - \left(1 - \left(q - k\cdot\frac{q^2}{2}\right) + \frac{\left(q-k\cdot\frac{q^2}{2}\right)^2}{2}\right) \geq$$
$$\left(q - k\cdot\frac{q^2}{2}\right) - \frac{q^2}{2} = q - (k+1)\cdot\frac{q^2}{2}.$$

Hence, $-\ln f(u) \geq q - u \cdot \frac{q^2}{2}$ and the probability that the honest prover fails is $\big(f(u)\big)^d \leq \exp\Big(-\big(q - u\cdot\frac{q^2}{2}\big)d\Big)$. Using the values for d and q, one can see that this is at most $2^{-\lambda_{\text{rel}}}$. Additionally, the probability that there exists a valid proof with a particular integer t is

$$1 - f(u) \geq e^{-\left(q-u\cdot\frac{q^2}{2}\right)} \geq$$
$$1 - \left(1 - \left(q - u\cdot\frac{q^2}{2}\right) + \frac{(q-u\cdot\frac{q^2}{2})^2}{2}\right) \geq q - (u+1)\cdot\frac{q^2}{2}.$$

□

Corollary 1. *Let*

$$u \geq \frac{\lambda_{sec} + \log\lambda_{rel} + 1 - \log\log e}{\log\frac{n_p}{n_f}}; d \geq \frac{2u\lambda_{rel}}{\log e}; q = \frac{2\lambda_{rel}}{d\log e}.$$

Then soundness error is $\leq 2^{-\lambda_{sec}}$ *and completeness error is* $\leq 2^{-\lambda_{rel}}$.

It is worth noting that the constant in d, and thus algorithm's running time, can be reduced. We show how to do this in the full version. Although the scheme still remains less efficient than the improved construction in Sect. 3.2, the optimizations can potentially be transferred over; we leave that for future work.

Running Time. In this section we analyze the prover's running time.

Assume S_{p} is a set with cardinality n_{p}. As mentioned above, all tuples $(j, s_1, ..., s_i)$ can be represented as d trees. We would like to analyze the number of "accessible" vertices in these trees. Let the indicator random variable

$$A_{j,s_1,...,s_i} = \begin{cases} 1 & \text{if for all } 1 \leq r \leq i, H_1(j, s_1, ..., s_r) = 1 \\ 0 & \text{otherwise.} \end{cases}$$

If $A_{j,s_1,\ldots,s_i} = 1$ we say the vertex $(j, s_1, \ldots, s_i)$ is accessible.

Let us first prove that the expected number of accessible vertices in a single tree at a particular height is 1.

Theorem 3. *For any j and $0 \leq i \leq u$,*

$$\mathbb{E}\left[\sum_{s_1,\ldots,s_i \in S_p} A_{j,s_1,\ldots,s_i}\right] = 1.$$

We present the proof in the full version [CKRZ23] of this work.

Assuming the prover runs DFS, Theorem 2 gives a bound on the expected number of evaluated trees. And by the above theorem, the algorithm invokes H_1 $n_\mathrm{p}u$ times and H_2 once in expectation per tree. Thus, the expected total number of hash evaluations shall be the product of the expected number of evaluated trees and $(n_\mathrm{p}u + 1)$. This, however, needs a more careful proof.

Theorem 4. *The expected number of hash evaluations is at most*

$$\left(q - (u+1) \cdot \frac{q^2}{2}\right)^{-1} (n_p u + 1)$$

We present the proof in the full version [CKRZ23] of this work.

Taking parameter values from Corollary 1 and letting $\lambda = \lambda_\mathrm{sec} = \lambda_\mathrm{rel}$, we thus obtain an expected number of hash evaluations of $O(n_\mathrm{p} \cdot \lambda^2)$.

We might also wish to have a tighter bound on the running time or on the number of accessible vertices to argue that an adversary cannot exploit an imperfect hash function or a PRF by making too many queries. Below we present a Chernoff style bound on the number of accessible non-root vertices in all d trees

$$Z = \sum_{\substack{1 \leq j \leq d, \\ 1 \leq i \leq u, \\ s_1,\ldots,s_i \in S_\mathrm{p}}} A_{j,s_1,\ldots,s_i}.$$

Note that $\mathbb{E}[Z] = du$.

Theorem 5.

$$\Pr[Z \geq (1+\delta)du] \leq \exp\left(-\frac{\delta^2}{4(1+\delta)} \cdot \frac{d}{u}\right).$$

We present the proof in the full version [CKRZ23] of this work.

Taking parameter values from Corollary 1 and letting $\lambda = \lambda_\mathrm{sec} = \lambda_\mathrm{rel}$, we thus conclude that the algorithm does $O(n_\mathrm{p} \cdot \lambda^3)$ hash evaluations with overwhelming probability.

3.2 Construction with Prehashing

The basic scheme described above has proving expected time $O(n_\mathrm{p} \cdot \lambda^2)$ and verification time $O(\lambda)$ if we let $\lambda = \lambda_\mathrm{sec} = \lambda_\mathrm{rel}$. The modification described in this section has proving expected time $O(n_\mathrm{p} + \lambda^2)$ and verification time is unchanged.

The improvement is inspired by balls-and-bins collisions. Whereas in the previous scheme for every tuple we tried each of n_p possible extensions, here we hash tuples to a uniform value in $[n_\mathrm{p}]$ and hash individual set elements to a uniform value in $[n_\mathrm{p}]$, and consider a valid extension to be such that the tuple and the extension both hash to the same value. More formally, we have random functions H_0 and H_1 producing a uniformly random value in $[n_\mathrm{p}]$ and hash function H_2 returning 1 with probability q and 0 otherwise, and consider a tuple $(t, s_1, ..., s_u)$ a valid proof if and only if

- $1 \leq t \leq d$;
- for all $1 \leq i \leq u$, $H_1(t, s_1, ..., s_{i-1}) = H_0(s_i)$;
- $H_2(t, s_1, ..., s_u) = 1$;

(see Sect. 3.3 how to implement H_1 efficiently). Define the Verify program accordingly.

As before, we have d valid tuples in expectation at each stage but by precomputing $H_0(\cdot)$ (balls to bins) we avoid trying all n_p extensions for a tuple. The analysis of completeness, however, is more complicated. Before, we assumed in the recursive formula that failure events for each element extension are all independent. Here, it is not true: the fact that one extension eventually succeeds can tell that the balls-to-bins are well distributed. Indeed, if each bin gets exactly one ball, then there will always be a tuple that succeeds except maybe for the requirement that $H_2(\cdot) = 1$. However, if all balls land in one bin, then the success probability is smaller. To get rid of this dependency, we can however fix the balls-to-bins arrangement. Then such events become independent again.

The proof has two parts: the first one says that if the arrangement of the balls is "nice", then with high probability the honest player succeeds. The second part proves that we get a "nice" distribution of balls with high probability. The "nice" property itself is artificial, but one can notice that if the number of bins of size s is exactly the expected number of bins of size s if the size of each bin is a Poisson random variable with mean 1, then the analysis of completeness becomes very similar to that of the previous scheme. By using Poisson approximation and Chernoff like analysis, we can show that the property we care about does hold with high probability. We need, however, assume that the number of set elements n_p is large enough (on the order of λ^3). Alternatively, we can generate multiple balls per set element.

We first analyze soundness. As mentioned previosly, we define soundness error to be the probability that a valid proof can be constructed using elements S_f with $|S_\mathrm{f}| = n_\mathrm{f}$ (simple soundness).

Theorem 6. *Let*

$$u \geq \frac{\lambda_{sec} + \log(qd)}{\log \frac{n_p}{n_f}}.$$

Then soundness error is $\leq 2^{-\lambda_{sec}}$.

We present the proof in the full version [CKRZ23] of this work.

Theorem 7. *Assume*

$$d \geq \frac{16u(\lambda_{rel} + \log 3)}{\log e}; q = \frac{2(\lambda_{rel} + \log 3)}{d \log e}; n_p \geq \frac{d^2 \log e}{9(\lambda_{rel} + \log 3)}. \qquad (3)$$

Then completeness error is $\leq 2^{-\lambda_{rel}}$.

We present the proof in the full version [CKRZ23] of this work.

Corollary 2. *Assume*

$$u \geq \frac{\lambda_{sec} + \log(\lambda_{rel} + \log 3) + 1 - \log\log e}{\log \frac{n_p}{n_f}}; d \geq \frac{16u(\lambda_{rel} + \log 3)}{\log e};$$

$$q = \frac{2(\lambda_{rel} + \log 3)}{d \log e}; n_p \geq \frac{d^2 \log e}{8(\lambda_{rel} + \log 3)}.$$

Then soundness is $\leq 2^{-\lambda_{sec}}$ *and completeness error is* $\leq 2^{-\lambda_{rel}}$.

It is worth noting that the $n_{\mathrm{p}} \geq \Omega(\lambda^3)$ requirement can be removed as follows. Instead of doing Chernoff like analysis, that requires large n_{p}, one can utilize Markov's inequality to show that the "nice" arrangement of balls into bins happens with moderate probability, e.g. $\frac{3}{4}$, to achieve a scheme with completeness $\frac{1}{2}$. Such a scheme can then be amplified to achieve arbitrary λ_{rel} by setting $\lambda_{\mathrm{sec}} := \lambda_{\mathrm{sec}} + \log \lambda_{\mathrm{rel}}$ and having the verifier accept any one of λ_{rel} independent proofs. As a result, the expected running time is no longer $\Omega(\lambda^3)$ but $O(n_{\mathrm{p}} + \lambda^2)$, but we need to apply H_0 to all elements twice in expectation as opposed to exactly once. Hence, for large n_{p} it still makes sense to use the algorithm as described at the beginning of the section. The Markov analysis can be found in the full version [CKRZ23] of this paper.

Running Time. In this section we analyze the prover's running time. Assume S_{p} is a set with cardinality n_{p}. As described in Sect. 3.1, all tuples $(j, s_1, ..., s_i)$ can be represented as d trees of height u. We would like to analyze the number of "accessible" vertices in these trees. Let the indicator random variable

$$A_{j,s_1,\ldots,s_i} = \begin{cases} 1 & \text{if for all } 1 \leq r \leq i, H_1(j, s_1, ..., s_{r-1}) = H_0(s_r) \\ 0 & \text{otherwise.} \end{cases}$$

If $A_{j,s_1,\ldots,s_i} = 1$ we say the vertex $(j, s_1, ..., s_i)$ is accessible.

Similarly to Sect. 3.1, one can prove that the expected number of accessible vertices in a single tree at a particular height is 1. This holds independently of the value of H_0!

Theorem 8. *For any j and $0 \le i \le u$,*

$$\mathbb{E}\left[\sum_{s_1,\ldots,s_i \in S_p} A_{j,s_1,\ldots,s_i} \middle| H_0\right] = 1.$$

We will now analyze the expected running time of the prover. The hash function H_0 is invoked exactly n_p times, so we will only upper bound the expected number of invocations of H_1 and H_2.

Theorem 9. *The expected number of invocations of H_1 and H_2 is at most*

$$\frac{u+1}{1-e^{-q+4uq^2}} + 2e^{-\frac{9}{4}n_p q^2} \cdot d(u+1).$$

We present the proof in the full version [CKRZ23] of this work.

Taking parameter values from Corollary 2 and letting $\lambda = \lambda_\text{sec} = \lambda_\text{rel}$, we thus get expected number of evaluations of H_1 and H_2 $O(\lambda^2)$. This is dominated by n_p invocations of H_0 since n_p is assumed to be $\Omega(\lambda^3)$.

Below we also present a tight bound on the number of accessible non-root vertices in all d trees

$$Z = \sum_{\substack{1 \le j \le d,\\ 1 \le i \le u,\\ s_1,\ldots,s_i \in S_\text{p}}} A_{j,s_1,\ldots,s_i}.$$

Note that $\mathbb{E}[Z] = du$.

Theorem 10. *Let*

$$\lambda > 0; \quad \lambda' = \frac{\lambda+2}{\log e}; \quad n_p \ge \frac{u^2\lambda'}{2};$$
$$u! \cdot \frac{u - e^{\frac{1}{3u}}}{u^3} \ge 72e^{-\frac{2}{3}} \cdot 2^\lambda; \quad \delta = e^{1+\sqrt{\frac{18u^2\lambda'}{n_p}}}\left(\frac{3u\lambda'}{d} + 1\right).$$

Then

$$\Pr[Z \ge \delta du] \le 2^{-\lambda}.$$

We present the proof in the full version [CKRZ23] of this work.

Taking parameter values from Corollary 2 and letting $\lambda = \lambda_\text{sec} = \lambda_\text{rel}$, we thus conclude that the prover algorithm evaluates H_1 and H_2 $O(\lambda^3)$ times with overwhelming probability.

3.3 Implementing Random Oracles with Long Inputs

We describe our protocols assuming a random oracle H_1 that can accommodate inputs of any length, which, in particular, implies independence of outputs for inputs of different lengths. However, to have an accurate accounting for running times, one has to charge for the cost of running a random oracle in proportion to the input length. Because the Telescope construction runs $H_1(j)$, $H_1(j, s_1)$, $H_1(j, s_1, s_2)$, $H_1(j, s_1, s_2 \ldots, s_u)$, the cost of just one u-tuple is quadratic in u. To reduce this cost to linear (thus saving a factor of u in running time), we will implement $H_1(j, s_1, \ldots, s_{i+1})$ to reuse most of the computation of $H_1(j, s_1, \ldots, s_i)$. The most natural way to do so is to slightly modify the Merkle-Damgård construction: use a two-input random oracle f ("compression function") with a sufficiently long output and a function g that maps the range of f to the distribution needed by H_1. We present the details of implementing g in the full version of this work. Inductively define $H_1'(j, s_1, \ldots, s_{i+1}) = f(H_1'(j, s_1, \ldots, s_i), s_{i+1})$ and let $H_1(x) = g(H_1'(x))$.

While not indifferentiable from a random oracle (see Coron et al. [CDMP05] for similar constructions that are), this construction suffices for our soundness and extractability arguments, because those arguments need independence only for a single chain (they handle multiple different chains by the union bound). Neither length extension attacks nor collisions are important. Completeness suffers very slightly by the probability of f-collisions, which can be made negligible by making the output of f large enough and using the bound on the number of queries made by the honest prover (Theorems 5 and 10).

3.4 Optimality of the Certificate Size

In this section, we show that the number of set elements u included in a proof is essentially optimal for our constructions. Because our construction works for a black-box weight function that formally is implemented via an oracle (and in reality may be implemented by MPC, a human judge, etc.), the verifier must query the weight function on some values; else the verifier has no knowledge of whether any values in the prover's possession have any weight.

Thus, for the sake of proving optimality, we consider only protocols that make this part of verification explicit. We define an algorithm Read (see the definition below) that takes a proof and returns set elements; these set elements must have been in the prover's possession. We bound the proof size in terms of the number of set elements returned by Read, showing that if it is too small, the protocol cannot be secure. We also note that the following definition can be used for upper bound results too, as demonstrated in Sect. 7 for the CRS model.

Definition 4. (Prove, Read, Verify) *is a* $(\lambda_{sec}, \lambda_{rel}, n_p, n_f)$*-ALBA scheme if and only if*

- Prove^H *is a probabilistic random oracle access program;*
- Verify^H *is a random oracle access program;*
- Read *is a program;*
- *completeness: consider the following experiment CompExp*(S_p)*:*

  $\pi \leftarrow \mathsf{Prove}^H(S_p)$*;*

  output 1 *iff* $\mathsf{Read}(\pi) \subseteq S_p$ *and* $\mathsf{Verify}^H(\pi) = 1$*;*

  we require that for all sets S_p *with size* $\geq n_p$*,* $\Pr[\mathit{CompExp}(S_p) = 1] \geq 1 - 2^{-\lambda_{rel}}$*.*

– *soundness: consider the following experiment SoundExp*(S_f)*:*

  output 1 *iff* $\exists \pi, \mathsf{Read}(\pi) \subseteq S_f \wedge \mathsf{Verify}^H(\pi) = 1$*;*

  we require that for all sets S_f *with size* $\leq n_f$*,* $\Pr[\mathit{SoundExp}(S_f) = 1] \leq 2^{-\lambda_{sec}}$*;*

We now prove a lower bound for a scheme satisfying this definition.

Theorem 11. *Assume* $\lambda_{rel} \geq 1$*, define* $\alpha = \frac{\lambda_{sec}-3}{\log(n_p/n_f)}$*, assume* $n_f \geq 3\alpha^2$*, let* S_p *be an arbitrary set of size* n_p*, and let* $(\mathsf{Prove}, \mathsf{Read}, \mathsf{Verify})$ *be a* $(\lambda_{sec}, \lambda_{rel}, n_p, n_f)$*-ALBA scheme. Then*

$$\Pr\left[\left|\mathsf{Read}(\mathsf{Prove}^H(S_p))\right| > \alpha\right] \geq \frac{1}{4}.$$

We present the proof in the full version [CKRZ23] of this work.

4 ALBAs with Decentralized Prover

In the previous section we assumed the ALBA prover has all the set elements at hand. In many applications however, such as threshold signatures, this is not the case. The set elements may be spread across numerous parties who will then jointly compute a proof. A trivial solution is to use a centralized protocol, by designating one of the parties as the lead prover and have all other parties communicate their set elements to that party. However, this incurs a communication cost equal to the size of the set, which we would rather avoid.

In this section we present protocols where the various parties holding set elements start out by performing computations locally and only conditionally communicate their elements to a designated prover or aggregator. Whilst our constructions we present in this section still use weights of 0 or 1, they can be generalized to integer weights as explained in Sect. 5. Finally, as in Sect. 3, instead of proof of knowledge we consider a simpler notion of soundness: the probability that a valid proof exists containing only elements from set S_f of size n_f. Section 6 demonstrates how to do knowledge extraction.

4.1 Simple Lottery Construction

The simple lottery scheme is parametrized by the expected number of network participants μ. Let H be a random oracle that outputs 1 with probability $p = \frac{\mu}{n_\mathrm{p}}$ and 0 otherwise. Each set element s is sent to the aggregator over the network if and only if $H(s) = 1$. Now let $r_\mathrm{s}, r_\mathrm{c} > 1$ such that $r_\mathrm{s} r_\mathrm{c} = \frac{n_\mathrm{p}}{n_\mathrm{f}}$ and set $u = r_\mathrm{s} \cdot p n_\mathrm{f}$ (or equivalently $u = \frac{p n_\mathrm{p}}{r_\mathrm{c}}$). The aggregator needs to collect and concatenate u set elements and the verifier accepts if it receives u values that each hash to 1.

Lemma 1. *Assuming*

$$u \geq \frac{\lambda_{sec} \cdot \ln 2}{\ln r_s - 1 + \frac{1}{r_s}},$$

soundness error of the scheme is $\leq 2^{-\lambda_{sec}}$*.*

We present the proof in the full version [CKRZ23] of this work.

Lemma 2. *Assuming*

$$u \geq \frac{\lambda_{rel} \cdot \ln 2}{r_c - 1 - \ln r_c},$$

completeness error of the scheme is $\leq 2^{-\lambda_{rel}}$.

We present the proof in the full version [CKRZ23] of this work.

Thus, to minimize u, we need to minimize

$$\max\left\{\frac{\lambda_{\mathrm{sec}} \cdot \ln 2}{\ln r_{\mathrm{s}} - 1 + \frac{1}{r_{\mathrm{s}}}}, \frac{\lambda_{\mathrm{rel}} \cdot \ln 2}{r_{\mathrm{c}} - 1 - \ln r_{\mathrm{c}}}\right\}.$$

Noting that the first term is decreasing with respect to r_{s} and the second term is decreasing with respect to r_{c}, the minimum is achieved when the two terms are equal. If $\lambda_{\mathrm{sec}} = \lambda_{\mathrm{rel}} = \lambda$, then setting $r_{\mathrm{c}} = \frac{n_{\mathrm{p}}}{n_{\mathrm{p}} - n_{\mathrm{f}}} \cdot \ln \frac{n_{\mathrm{p}}}{n_{\mathrm{f}}}$ and $r_{\mathrm{s}} = \frac{n_{\mathrm{p}} - n_{\mathrm{f}}}{n_{\mathrm{f}}} \cdot \frac{1}{\ln \frac{n_{\mathrm{p}}}{n_{\mathrm{f}}}}$ gives the smallest u.

We note the interesting fact that choosing r_{s} and r_{c} that minimize u also minimizes μ. Since $\mu = pn_{\mathrm{p}} = ur_{\mathrm{c}}$, we have

$$\mu \geq \max\left\{\frac{\lambda_{\mathrm{sec}} \cdot \ln 2}{\ln r_{\mathrm{s}} - 1 + \frac{1}{r_{\mathrm{s}}}} \cdot r_{\mathrm{c}}, \frac{\lambda_{\mathrm{rel}} \cdot \ln 2}{r_{\mathrm{c}} - 1 - \ln r_{\mathrm{c}}} \cdot r_{\mathrm{c}}\right\}.$$

The first term is decreasing with respect to r_{s} since r_{c} is, and it can be seen that the second term is decreasing with respect to r_{c}. Hence, μ is minimized when the two terms are equal which is the same as the condition for minimizing u.

4.2 Decentralized Telescope

The next logical step to minimize the size of the proof is to run a smarter aggregator, Telescope, and calculate an appropriate increase to the security and reliability parameters. While combining a simple lottery with an ALBA aggregator is a generic technique, the generic analysis requires one to calculate two lottery tail bounds: one for soundness and one for completeness. By using Telescope for the aggregator, we benefit from omitting the soundness tail bounds from analysis; this section has all details.

As previously, we have parameter μ and select each element to be transmitted over the network with probability μ/n_{p}. After receiving enough elements selected by the simple lottery, the aggregator runs the algorithm from Sect. 3.2. Note that it assumes that the number of set elements is large enough, see a bound on n_{p} in Theorem 7. Since μ can be much smaller than said bound, we artificially increase the number of set elements by producing k sub-elements for each element, for an appropriate k. This is a purely technical hindrance that goes away in the full version [CKRZ23] of this paper which describes an ALBA scheme without this requirement.

We employ threshold analysis here: calculate the number of set elements selected by the simple lottery such that 1) this number is achievable with probability $1-\frac{1}{4}\cdot 2^{-\lambda_{\mathrm{rel}}}$ and 2) the Telescope aggregator will produce a valid certificate with probability $1-\frac{3}{4}\cdot 2^{-\lambda_{\mathrm{rel}}}$.

For all $1 \le i \le n_{\mathrm{p}}$, let X_i be 1 if and only if element s_i is selected and 0 otherwise. Let $X = \sum_{i=1}^{n_{\mathrm{p}}} X_i$; then $\mathbb{E}[X] = \mu$. Assume $\rho \in \mathbb{N}$ satisfies $\Pr[X \ge \rho] \ge 1 - 2^{-\lambda_{\mathrm{rel}}-2}$. Reducing the honest-malicious gap from $\frac{n_{\mathrm{p}}}{n_{\mathrm{f}}}$ to $\frac{\rho}{\frac{\mu}{n_{\mathrm{p}}}\cdot n_{\mathrm{f}}} = \frac{n_{\mathrm{p}}}{n_{\mathrm{f}}}\cdot\frac{\rho}{\mu}$ results in increasing the certificate size to

$$\frac{\lambda_{\mathrm{sec}} + \log(\lambda_{\mathrm{rel}}+2) + 1 + \log e - \log\log e}{\log\frac{n_{\mathrm{p}}}{n_{\mathrm{f}}} + \log\frac{\rho}{\mu}}$$

(we have $\lambda_{\mathrm{sec}} + \log(\lambda_{\mathrm{rel}}+2)+1+\log e - \log\log e$ instead of $\lambda_{\mathrm{sec}} + \log(\lambda_{\mathrm{rel}} + \log 3) + 1 - \log\log e$ in Theorem 2 because we instantiate it with $\lambda_{\mathrm{sec}} := \lambda_{\mathrm{sec}} + \log e$ and $\lambda_{\mathrm{rel}} := \lambda_{\mathrm{rel}} + \log\frac{4}{3}$ for technical reasons).

One can think of the gap $\frac{\rho n_{\mathrm{p}}}{\mu n_{\mathrm{f}}}$ as $\frac{(1-\delta)n_{\mathrm{p}}}{n_{\mathrm{f}}}$ if we set $\rho = (1-\delta)\mu$. Note that we only decrease n_{p} in the $\frac{n_{\mathrm{p}}}{n_{\mathrm{f}}}$ gap. n_{f} remains the same since the union bound argument for soundness still works, but with some modifications. Particularly, it requires μ to be on the order of u^2. If we wanted to decrease μ even further, we could improve the proof below or employ a two-sided threshold analysis as well.

Let Lottery : $\{0,1\}^* \to \{0,1\}$ be an oracle returning 1 with probability $\frac{\mu}{n_{\mathrm{p}}}$ and assume $H = (H_0, H_1, H_2, \text{Lottery})$ where H_0, H_1, H_2 are as defined in Sect. 3.2. Also let $A.\mathsf{Prove}^H$, $A.\mathsf{Verify}^H$ be as in Sect. 3.2 and define the following.

```
procedure $B.\mathsf{Prove}^H(s)$
    if Lottery$(s) = 1$ then
        return $s$;
    else
        return empty string;

procedure $B.\mathsf{Aggregate}^H(S)$
    return $A.\mathsf{Prove}^H(S)$;
procedure $B.\mathsf{Verify}^H(\pi)$
    parse $(t, s_1, ..., s_u) = \pi$;
    return 1 iff $A.\mathsf{Verify}^H(\pi) = 1 \wedge$
    $\forall 1 \le i \le u$, Lottery$(s_i) = 1$;
```

Theorem 12. *Assume*

$$k \ge \frac{d^2 \log e}{9\rho(\lambda_{rel}+2)}$$

and instantiate the algorithm in Sect. 3.2 with $d \ge \frac{16u(\lambda_{rel}+2)}{\log e}$, $q := \frac{2(\lambda_{rel}+2)}{d\log e}$, *and* $n_p := k\rho$. *Then completeness error is* $\le 2^{-\lambda_{rel}}$.

Proof. As assumed above, the simple lottery chooses at least ρ set elements with probability at least $1 - 2^{-\lambda_{\mathrm{rel}}-2}$. Given this event, by Theorem 7, the algorithm outputs a valid certificate with probability at least $1 - 2^{-\lambda_{\mathrm{rel}} - \log\frac{4}{3}}$. Therefore, completeness error is $\le 2^{-\lambda_{\mathrm{rel}}}$. □

We now calculate soundness error defined as the probability that a valid proof can be constructed using elements S_{f} with $|S_{\mathrm{f}}| = n_{\mathrm{f}}$.

Theorem 13. *Assume*

$$\mu \geq \frac{n_p u^2}{n_f}; \qquad \frac{\rho n_p}{\mu n_f} > 1;$$
$$u \geq \frac{\lambda_{sec} + \log(\lambda_{rel} + 2) + 1 + \log e - \log\log e}{\log \frac{n_p}{n_f} + \log \frac{\rho}{\mu}}.$$

Then soundness error is $\leq 2^{-\lambda_{sec}}$.

We present the proof in the full version [CKRZ23] of this work.

Theorem 12 and 13 give

Corollary 3. *Assume*

$$\mu \geq \frac{n_p u^2}{n_f}; \qquad \frac{\rho n_p}{\mu n_f} > 1; \qquad k \geq \frac{d^2 \log e}{9\rho(\lambda_{rel} + 2)};$$
$$u \geq \frac{\lambda_{sec} + \log(\lambda_{rel} + 2) + 1 + \log e - \log\log e}{\log \frac{n_p}{n_f} + \log \frac{\rho}{\mu}}$$

and instantiate the algorithm in Sect. 3.2 with $u := u$, $d \geq \frac{16u(\lambda_{rel}+2)}{\log e}$, $q := \frac{2(\lambda_{rel}+2)}{d \log e}$, *and* $n_p := k\rho$. *Then soundness error is* $\leq 2^{-\lambda_{sec}}$ *and completeness error is* $\leq 2^{-\lambda_{rel}}$.

Using this, we can see how big μ needs to be if we increase $u \log \frac{n_\mathrm{p}}{n_\mathrm{f}}$ only by some amount C. To calculate a suitable ρ, we just use a Chernoff bound: $\Pr[X \leq (1-\delta)\mu] \leq e^{-\frac{\mu\delta^2}{2}}$. Setting this to $2^{-\lambda_\mathrm{rel}-2}$, we get $\delta = \sqrt{\frac{2(\lambda_\mathrm{rel}+2)}{\mu \log e}}$. We now set $\rho = \lceil (1-\delta)\mu \rceil$.

Corollary 4. *Assume*

$$C > 0; \quad u \geq \frac{\lambda_{sec} + \log(\lambda_{rel} + 2) + 1 + \log e - \log\log e + C}{\log \frac{n_p}{n_f}}; \quad k \geq \frac{d^2 \log e}{9\rho(\lambda_{rel} + 2)};$$
$$\mu \geq \max\left\{ \frac{8(\lambda_{rel} + 2)}{\log e}, \frac{n_p u^2}{n_f}, \frac{9u^2(\lambda_{rel} + 2)\log e}{2C^2} \right\}; \mu > \frac{2(\lambda_{rel} + 2)}{\left(1 - \frac{n_f}{n_p}\right)^2 \log e};$$

and instantiate the algorithm in Sect. 3.2 just like in corollary 3. Then soundness error is $\leq 2^{-\lambda_{sec}}$ *and completeness error is* $\leq 2^{-\lambda_{rel}}$.

We present the proof in the full version [CKRZ23] of this work.

Thus, if we let $\lambda = \lambda_\mathrm{sec} = \lambda_\mathrm{rel}$ and let u only be a constant larger than optimal, we have $\mu = O(\lambda^3)$ as well as the time complexity of the centralized algorithm also $O(\lambda^3)$. Moreover, μ is proportional to $\frac{1}{C^2}$. We note, however, that setting $\lambda_\mathrm{rel} := 1$ and $\lambda_\mathrm{sec} := \lambda_\mathrm{sec} + \log \lambda_\mathrm{rel}$ and amplifying the completeness via parallel repetitions as mentioned in Sect. 3.2 lets us reduce the expected communication complexity to $O(\lambda^2)$. However, it requires some network engineering to avoid redundant communication, specifically delaying repetitions until the previous ones have certainly failed.

In the full version [CKRZ23], we also present a different corollary showing what u needs to be in terms of μ.

4.3 Optimality of the Certificate Size - Communication Tradeoff

We can attempt to find a lower bound for the tradeoff between the certificate size u and μ. For this purpose, we use the following definition.

Definition 5. $(\mathsf{Prove}, \mathsf{Read}, \mathsf{Verify})$ *is a* $(\lambda_{sec}, \lambda_{rel}, n_p, n_f, \mu)$*-lottery based ALBA scheme if and only if*

- Prove^H *is a probabilistic random oracle access program;*
- Verify^H *is a random oracle access program;*
- Read *is a program;*
- *if* L *is a random function such that for all* x, $\Pr[L(x) = 1] = \frac{\mu}{n_p}$ *and we define* $Lottery(S) = \{x \in S : L(x) = 1\}$, *then*
  - *completeness: consider the following experiment* $CompExp(S_p)$*:*

    $\pi \leftarrow \mathsf{Prove}^H(Lottery(S_p))$;

    output 1 *iff* $\mathsf{Read}(\pi) \subseteq Lottery(S_p)$ *and* $\mathsf{Verify}^H(\pi) = 1$;

    we require that for all sets S_p *with size* $\geq n_p$, $\Pr[CompExp(S_p) = 1] \geq 1 - 2^{-\lambda_{rel}}$.
  - *soundness: consider the following experiment* $SoundExp(S_f)$*:*

    output 1 *iff* $\exists \pi, \mathsf{Read}(\pi) \subseteq Lottery(S_f) \wedge \mathsf{Verify}^H(\pi) = 1$;

    we require that for all sets S_f *with size* $\leq n_f$, $\Pr[SoundExp(S_f) = 1] \leq 2^{-\lambda_{sec}}$;

The following theorem presents our lower bound.

Theorem 14. *Assume* ρ *satisfies* $\Pr\left[B(n_p, \frac{\mu}{n_p}) \leq \rho\right] \geq 2^{-\lambda_{rel}+1}$ *where* $B(n,p)$ *is a binomial random variable with* n *experiments each with probability of success* p*. Also assume*

$$\frac{\rho n_p}{\mu n_f} > 1; \quad \mu \geq \frac{3u^2 n_p \log e}{2n_f}; \quad n_f \geq \rho,$$

let S_p *be an arbitrary set of size* n_p *and let* $(\mathsf{Prove}, \mathsf{Read}, \mathsf{Verify})$ *be a* $(\lambda_{sec}, \lambda_{rel}, n_p, n_f, \mu)$*-lottery based ALBA scheme such that*

$$\Pr\left[\left|\mathsf{Read}(\mathsf{Prove}^H(Lottery(S_p)))\right| \leq u\right] = 1.$$

Then

$$u > \frac{\lambda_{sec} - 4}{\log \frac{n_p}{n_f} + \log \frac{\rho}{\mu}}.$$

We present the proof in the full version [CKRZ23] of this work.

Using this, we can establish a lower bound similar to the upper bound Corollary 4.

Corollary 5. *Let $C > 0$, define*

$$\alpha = \frac{\lambda_{sec} - 4 + C}{\log \frac{n_p}{n_f}}; u = \lfloor \alpha \rfloor$$

and assume

$$\max\left\{\frac{4}{\lambda_{rel}}, \frac{\lambda_{rel}}{(1-\frac{n_f}{n_p})^2}, \frac{3u^2 n_p \log e}{2n_f}\right\} \le \mu \le \min\left\{\frac{\alpha^2 \lambda_{rel} \log^2 e}{4\,C^2}, \frac{(\frac{4}{e})^{\lambda_{rel}}}{4e^{10}}\right\};$$

$$n_f \ge 2\mu.$$

Let S_p be an arbitrary set of size n_p and let $(\mathsf{Prove}, \mathsf{Read}, \mathsf{Verify})$ *be a* $(\lambda_{sec}, \lambda_{rel}, n_p, n_f, \mu)$*-lottery based ALBA scheme. Then*

$$\Pr\left[\left|\mathsf{Read}(\mathsf{Prove}^H(\mathit{Lottery}(S_p)))\right| > \alpha\right] > 0.$$

We present the proof in the full version [CKRZ23] of this work.

Alternatively, in the full version we also present a corollary showing a lower bound on the certificate size as of function of μ.

5 Adding Weights

We will assume, without loss of generality, that the weight function W outputs integers. A naive way to handle weights other than 0 and 1 is to interpret each set element s as $W(s)$ elements $(s, 1), \dots, (s, W(s))$ and apply schemes designed for the unweighted case to (s, i) pairs. Unfortunately, this approach makes the prover running time linear in the total weight which could be in the order of 2^{64}.

Fortunately, any lottery-based scheme in which the number of lottery winners is independent of n_p (or at most polylogarithmic in n_p) is amenable to a more efficient solution (and the Telescope scheme in Sect. 3 can be turned into a lottery-based scheme first using Sect. 4.2). We simply view $(s, 1), \dots, (s, W(s))$ pairs as $W(s)$ different lottery participants. For efficiency, instead of having each of them play the lottery individually with probability p, we sample the number of winners from the binomial distribution $\mathrm{Binom}(W(s), p)$ (similar to the sortition algorithm used in Algorand [GHM+17]). We do so because it does not matter which i values win—what matters is only the number of winners. If the binomial sampling returns k, then $(s, 1), \dots, (s, k)$ are considered winners. This does not increase the complexity compared to the unweighted-lottery-based scheme, except for binomial sampling rather than lottery applied to each element.

Using Corollary 4 with $\lambda_\mathrm{sec} := \lambda_\mathrm{sec} + \log \lambda_\mathrm{rel}$, $\lambda_\mathrm{rel} := 1$ and $C := 1$, we achieve a weighted $(\lambda_\mathrm{sec}, \lambda_\mathrm{rel}, n_\mathrm{p}, n_\mathrm{f})$-ALBA scheme with proof size

$$\frac{\lambda_\mathrm{sec} + \log \lambda_\mathrm{rel} + 2 + \log 3 + \log e - \log\log e}{\log \frac{n_\mathrm{p}}{n_\mathrm{f}}}$$

and expected prover running time $O(n + \lambda_\mathrm{sec}^2)$, where n is the number of weighted elements in the input.

6 Knowledge Extraction for NIROPK

In this section we show how Definitions 1 and 2 can be realized. While Sects. 3 and 4 provide intuitive constructions with clean combinatorial analysis, they have a missing piece — a knowledge extractor. As we will see, the simple soundness proven there does not immediately imply proof of knowledge, and more reasoning is needed. We also remind that the knowledge extractor must be straight-line; i.e., rewinding of the prover is not allowed but observing its queries to the oracles is. Here we describe the full NIROPK scheme including its knowledge extactor for the case of the basic Telescope construction from Sect. 3.1 while other Telescope constructions can be made NIROPK in a similar fashion. For $H = (H_1, H_2)$, define

```
procedure Prove^{H,W}(S_p)
    run DFS as described in Section 3.1
procedure Verify^{H,W}(π)
    parse $(t, s_1, ..., s_u) = \pi$
    return 1 iff
        – $1 \le t \le d$;
        – $\forall 1 \le i \le u$, $H_1(t, s_1, ..., s_i) = 1$;
        – $H_2(t, s_1, ..., s_u) = 1$;
        – $\forall 1 \le i \le u$, $W(s_i) = 1$

procedure Extract^{H,W,A}
    function $A_1^{H,W}$
        $\pi \leftarrow \mathcal{A}^{H,W}()$;
        $v \leftarrow \mathsf{Verify}^{H,W}(\pi)$;
        return $\pi$;
    run $\mathcal{A}_1^{H,W}()$ and observe its oracles transcript $\tau$;
    $S_f := \emptyset$;
    for $x$ queried to $H_1$ or $H_2$ in $\tau$ do
        if $W(x) = 1$ then
            add $x$ to $S_f$;
    return $S_f$.
```

Theorem 15. *Define parameters as in Theorem 1. Then algorithms* $\mathsf{Verify}^{H,W}$ *with* $\mathsf{Extract}^{H,W,\mathcal{A}}$ *satisfy the proof of knowledge property of Definition 1.*

Proof. The extractor succeeds whenever $\mathcal{A}$ succeeds, unless $\mathcal{A}$ succeeds after querying fewer than n_f elements of S, which happens with probability at most $2^{-\lambda_{sec}}$ by the following lemma. Thus, the proof of knowledge property follows by the union bound.

We present the proof in the full version [CKRZ23] of this work.

The following lemma resembles the simple soundness result in Theorem 1, but unfortunately is harder to prove. Whereas the proof of Theorem 1 is a simple application of union bound, the fact that the adversary can choose what weight-1 elements to query adaptively based on past RO responses makes the "vanilla" union bound argument inapplicable. Fortunately, there exists a way around this problem.

Lemma 3. *Define parameters as in Theorem 1 and let E be the event that a valid proof can be made from the first n_f (or less) weight-1 elements that $\mathcal{A}_1^{H,W}$ queries to H. Then $\Pr[E] \le 2^{-\lambda_{sec}}$.*

We present the proof in the full version [CKRZ23] of this work.

Combining the proof of knowledge property with completeness proven in Sect. 3.1, we now state the main result of this section.

Corollary 6. *Using parameters from Corollary 1,* $(\mathsf{Prove}, \mathsf{Verify}, \mathsf{Extract})$ *is a* $(\lambda_{sec}, \lambda_{rel}, n_p, n_f)$*-NIROPK ALBA scheme.*

In summary, we achieve information-theoretic but non-adaptive security; i.e., additional computational power does not help the adversary avoid knowledge extraction but he is not allowed to choose the predicate/weight function based on the random oracle. Adaptive security can be achieved the traditional way: rerandomize the random oracle by including a commitment to the weight function as additional input to the random oracle. However, the security downgrades to computational: assuming that adversary makes at most 2^q RO queries, we need to increase ALBA's λ_{sec} parameter by q.

7 Replacing the Random Oracle with PRF

In this section we show how to remove the need for the random oracle and instantiate our scheme in the Common Reference String model (or alternatively, the Uniform Random String model). This is a novel feature of our scheme in comparison to compact certificates which inherently rely on the random oracle because of Fiat-Shamir. We utilize a PRF for the hash function H with the CRS being a random PRF key (or alternatively, uniformly random bits sufficient to generate one). We note that although the PRF is only secure against computationally bounded distinguishers, our ALBA scheme retains information-theoretic security.

Assume (GenKey, F) is a PRF such that for any oracle access program $\mathcal{A}^O$ with running time bounded by T,

$$\left| \Pr\left[\mathcal{A}^H() = 1\right] - \Pr\left[\mathcal{A}^{F(\mathsf{GenKey}(),\cdot)}() = 1\right] \right| \leq \varepsilon_{\text{prf}}(T). \tag{4}$$

We will assume the unweighted case, but the following can extended to support weights as well. Combining the improved Telescope construction from Sect. 3.2 with the tight bound on the number of accessible vertices (Theorem 10) and instantiating the scheme with the standard (binary) random oracle (the latter is described in the full version) one can build a Telescope scheme such that for some $B \in O(\lambda^3)$,

- the honest prover's DFS visits at most B vertices and outputs a valid proof with probability $\geq 1 - 2^{-\lambda}$;
- there exists a valid proof containing elements from S_{f} or the number of accessible vertices exceeds B with probability $\leq 2^{-\lambda}$.

Implement $\mathsf{Prove}^H(S_{\text{p}})$ as the standard DFS that visits at most B vertices and define $\mathsf{Verify}^H(\pi)$ in a natural way. We show an ALBA scheme under Definition 4 where the random oracle is replaced with CRS. Below is the new definition and a Telescope construction for it.

Definition 6. $(\mathsf{Prove}, \mathsf{Read}, \mathsf{Verify}, \mathsf{GenCRS})$ *is a* $(\lambda_{sec}, \lambda_{rel}, n_p, n_f)$*-CRS ALBA scheme if and only if*

- Prove *is a probabilistic program;*
- Verify *is a program;*
- Read *is a program;*
- GenCRS *is a probabilistic program;*
- *completeness: consider the following experiment* $CompExp(S_p)$*:*
  we require that for all sets S_p *with size* $\geq n_p$, $\Pr[CompExp(S_p) = 1] \geq 1 - 2^{-\lambda_{rel}}$.
- *soundness: consider the following experiment* $SoundExp(S_f)$*:*
  $crs \leftarrow \mathsf{GenCRS}()$;
  output 1 *iff* $\exists \pi, \mathsf{Read}(\pi) \subseteq S_f \wedge \mathsf{Verify}(crs, \pi) = 1$;
  we require that for all sets S_f *with size* $\leq n_f$, $\Pr[SoundExp(S_f) = 1] \leq 2^{-\lambda_{sec}}$*;*

```
procedure $R$.Prove(crs, $S_p$)
    $\pi \leftarrow \mathsf{Prove}^{F(\text{crs},\cdot)}(S_p)$;
    return $\pi$;
procedure $R$.Verify(crs, $\pi$)
    $r \leftarrow \mathsf{Verify}^{F(\text{crs},\cdot)}(\pi)$;
    return $r$;
procedure $R$.Read($\pi$)
    parse $(t, s_1, ..., s_u) = \pi$;
    return $\{s_1, ..., s_u\}$;
procedure $R$.GenCRS
    $k \leftarrow \mathsf{GenKey}()$;
    return $k$;
```

Theorem 16. *R is a* $(\lambda'_{sec}, \lambda'_{rel}, n_p, n_f)$*-CRS ALBA scheme where* $\lambda'_{sec} = \lambda'_{rel} = -\log\left(2^{-\lambda} + \varepsilon_{prf}(O(n_p + \lambda^3))\right)$.

Proof. Completeness follows from the fact that Prove's running time is bounded by $O(n_\text{p} + B) = O(n_\text{p} + \lambda^3)$ steps and that $\mathsf{Prove}^H(S_\text{p})$, when instantiated with the random oracle H, finds a valid proof with probability $\geq 1 - 2^{-\lambda}$. Acting as a PRF distinguisher, we conclude that $\mathsf{Prove}^{F(\mathsf{GenKey}(),\cdot)}$ outputs a valid proof with probability $\geq 1 - 2^{-\lambda} - \varepsilon_\text{prf}(O(n_\text{p} + \lambda^3))$.

To prove soundness, we can observe whether a DFS on set S_f finds a valid proof or does not terminate after visiting B vertices. In the random oracle case, one or both happen with probability $\leq 2^{-\lambda}$, so in the PRF case it is $\leq 2^{-\lambda} + \varepsilon_\text{prf}(O(n_\text{p} + \lambda^3))$. But the probability that there *exists* a valid proof in the PRF case cannot be larger. □

We present the complete version of the proof in the full version [CKRZ23] of this work.

7.1 Knowledge Extraction For Definition 6/4

In this section we show how to generically convert an ALBA scheme under Definition 6 to a proof of knowledge scheme under Definition 3. We still assume the unweighted scenario ($W : \{0,1\}^* \to \{0,1\}$) but the following can be generalized to add weights. Sometimes it will be convenient to treat W as a set: $\{s : W(s) = 1\}$.

Let $X = (X.\mathsf{Prove}, X.\mathsf{Read}, X.\mathsf{Verify}, X.\mathsf{GenCRS})$ be a $(\lambda_{\mathrm{sec}}, \lambda_{\mathrm{rel}}, n_{\mathrm{p}}, n_{\mathrm{f}})$-CRS ALBA scheme (as in Definition 6) and define $Y = (Y.\mathsf{Prove}, Y.\mathsf{Verify}, Y.\mathsf{Extract}, Y.\mathsf{GenCRS})$ as follows.

```
procedure $Y.\mathsf{GenCRS}$
    return $X.\mathsf{GenCRS}()$;
procedure $Y.\mathsf{Prove}^W(\mathrm{crs}, S_{\mathrm{p}})$
    return $X.\mathsf{Prove}(\mathrm{crs}, S_{\mathrm{p}} \cap W)$;
procedure $Y.\mathsf{Verify}^W(\mathrm{crs}, \pi)$
    $S := X.\mathsf{Read}(\pi)$;
    return 1 iff $S \subseteq W \wedge$
    $X.\mathsf{Verify}(\mathrm{crs}, \pi) = 1$;

procedure $Y.\mathsf{Extract}^W(\mathcal{A})$
    $S_{\mathrm{f}} := \emptyset$;
    while $|S_{\mathrm{f}}| \leq n_{\mathrm{f}}$ do
        $\mathrm{crs} \leftarrow X.\mathsf{GenCRS}()$;
        $\pi \leftarrow \mathcal{A}^W(\mathrm{crs})$;
        $S := X.\mathsf{Read}(\pi)$;
        $S_{\mathrm{f}} := S_{\mathrm{f}} \cup (S \cap W)$;
    return $S_{\mathrm{f}}$;
```

Theorem 17. *Y is $(\lambda_{sec}, \lambda_{rel}, n_p, n_f)$-CRS proof of knowledge ALBA scheme.*

Proof. It is easy to see that Y satisfies the completeness property. We are left to prove the proof of knowledge property.

First, notice that $Y.\mathsf{Extract}$ can only output a set S_{f} such that $S_{\mathrm{f}} \subseteq W$ and $|S_{\mathrm{f}}| > n_{\mathrm{f}}$. Now examine a single loop iteration in $Y.\mathsf{Extract}$. We know that $\varepsilon = \Pr\left[Y.\mathsf{Verify}^W(\mathrm{crs}, \pi) = 1\right] - 2^{-\lambda_{\mathrm{sec}}} > 0$ and $Y.\mathsf{Verify}^W(\mathrm{crs}, \pi) = 1$ implies that $S \subseteq W$ and $X.\mathsf{Verify}(\mathrm{crs}, \pi) = 1$. So,

$$2^{-\lambda_{\mathrm{sec}}} + \varepsilon = \Pr[Y.\mathsf{Verify}^W(\mathrm{crs}, \pi) = 1] \leq \Pr[S \subseteq W \wedge X.\mathsf{Verify}(\mathrm{crs}, \pi) = 1].$$

At the same time, since $|S_{\mathrm{f}}| \leq n_{\mathrm{f}}$, by the soundness of X (considering the experiment $\mathrm{SoundExp}(S_{\mathrm{f}})$ from Definition 6), $\Pr[S \subseteq S_{\mathrm{f}} \wedge X.\mathsf{Verify}(\mathrm{crs}, \pi) = 1] \leq 2^{-\lambda_{\mathrm{sec}}}$. Therefore,

$$\begin{gathered}\varepsilon = (2^{-\lambda_{\mathrm{sec}}} + \varepsilon) - 2^{-\lambda_{\mathrm{sec}}} \leq \\ \Pr[S \subseteq W \wedge X.\mathsf{Verify}(\mathrm{crs}, \pi) = 1] - \Pr[S \subseteq S_{\mathrm{f}} \wedge X.\mathsf{Verify}(\mathrm{crs}, \pi) = 1] \leq \\ \Pr[(S \subseteq W \wedge X.\mathsf{Verify}(\mathrm{crs}, \pi) = 1) \wedge \neg(S \subseteq S_{\mathrm{f}} \wedge X.\mathsf{Verify}(\mathrm{crs}, \pi) = 1)] = \\ \Pr[S \subseteq W \wedge S \not\subseteq S_{\mathrm{f}} \wedge X.\mathsf{Verify}(\mathrm{crs}, \pi) = 1] \leq \\ \Pr[S \subseteq W \wedge S \not\subseteq S_{\mathrm{f}}] \leq \\ \Pr[\exists x \in (S \cap W) \setminus S_{\mathrm{f}}].\end{gathered}$$

So, a single iteration of the loop adds at least one new element of W to S_{f} with probability at least ε. Therefore, in expectation, the loop runs for at most $(n_{\mathrm{f}} + 1) \cdot \frac{1}{\varepsilon}$ iterations. Then it is easy to see that $Y.\mathsf{Extract}$ runs in expected time $\mathrm{poly}(T, 1/\varepsilon)$ (treating n_{f} as constant). □

In summary, we achieve information-theoretic but non-adaptive security; i.e., additional computational power does not help the adversary avoid knowledge

$n_{\mathrm{p}}/n_{\mathrm{f}}$	60/40		66/33		80/20	
ALBA Protocol	Size	Comms	Size	Comms	Size	Comms
GS [GS86]	82944σ		16384σ		3237σ	
C. Cert. [MRV⁺21] (2^{80})	$356\sigma + 356\eta$		$208\sigma + 208\eta$		$104\sigma + 104\eta$	
C. Cert. [MRV⁺21] (2^{128})	$438\sigma + 438\eta$		$256\sigma + 256\eta$		$128\sigma + 128\eta$	
Telescope, no weights (Sect. 3)	232σ		136σ		68σ	
Telescope, weights (Sect. 4.2,5)	239σ		140σ		70σ	
Simple Lottery (Sect. 4.1)	4157σ	5058σ	1428σ	1981σ	364σ	675σ
Simple Lottery ($\lambda_{\mathrm{rel}} = 64$)	3060σ	3591σ	1069σ	1395σ	283σ	466σ
Dec. Telescope (Sect. 4.2)	262σ	114264σ	151σ	49929σ	74σ	23104σ

Fig. 1. Certificate sizes and expected communication cost, expressed in revealed/sent set elements (σ) and, in the case of [MRV+21], secondary reveals of the same elements in the form of Merkle Tree paths (η). The parameters $\lambda_{\mathrm{sec}}, \lambda_{\mathrm{rel}}$ are set to 128 unless otherwise indicated.

extraction but he is not allowed to choose the predicate/weight function based on the CRS. Even then, this can be useful; one example is applications where PRF seed is chosen by a randomness beacon after the statement to be proven is already decided. As a last resort, adaptive security can be achieved by rerandomizing the PRF using the random oracle: let the CRS be the output of the random oracle on the description of the weight function. This can be beneficial to instantiating ALBA purely in the random oracle model, for example, when the knowledge of an ALBA proof is proven by a SNARK. In that case, calculating the CRS outside of the SNARK circuit and using PRF inside the circuit lets one avoid heuristically instantiating RO in the circuit.

8 Performance Comparisons

In terms of prover computation, the Simple Lottery scheme requires negligible effort from the aggregator (apart from verifying membership and eligibility of the received set elements). Compact certificates require the prover to build a commitment to the set of received set items in the form of a Merkle tree, requiring $O(n)$ hash evaluations, where n is the number of weighted elements in prover's input. Telescope in turn requires $O(n + \lambda^2)$ hashes in expectation and Goldwasser-Sipser requires $O(n_{\mathrm{p}} \cdot \lambda)$ hashes.

In terms of number of revealed elements, compact certificates need to reveal at least $\frac{\lambda_{\mathrm{sec}}}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$ set elements (denoted by σ) but they additionally need to reveal the Merkle tree path of each element (denoted by η) with regards to the commitment constructed by the prover. The Simple Lottery scheme only reveals set elements and the number of reveals has the same, linear dependency on λ_{sec}, but has a more complex (and more costly) dependency on $(n_{\mathrm{p}}/n_{\mathrm{f}})$. Telescope combines the best of both worlds, as it only needs to reveal close to $\frac{\lambda_{\mathrm{sec}}}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$ set elements and the integer t with no need for secondary openings. Goldwasser-Sipser requires $8\lambda \cdot (n_{\mathrm{p}}/n_{\mathrm{f}})^4 \cdot (n_{\mathrm{p}}/n_{\mathrm{f}} - 1)^{-4}$ reveals.

In Fig. 1 we compare proof sizes and communication costs of our constructions with those of existing protocols: compact certificates [MRV+21] and the

Goldwasser-Sipser [GS86] scheme. Our analysis of the simple lottery scheme of Sect. 4.1 is also applicable to Mithril [CK21] as the combinatorics are very similar. Compact certificates have computational security and we provide proof sizes secure against adversaries making 2^{80} and 2^{128} queries; Telescope, on the other hand, has information-theoretic security and smaller number of revealed elements, but becomes computationally secure with number of revealed elements similar to compact certificates when the adversary is allowed to choose the weight function.

We consider communication costs only where they are meaningful, i.e. in decentralized schemes. We note that these costs may be significantly lower in the case of weighted sets where the same element may appear multiple times with different indices. For compact certificates, we derive values using the formula from [MRV+21]. For the simple lottery we use direct calculation, slightly improving on the bounds of Sect. 4.1, for Goldwasser-Sipser we use the analysis performed in the full version, and for Telescope and Decentralized Telescope we use the bounds from Corollaries 2 and 5. For the weighted Telescope scheme we use the formula in Sect. 5.

Acknowledgement. We are grateful to Mahak Pancholi and Akira Takahashi for discussion about UC SNARKs. This material is based upon work supported in part by a gift from Input Output - IOG and by DARPA under Agreements No. HR00112020021 and HR00112020023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA.

References

[Bab85] Babai, L.: Trading group theory for randomness. In: 17th ACM STOC, pp. 421–429. ACM Press, May 1985

[BCS16] Ben-Sasson, E., Chiesa, A., Spooner, N.: Interactive oracle proofs. In: Hirt, M., Smith, A. (eds.) TCC 2016. LNCS, vol. 9986, pp. 31–60. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_2

[BR93] Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS 1993, pp. 62–73. ACM Press (1993)

[Can00] Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptol. **13**(1), 143–202 (2000)

[CDMP05] Coron, J.-S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård revisited: how to construct a hash function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)

[CK21] Chaidos, P., Kiayias, A.: Mithril: stake-based threshold multisignatures. Cryptology ePrint Archive, Report 2021/916 (2021). https://eprint.iacr.org/2021/916

[CKRZ23] Chaidos, P., Kiayias, A., Reyzin, L., Zinovyev, A.: Approximate lower bound arguments. Cryptology ePrint Archive, Paper 2023/1655 (2023). https://eprint.iacr.org/2023/1655

[CW79] Carter, L., Wegman, M.N.: Universal classes of hash functions. J. Comput. Syst. Sci. **18**(2), 143–154 (1979)

[DCX+23] Das, S., Camacho, P., Xiang, Z., Nieto, J., Bunz, B., Ren, L.: Threshold signatures from inner product argument: succinct, weighted, and multi-threshold. Cryptology ePrint Archive, Paper 2023/598 (2023). https://eprint.iacr.org/2023/598

[Fis05] Fischlin, M.: Communication-efficient non-interactive proofs of knowledge with online extractors. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 152–168. Springer, Heidelberg (2005)

[GHM+17] Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: scaling byzantine agreements for cryptocurrencies. In: Proceedings of the 26th Symposium on Operating Systems Principles, pp. 51–68 (2017)

[GJM+23] Garg, S., Jain, A., Mukherjee, P., Sinha, R., Wang, M., Zhang, Y.: hints: threshold signatures with silent setup. Cryptology ePrint Archive, Paper 2023/567 (2023). https://eprint.iacr.org/2023/567

[GKO+23] Ganesh, C., Kondi, Y., Orlandi, C., Pancholi, M., Takahashi, A., Tschudi, D.: Witness-succinct universally-composable SNARKs. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part II. LNCS, vol. 14005, pp. 315–346. Springer, Heidelberg (2023). https://doi.org/10.1007/978-3-031-30617-4_11

[Gro16] Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 305–326. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_11

[GS86] Goldwasser, S., Sipser, M.: Private coins versus public coins in interactive proof systems. In: 18th ACM STOC, pp. 59–68. ACM Press (1986)

[GWC19] Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953 (2019). https://eprint.iacr.org/2019/953

[MRV+21] Micali, S., Reyzin, L., Vlachos, G., Wahby, R.S., Zeldovich, N.: Compact certificates of collective knowledge. In: 2021 IEEE Symposium on Security and Privacy, pp. 626–641. IEEE Computer Society Press (2021)

[Pas03] Pass, R.: On deniability in the common reference string and random oracle model. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 316–337. Springer, Heidelberg (2003)

[Sip83] Sipser, M.: A complexity theoretic approach to randomness. In: 15th ACM STOC, pp. 330–335. ACM Press (1983)