Zero-Knowledge Proofs for Set Membership: Efficient, Succinct, Modular

Daniel Benarroch[1], Matteo Campanelli[2], Dario Fiore[2], Kobi Gurkan[4,5], and Dimitris Kolonelos[2,3]

[1] QEDIT, Israel
[2] IMDEA Software Institute, Madrid, Spain
[3] Universidad Politécnica de Madrid, Spain
[4] Ethereum Foundation
[5] cLabs

Abstract. We consider the problem of proving in zero knowledge that an element of a public set satisfies a given property without disclosing the element, i.e., for some u, "$u \in S$ and $P(u)$ holds". This problem arises in many applications (anonymous cryptocurrencies, credentials or whitelists) where, for privacy or anonymity reasons, it is crucial to hide certain data while ensuring properties of such data.
We design new *modular* and *efficient* constructions for this problem through new *commit-and-prove zero-knowledge systems for set membership*, i.e. schemes proving $u \in S$ for a value u that is in a public commitment c_u. We also extend our results to support *non-membership proofs*, i.e. proving $u \notin S$.
Being commit-and-prove, our solutions can act as plug-and-play modules in statements of the form "$u \in S$ and $P(u)$ holds" by combining our set (non-)membership systems with any other commit-and-prove scheme for $P(u)$. Also, they work with Pedersen commitments over prime order groups which makes them compatible with popular systems such as Bulletproofs or Groth16.
We implemented our schemes as a software library, and tested experimentally their performance. Compared to previous work that achieves similar properties—the clever techniques combining zkSNARKs and Merkle Trees in Zcash—our solutions offer more flexibility, shorter public parameters and 4.7×–42× faster proving time for a set of size 2^{64}.

1 Introduction

The problem of proving set membership—that a given element x belongs to some set S—arises in many applications, including governmental white-lists to prevent terrorism or money-laundering, voting and anonymous credentials, among others. More recently, this problem also appears at the heart of currency transfer and identity systems over blockchains. In this setting, parties can first publicly commit to sets of data (through the blockchain itself) and then, by proving set membership, can claim ownership of assets or existence of identity attributes, while ensuring privacy.

A naive approach to check if an element is in a set is to go through all its entries. The complexity of this approach, however, is unacceptable in many scenarios. This is especially true for blockchains, where most of the parties (the verifiers) should run quickly.

How to efficiently verify set membership then? Cryptographic *accumulators* [Bd94] provide a nice solution to this problem. They allow a set of elements to be compressed into a short value (the accumulator) and to generate membership proofs that are short and fast to verify. As a security guarantee they require it should be computationally infeasible to generate a false membership proof.

As of today, we can divide constructions for accumulators into three main categories: Merkle Trees [Mer88]; RSA-based [BP97, CL02, LLX07, BBF18a]; pairing-based [Ngu05, DT08, CKS09, ZKP17]. Approaches based on Merkle Trees [1] allow for short (i.e., $O(1)$) public parameters and

[1] We can include under this class currently known lattice-based accumulators such as [PSTY13, LLNW16].

accumulator values, whereas the witness for membership proofs is of size $\log(n)$, where n is the size of the set. In RSA-based constructions (which can be actually generalized to any group of unknown order [Lip12], including class groups) both the accumulator and the witness are each a single element in a relatively large hidden-order group $\mathbb{G}$,[2] and thus of constant-size. Schemes that use pairings in elliptic curves such as [Ngu05, CKS09] offer small accumulators and small witnesses (which can each be a single element of a prime order bilinear group, e.g., 256 bits) but require large parameters (approximately $O(n)$) and a trusted setup.

In anonymous cryptocurrencies, e.g. Zerocash [BCG+14] (but also in other applications such as Anonymous Credentials [Cha85] and whitelists), we also require *privacy*. That is, parties in the system would not want to disclose *which* element in the set is being used to prove membership. Phrased differently, one desires to prove that $u \in S$ without revealing u, or: the proof should be *zero-knowledge* [GMR89] for u. As an example, in Zerocash users want to prove that a coin exists (i.e. belongs to the set of previously sent coins) without revealing which coin it is that they are spending.

In practice it is common that this privacy requirement goes beyond proving membership. In fact, these applications often require proving further properties about the accumulated elements, e.g., that for some element u in the set, property $P(u)$ holds. And this without leaking any more information about u other than what is entailed by P. In other words, we desire zero-knowledge for the statement $R^*(S, u) :=$ "$u \in S$ and $P(u)$".

One way to solve the problem, as done in Zerocash, is to directly apply general-purpose zero-knowledge proofs for R^*, e.g., [PHGR13, Gro16]. This approach, however, tends to be expensive and ad-hoc. One of the questions we aim to tackle is that of providing a more efficient proof systems for set membership relations, that can also be modular.

Specifically, as observed in [CFQ19], the design of practical proof systems can benefit from a more modular vision. A modular framework such as the one in [CFQ19] not only allows for separation of concerns, but also increases reusability and compatibility in a plug-and-play fashion: the same proof system is designed once and can be reused for the same sub-problem regardless of the context[3]; it can be replaced with a component for the same sub-problem at any time. Also, as [CFQ19] shows, this can have a positive impact on efficiency since designing a special-purpose proof system for a specific relation can lead to significant optimizations. Finally, this compositional approach can also be leveraged to build general-purpose proof systems.

In this work we focus on applying this modular vision to designing *succinct zero-knowledge proofs for set membership*. Following the abstract framework in [CFQ19] we investigate how to apply commit-and-prove techniques [CLOS02] to our setting. Our approach uses commitments for composability as follows. Consider an efficient zero-knowledge proof system Π for property $P(u)$. Let us also assume it is commit-and-prove, i.e. the verifier can test $P(u)$ by simply holding a commitment $c(u)$ to u. Such Π could be for example a commit-and-prove NIZK such as Bulletproofs [BBB+18] or a commit-and-prove zkSNARK such as LegoGroth16 from [CFQ19] that are able to operate on Pedersen commitments $c(\cdot)$ over elliptic curves. In order to obtain a proof gadget for set membership, all one needs to design is a commit-and-prove scheme for the relations "$u \in S$" where

[2] The group $\mathbb{G}$ is typically $\mathbb{Z}_N^*$ where N is an RSA modulus. The size of an element in this group for a standard 128-bit security parameter is of 3072 bits.

[3] For instance, one can plug a proof system for matrix product $C = A \cdot B$ in any larger context of computation involving matrix multiplication. This regardless of whether, say, we then hash C or if A, B are in turn the output of a different computation

both u and S are committed: u through $c(u)$ and S through some other commitment for sets, such as an accumulator.

Our main contribution is to propose a formalization of this approach and new constructions of succinct zero-knowledge commit-and-prove systems for set membership. In addition, as we detail later, we also extend our results to capture proofs of *non-membership*, i.e., to show that $u \notin S$. For our constructions we focus on designing schemes where $c(u)$ is a Pedersen commitment in a prime order group $\mathbb{G}_q$. We focus on linking through Pedersen commitments as these can be (re)used in some of the best state-of-the-art zero-knowledge proof systems for general-purpose relations that offer for example the shortest proofs and verification time (see, e.g.,[Gro16] and its efficient commit-and-prove variant [CFQ19]), or transparent setup and logarithmic-size proofs [BBB+18].

Before describing our results in more detail, we review existing solutions and approaches to realize commit-and-prove zkSNARKs for set membership.

Existing Approaches for Proving Set Membership for Pedersen Commitments. The accumulator of Nguyen [Ngu05], by the simple fact of having a succinct pairing-based verification equation, can be combined with standard zero-knowledge proof techniques (e.g., Sigma protocols or the celebrated Groth-Sahai proofs [GS08]) to achieve a succinct system with reasonable proving and verification time. The main drawbacks of using [Ngu05], however, are the large public parameters (i.e. requiring as many prime group elements as the elements in the set) and a high cost for updating the accumulator to the set, in order to add or remove elements (essentially requiring to recompute the accumulator from scratch).

By using general-purpose zkSNARKs one can obtain a solution with constant-size proofs based on Merkle Trees: prove that there exists a valid path which connects a given leaf to the root; this requires proving correctness of about $\log n$ hash function computations (e.g., SHA256). This solution yields a constant-size proof and requires $\log n$-size public parameters if one uses preprocessing zkSNARKs such as [PHGR13, Gro16]. On the other hand, often when proving a relation such as $R^*(S, u) :=$ "$u \in S$ and $P(u)$" the bulk of the work stems from the set membership proof. This is the case in Zcash or Filecoin[4] where the predicate $P(\cdot)$ is sufficiently small.

Finally, another solution that admits constant-size public parameters and proofs is the protocol of [CL02]. Specifically, Camenisch and Lysyanskaya showed how to prove in zero-knowledge that an element u committed in a Pedersen commitment over a prime order group $\mathbb{G}_q$ is a member of an RSA accumulator. In principle this solution would fit the criteria of the gadget we are looking for. Nonetheless, its concrete instantiations show a few limitations in terms of efficiency and flexibility. The main problem is that, for its security to hold, we need a prime order group (the commitment space) and the primes (the message space) to be quite large, for example[5] $q > 2^{519}$. But having such a large prime order group may be undesirable in practice for efficiency reasons. In fact the group $\mathbb{G}_q$ is the one that is used to instantiate more proof systems that need to interact and be linked with the Pedersen commitment.

1.1 Our Contributions

We investigate the problem of designing commit-and-prove zero-knowledge systems for set membership and non-membership that can be used in a modular way and *efficiently* composed with

[4] `https://filecoin.io`

[5] More specifically: the elements of a set need to be prime numbers in a range (A, B) such that $q/2 > A^2 - 1 > B \cdot 2^{2\lambda_{st}+2}$. If aiming at 128 bits of security level one can meet this constraint by choosing for example $A = 2^{259}$, $B = 2^{260}$ and $q > 2^{519}$.

other zero-knowledge proof systems for potentially arbitrary relations. Our main results are the following.

First, building upon the view of recent works on composable proofs [AGM18, CFQ19], we define a formal framework for commit-and-prove zkSNARKs (CP-SNARKs) for set (non-)membership. The main application of this framework is a compiler that, given a CP-SNARK $\mathsf{CP}_{\mathsf{mem}}$ for set membership and any other CP-SNARK CP_R for a relation R, yields a CP-SNARK CP for the composed relation "$u \in S \wedge \exists \omega : R(u, \omega)$". As a further technical contribution, our framework extends the one in [CFQ19] in order to work with commitments from multiple schemes (including set commitments, e.g., accumulators).

Second, we propose new efficient constructions of CP-SNARKs for set membership and non-membership, in which elements of the accumulated set can be committed with a Pedersen commitment in a prime order group $\mathbb{G}_q$—a setting that, as argued before, is of practical relevance due to the widespread use of these commitments and of proof systems that operate on them. More in detail, we propose: four schemes (two for set membership and two for non-membership) that enjoy constant-size public parameters and are based on RSA accumulators for committing to sets, and a scheme over pairings that has public parameters linear in the size of the set, but where the set can remain hidden.

Finally, we implement our solutions in a software library and experimentally evaluate their performance.

Like the recent works [AGM18] and [CFQ19], our work can be seen as showing yet another setting—set membership— where the efficiency of SNARKs can benefit from a modular design.

RSA-based constructions. Our first scheme, a CP-SNARK for set membership based on RSA accumulators, supports a large domain for the set of accumulated elements, represented by binary strings of a given length η. Our second scheme, also based on RSA accumulators, supports elements that are prime numbers of exactly μ bits (for a given μ). Neither scheme requires an a-priori bound on the cardinality of the set. Both schemes improve the proof-of-knowledge protocol by Camenisch and Lysyanskaya [CL02]: (i) we can work with a prime order group $\mathbb{G}_q$ of "standard" size, e.g., 256 bits, whereas [CL02] needs a much larger $\mathbb{G}_q$ (see above). We note that the size of $\mathbb{G}_q$ affects not only the efficiency of the set membership protocol but also the efficiency of any other protocol that needs to interact with commitments to alleged set members; (ii) we can support flexible choices for the size of set elements. For instance, in the second scheme, we could work with primes of about 50 or 80 bits, which in practice captures virtually unbounded sets and can make the accumulator operations 4–5× faster compared to using $\approx$ 256-bits primes as in [CL02].

Our main technical contribution here involves a new way to link a proof of membership for RSA accumulators to a Pedersen commitment in a prime order group, together with a careful analysis showing this can be secure under parameters *not requiring a larger prime order group* (as in [CL02]). See Section 4 for further details.

Pairing-based construction. Our pairing-based scheme for set membership supports set elements in $\mathbb{Z}_q$, where q is the order of bilinear groups, while the sets are arbitrary subsets of $\mathbb{Z}_q$ of cardinality less than a fixed a-priori bound n. This scheme has the disadvantage of having public parameters linear in n, but has other advantages in comparison to previous schemes with a similar limitation (and also in comparison to the RSA-based schemes above). First, the commitment to the set can be hiding and untrusted for the verifier, i.e., the set can be kept hidden and it is not needed to check the opening of the commitment to the set; this makes it composable with proof systems that could for example prove global properties on the set, i.e., that $P(S)$ holds. Second, the scheme works entirely

in bilinear groups, i.e., no need of operating over RSA groups. The main technical contribution here is a technique to turn the EDRAX vector commitment [CPZ18] into an accumulator admitting efficient zero-knowledge membership proofs.

Extensions to Set Non-Membership. We propose extensions of both our CP-SNARK framework and RSA constructions to deal with proving *set non-membership*, namely proving in zero-knowledge that $u \notin S$ with respect to a commitment $c(u)$ and a committed set S. Our two RSA-based schemes for non-membership have the same features as the analogous membership schemes mentioned above: the first scheme supports sets whose elements are strings of length η, the second one supports elements that are prime numbers of μ bits, and both work with elements committed using Pedersen in a prime order group and sets committed with RSA accumulators. A byproduct of sharing the same parameters is that we can easily compose the set-membership and non-membership schemes, via our framework, in order to prove statements like $u \in S_1 \wedge u \notin S_2$. Our technical contribution in the design of these schemes is a zero-knowledge protocol for non-membership witnesses of RSA accumulators that is linked to Pedersen commitments in prime order groups.

Implementation and Experiments. We have implemented our RSA-based[6] schemes for membership and non-membership as a Rust library which will be made publicly available (link not provided for anonymity). Our library is implemented in a modular fashion to work with any elliptic curve from **libzexe** [SCI] and Ristretto from **curve25519-dalek** [LdV]. This choice enables everyone to easily and efficiently combine our CP-SNARKs in a modular way with other CP-SNARKs implemented over these elliptic curves, such as Bulletproofs [BBB+18] and LegoGroth16 [CFQ19].

We evaluated our RSA-based constructions and compared them against highly optimized solutions based on Merkle Trees[7]. Our schemes achieve significantly better performance in proving time while slightly compromising on proof size and verification time. Our implementation is fast, yet we have not heavily optimized it and thus expect the results can be further improved.

Our solutions supporting sets of arbitrary elements achieve a proving time that is up to[8] 4.75× faster for set membership and up to 9.5× faster for set non-membership[9]

Our solutions where elements of the set are large prime numbers (i.e., of 252-bit size) offer even better results: our proving time is 7×–42× faster for membership and 9×–57× faster for non-membership (depending on the depth of the Merkle tree used in the comparison). This scenario can for example capture the case of sets made of hiding commitments that are prime numbers. In Section 8 we discuss how this can be relevant for a slight variant of the Zerocash protocol where commitments can be made prime numbers.

More details on the implementation and the benchmarks are available in Section 7.

1.2 Other Related Work

Ozdemir et al. [OWWB19] recently proposed a solution to scale operations on RSA accumulators inside a SNARK. In particular, their approach scales when these operations are *batched* (i.e., when

[6] For the implementation we focused on schemes where the public parameters do not depend on the set size; hence, we did not implement the pairing-based solutions.

[7] For our experiments we consider Merkle Trees using Pedersen Hash over the JubJub curve [HBHW16]

[8] We stress the proving time for our construction does not vary when the set grows. On the other hand this time varies for solutions based on Merkle trees.

[9] These ratios refer to a comparison against Interval Merkle Trees which require opening two paths to prove non-membership. When compared against Sparse Merkle Trees, our solutions show similar improvement ratios.

proving membership of many elements at the same time); for example, they surpass a 2^{20}-large Merkle tree when proving batches of at least 600 elements. This approach is attractive in settings where we can delegate a *large* quantity of these checks to an untrusted server as there is a high constant proving cost. In contrast, our approach can achieve faster proving time than Merkle trees already for a single membership check. It is an interesting open problem to adapt our techniques for modular set (non-)membership for the case of batched membership while keeping the tested elements hidden.

1.3 Organization

We give basic definitions in Section 2. In Section 3 we formalize commit-and-prove zkSNARKs for set (non-)membership. We describe our main constructions based on RSA accumulators for set membership and non-membership respectively in Sections 4 and 5. We describe our construction for set membership based on bilinear pairings in Section 6. Finally, in Sections 7 and 8 we discuss our implementation, experiments and applications.

2 Preliminaries

Notation. We denote the security parameter with $\lambda \in \mathbb{N}$ and its unary representation with 1^λ. Throughout the paper we assume that all the algorithms of the cryptographic schemes take as input 1^λ, which is thus omitted from the list of inputs. If D is a distribution, we denote by $x \leftarrow D$ the process of sampling x according to D. An ensemble $\mathcal{X} = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ is a family of probability distributions over a family of domains $\mathcal{D} = \{D_\lambda\}_{\lambda\in\mathbb{N}}$, and we say that two ensembles $\mathcal{D} = \{D_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{D}' = \{D'_\lambda\}_{\lambda\in\mathbb{N}}$ are statistically indistinguishable (denoted by $\mathcal{D} \approx_s \mathcal{D}'$) if $\frac{1}{2}\sum_x |D_\lambda(x) - D'_\lambda(x)| < \mathsf{negl}(\lambda)$. If $\mathcal{A} = \{\mathcal{A}_\lambda\}$ is a (possibly non-uniform) family of circuits and $\mathcal{D} = \{D_\lambda\}_{\lambda\in\mathbb{N}}$ is an ensemble, then we denote by $\mathcal{A}(\mathcal{D})$ the ensemble of the outputs of $\mathcal{A}_\lambda(x)$ when $x \leftarrow D_\lambda$. We say two ensembles $\mathcal{D} = \{D_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{D}' = \{D'_\lambda\}_{\lambda\in\mathbb{N}}$ are computationally indistinguishable (denoted by $\mathcal{D} \approx_c \mathcal{D}'$) if for every non-uniform polynomial time distinguisher $\mathcal{A}$ we have $\mathcal{A}(\mathcal{D}) \approx_s \mathcal{A}(\mathcal{D}')$.

We use $[n]$ to denote the set of integers $\{1, \ldots, n\}$, and $[0, n]$ for $\{0, 1, \ldots, n\}$. We denote by $(u_j)_{j\in[\ell]}$ the tuple of elements $(u_1, \ldots, u_\ell)$.

We denote $\mathsf{Primes} := \{e \in \mathbb{N} : e \text{ is prime}\}$ the set of all positive integers $e > 1$ such that they do not have non-trivial (i.e. different than e and 1) factors. More specifically, given two positive integers $A, B > 0$ such that $A < B$, we denote with $\mathsf{Primes}(A, B)$ the subset of Primes of numbers lying in the interval (A, B), i.e., $\mathsf{Primes}(A, B) := \{e \in \mathbb{Z} : e \text{ is prime } \wedge\ A < e < B\}$. According to the well known prime number theorem $|\mathsf{Primes}(1, B)| = O(\frac{B}{\log B})$ which results to $|\mathsf{Primes}(A, B)| = O(\frac{B}{\log B}) - O(\frac{A}{\log A})$.

2.1 RSA Groups

We say that $N = pq$ is an RSA modulus for some primes p, q, such that $|p| = |q|$. We further say that N is a strong RSA modulus if there are primes p', q' such that $p = 2p' + 1, q = 2q' + 1$. We call $\mathbb{Z}_N^*$ for an RSA modulus an RSA group. With $\phi : \mathbb{N} \to \mathbb{N}$ we denote the Euler's totient function, $\phi(N) := |\mathbb{Z}_N^*|$. In particular for RSA modulus $\phi(N) = (p - 1)(q - 1)$. An RSA Group generator $N \leftarrow_\$ \mathsf{GenSRSAmod}(1^\lambda)$ is a probabilistic algorithm that outputs a strong RSA modulus N of bit-length $\ell(\lambda)$ for an appropriate polynomial $\ell(\cdot)$.

For any N we denote by $\mathsf{QR}_N \coloneqq \{Y : \exists X \in \mathbb{Z}_N^* \text{ such that } Y = X^2 \pmod N\}$, the set of all the quadratic residues modulo N. QR_N is a subgroup (and thus closed under multiplication) of $\mathbb{Z}_N^*$ with order $|\mathsf{QR}_N| = |\mathbb{Z}_N^*|/2$. In particular for a strong RSA modulus $|\mathsf{QR}_N| = \frac{4p'q'}{2} = 2p'q'$.

Computational Assumptions in RSA Groups. The most fundamental assumption for RSA groups is the factoring assumption which states that given an RSA modulus $N \leftarrow \mathsf{GenSRSAmod}(1^\lambda)$ it is hard to compute its factors p and q. We further recall the Discrete Logarithm and strong RSA [BP97] assumptions:

Definition 2.1 (DLOG Assumption for RSA groups). *We say that the* Discrete Logarithm *(DLOG) assumption holds for* $\mathsf{GenSRSAmod}$ *if for any PPT adversary* $\mathcal{A}$*:*

$$\Pr\left[G^{x'} = Y \pmod N \;:\; \begin{array}{l} N \leftarrow \mathsf{GenSRSAmod}(1^\lambda) \\ G \leftarrow_\$ \mathbb{Z}_N^*; x \leftarrow_\$ \mathbb{Z} \\ Y \leftarrow G^x \pmod N \\ x' \leftarrow \mathcal{A}(\mathbb{Z}_N^*, G, Y) \end{array}\right] = \mathsf{negl}(\lambda)$$

Definition 2.2 (Strong-RSA Assumption [BP97]). *We say that the* strong RSA assumption *holds for* $\mathsf{GenSRSAmod}$ *if for any PPT adversary* $\mathcal{A}$*:*

$$\Pr\left[U^e = G \;:\; \begin{array}{l} N \leftarrow \mathsf{GenSRSAmod}(1^\lambda) \\ G \leftarrow_\$ \mathbb{Z}_N^* \\ (U, e) \leftarrow \mathcal{A}(\mathbb{Z}_N^*, G) \end{array}\right] = \mathsf{negl}(\lambda)$$

2.2 Non-Interactive Zero-Knowledge (NIZK)

We recall the definition of zero-knowledge non-interactive arguments of knowledge (NIZKs, for short).

Definition 2.3 (NIZK). *A NIZK for* $\{\mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$ *is a tuple of three algorithms* $\Pi = (\mathsf{KeyGen}, \mathsf{Prove}, \mathsf{VerProof})$ *that work as follows and satisfy the notions of* completeness*,* knowledge soundness *and (composable)* zero-knowledge *defined below.*

- $\mathsf{KeyGen}(R) \to (\mathsf{ek}, \mathsf{vk})$ *takes the security parameter* λ *and a relation* $R \in \mathcal{R}_\lambda$*, and outputs a common reference string consisting of an evaluation and a verification key.*
- $\mathsf{Prove}(\mathsf{ek}, x, w) \to \pi$ *takes an evaluation key for a relation* R*, a statement* x*, and a witness* w *such that* $R(x, w)$ *holds, and returns a proof* π*.*
- $\mathsf{VerProof}(\mathsf{vk}, x, \pi) \to b$ *takes a verification key, a statement* x*, and either accepts (*$b = 1$*) or rejects (*$b = 0$*) the proof* π*.*

Completeness. *For any* $\lambda \in \mathbb{N}$*,* $R \in \mathcal{R}_\lambda$ *and* (x, w) *such that* $R(x, w)$*, it holds* $\Pr[(\mathsf{ek}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(R), \pi \leftarrow \mathsf{Prove}(\mathsf{ek}, x, w) : \mathsf{VerProof}(\mathsf{vk}, x, \pi) = 1] = 1$*.*

Knowledge Soundness. *Let* $\mathcal{RG}$ *be a relation generator such that* $\mathcal{RG}_\lambda \subseteq \mathcal{R}_\lambda$*.* Π *has computational knowledge soundness for* $\mathcal{RG}$ *and auxiliary input distribution* $\mathcal{Z}$*, denoted* $\mathsf{KSND}(\mathcal{RG}, \mathcal{Z})$ *for brevity, if for every (non-uniform) efficient adversary* $\mathcal{A}$ *there exists a (non-uniform) efficient extractor* $\mathcal{E}$ *such that* $\Pr[\mathsf{Game}^{\mathsf{KSND}}_{\mathcal{RG},\mathcal{Z},\mathcal{A},\mathcal{E}} = 1] = \mathsf{negl}$*. We say that* Π *is knowledge sound if there exists benign* $\mathcal{RG}$ *and* $\mathcal{Z}$ *such that* Π *is* $\mathsf{KSND}(\mathcal{RG}, \mathcal{Z})$*.*

$\mathsf{Game}^{\mathsf{KSND}}_{\mathcal{RG},\mathcal{Z},\mathcal{A},\mathcal{E}} \to b$

$(R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda)$; $\mathsf{crs} := (\mathsf{ek}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(R)$

$\mathsf{aux}_Z \leftarrow \mathcal{Z}(R, \mathsf{aux}_R, \mathsf{crs})$; $(x, \pi) \leftarrow \mathcal{A}(R, \mathsf{crs}, \mathsf{aux}_R, \mathsf{aux}_Z)$

$w \leftarrow \mathcal{E}(R, \mathsf{crs}, \mathsf{aux}_R, \mathsf{aux}_Z)$; $b = \mathsf{VerProof}(\mathsf{vk}, x, \pi) \wedge \neg R(x, w)$

Composable Zero-Knowledge. *A scheme* Π *satisfies composable zero-knowledge for a relation generator* $\mathcal{RG}$ *if there exists a simulator* $\mathcal{S} = (\mathcal{S}_{\mathsf{kg}}, \mathcal{S}_{\mathsf{prv}})$ *such that both following conditions hold:*

Keys Indistinguishability *For all adversaries* $\mathcal{A}$

$$\Pr\left[\begin{array}{l}(R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda)\\ \mathsf{crs} \leftarrow \mathsf{KeyGen}(R)\\ \mathcal{A}(\mathsf{crs}, \mathsf{aux}_R) = 1\end{array}\right] \approx \Pr\left[\begin{array}{l}(R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda)\\ (\mathsf{crs}, \mathsf{td}_\mathsf{k}) \leftarrow \mathcal{S}_{\mathsf{kg}}(R)\\ \mathcal{A}(\mathsf{crs}, \mathsf{aux}_R) = 1\end{array}\right]$$

Proof Indistinguishability *For all adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$

$$\Pr\begin{bmatrix}(R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda)\\ (\mathsf{crs}, \mathsf{td}_\mathsf{k}) \leftarrow \mathcal{S}_{\mathsf{kg}}(R)\\ (x, w, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{crs}, \mathsf{aux}_R) \ : \ R(x, w)\\ \pi \leftarrow \mathsf{Prove}(\mathsf{ek}, x, w)\\ \mathcal{A}_2(\mathsf{st}, \pi) = 1\end{bmatrix} \approx \Pr\begin{bmatrix}(R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda)\\ (\mathsf{crs}, \mathsf{td}_\mathsf{k}) \leftarrow \mathcal{S}_{\mathsf{kg}}(R)\\ (x, w, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{crs}, \mathsf{aux}_R) \ : \ R(x, w)\\ \pi \leftarrow \mathcal{S}_{\mathsf{prv}}(\mathsf{crs}, \mathsf{td}_\mathsf{k}, x)\\ \mathcal{A}_2(\mathsf{st}, \pi) = 1\end{bmatrix}$$

Definition 2.4 (zkSNARKs). *A NIZK* Π *is called* zero-knowledge succinct non-interactive argument of knowledge *(zkSNARK) if* Π *is a NIZK as per Definition 2.3 enjoying an additional property,* succinctness*, i.e., if the running time of* $\mathsf{VerProof}$ *is* $\mathsf{poly}(\lambda + |x| + \log|w|)$ *and the proof size is* $\mathsf{poly}(\lambda + \log|w|)$*.*

Remark 2.1 (On Knowledge-Soundness). In the NIZK definition above we use a non black-box notion of extractability. Although this is virtually necessary in the case of zkSNARKs [GW11], NIZKs can also satisfy stronger (black-box) notions of knowledge-soundness.

2.3 Type-Based Commitments

We recall the notion of Type-Based Commitment schemes introduced by Escala and Groth [EG14]. In brief, a Type-Based Commitment scheme is a normal commitment scheme with the difference that it allows one to commit to values from different domains. More specifically, the Commit algorithm (therefore the $\mathsf{VerCommit}$ algorithm also) depends on the domain of the input, while the commitment key remains the same. For example, as in the original motivation of [EG14], the committer can use the same scheme and key to commit to elements that may belong to two different groups $\mathbb{G}_1, \mathbb{G}_2$ or a field $\mathbb{Z}_p$. In our work we use type-based commitments. The main benefit of this formalization is that it can unify many commitment algorithms into one scheme. In our case this is useful to formalize the notion of commit-and-prove NIZKs that work with commitments from different groups and schemes.

More formally, a Type-Based Commitment is a tuple of algorithms $\mathsf{Com} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{VerCommit})$ that works as a Commitment scheme defined above with the difference that Commit and $\mathsf{VerCommit}$ algorithms take an extra input t that represent the type of u. All the possible types are included in the type space $\mathcal{T}$[10].

Definition 2.5. *A type-based commitment scheme for a set of types $\mathcal{T}$ is a tuple of algorithms $\mathsf{Com} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{VerCommit})$ that work as follows:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{ck}$ *takes the security parameter and outputs a commitment key* ck*. This key includes $\forall \mathsf{t} \in \mathcal{T}$ descriptions of the input space $\mathcal{D}_\mathsf{t}$, commitment space $\mathcal{C}_\mathsf{t}$ and opening space $\mathcal{O}_\mathsf{t}$.*
- $\mathsf{Commit}(\mathsf{ck}, \mathsf{t}, u) \to (c, o)$ *takes the commitment key* ck*, the type* t *of the input and a value $u \in \mathcal{D}_\mathsf{t}$, and outputs a commitment c and an opening o.*
- $\mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}, c, u, o) \to b$ *takes as a type* t*, a commitment c, a value u and an opening o, and accepts ($b = 1$) or rejects ($b = 0$).*

Furthermore, the security properties depend on the type, in the sense that binding and hiding should hold with respect to a certain type.

Definition 2.6. *Let $\mathcal{T}$ be a set of types, and* Com *be a type-based commitment scheme for $\mathcal{T}$. Correctness,* t*-Type Binding and* t*-Type Hiding are defined as follows:*

Correctness. *For all $\lambda \in \mathbb{N}$ and any input $(\mathsf{t}, u) \in (\mathcal{T}, \mathcal{D}_\mathsf{t})$ we have:*

$$\Pr[\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda), (c, o) \leftarrow \mathsf{Commit}(\mathsf{ck}, \mathsf{t}, u) : \mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}, c, u, o) = 1] = 1.$$

t-Type Binding. *Given $\mathsf{t} \in \mathcal{T}$, for every polynomial-time adversary $\mathcal{A}$:*

$$\Pr\begin{bmatrix}\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda) \\ (c, u, o, u', o') \leftarrow \mathcal{A}(\mathsf{ck}, \mathsf{t})\end{bmatrix} : \begin{matrix} u \neq u' \wedge \ \mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}, c, u, o) = 1 \\ \wedge \ \mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}, c, u', o') = 1 \end{matrix} = \mathsf{negl}$$

In case Com *is* t*-Type Bidning for all $\mathsf{t} \in \mathcal{T}$ we will say that it is Binding.*

t-Type Hiding. *Given a $\mathsf{t} \in \mathcal{T}$, for $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda)$ and every pair of values $u, u' \in \mathcal{D}_\mathsf{t}$, the following two distributions are statistically close:* $\mathsf{Commit}(\mathsf{ck}, \mathsf{t}, u) \approx \mathsf{Commit}(\mathsf{ck}, \mathsf{t}, u')$.
In case Com *is* t*-Type Hiding for all $\mathsf{t} \in \mathcal{T}$ we say it is Hiding.*

Composing Type-Based Commitments. For simplicity we now define an operator that allows to compose type-based commitment schemes in a natural way.

Definition 2.7. *Let* C *and* C' *be two commitment schemes respectively for (disjoint) sets of types $\mathcal{T}$ and $\mathcal{T}'$. Then we denote by $\mathsf{C} \bullet \mathsf{C}'$ the commitment scheme $\bar{\mathsf{C}}$ for $\mathcal{T} \cup \mathcal{T}'$ such as:*

- $\bar{\mathsf{C}}.\mathsf{Setup}(\mathsf{secpar}, \mathsf{secpar}') \to \overline{\mathsf{ck}}$ *: compute* $\mathsf{ck} \leftarrow \mathsf{C}.\mathsf{Setup}(\mathsf{secpar})$ *and* $\mathsf{ck}' \leftarrow \mathsf{C}'.\mathsf{Setup}(\mathsf{secpar}'); \overline{\mathsf{ck}} := (\mathsf{ck}, \mathsf{ck}')$.
- $\bar{\mathsf{C}}.\mathsf{Commit}(\overline{\mathsf{ck}} := (\mathsf{ck}, \mathsf{ck}'), \mathsf{t}, u)$ *: If $t \in \mathcal{T}$ then output* $\mathsf{C}.\mathsf{Commit}(\mathsf{ck}, \mathsf{t}, u)$*; otherwise return* $\mathsf{C}'.\mathsf{Commit}(\mathsf{ck}', \mathsf{t}, u)$.
- $\bar{\mathsf{C}}.\mathsf{VerCommit}(\overline{\mathsf{ck}} := (\mathsf{ck}, \mathsf{ck}'), \mathsf{t}, c, u, o)$ *: If $t \in \mathcal{T}$ then return* $\mathsf{C}.\mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}, c, u, o)$*; otherwise return* $\mathsf{C}'.\mathsf{VerCommit}(\mathsf{ck}', \mathsf{t}, c, u, o)$.

[10] Normally $\mathcal{T}$ is finite and includes a small number of type, e.g. $\mathcal{T} = \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{Z}_p\}$.

The following property of $\bullet$ follows immediately from its definition.

Lemma 2.1. *Let* C *and* C' *be two commitment schemes with disjoint sets of types. For all types* t *if* C *or* C' *is* t*-hiding (resp.* t*-binding) then* $\mathsf{C} \bullet \mathsf{C}'$ *is* t*-hiding (resp.* t*-binding).*

Remark 2.2. We observe that a standard non type-based commitment scheme with input space $\mathcal{D}$ induces directly a type-based commitment scheme with the same input space and a type we denote by $\mathbb{T}[\mathcal{D}]$.

2.4 Commit-And-Prove NIZKs

We give the definition of *commit-and-prove NIZKs* (CP-NIZKs). We start from the definition given in [CFQ19, BCF19] and we extend it to type-based commitments. The main benefit of such extension is that we can formalize CP-NIZKs working with commitments over different domains. In a nutshell, a CP-NIZK is a NIZK that can prove knowledge of (x, w) such that $R(x, w)$ holds with respect to a witness $w = (u, \omega)$ such that u opens a commitment c_u. As done in [CFQ19], we explicitly considers the input domain $\mathcal{D}_u$ at a more fine grained-level splitting it over ℓ subdomains. We call them *commitment slots* as each of the $\mathcal{D}_i$-s intuitively corresponds to a committed element[11]. The description of the splitting is assumed part of R's description.

In the remainder of this work we use the following shortcut definition. If C is a type-based commitment scheme over set of types $\mathcal{T}$, we say that a relation R over $(\mathcal{D}_1 \times \cdots \times \mathcal{D}_\ell)$ is $\mathcal{T}$-compatible if for all $j \in [\ell]$ it holds that $\mathbb{T}[\mathcal{D}_j] \in \mathcal{T}$. We say a relation family $\mathcal{R}$ is $\mathcal{T}$-compatible if every R in $\mathcal{R}$ is $\mathcal{T}$-compatible; a relation generator $\mathcal{RG}$ is $\mathcal{T}$-compatible if $\mathsf{Range}(\mathcal{RG})$ is $\mathcal{T}$-compatible.

Definition 2.8 (CP-NIZKs [CFQ19]). *Let* $\{\mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$ *be a family of relations* R *over* $\mathcal{D}_x \times \mathcal{D}_u \times \mathcal{D}_\omega$ *such that* $\mathcal{D}_u$ *splits over* ℓ *arbitrary domains* $(\mathcal{D}_1 \times \cdots \times \mathcal{D}_\ell)$ *for some arity parameter* $\ell \geq 1$*. Let* $\mathsf{C} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{VerCommit})$ *be a commitment scheme (as per Definition 2.5) over set of types* $\mathcal{T}$ *such that* $\{\mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$ *is* $\mathcal{T}$*-compatible. A commit and prove NIZK for* C *and* $\{\mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$ *is a NIZK for a family of relations* $\{\mathcal{R}^{\mathsf{C}}_\lambda\}_{\lambda\in\mathbb{N}}$ *such that:*

- *every* $\mathbf{R} \in \mathcal{R}^{\mathsf{C}}$ *is represented by a pair* (ck, R) *where* $\mathsf{ck} \in \mathsf{C.Setup}(1^\lambda)$ *and* $R \in \mathcal{R}_\lambda$*;*
- $\mathbf{R}$ *is over pairs* $(\mathbf{x}, \mathbf{w})$ *where the statement is* $\mathbf{x} := (x, (c_j)_{j\in[\ell]}) \in \mathcal{D}_x \times \mathcal{C}^\ell$*, the witness is* $\mathbf{w} := ((u_j)_{j\in[\ell]}, (o_j)_{j\in[\ell]}, \omega) \in \mathcal{D}_1 \times \cdots \times \mathcal{D}_\ell \times \mathcal{O}^\ell \times \mathcal{D}_\omega$*, and the relation* $\mathbf{R}$ *holds iff*

$$\bigwedge\nolimits_{j\in[\ell]} \mathsf{VerCommit}(\mathsf{ck}, \mathbb{T}[\mathcal{D}_j], c_j, u_j, o_j) = 1 \wedge R(x, (u_j)_{j\in[\ell]}, \omega) = 1$$

We denote knowledge soundness of a CP-NIZK for commitment scheme C *and relation and auxiliary input generators* $\mathcal{RG}$ *and* $\mathcal{Z}$ *as* $\mathsf{CP\text{-}KSND}(\mathsf{C}, \mathcal{RG}, \mathcal{Z})$*.*

We denote a CP-NIZK as a tuple of algorithms $\mathsf{CP} = (\mathsf{KeyGen}, \mathsf{Prove}, \mathsf{VerProof})$. For ease of exposition, in our constructions we adopt the following explicit syntax for CP's algorithms.

- $\mathsf{KeyGen}(\mathsf{ck}, R) \to \mathsf{crs} := (\mathsf{ek}, \mathsf{vk})$
- $\mathsf{Prove}(\mathsf{ek}, x, (c_j)_{j\in[\ell]}, (u_j)_{j\in[\ell]}, (o_j)_{j\in[\ell]}, \omega) \to \pi$
- $\mathsf{VerProof}(\mathsf{vk}, x, (c_j)_{j\in[\ell]}, \pi) \to b \in \{0, 1\}$

[11] Each of the "open" elements in the $\mathcal{D}_i$-s (together with any auxiliary opening information) should also be thought of as the witness to the relation as we require them to be extractable. On the other hand, the commitments themselves are part of the public input.

2.5 Commit-and-Prove NIZKs with Partial Opening

We now define a variant of commit-and-prove NIZKs with a weaker notion of knowledge-soundness. In particular we consider the case where part of the committed input is not assumed to be extractable (or hidden)[12], i.e., such input is assumed to be opened by the adversary. This models scenarios where we do not require this element to be input of the verification algorithm (the verifier can directly use a digest to it).

The motivation to define and use this notion is twofold. First, in some constructions commitments on sets are compressing but not knowledge-extractable. Second, in many applications this definition is sufficient since the set is public (e.g., the set contain the valid coins).

The definition below is limited to a setting where the adversary opens only one input in this fashion[13]. We will assume, as a convention, that in a scheme with partial opening this special input is always the first committed input of the relation, i.e. the one denoted by u_1 and corresponding to $\mathcal{D}_1$. We note that the commitment to u_1 does not require hiding for zero-knowledge to hold.

Definition 2.9 (CP-NIZK with Partial Opening). *A commit and prove NIZK with partial opening for* C *and* $\{\mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$ *is a NIZK for a family of relations* $\{\mathcal{R}^{\mathsf{C}}_\lambda\}_{\lambda\in\mathbb{N}}$ *(defined as in Definition 2.8)such that the property of knowledge soundness is replaced by* knowledge soundness with partial opening *below.*

Knowledge Soundness with Partial Opening. *Let* $\mathcal{RG}$ *be a relation generator such that* $\mathcal{RG}_\lambda \subseteq \mathcal{R}_\lambda$. Π *has knowledge soundness with partial opening for* C, $\mathcal{RG}$ *and auxiliary input distribution* $\mathcal{Z}$, *denoted* $\mathsf{CP\text{-}poKSND}(\mathsf{C},\mathcal{RG},\mathcal{Z})$ *for brevity, if for every (non-uniform) efficient adversary* $\mathcal{A}$ *there exists a (non-uniform) efficient extractor* $\mathcal{E}$ *such that* $\Pr[\mathsf{Game}^{\mathsf{CP\text{-}poKSND}}_{\mathsf{C},\mathcal{RG},\mathcal{Z},\mathcal{A},\mathcal{E}} = 1] = \mathsf{negl}$. *We say that* Π *is knowledge sound for* C *if there exists benign* $\mathcal{RG}$ *and* $\mathcal{Z}$ *such that* Π *is* $\mathsf{CP\text{-}poKSND}(\mathsf{C},\mathcal{RG},\mathcal{Z})$[14].

$\mathsf{Game}^{\mathsf{CP\text{-}poKSND}}_{\mathsf{C},\mathcal{RG},\mathcal{Z},\mathcal{A},\mathcal{E}} \to b$

$\mathsf{ck} \leftarrow \mathsf{C.Setup}(1^\lambda);\ (R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda);\ \mathbf{R} := (\mathsf{ck}, R)$

$\mathsf{crs} := (\mathsf{ek}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(\mathbf{R})$

$\mathsf{aux}_Z \leftarrow \mathcal{Z}(\mathbf{R}, \mathsf{aux}_R, \mathsf{crs})$

$\big(x, (c_j)_{j\in[\ell]}, u_1, o_1, \pi\big) \leftarrow\ \mathcal{A}(\mathbf{R}, \mathsf{crs}, \mathsf{aux}_R, \mathsf{aux}_Z)$

$\big((u_j)_{j\in[\ell]}, (o_j)_{j\in[\ell]}, \omega\big) \leftarrow\ \mathcal{E}(\mathbf{R}, \mathsf{crs}, \mathsf{aux}_R, \mathsf{aux}_Z)$

$b = \mathsf{VerProof}(\mathsf{vk}, x, (c_j)_{j\in[\ell]}, \pi) \wedge$

$\neg\big(\bigwedge_{j\in[\ell]} \mathsf{C.VerCommit}(\mathsf{ck}, \mathbb{T}[\mathcal{D}_j], c_j, u_j, o_j) = 1 \wedge R(x, (u_j)_{j\in[\ell]}, \omega) = 1\big)$

Remark 2.3 (On Weaker ZK in the Context of Partial Opening). The notion of zero-knowledge for CP-NIZKs with partial opening that is implied by our definition above implies that the simulator does not have access to the opening of the first input (as it is the case in zero-knowledge for CP-NIZKs in general). Since this first commitment is opened, in principle one could also consider and

[12] This is reminiscent of the soundness notions considered in [FFG+16]

[13] We can easily generalize the notion for an adversary opening an arbitrary subset of the committed inputs.

[14] We point out that, although in the game below we make explicit the commitment opening in the relation, this is essentially the same notion of knowledge soundness as in CP-NIZKs (i.e. Definition 2.3) where the only tweak is that the adversary gives explicitly the first input in the commitment slot. We make commitments explicit hoping for the definition to be clearer. This is, however, in contrast to the definition of CP-NIZKs where the commitment opening is completely abstracted away inside the relation.

define a weaker notion of zero-knowledge where the simulator has access to the first opened input. We leave it as an open problem to investigate if it can be of any interest.

Remark 2.4 (Full Extractability). If a CP-NIZK has an empty input u_1 opened by the adversary in the game above, then we say that it is *fully extractable*. This roughly corresponds to the notion of knowledge soundness in Definition 2.3.

Composition Properties of Commit-and-Prove Schemes In [CFQ19] Campanelli et al. show a compiler for composing commit-and-prove schemes that work for the same commitment scheme in order to obtain CP systems for conjunction of relations. In this section we generalize their results to the case of typed relations and type-based commitments. This generalization in particular can model the composition of CP-NIZKs that work with different commitments, as is the case in our constructions for set membership in which one has a commitment to a set and a commitment to an element.

We begin by introducing the following compact notation for an augmented relation generator.

Definition 2.10 (Augmented Relation Generator). *Let $\mathcal{RG}$ be a relation generator and $\mathcal{F}(1^\lambda)$ an algorithm taking as input a security parameter. Then we denote by $\mathcal{RG}[\mathcal{F}]$ the relation generator returning $(R, (\mathsf{aux}_R, \mathsf{out}_\mathcal{F}))$ where $\mathsf{out}_\mathcal{F} \leftarrow \mathcal{F}(1^\lambda)$ and $(R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda)$.*

The next lemma states that we can (with certain restrictions) trivially extend a CP-NIZK for commitment scheme C to an extended commitment scheme $\mathsf{C} \bullet \mathsf{C}'$.

Lemma 2.2 (Extending to Commitment Composition). *Let C, C' be commitment schemes defined over disjoint type sets $\mathcal{T}$ and $\mathcal{T}'$. If CP is $\mathsf{CP\text{-}poKSND}(\mathsf{C}, \mathcal{RG}[\mathsf{C.Setup}], \mathcal{Z})$ for some relation and auxiliary input generators $\mathcal{RG}, \mathcal{Z}$. Then CP is $\mathsf{CP\text{-}poKSND}(\mathsf{C} \bullet \mathsf{C}', \mathcal{RG}[\mathsf{C.Setup}], \mathcal{Z})$ if $\mathcal{RG}$ is $\mathcal{T}$-compatible.*

We now define relation generators and auxiliary input generators for our composition constructions.

$\underline{\mathsf{Aux}^{\mathcal{RG}}(1^\lambda):}$

$(R_1, \mathsf{aux}_R^{(1)}) \leftarrow \mathcal{RG}_1(1^\lambda)$

$(R_2, \mathsf{aux}_R^{(2)}) \leftarrow \mathcal{RG}_2(1^\lambda)$

return $(R_b, \mathsf{aux}_R^{(b)})_{b\in\{1,2\}}$

$\underline{\mathcal{RG}^*(1^\lambda):}$

$(R_b, \mathsf{aux}_R^{(b)})_{b\in\{1,2\}} \leftarrow \mathsf{Aux}^{\mathcal{RG}}(1^\lambda)$

return $(R^\wedge_{R_1,R_2}, (\mathsf{aux}_R^{(b)})_{b\in\{1,2\}})$

$\underline{\overline{\mathcal{RG}}_b(1^\lambda):}$

$(R_b, \mathsf{aux}_R^{(b)})_{b\in\{1,2\}} \leftarrow \mathsf{Aux}^{\mathcal{RG}}(1^\lambda)$

return $(R_b, \overline{\mathsf{aux}}_R^{(b)} := (R_{3-b}, (\mathsf{aux}_R^{(b)})_{b\in\{1,2\}}))$

$\underline{\mathsf{Aux}^{\mathcal{Z}}(\mathsf{ck}, (\mathsf{crs}_b, R_b, \mathsf{aux}_R^{(b)})_{b\in\{1,2\}}):}$

$\mathsf{aux}_Z^{(1)} \leftarrow \mathcal{Z}_1(\mathsf{ck}, R_1, \mathsf{crs}_1, \mathsf{aux}_R^{(1)})$

$\mathsf{aux}_Z^{(2)} \leftarrow \mathcal{Z}_2(\mathsf{ck}, R_2, \mathsf{crs}_2, \mathsf{aux}_R^{(2)})$

return $(\mathsf{aux}_Z^{(b)})_{b\in\{1,2\}}$

$\underline{\mathcal{Z}^*((\mathsf{ck}, R^\wedge_{R_1,R_2}), (\mathsf{ek}^*, \mathsf{vk}^*), (\mathsf{aux}_R, \mathsf{aux}'_R)):}$

$(\mathsf{aux}_Z^{(b)})_{b\in\{1,2\}} \leftarrow \mathsf{Aux}^{\mathcal{Z}}(\mathsf{ck}, (\mathsf{crs}_b, R_b, \mathsf{aux}_R^{(b)})_{b\in\{1,2\}})$

return $(\mathsf{aux}_Z^{(b)})_{b\in\{1,2\}}$

$\underline{\overline{\mathcal{Z}}_b(\mathsf{ck}, R_b, \mathsf{crs}_b, \overline{\mathsf{aux}}_R^{(b)}):}$

Parse $\overline{\mathsf{aux}}_R$ as $(R_{3-b}, (\mathsf{aux}_R^{(b)})_{b\in\{1,2\}})$

$\mathsf{crs}_{3-b} \leftarrow \mathsf{CP}_{3-b}.\mathsf{KeyGen}(\mathsf{ck}, R_{3-b})$

$(\mathsf{aux}_Z^{(b)})_{b\in\{1,2\}} \leftarrow \mathsf{Aux}^{\mathcal{Z}}(\mathsf{ck}, \ldots, (\mathsf{crs}_b, R_b, \mathsf{aux}_R^{(b)})_{b\in\{1,2\}})$

$\overline{\mathsf{aux}}_Z^{(b)} := (\mathsf{crs}_{3-b}, (\mathsf{aux}_Z^{(b)})_{b\in\{1,2\}})$

return $\overline{\mathsf{aux}}_Z^{(b)}$

Fig. 1: Relation and Auxiliary Input Generators for AND Composition Construction

The following lemma shows how we can compose CP-NIZKs even when one of them is fully extractable but the other is not. We are interested in the conjunction $R^\wedge_{asym}$ of relations of type $R_1(x_1, (u_0, u_1, u_3), \omega_1)$ and $R_2(x_2, (u_2, u_3), \omega_2)$ where

$$R^\wedge_{asym}(x_1, x_2, (u_0, u_1, u_2, u_3), \omega_1, \omega_2) := R_1(x_1, (u_0, u_1, u_3), \omega_1) \wedge R_2(x_2, (u_2, u_3), \omega_2)$$

Lemma 2.3 (Composing Conjunctions (with asymmetric extractability). *Let* C *be a computationally binding commitment scheme.If* CP_1 *is* $\mathsf{CP\text{-}poKSND}(\mathsf{C}, \overline{\mathcal{RG}}_1, \overline{\mathcal{Z}}_1)$ *and* CP_2 *is* $\mathsf{KSND}(\mathsf{C}, \overline{\mathcal{RG}}_2, \overline{\mathcal{Z}}_2)$ *(where* $\overline{\mathcal{RG}}_b, \overline{\mathcal{Z}}_b$ *are defined in terms of* $\mathcal{RG}_b, \mathcal{Z}_b$ *in Figure 1 for* $b \in \{1,2\}$*), then the scheme* $\mathsf{CP}^\wedge_{asym}$ *in Figure 2 is* $\mathsf{CP\text{-}poKSND}(\mathsf{C}, \mathcal{RG}^*, \mathcal{Z}^*)$ *where* $\mathcal{RG}^*, \mathcal{Z}^*$ *are as defined in Figure 1.*

The following lemma is a symmetric variant of Lemma 2.3, i.e. the CP-NIZKs we are composing are both secure over the same commitment scheme and support partial opening, that is they both handle relations with and adversarially open input u_0. This time we are interested in the conjunction $R^\wedge_{sym}$ of relations of type $R_1(x_1, (u_0, u_1, u_3), \omega_1)$ and $R_2(x_2, (u_0, u_2, u_3), \omega_2)$ where

$$R^\wedge_{sym}(x_1, x_2, (u_0, u_1, u_2, u_3), \omega_1, \omega_2) := R_1(x_1, (u_0, u_1, u_3), \omega_1) \wedge R_2(x_2, (u_0, u_2, u_3), \omega_2)$$

Lemma 2.4 (Composing Conjunctions (symmetric case)). *Let* C *be a (type-based) computationally binding commitment scheme. If* CP_b *is* $\mathsf{CP\text{-}poKSND}(\mathsf{C}, \overline{\mathcal{RG}}_b, \overline{\mathcal{Z}}_b)$ *(where* $\overline{\mathcal{RG}}_b, \overline{\mathcal{Z}}_b$ *are defined in terms of* $\mathcal{RG}_b, \mathcal{Z}_b$ *in Figure 1) for* $b \in \{1,2\}$*, then the scheme* $\mathsf{CP}^\wedge_{sym}$ *in Figure 3 is* $\mathsf{CP\text{-}poKSND}(\mathsf{C}, \mathcal{RG}^*, \mathcal{Z}^*)$ *where* $\mathcal{RG}^*, \mathcal{Z}^*$ *are as defined in Figure 1.*

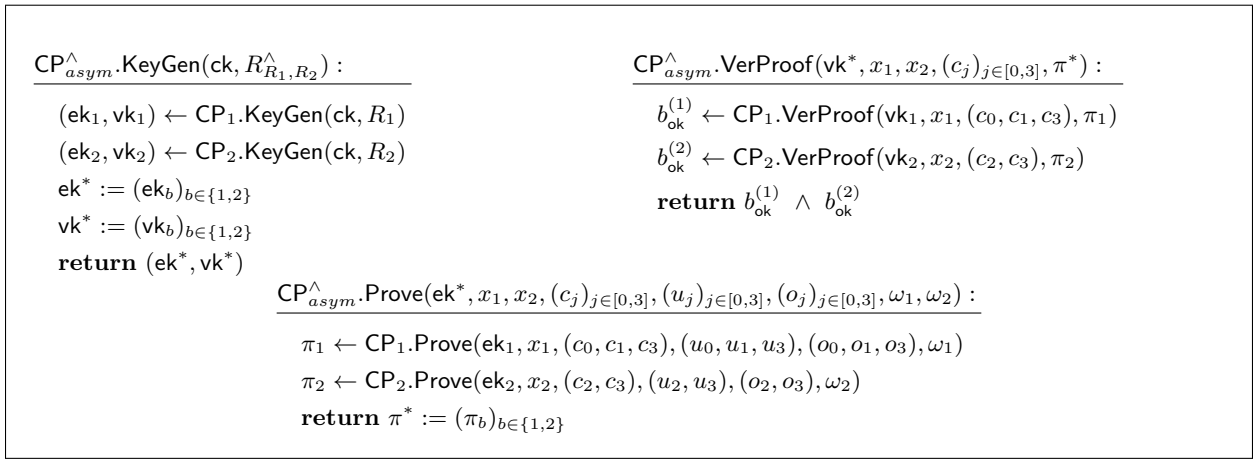

Fig. 2: CP-NIZK construction for AND composition (asymmetric case)

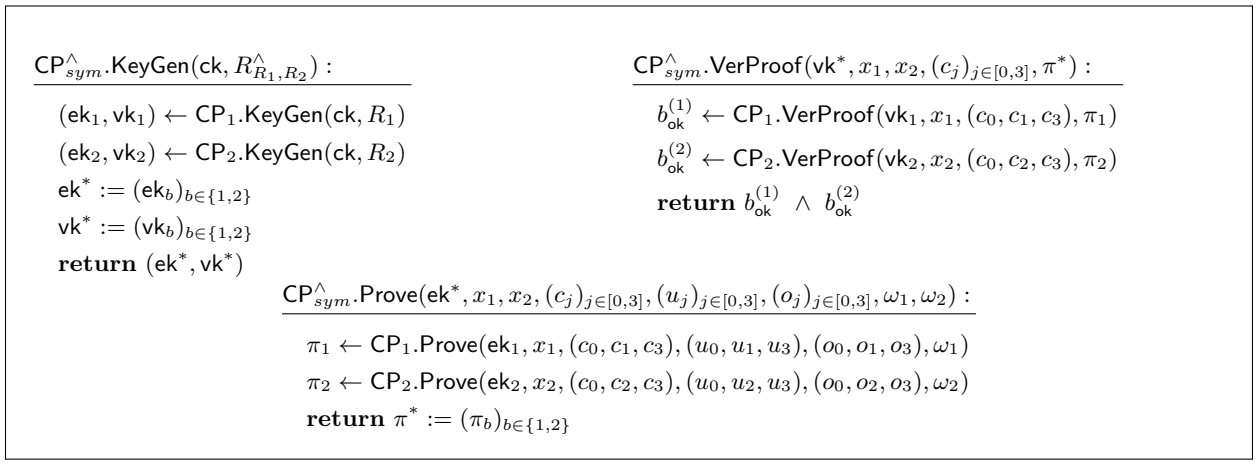

Fig. 3: CP-NIZK construction for AND composition (symmetric case)

3 CP-SNARKs for Set Membership (and non-Membership)

In this section we discuss a specialization of CP-SNARKs for the specific NP relation that models membership (resp. non-membership) of an element in a set, formally defined below.

Set membership relations. Let $\mathcal{D}_{\mathsf{elm}}$ be some domain for set elements, and let $\mathcal{D}_{\mathsf{set}} \subseteq 2^{\mathcal{D}_{\mathsf{elm}}}$ be a set of possible sets over $\mathcal{D}_u$. We define the set membership relation $R_{\mathsf{mem}} : \mathcal{D}_{\mathsf{elm}} \times \mathcal{D}_{\mathsf{set}}$ as

$$R_{\mathsf{mem}}(U, u) = 1 \iff u \in U$$

This is the fundamental relation that we deal with in the rest of this work.
The non-membership relation $R_{\mathsf{nmem}} : \mathcal{D}_{\mathsf{elm}} \times \mathcal{D}_{\mathsf{set}}$ can be defined analogously as

$$R_{\mathsf{nmem}}(U, u) = 1 \iff u \notin U$$

CP-SNARKs for set membership. Intuitively, a commit-and-prove SNARK for set membership allows one to commit to a set U and to an element u, and then to prove in zero-knowledge that $R_{\mathsf{mem}}(U, u) = 1$. More formally, let $R_{\mathsf{mem}} : \mathcal{D}_{\mathsf{elm}} \times \mathcal{D}_{\mathsf{set}}$ be a set membership relation as defined above where $\mathbb{T}[\mathcal{D}_{\mathsf{elm}}] = \mathsf{t}_{\mathsf{elm}}$ and $\mathbb{T}[\mathcal{D}_{\mathsf{set}}] = \mathsf{t}_{\mathsf{set}}$, and let $\mathsf{Com}_{\mathsf{S} \cup \mathsf{elm}}$ be a type-based commitment scheme for $\mathcal{T}$ such that $\mathsf{t}_{\mathsf{set}}, \mathsf{t}_{\mathsf{elm}} \in \mathcal{T}$. Basically, $\mathsf{Com}_{\mathsf{S} \cup \mathsf{elm}}$ allows one to either commit an element of $\mathcal{D}_{\mathsf{elm}}$ or to a set of values of $\mathcal{D}_{\mathsf{elm}}$. Then a CP-SNARK for set membership is a CP-SNARK for the family of relations $\{\mathcal{R}_\lambda^{\mathsf{mem}}\}$ and a type-based commitment scheme $\mathsf{Com}_{\mathsf{S} \cup \mathsf{elm}}$. It is deduced from definition 2.8 that this is a zkSNARK for the relation:

$$\mathbf{R} = (ck, R_{\mathsf{mem}}) \text{ over } (\boldsymbol{x}, \boldsymbol{w}) = ((x, c), (u, o, \omega)) := (\,(\,\varnothing\,,\,(c_U, c_u)\,)\,,\,(\,(U, u)\,,\,(o_U, o_u)\,,\,\varnothing\,)\,)$$

such that $\mathbf{R}$ holds iff:

$$R_{\mathsf{mem}}(U, u) = 1 \wedge \mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_{\mathsf{set}}, c_U, U, o_U) = 1 \wedge \mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_{\mathsf{elm}}, c_u, u, o_u) = 1$$

A commit-and-prove version of R_{nmem} can be defined as a natural variant of the relation above.

Notice that for the relation R_{mem} it is relevant for the proof system to be succinct so that proofs can be at most polylogarithmic (or constant) in the the size of the set (that is part of the witness). This is why for set membership we are mostly interested in designing CP-SNARKs.

Proving arbitrary relations involving set (non-)membership. As discussed in the introduction, a primary motivation of proving set membership in zero-knowledge is to prove additional properties about an alleged set member. In order to make our CP-SNARK for set membership a reusable gadget, we discuss a generic and simple method for composing CP-SNARKs for set membership (with partial opening) with other CP-SNARKs (with full extractability) for arbitrary relations. More formally, let R_{mem} be the set membership relation over pairs $(U, u) \in \mathbb{X} \times \mathcal{D}_u$ as R be an arbitrary relation over pairs (u, ω), then we define as R^* the relation:

$$R^*(U, u, \omega) := R_{\mathsf{mem}}(U, u) \wedge R(u, \omega)$$

The next corollary (direct consequence of Lemmas 2.2, 2.3) states we can straightforwardly compose a CP-SNARK for set membership with a CP-SNARK for an arbitrary relation on elements of the set.

Corollary 3.1 (Extending Relations with Set Membership). *Let* $\mathsf{C_S}, \mathsf{C}_u$ *be two computationally binding commitment schemes defined over disjoint type sets* $\mathcal{T}_\mathsf{S}$ *and* $\mathcal{T}_u$*. Let* $\mathsf{CP}_\mathsf{mem}, \mathsf{CP}_u$ *be two CP-SNARKs and* $R_\mathsf{mem}, \mathcal{RG}_u$ *(resp.* $\mathcal{Z}_\mathsf{mem}, \mathcal{Z}_u$*) be two relation (resp. auxiliary input) generators. If* CP_mem *is* $\mathsf{CP\text{-}poKSND}(\mathsf{C_S} \bullet \mathsf{C}_u, R_\mathsf{mem}, \mathcal{Z}_\mathsf{mem})$ *and* CP_u *is* $\mathsf{KSND}(\mathsf{C}_u, \mathcal{RG}_u, \mathcal{Z}_u)$ *then there exists a* CP^* *that is* $\mathsf{CP\text{-}poKSND}(\mathsf{C_S} \bullet \mathsf{C}_u, \mathcal{RG}^*, \mathcal{Z}^*)$ *where* $\mathcal{RG}^*, \mathcal{Z}^*$ *are as defined in Figure 1.*

In a similar fashion, we can combine an arbitrary relation R with the relation for non-membership obtaining relation $\bar{R}^*$ defined as:

$$\bar{R}^*(U, u, \omega) := R_\mathsf{nmem}(U, u) \wedge R(u, \omega)$$

The next corollary states we can straightforwardly compose a CP-SNARK for set non-membership with a CP-SNARK for an arbitrary relation on elements in the universe of the set.

Corollary 3.2 (Extending Relations with Set non-Membership). *Let* $\mathsf{C_S}, \mathsf{C}_u$ *be two computationally binding commitment schemes defined over disjoint type sets* $\mathcal{T}_\mathsf{S}$ *and* $\mathcal{T}_u$*. Let* $\mathsf{CP}_\mathsf{nmem}, \mathsf{CP}_u$ *be two CP-SNARKs and* $R_\mathsf{nmem}, \mathcal{RG}_u$ *(resp.* $\mathcal{Z}_\mathsf{nmem}, \mathcal{Z}_u$*) be two relation (resp. auxiliary input) generators. If* $\mathsf{CP}_\mathsf{nmem}$ *is* $\mathsf{CP\text{-}poKSND}(\mathsf{C_S} \bullet \mathsf{C}_u, R_\mathsf{nmem}, \mathcal{Z}_\mathsf{nmem})$ *and* CP_u *is* $\mathsf{KSND}(\mathsf{C}_u, \mathcal{RG}_u, \mathcal{Z}_u)$ *then there exists a* CP^* *that is* $\mathsf{CP\text{-}poKSND}(\mathsf{C_S} \bullet \mathsf{C}_u, \mathcal{RG}^*, \mathcal{Z}^*)$ *where* $\mathcal{RG}^*, \mathcal{Z}^*$ *are as defined in Figure 1.*

CP-SNARKs for set membership from accumulators with proofs of knowledge. As discussed in the introduction, CP-SNARKs for set membership are simply a different lens through which we can approach accumulators that have a protocol for proving in zero-knowledge that a committed value is in the accumulator (i.e., it is in the set succinctly represented by the accumulator). To strengthen this intuition in Appendix B we formally show that a CP-SNARK for set membership can be constructed from an accumulator scheme that has a zero-knowledge proof for committed values. This allows us to capture existing schemes such as [CL02] and [Ngu05].

4 A CP-SNARK for Set Membership with Short Parameters

In this section we describe CP-SNARKs for set membership in which the elements of the sets can be committed using a Pedersen commitment scheme defined in a prime order group, and the sets are committed using an RSA accumulator. The advantage of having elements committed with Pedersen in a prime order group is that our CP-SNARKs can be composed with any other CP-SNARK for Pedersen commitments and relations R that take set elements as inputs. The advantage of committing to sets using RSA accumulators is instead that the public parameters (i.e., the CRS) of the CP-SNARKs presented in this section are *short*, virtually independent of the size of the sets. Since RSA accumulators are not extractable commitments, the CP-SNARKs presented here are secure in a model where the commitment to the set is assumed to be checked at least once, namely they are knowledge-sound with partial opening of the set commitment.

A bit more in detail, we propose two CP-SNARKs. Our first scheme, called $\mathsf{MemCP}_\mathsf{RSA}$, works for set elements that are arbitrary strings of length η, i.e., $\mathcal{D}_\mathsf{elm} = \{0,1\}^\eta$, and for sets that are any subset of $\mathcal{D}_\mathsf{elm}$, i.e., $\mathcal{D}_\mathsf{set} = 2^{\mathcal{D}_\mathsf{elm}}$. Our second scheme, $\mathsf{MemCP}_\mathsf{RSAPrm}$, instead works for set elements that are prime numbers of exactly μ bits, and for sets that are any subset of such prime numbers. This second scheme is a simplified variant of the first one that requires more structure on the set

elements (they must be prime numbers) but in exchange of that offers better efficiency. So it is preferable in those applications that can work with prime representatives.

An High-Level Overview of Our Constructions. We provide the main idea behind our scheme, and to this end we use the simpler scheme $\mathsf{MemCP}_{\mathsf{RSAPrm}}$ in which set elements are prime numbers in $\left(2^{\mu-1}, 2^{\mu}\right)$. The commitment to the set $P = \{e_1, \ldots, e_n\}$ is an RSA accumulator [Bd94, BP97] that is defined as $\mathsf{Acc} = G^{\prod_{e_i \in P} e_i}$ for a random quadratic residue $G \in \mathsf{QR}_N$. The commitment to a set element e is instead a Pedersen commitment $c_e = g^e h^{r_q}$ in a group $\mathbb{G}_q$ of prime order q, where q is of ν bits and $\mu < \nu$. For public commitments Acc and c_e, our scheme allows to prove in zero-knowledge the knowledge of e committed in c_e such that $e \in P$ and $\mathsf{Acc} = G^{\prod_{e_i \in P} e_i}$. A public coin protocol for this problem was proposed by Camenisch and Lysyanskaya [CL02]. Their protocol however requires various restrictions. For instance, the accumulator must work with at least 2λ-bit long primes, which slows down accumulation time, and the prime order group must be more than 4λ-bits (e.g., of 512 bits), which is undesirable for efficiency reasons, especially if this prime order group is used to instantiate more proof systems to create other proofs about the committed element. In our scheme the goal is instead to keep the prime order group of "normal" size (say, 2λ bits), so that it can be for example a prime order group in which we can efficiently instantiate another CP-SNARK that could be composed with our $\mathsf{MemCP}_{\mathsf{RSAPrm}}$. And we can also allow flexible choices of the primes size that can be tuned to the application so that applications that work with moderately large sets can benefit in efficiency. In order to achieve these goals, our idea to create a membership proof is to compute the following:

- An accumulator membership witness $W = G^{\prod_{e_i \in P \setminus \{e\}} e_i}$, and an integer commitment to e in the RSA group, $C_e = G^e H^r$.
- A ZK proof of knowledge $\mathsf{CP}_{\mathsf{Root}}$ of a committed root for Acc, i.e. a proof of knowledge of e and W such that $W^e = \mathsf{Acc}$ and $C_e = G^e H^r$. Intuitively, this gives that C_e commits to an integer that is accumulated in Acc (at this point, however, the integer may be a trivial root, i.e., 1).
- A ZK proof $\mathsf{CP}_{\mathsf{modEq}}$ that C_e and c_e commit to the same value modulo q.
- A ZK proof $\mathsf{CP}_{\mathsf{Range}}$ that c_e commits to an integer in the range $\left(2^{\mu-1}, 2^{\mu}\right)$.

From the combination of the above proofs we would like to conclude that the integer committed in c_e is in P. Without further restrictions, however, this may not be the case; in particular, since for the value committed in C_e we do not have a strict bound it may be that the integer committed in c_e is another e_q such $e = e_q \pmod q$ but $e \neq e_q$ over the integers. In fact, the proof $\mathsf{CP}_{\mathsf{Root}}$ does not guarantee us that C_e commits to a single prime number e, but only that e divides $\prod_{e_i \in P} e_i$, namely e might be a product of a few primes in P or the corresponding negative value, while its residue modulo q may be some value that is not in the set—what we call a "collision". We solve this problem by taking in consideration that e_q is guaranteed by π_{Range} to be in $\left(2^{\mu-1}, 2^{\mu}\right)$ and by enhancing π_{Root} to also prove a bound on e: roughly speaking $|e| < 2^{2\lambda_s + \mu}$ for a statistical security parameter λ_s. Using this information we develop a careful analysis that bounds the probability that such collisions can happen for a malicious e (see Section 4.2 for more intuition).

In the following section we formally describe the type-based commitment scheme supported by our CP-SNARK, and a collection of building blocks. Then we present the $\mathsf{MemCP}_{\mathsf{RSA}}$ and $\mathsf{MemCP}_{\mathsf{RSAPrm}}$ CP-SNARKs in Sections 4.2 and 4.3 respectively, and finally we give instantiations for some of our building blocks in Section 4.4.

$\mathsf{Setup}(1^\lambda)$: Choose a prime order group $\mathbb{G}_q$ of order $q \in (2^{\nu-1}, 2^\nu)$ and generators $g, h \leftarrow\$ \mathbb{G}_q$. Return $\mathsf{ck} := (\mathbb{G}_q, g, h)$

$\mathsf{Commit}(\mathsf{ck}, \mathsf{t}_q, u)$: sample $r \leftarrow\$ \mathbb{G}_q$; return $(c, o) := (g^u h^r, r)$.

$\mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_q, c, u, r)$: Output 1 if $c = g^u h^r$; output 0 otherwise.

(a) PedCom

$\mathsf{Setup}(1^\lambda, 1^\mu)$: Let $N \leftarrow \mathsf{GenSRSAmod}(1^\lambda)$, $F \leftarrow\$ \mathbb{Z}_N^*$, and $\mathsf{H}_{\mathsf{prime}} \leftarrow\$ \mathcal{H}$; compute $G \leftarrow F^2 \bmod N \in \mathsf{QR}_N$. Return $\mathsf{ck} := (N, G, \mathsf{H}_{\mathsf{prime}})$.

$\mathsf{Commit}(\mathsf{ck}, \mathsf{t}_U, U)$: compute $P := \{\mathsf{H}_{\mathsf{prime}}(u) \mid u \in U\}$, $\mathsf{Acc} \leftarrow G^{\mathsf{prod}_P}$. Return $(c, o) := (\mathsf{Acc}, \varnothing)$.

$\mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_U, \mathsf{Acc}, U, \varnothing)$: compute $P := \{\mathsf{H}_{\mathsf{prime}}(u) \mid u \in U\}$ and return 1 iff $\mathsf{Acc} = G^{\mathsf{prod}_P} \bmod N$.

(b) $\mathsf{SetCom}_{\mathsf{RSA}}$

Fig. 4: RSA Accumulator and Pedersen commitment schemes for $\mathsf{RSAHashmem}$.

4.1 Preliminaries and Building Blocks

Notation. Given a set $U = \{u_1, \ldots, u_n\} \subset \mathbb{Z}$ of cardinality n we denote compactly with $\mathsf{prod}_U := \prod_{i=1}^n u_i$ the product of all its elements. We use capital letters for elements in an RSA group $\mathbb{Z}_N^*$, e.g., $G, H \in \mathbb{Z}_N^*$. Conversely, we use small letters for elements in a prime order group $\mathbb{G}_q$, e.g., $g, h \in \mathbb{G}_q$. Following this notation, we denote a commitment in a prime order group as $c \in \mathbb{G}_q$, while a commitment in an RSA group as $C \in \mathbb{Z}_N^*$.

Commitment Schemes. Our first CP-SNARK, called $\mathsf{MemCP}_{\mathsf{RSA}}$, is for a family of relations $R_{\mathsf{mem}} : \mathcal{D}_{\mathsf{elm}} \times \mathcal{D}_{\mathsf{set}}$ such that $\mathcal{D}_{\mathsf{elm}} = \{0,1\}^\eta$, $\mathcal{D}_{\mathsf{set}} = 2^{\mathcal{D}_{\mathsf{elm}}}$, and for a type-based commitment scheme that is the canonical composition $\mathsf{SetCom}_{\mathsf{RSA}} \bullet \mathsf{PedCom}$ of the two commitment schemes given in Fig. 4. PedCom is essentially a classical Pedersen commitment scheme in a group $\mathbb{G}_q$ of prime order q such that $q \in (2^{\nu-1}, 2^\nu)$ and $\eta < \nu$. PedCom is used to commit to set elements and its type is t_q. $\mathsf{SetCom}_{\mathsf{RSA}}$ is a (non-hiding) commitment scheme for sets of η-bit strings, that is built as an RSA accumulator [Bd94, BP97] to a set of μ-bit primes, each derived from an η-bit string by a deterministic function $\mathsf{H}_{\mathsf{prime}} : \{0,1\}^\eta \to \mathsf{Primes}\left(2^{\mu-1}, 2^\mu\right)$. $\mathsf{SetCom}_{\mathsf{RSA}}$ is computationally binding under the factoring assumption[15] and the collision resistance of $\mathsf{H}_{\mathsf{prime}}$. Its type for sets is t_U.

Hashing to primes. The problem of mapping arbitrary values to primes in a collision-resistant manner has been studied in the past, see e.g., [GHR99, CMS99, CS99], and in [FT14] a method to generate random primes is presented. Although the main idea of our scheme would work with any instantiation of $\mathsf{H}_{\mathsf{prime}}$, for the goal of significantly improving efficiency, our construction considers a specific class of $\mathsf{H}_{\mathsf{prime}}$ functions that work as follows. Let $\mathsf{H} : \{0,1\}^\eta \times \{0,1\}^\iota \to \{0,1\}^{\mu-1}$ be a collision-resistant function, and define $\mathsf{H}_{\mathsf{prime}}(u)$ as the function that starting with $j = 0$, looks for the first $j \in [0, 2^\iota - 1]$ such that the integer represented by the binary string $1|\mathsf{H}(u, j)$ is prime. In case it reaches $j = 2^\iota - 1$ it failed to find a prime and outputs $\perp$ [16]. We consider two main candidates of such function H (and thus $\mathsf{H}_{\mathsf{prime}}$):

- Pseudorandom function. Namely $\mathsf{H}(u, j) := \mathsf{F}_\kappa(u, j)$ where $\mathsf{F}_\kappa : \{0,1\}^{\eta+\iota}$ is a PRF with public seed κ and $\iota = \lceil \log \mu\lambda \rceil$. Due to the density of primes, the corresponding $\mathsf{H}_{\mathsf{prime}}$ runs in the expected running time $O(\mu)$ and $\perp$ is returned with probability $\leq \mathsf{exp}(-\lambda) = \mathsf{negl}(\lambda)$.[17] Under the random oracle heuristic, F can be instantiated with a hash function like SHA256.

[15] Here is why: finding two different sets of primes $P, P', P \neq P'$ such that $G^{\mathsf{prod}_P} = \mathsf{Acc} = G^{\mathsf{prod}_{P'}}$ implies finding an integer $\alpha = \mathsf{prod}_P - \mathsf{prod}_{P'} \neq 0$ such that $G^\alpha = 1$. This is known to lead to an efficient algorithm for factoring N.
[16] For specific instantiations of H, ι can be set so that $\perp$ is returned with negligible probability.
[17] We assume for simplicity that the function never outputs $\perp$, though it can happen with negligible probability.

– Deterministic map. $\mathsf{H}(u, j) := f(u) + j$ with $u > 2^{\eta-1}$ and $j \in (f(u), f(u+1))$, where $f(u) := \overline{2(u+2)\log_2(u+1)^2}$. The corresponding function $\mathsf{H}_{\mathsf{prime}}(u)$ is essentially the function that maps to the next prime after $f(u)$. This function is collision-free (indeed it requires to take $\mu > \eta$) and generates primes that can be smaller (in expectation) than the function above. Cramer's conjecture implies that the interval $(f(u), f(u+1))$ contains a prime when u is sufficiently large.

CP-NIZK for H computation and PedCom. We use a CP-NIZK $\mathsf{CP}_{\mathsf{HashEq}}$ for the relation $R_{\mathsf{HashEq}} : \{0,1\}^{\mu} \times \{0,1\}^{\eta} \times \{0,1\}^{\iota}$ defined as

$$R_{\mathsf{HashEq}}(u_1, u_2, \omega) = 1 \iff u_1 = (1|\mathsf{H}(u_2, \omega))$$

and for the commitment scheme PedCom. Essentially, with this scheme one can prove that two commitments c_e and c_u in $\mathbb{G}_q$ are such that $c_e = g^e h^{r_q}$, $c_u = g^u h^{r_u}$ and there exists j such that $e = (1|\mathsf{H}(u, j))$. As it shall become clear in our security proof, we do not have to prove all the iterations of H until finding j such that $(1|\mathsf{H}(u, j)) = \mathsf{H}_{\mathsf{prime}}(u)$ is prime, which saves significantly on the complexity of this CP-NIZK.

Integer Commitments. We use a scheme for committing to arbitrarily large integer values in RSA groups introduced by Fujisaki and Okamoto [FO97] and later improved in [DF02]. We briefly recall the commitment scheme. Let $\mathbb{Z}_N^*$ be an RSA group. The commitment key consists of two randomly chosen generators $G, H \in \mathbb{Z}_N^*$; to commit to any $x \in \mathbb{Z}$ one chooses randomly an $r \leftarrow_{\$} [1, N/2]$ and computes $C \leftarrow G^x H^r$; the verifier checks whether or not $C = \pm G^x H^r$. This commitment scheme is statistically hiding and computationally binding under the assumption that factoring is hard in $\mathbb{Z}_N^*$. Furthermore, a proof of knowledge of an opening was presented in [DF02], its knowledge soundness was based on the strong RSA assumption, and later found to be reducible to the plain RSA assumption in [CPP17]. We denote this commitment scheme as IntCom.

Strong-RSA Accumulators. As observed earlier, our commitment scheme for sets is an RSA accumulator Acc computed on the set of primes P derived from U through the map to primes, i.e., $P := \{\mathsf{H}_{\mathsf{prime}}(s) | s \in U\}$. In our construction we use the accumulator's feature for computing succinct membership witnesses, which we recall works as follows. Given $\mathsf{Acc} = G^{\prod_{e_i \in P} e_i} := G^{\mathsf{prod}_P}$, the membership witness for e_k is $W_k = G^{\prod_{e_i \in P \setminus \{e_k\}} e_i}$, which can be verified by checking if $W_k^{e_k} = \mathsf{Acc}$.

Argument of Knowledge of a Root. We make use of a zero-knowledge non-interactive argument of knowledge of a root of a public RSA group element $\mathsf{Acc} \in \mathsf{QR}_N$. This NIZK argument is called $\mathsf{CP}_{\mathsf{Root}}$. More precisely, it takes in an integer commitment to a $e \in \mathbb{Z}$ and then proves knowledge of an e-th root of Acc, i.e., of $W = \mathsf{Acc}^{\frac{1}{e}}$. More formally, $\mathsf{CP}_{\mathsf{Root}}$ is a NIZK for the relation $R_{\mathsf{Root}} : (\mathbb{Z}_N^* \times \mathsf{QR}_N \times \mathbb{N}) \times (\mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}_N^*)$ defined as
$R_{\mathsf{Root}}((C_e, \mathsf{Acc}, \mu), (e, r, W)) = 1$ iff

$$C_e = \pm G^e H^r \bmod N \ \wedge \ W^e = \mathsf{Acc} \bmod N \ \wedge \ |e| < 2^{\lambda_z + \lambda_s + \mu + 2}$$

where λ_z and λ_s are the statistical zero-knowledge and soundness security parameters respectively of the protocol $\mathsf{CP}_{\mathsf{Root}}$. $\mathsf{CP}_{\mathsf{Root}}$ is obtained by applying the Fiat-Shamir transform to a public-coin protocol that we propose based on ideas from the protocol of Camenisch and Lysysanskaya for proving knowledge of an accumulated value [CL02]. In [CL02], the protocol ensures that the committed integer e is in a specific range, different from 1 and positive. In our $\mathsf{CP}_{\mathsf{Root}}$ protocol we instead removed these constraints and isolated the portion of the protocol that only proves

knowledge of a root. We present the $\mathsf{CP}_{\mathsf{Root}}$ protocol in Section 4.4; its interactive public coin version is knowledge sound under the RSA assumption and statistical zero-knowledge. Finally, we notice that the relation R_{Root} is defined for statements where $\mathsf{Acc} \in \mathsf{QR}_N$, which may not be efficiently checkable given only N if Acc is adversarially chosen. Nevertheless $\mathsf{CP}_{\mathsf{Root}}$ can be used in larger cryptographic constructions that guarantee $\mathsf{Acc} \in \mathsf{QR}_N$ through some extra information, as is the case in our scheme.

Proof of Equality of Commitments in $\mathbb{Z}_N^*$ and $\mathbb{G}_q$. Our last building block, called $\mathsf{CP}_{\mathsf{modEq}}$, proves in zero-knowledge that two commitments, a Pedersen commitment in a prime order group and an integer commitment in an RSA group, open to the same value modulo the prime order $q = \mathsf{ord}(\mathbb{G})$. This is a conjunction of a classic Pedersen Σ-protocol and a proof of knowledge of opening of an integer commitment [DF02], i.e. for the relation

$$R_{\mathsf{modEq}}\left((C_e, c_e), (e, e_q, r, r_q)\right) = 1 \text{ iff } e = e_q \bmod q \wedge C_e = \pm G^e H^r \bmod N \wedge c_e = g^{e_q \bmod q} h^{r_q \bmod q}$$

We present $\mathsf{CP}_{\mathsf{modEq}}$ in Section 4.4.

4.2 Our CP-SNARK $\mathsf{MemCP}_{\mathsf{RSA}}$

We are now ready to present our CP-SNARK $\mathsf{MemCP}_{\mathsf{RSA}}$ for set membership. The scheme is fully described in Figure 5 and makes use of the building blocks presented in the previous section.

The KeyGen algorithm takes as input the commitment key of Com_1 and a description of R_{mem} and does the following: it samples a random generator $H \leftarrow_{\$} \mathsf{QR}_N$ so that (G, H) define a key for the integer commitment, and generate a CRS $\mathsf{crs}_{\mathsf{HashEq}}$ of the $\mathsf{CP}_{\mathsf{HashEq}}$ CP-NIZK.

For generating a proof, the ideas are similar to the ones informally described at the beginning of Section 4 for the case when set elements are prime numbers. In order to support sets U of arbitrary strings the main differences are the following: (i) we use $\mathsf{H}_{\mathsf{prime}}$ in order to derive a set of primes P from U, (ii) given a commitment c_u to an element $u \in \{0,1\}^\eta$, we commit to $e = \mathsf{H}_{\mathsf{prime}}(u)$ in c_e; (iii) we use the previously mentioned ideas to prove that c_e commits to an element in P (that is correctly accumulated), except that we replace the range proof π_{Range} with a proof π_{HashEq} that c_u and c_e commits to u and e respectively, such that $\exists j : e = (1|\mathsf{H}(u, j))$.

Remark 4.1 (On the support of larger η.). In order to commit to a set element $u \in \{0,1\}^\eta$ with the PedCom scheme we require $\eta < \nu$. This condition is actually used for ease of presentation. It is straightforward to extend our construction to the case $\eta \geq \nu$, in which case every u should be split in blocks of less than ν bits that can be committed using the vector Pedersen commitment.

The correctness of $\mathsf{MemCP}_{\mathsf{RSA}}$ can be checked by inspection: essentially, it follows from the correctness of all the building blocks and the condition that $\eta, \mu < \nu$. For succinctness, we observe that the commitments C_U, c_u and all the three proofs have size that does not depend on the cardinality of the set U, which is the only portion of the witness whose size is not a-priori fixed.

Proof of Security. Recall that the goal is to prove in ZK that c_u is a commitment to an element $u \in \{0,1\}^\eta$ that is in a set U committed in C_U. Intuitively, we obtain the security of our scheme from the conjunction of proofs for relations $R_{\mathsf{Root}}, R_{\mathsf{modEq}}$ and R_{HashEq}: (i) π_{HashEq} gives us that c_e commits to $e_q = (1|\mathsf{H}(u, j))$ for some j and for u committed in c_u. (ii) π_{modEq} gives that C_e commits to an integer e such that $e \mod q = e_q$ is committed in c_e. (iii) π_{Root} gives us that the integer e committed in C_e divides prod_P, where $C_U = G^{\mathsf{prod}_P}$ with $P = \{\mathsf{H}_{\mathsf{prime}}(u_i) : u_i \in U\}$.

$\mathsf{KeyGen}(\mathsf{ck}, R^{\in})$: parse $\mathsf{ck} := ((N, G, \mathsf{H}_{\mathsf{prime}}), (\mathbb{G}_q, g, h))$ as the commitment keys of $\mathsf{SetCom}_{\mathsf{RSA}}$ and PedCom respectively. Sample a random generator H.
Generate $\mathsf{crs}_{\mathsf{HashEq}} \leftarrow\$ \mathsf{CP}_{\mathsf{HashEq}}.\mathsf{KeyGen}((\mathbb{G}_q, g, h), R_{\mathsf{HashEq}})$, a crs for $\mathsf{CP}_{\mathsf{HashEq}}$.
Return $\mathsf{crs} := (N, G, H, \mathsf{H}_{\mathsf{prime}}, \mathbb{G}_q, g, h, \mathsf{crs}_{\mathsf{HashEq}})$.
Given crs, one can define $\mathsf{crs}_{\mathsf{Root}} := (N, G, H)$, $\mathsf{crs}_{\mathsf{modEq}} := (N, G, H, \mathbb{G}_q, g, h)$.

$\mathsf{Prove}(\mathsf{crs}, (C_U, c_u), (U, u), (\varnothing, r_u))$: $e \leftarrow \mathsf{H}_{\mathsf{prime}}(u) = (1|\mathsf{H}(u, j))$, $(c_e, r_q) \leftarrow \mathsf{Com}_1.\mathsf{Commit}(\mathsf{ck}, \mathsf{t}_q, e)$.
$(C_e, r) \leftarrow \mathsf{IntCom}.\mathsf{Commit}((G, H), e)$; $P \leftarrow \{\mathsf{H}_{\mathsf{prime}}(u) : u \in U\}$, $W = G^{\prod_{e_i \in P \setminus \{e\}} e_i}$.
$\pi_{\mathsf{Root}} \leftarrow \mathsf{CP}_{\mathsf{Root}}.\mathsf{Prove}(\mathsf{crs}_{\mathsf{Root}}, (C_e, C_U, \mu), (e, r, W))$
$\pi_{\mathsf{modEq}} \leftarrow \mathsf{CP}_{\mathsf{modEq}}.\mathsf{Prove}(\mathsf{crs}_{\mathsf{modEq}}, (C_e, c_e), (e, e, r, r_q))$
$\pi_{\mathsf{HashEq}} \leftarrow \mathsf{CP}_{\mathsf{HashEq}}.\mathsf{Prove}(\mathsf{crs}_{\mathsf{HashEq}}, (c_e, c_u), (e, u), (r_q, r_u), j)$
Return $\pi := (C_e, c_e, \pi_{\mathsf{Root}}, \pi_{\mathsf{modEq}}, \pi_{\mathsf{HashEq}})$.

$\mathsf{VerProof}(\mathsf{crs}, (C_U, c_u), \pi)$: Return 1 iff $\mathsf{CP}_{\mathsf{Root}}.\mathsf{VerProof}(\mathsf{crs}_{\mathsf{Root}}, (C_e, C_U, \mu), \pi_{\mathsf{Root}}) = 1 \wedge$
$\mathsf{CP}_{\mathsf{modEq}}.\mathsf{VerProof}(\mathsf{crs}_{\mathsf{modEq}}, (C_e, c_e), \pi_{\mathsf{modEq}}) = 1 \wedge \mathsf{CP}_{\mathsf{HashEq}}.\mathsf{VerProof}(\mathsf{crs}_{\mathsf{HashEq}}, (c_e, c_u), \pi_{\mathsf{HashEq}}) = 1$.

Fig. 5: $\mathsf{MemCP}_{\mathsf{RSA}}$ CP-SNARK for set membership

By combining these three facts we would like to conclude that $e_q \in P$ that, together with π_{HashEq}, should also guarantee $u \in U$. A first problem to analyze, however, is that for e we do not have guarantees of a strict bound in $\left(2^{\mu-1}, 2^{\mu}\right)$; so it may in principle occur that $e = e_q \pmod q$ but $e \neq e_q$ over the integers. Indeed, the relation R_{Root} does not guarantee us that e is a single prime number, but only that e divides the product of primes accumulated in C_U. Assuming the hardness of Strong RSA we may still have that e is the product of a few primes in P or even is a negative integer. We expose a simple attack that could arise from this: an adversary can find a product of primes from the set P, let it call e, such that $e = e_q \pmod q$ but $e \neq e_q$ over the integers. Since e is a legitimate product of members of P, the adversary can efficiently compute the e-th root of C_U and provide a valid π_{Root} proof. This is what we informally call a "collision". Another simple attack would be that an adversary takes a single prime e and then commits to its opposite $e_q \leftarrow -e \bmod q$ in the prime order group. Again, since $e \in P$ the adversary can efficiently compute the e-th root of C_U, $W^e = C_U$, and then the corresponding $-e$-th root of C_U, $\left(W^{-1}\right)^{-e} = C_U$. This is a second type of attack to achieve what we called "collision". With a careful analysis we show that with appropriate parameters the probability that such collisions occur can be either 0 or negligible.

One key observation is that R_{Root} does guarantee a lower and an upper bound, $-2^{\lambda_z+\lambda_s+\mu+2}$ and $2^{\lambda_z+\lambda_s+\mu+2}$ respectively, for e committed in C_e. From these bounds (and that $e \mid \mathsf{prod}_P$) we get that an adversarial e can be the product of *at most* $d = 1 + \lfloor \frac{\lambda_z+\lambda_s+2}{\mu} \rfloor$ primes in P (or their corresponding negative product). Then, if $2^{d\mu} \leq 2^{\nu-2} < q$, or $d\mu + 2 \leq \nu$, we get that $e < 2^{d\mu} < q$. In case $e > 0$ and since q is prime, $e = e_q \bmod q \wedge e < q$ implies that $e = e_q$ over $\mathbb{Z}$, namely no collision can occur at all. In the other case $e < 0$ we have $e > -2^{d\mu}$ and $e = e_q \pmod q$ implies $e = -q + e_q < -2^{\nu-1} + 2^{\mu} < -2^{\nu-1} + 2^{\nu-2} = -2^{\nu-2}$. Therefore, $-2^{d\mu} < -2^{\nu-2}$, which is a contradiction since we assumed $d\mu + 2 \leq \nu$. So this type of collision cannot happen.

If on the other hand we are in a parameters setting where $d\mu > \nu - 2$, we give a concrete bound on the probability that such collisions occur. More precisely, for this case we need to assume that the integers returned by H are random, i.e., H is a random oracle, and we also use the implicit fact that R_{HashEq} guarantees that $e_q \in \left(2^{\mu-1}, 2^{\mu}\right)$. Then we give a concrete bound on the probability that the product of d out of $\mathsf{poly}(\lambda)$ random integers lies in a specific range $\left(2^{\mu-1}, 2^{\mu}\right)$, which turns out to be negligible when d is constant and $2^{\mu-\nu}$ is negligible.

Since the requirements of security are slightly different according to the setting of parameters mentioned above, we state two separate theorems, one for each case.

Theorem 4.1. *Let* PedCom*,* $\mathsf{SetCom}_{\mathsf{RSA}}$ *and* IntCom *be computationally binding commitments,* $\mathsf{CP}_{\mathsf{Root}}$*,* $\mathsf{CP}_{\mathsf{modEq}}$ *and* $\mathsf{CP}_{\mathsf{HashEq}}$ *be knowledge-sound NIZK arguments, and assume that the Strong RSA assumption hold, and that* H *is collision resistant. If* $d\mu+2 \leq \nu$*, then* $\mathsf{MemCP}_{\mathsf{RSA}}$ *is knowledge-sound with partial opening of the set commitments* C_U*.*

Theorem 4.2. *Let* PedCom*,* $\mathsf{SetCom}_{\mathsf{RSA}}$ *and* IntCom *be computationally binding commitments,* $\mathsf{CP}_{\mathsf{Root}}$*,* $\mathsf{CP}_{\mathsf{modEq}}$ *and* $\mathsf{CP}_{\mathsf{HashEq}}$ *be knowledge-sound NIZK arguments, and assume that the Strong RSA assumption hold, and that* H *is collision resistant. If* $d\mu+2 > \nu$*,* $d = O(1)$ *is a small constant,* $2^{\mu-\nu} \in \mathsf{negl}(\lambda)$ *and* H *is modeled as a random oracle, then* $\mathsf{MemCP}_{\mathsf{RSA}}$ *is knowledge-sound with partial opening of the set commitments* C_U*.*

Remark 4.2. It is worth noting that Theorem 4.2 where we assume H to be a random oracle requires a random oracle assumption stronger than usual; this has to do with the fact that while we assume H to be a random oracle we also assume that $\mathsf{CP}_{\mathsf{modEq}}$ can create proof about correct computations of H. Similar assumptions have been considered in previous works, see, e.g, [Val08, Remark 2].

Finally, we state the theorem about the zero-knowledge of $\mathsf{MemCP}_{\mathsf{RSA}}$.

Theorem 4.3. *Let* PedCom*,* $\mathsf{SetCom}_{\mathsf{RSA}}$ *and* IntCom *be statistically hiding commitments,* $\mathsf{CP}_{\mathsf{Root}}$*,* $\mathsf{CP}_{\mathsf{modEq}}$ *and* $\mathsf{CP}_{\mathsf{HashEq}}$ *be zero-knowledge arguments. Then* $\mathsf{MemCP}_{\mathsf{RSA}}$ *is zero-knowledge.*

Proof [sketch] The proof is rather straightforward, so we only provide a sketch. We define the simulator $\mathcal{S}$ that takes as input (crs, C_U, c_u) and does the following:

- Parses $\mathsf{crs} := (N, G, H, \mathsf{H}_{\mathsf{prime}}, \mathbb{G}_q, g, h, \mathsf{crs}_{\mathsf{HashEq}})$, from which it computes the corresponding $\mathsf{crs}_{\mathsf{Root}} := (N, G, H)$ and $\mathsf{crs}_{\mathsf{modEq}} := (N, G, H, \mathbb{G}_q, g, h)$.
- Samples at random $C_e^* \leftarrow_{\$} \mathbb{Z}_N^*$ and $c_e^* \leftarrow_{\$} \mathbb{G}_q$.
- Invokes $\mathcal{S}_{\mathsf{Root}}(\mathsf{crs}_{\mathsf{Root}}, C_e^*, C_U)$, $\mathcal{S}_{\mathsf{modEq}}(\mathsf{crs}_{\mathsf{modEq}}, C_e^*, c_e^*)$ and $\mathcal{S}_{\mathsf{HashEq}}(\mathsf{crs}_{\mathsf{HashEq}}, c_e^*, c_u)$ the corresponding simulators of $\mathsf{CP}_{\mathsf{Root}}$, $\mathsf{CP}_{\mathsf{modEq}}$ and $\mathsf{CP}_{\mathsf{HashEq}}$ respectively. They output simulated proof π^*_{Root}, π^*_{modEq} and π^*_{HashEq} respectively.
- $\mathcal{S}$ outputs $(C_e^*, c_e^*, \pi^*_{\mathsf{Root}}, \pi^*_{\mathsf{modEq}}, \pi^*_{\mathsf{HashEq}})$.

Let $\pi := (C_e, c_e, \pi_{\mathsf{Root}}, \pi_{\mathsf{modEq}}, \pi_{\mathsf{HashEq}}) \leftarrow \mathsf{Prove}(\mathsf{crs}, (C_U, c_u), (U, u), (\varnothing, r_u))$ be the output of a real proof. Since IntCom and PedCom are statistically hiding C_e^* and c_e^* are indistinguishable from C_e and c_e resp. Finally, since $\mathsf{CP}_{\mathsf{Root}}$, $\mathsf{CP}_{\mathsf{modEq}}$ and $\mathsf{CP}_{\mathsf{HashEq}}$ are zero knowledge arguments π^*_{Root}, π^*_{modEq} and π^*_{HashEq} are indistinguishable from π_{Root}, π_{modEq} and π_{HashEq} resp. □

NOTATION. We introduce some notation that eases our proofs exposition. Let $U = \{u_1, \ldots, u_n\} \subset \mathbb{Z}$ be a set of cardinality n. We denote as prod a product of (an arbitrary number of) elements of U, $\mathsf{prod} = \prod_{i \in I} u_i$, for some $I \subseteq [n]$. Furthermore, $\Pi_U = \{\mathsf{prod}_1, \ldots, \mathsf{prod}_{2^n-1}\}$ is the set of all possible products and more specifically $\Pi_{U,d} \subseteq \Pi_U$ denotes the set of possible products of exactly d elements of U, $|I| = d$, while for the degenerate case of $d > n$ we define $\Pi_{U,d} = \emptyset$. We note that $|\Pi_{U,d}| = \binom{n}{d}$ (except for the degenerate case where $|\Pi_{U,d}| = 0$). For convenience, in the special case of $\mathsf{prod} \in \Pi_{U,|U|}$, i.e. the (unique) product of all elements of U, we will simply write prod_U. Finally, for a $J \subseteq [n]$ we let $\Pi_{U,J} = \cup_{j \in J} \Pi_{U,j}$; for example $\Pi_{U,[1,\ldots,d]} = \cup_{j=1}^{d} \Pi_{U,j}$ is the set of all possible

products of up to d elements of U. For all of the above we also denote with "$-$" the corresponding set of the opposite element, e.g. $-\Pi_U = \{-\mathsf{prod}_1, \ldots, -\mathsf{prod}_{2^n-1}\}$

Proof [of Theorem 4.1] Let a malicious prover $\mathcal{P}^*$, a PPT adversary of Knowledge Soundness with Partial Opening (see the definition in section 2.5) that on input $(\mathsf{ck}, R_{\mathsf{mem}}, \mathsf{crs}, \mathsf{aux}_R, \mathsf{aux}_Z)$ outputs (C_U, c_u, U, π) such that the verifier $\mathcal{V}$ accepts, i.e. $\mathsf{VerProof}(\mathsf{crs}, C_U, c_u), \pi) = 1$ and $\mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_U, C_U, U, \varnothing) = 1$ with non-negligible probability ϵ. We will construct a PPT extractor $\mathcal{E}$ that on the same input outputs a partial witness (u, r_q) such that $R_{\mathsf{mem}}(U, u) = 1 \wedge \mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_q, c_u, u, r_q) = 1$.

For this we rely on the Knowledge Soundness of $\mathsf{CP}_{\mathsf{Root}}, \mathsf{CP}_{\mathsf{modEq}}$ and $\mathsf{CP}_{\mathsf{HashEq}}$ protocols. $\mathcal{E}$ parses $\pi := (C_e, c_e, \pi_{\mathsf{Root}}, \pi_{\mathsf{modEq}}, \pi_{\mathsf{HashEq}})$ and $\mathsf{crs} := (N, G, H, \mathsf{H}_{\mathsf{prime}}, \mathbb{G}_q, g, h, \mathsf{crs}_{\mathsf{HashEq}})$, from which it computes the corresponding $\mathsf{crs}_{\mathsf{Root}} := (N, G, H)$ and $\mathsf{crs}_{\mathsf{modEq}} := (N, G, H, \mathbb{G}_q, g, h)$. Then constructs an adversary $\mathcal{A}_{\mathsf{Root}}$ for $\mathsf{CP}_{\mathsf{Root}}$ Knowledge Soundness that outputs $(C_e, C_U, \mu, \pi_{\mathsf{Root}})$. It is obvious that since $\mathcal{V}$ accepts π then it also accepts π_{Root}, i.e., $\mathsf{CP}_{\mathsf{Root}}.\mathsf{VerProof}(\mathsf{crs}_{\mathsf{Root}}, (C_e, C_U, \mu), \pi_{\mathsf{Root}}) = 1$. From Knowledge Soundness of $\mathsf{CP}_{\mathsf{Root}}$ we know that there is an extractor $\mathcal{E}_{\mathsf{Root}}$ that outputs (e, r, W) such that $C_e = \pm G^e H^r \pmod N \wedge W^e = C_U \pmod N \wedge |e| < 2^{\lambda_z+\lambda_s+\mu+2}$. Similarly, $\mathcal{E}$ constructs adversaries $\mathcal{A}_{\mathsf{modEq}}$ and $\mathcal{A}_{\mathsf{HashEq}}$ of protocols $\mathsf{CP}_{\mathsf{modEq}}$ and $\mathsf{CP}_{\mathsf{HashEq}}$ respectively. And similarly there are extractors $\mathcal{E}_{\mathsf{modEq}}$ and $\mathcal{E}_{\mathsf{HashEq}}$ that output (e', e_q, r', r_q) such that $e' = e_q \pmod q \wedge C_{e'} = \pm G^{e'} H^{r'} \pmod N \wedge c_{e_q} = g^{e_q \bmod q} h^{r_q \bmod q}$ and (e'_q, u, r'_q, r_u, j) such that $c_e = g^{e'_q} h^{r'_q} \wedge e'_q = (1|\mathsf{H}(u, j))$ respectively.

From the Binding property of the integer commitment scheme we get that $e = e'$ and $r = r'$ (over the integers), unless with a negligible probability. Similarly, from the Binding property of the Pedersen commitment scheme we get that $e_q = e'_q \pmod q$ and $r_q = r'_q \pmod q$, unless with a negligible probability. So if we put everything together the extracted values are $(e, r, W, e_q, r_q, u, r_u, j)$ such that:

$$W^e = C_U \pmod N \wedge |e| < 2^{\lambda_z+\lambda_s+\mu+2} \wedge e = e_q \pmod q \wedge e_q = (1|\mathsf{H}(u, j))$$

and additionally

$$C_e = \pm G^e H^r \wedge c_e = g^{e_q \bmod q} h^{r_q \bmod q} \wedge \mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_q, c_u, u, r_u) = 1$$

From $\mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_U, C_U, U, \varnothing) = 1$ we infer that $C_U = G^{\mathsf{prod}_P}$, where $P := \{\mathsf{H}_{\mathsf{prime}}(u) \mid u \in U\}$. From the strong RSA assumption since $W^e = C_U = G^{\mathsf{prod}_P} \pmod N$ we get $e \in \Pi_P$ or $e \in -\Pi_P$, unless with a negligible probability (see appendix A.1).

Since, all the elements of P are outputs of $\mathsf{H}_{\mathsf{prime}}$ they have exactly bitlength μ, that is $2^{\mu-1} < e_i < 2^\mu$ for each $e_i \in P$. This means that e is a $(\pm)$ product of μ-sized primes. Let $|e|$ be a product of ℓ primes, meaning that $2^{\ell(\mu-1)} < |e| < 2^{\ell\mu}$, and $d := \lfloor \frac{\lambda_z+\lambda_s+\mu+2}{\mu} \rfloor$. From $|e| < 2^{\lambda_z+\lambda_s+\mu+2}$ we get that $2^{\ell\mu} < 2^{\lambda_z+\lambda_s+\mu+2} \Rightarrow \ell < d$ which means that $e \in \Pi_{P,[1,\ldots,d]}$ or $e \in -\Pi_{P,[1,\ldots,d]}$ (i.e. e is a $(\pm)$ product of at most d primes).

First we show that $e \in \Pi_P$, i.e., that e cannot be negative. Let $e \in -\Pi_{P,[1,\ldots,d]}$. We use the fact that $e = e_q \pmod q$, so $e \leq -q + e_q < -2^{\nu-1} + 2^\mu < -2^{\nu-1} + 2^{\nu-2} = -2^{\nu-2}$. Since $-2^{d\mu} < e$ this leads to $-2^{d\mu} < -2^{\nu-2}$ which contradicts the assumption $d\mu + 2 \leq \nu$ (we used the fact that $e_q = (1|\mathsf{H}(u, j))$ to conclude that $2^{\mu-1} < e_q < 2^\mu$, which comes from the definition of H). So $e > 0$ or $e \in \Pi_{P,[1,\ldots,d]}$.

Recall that $e < 2^{d\mu}$. From the assumption $d\mu + 2 \leq \nu$ which means that $e < 2^{d\mu} < 2^{\nu-2} < q \Rightarrow e < q$. Since $e = e_q \pmod q$ and $e < q$ this means that $e = e_q$ over the integers. Again we are using

the fact that $e_q = (1|\mathsf{H}(u,j))$ to conclude that $2^{\mu-1} < e_q < 2^{\mu}$, which comes from the definition of H, and combined with $e = e_q$ we get that $2^{\mu-1} < e < 2^{\mu}$. The last fact means that $e \in \Pi_{P,\{1\}}$ (i.e. e is exactly one prime from P) otherwise it would exceed 2^{μ}, so $e \in P$.

Finally, $e = e_q = (1|\mathsf{H}(u,j)) = \mathsf{H}_{\mathsf{prime}}(u) \in P = \{\mathsf{H}_{\mathsf{prime}}(u_1), \ldots, \mathsf{H}_{\mathsf{prime}}(u_n)\}$, where $U := \{u_1, \ldots, u_n\}$. This means that there is an i such that $\mathsf{H}_{\mathsf{prime}}(u) = \mathsf{H}_{\mathsf{prime}}(u_i)$. From collision resistance of $\mathsf{H}_{\mathsf{prime}}$ we infer that $u = u_i$. So we conclude that $u \in U$ or $R_{\mathsf{mem}}(U,u) = 1$ and as shown above $\mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_q, c_u, u, r_u) = 1$. □

Collision Finding Analysis For the second theorem we cannot count on the formula $d\mu + 2 \leq \nu$ that ensures that the extracted integer e lies inside $[0, q-1]$. As explained above, we can only rely on the randomness of each prime to avoid the described "collisions". First, we formally define what a "collision" is through a probabilistic experiment, $\mathsf{CollisionFinding}$, and then we compute a concrete bound for the probability that this event happens, i.e. the experiment outputs 1. Finally, we state a theorem that shows this probability is asymptotically negligible under the assumption that $2^{\mu-\nu}$ is a negligible value (and d is a constant).

$\mathsf{CollisionFinding}(\mu, d, \mathbb{G}_q, n)$

Let $P \subseteq \mathsf{Primes}\left(2^{\mu-1}, 2^{\mu}\right)$ be a randomly chosen set of cardinality n, i.e. each $e \in P$ is chosen uniformly at random, $e_i \leftarrow\!\!\$\, \mathsf{Primes}\left(2^{\mu-1}, 2^{\mu}\right)$ meaning that:

1. e_i is prime.
2. $2^{\mu-1} \leq e_i \leq 2^{\mu}$
3. $Pr[e_i = x] = \frac{\mu}{2^{\mu}} + \mathsf{negl}(\lambda)$ for each $x \in \mathsf{Primes}\left(2^{\mu-1}, 2^{\mu}\right)$

The output of the experiment is 1 iff there exists $\mathsf{prod} \in \left(\Pi_{P,[2,d]} \cup -\Pi_{P,[2,d]}\right)$ such that $(\mathsf{prod} \mod q) \in \left(2^{\mu-1}, 2^{\mu}\right)$

Lemma 4.1. *Let $\mathbb{G}_q$ be a prime order group of order $q \in \left(2^{\nu-1}, 2^{\nu}\right)$ and μ such that $\mu < \nu$ then* $Pr[\mathsf{CollisionFinding}(\mu, d, \mathbb{G}_e, n) = 1] \leq 2 \cdot \sum_{j=2}^{d} \frac{\binom{n}{j} 2^{(j+1)\mu - j - \nu}(2^j - 1)}{\frac{2^{j\mu-j}}{(\mu-1)^j} - \binom{n}{j}}$

Proof First we will prove it for positive products, that is we bound the probability $Pr[\mathsf{CollisionFinding}(\mu, d, \mathbb{G}_e, n) = 1 | \mathsf{prod} \in \Pi_{P,[2,d]}]$. Let $\mathsf{prod} = q_1 ... q_j$ be a product of exactly j primes for a $2 \leq j \leq d$. Since $q_i \in \left(2^{\mu-1}, 2^{\mu}\right)$ we get $\mathsf{prod} = q_1 ... q_j \in \left(2^{j\mu-j}, 2^{j\mu}\right)$. Also $\mathbb{Z}_q^*$ is cyclic so we know that at most

$$\left\lceil \frac{\left|\left(2^{j\mu-j}, 2^{j\mu}\right)\right|}{q} \right\rceil = \left\lceil \frac{2^{j\mu} - 2^{j\mu-j}}{q} \right\rceil = \left\lceil \frac{2^{j\mu-j} \cdot (2^j - 1)}{q} \right\rceil \leq 2^{j\mu-j-\nu+1} \cdot (2^j - 1)$$

integers in $\left(2^{j\mu-j}, 2^{j\mu}\right)$ are equal to c modulo q, for any $c \in \{0, 1, ..., q-1\}$.

We are interested in the interval $\left(2^{\mu-1}, 2^{\mu}\right)$ modulo q. From the previous we get that at most $2^{j\mu-j-\nu+1} \cdot (2^j - 1) \cdot \left|\left(2^{\mu-1}, 2^{\mu}\right)\right| = 2^{j\mu-j-\nu+1} \cdot (2^j - 1) \cdot 2^{\mu-1} = 2^{(j+1)\mu-j-\nu}(2^j - 1)$ integers in the

range of $\left(2^{j\mu-j}, 2^{j\mu}\right)$ are "winning" integers for the adversary, meaning that after modulo q they are mapped to the winning interval $\left(2^{\mu-1}, 2^{\mu}\right)$.

From the distribution of primes we know that the number of primes in $\left(2^{\mu-1}, 2^{\mu}\right)$ is approximately $\frac{2^{\mu-1}}{\mu-1}$. So there are (approximately) $\left(\frac{2^{\mu-1}}{\mu-1}\right)^j = \frac{2^{j\mu-j}}{(\mu-1)^j}$ different products of j primes from $\mathsf{Primes}\left(2^{\mu-1}, 2^{\mu}\right)$ in $\left(2^{j\mu-d}, 2^{j\mu}\right)$. This leads us to the combinatorial experiment of choice of $B = \frac{2^{j\mu-j}}{(\mu-1)^j}$ "balls", with $T = 2^{(j+1)\mu-j-\nu}(2^j-1)$ "targets" and $X = \binom{n}{j}$ "tries" without replacement, where "balls" are all possible products, "targets" are the ones that go to $\left(2^{\mu-1}, 2^{\mu}\right)$ modulo q (the winning ones) and tries are the number of products (for a constant j) that the adversary can try. The "without replacement" comes from the fact that all products are different. The final winning probability is:

$$\begin{aligned} Pr[\mathsf{prod} \mod q \in \left(2^{\mu-1}, 2^{\mu}\right) \wedge \mathsf{prod} \in \Pi_{P,j}] &\leq \frac{T}{B} + \frac{T}{B-1} + \frac{T}{B-2} + \ldots + \frac{T}{B-X} \\ &\leq X \cdot \frac{T}{B-X} \\ &= \frac{\binom{n}{j} 2^{(j+1)\mu-j-\nu}(2^j-1)}{\frac{2^{j\mu-j}}{(\mu-1)^j} - \binom{n}{j}} \end{aligned}$$

By applying the union bound for all j's we get:

$$Pr[\mathsf{prod} \mod q \in \left(2^{\mu-1}, 2^{\mu}\right) \wedge \mathsf{prod} \in \Pi_{P,[2,d]}] \leq \sum_{j=2}^{d} \frac{\binom{n}{j} 2^{(j+1)\mu-j-\nu}(2^j-1)}{\frac{2^{j\mu-j}}{(\mu-1)^j} - \binom{n}{j}}$$

By using the same arguments for negative products we would conclude that

$$Pr[\mathsf{prod} \mod q \in \left(2^{\mu-1}, 2^{\mu}\right) \wedge \mathsf{prod} \in -\Pi_{P,[2,d]}] \leq \sum_{j=2}^{d} \frac{\binom{n}{j} 2^{(j+1)\mu-j-\nu}(2^j-1)}{\frac{2^{j\mu-j}}{(\mu-1)^j} - \binom{n}{j}}$$

Therefore

$$\begin{aligned} Pr[\mathsf{CollisionFinding}(\mu, d, \mathbb{G}_e, n) = 1] =& Pr[\mathsf{CollisionFinding}(\mu, d, \mathbb{G}_e, n) = 1 \wedge \mathsf{prod} \in \Pi_{P,[2,d]}] + \\ &+ Pr[\mathsf{CollisionFinding}(\mu, d, \mathbb{G}_e, n) = 1 \wedge \mathsf{prod} \in -\Pi_{P,[2,d]}] = \\ \leq& 2 \cdot \sum_{j=2}^{d} \frac{\binom{n}{j} 2^{(j+1)\mu-j-\nu}(2^j-1)}{\frac{2^{j\mu-j}}{(\mu-1)^j} - \binom{n}{j}} \end{aligned}$$

□

Theorem 4.4. *Let $\mathbb{G}_q$ be a prime order group of order $q \in \left(2^{\nu-1}, 2^{\nu}\right)$, μ such that $2^{\mu-\nu} \in \mathsf{negl}(\lambda)$, d constant and $n = \mathsf{poly}(\lambda)$ then $Pr[\mathsf{CollisionFinding}(\mu, d, \mathbb{G}_q, n) = 1] \in \mathsf{negl}(\lambda)$*

Proof Now $n = \mathsf{poly}(\lambda)$ so the set P is polynomially bounded. Due to lemma 4.1 it is straightforward that $Pr[\mathsf{CollisionFinding}(\mu, d, \mathbb{G}_q, n) = 1] \leq \sum_{j=2}^{d} \frac{\binom{n}{j} 2^{(j+1)\mu-j-\nu}(2^j-1)}{\frac{2^{j\mu-j}}{(\mu-1)^j} - \binom{n}{j}}$. Since d is constant, for any $j \in [2, d]$ $\binom{n}{j} = O(n^j)$ and we get:

$$2 \cdot \frac{\binom{n}{j} 2^{(j+1)\mu-j-\nu}(2^j-1)}{\frac{2^{j\mu-j}}{(\mu-1)^j} - \binom{n}{j}} = 2 \cdot \frac{O(n^j)2^{(j+1)\mu-j-\nu}(2^j-1)}{\frac{2^{j\mu-j}}{(\mu-1)^j} - O(n^j)}$$
$$= 2 \cdot \frac{O(n^j)(2^j-1)(\mu-1)^j}{\frac{2^{j\mu-j}}{2^{(j+1)\mu-j-\nu}} - \frac{O(n^j)(\mu-1)^j}{2^{(j+1)\mu-j-\nu}}}$$

$O(n^j)(2^j-1)(\mu-1)^j = \mathsf{poly}(\lambda)$ and $\frac{O(n^j)(\mu-1)^j}{2^{(j+1)\mu-j-\nu}} = \mathsf{negl}(\lambda)$. Also $\frac{2^{j\mu-j}}{2^{(j+1)\mu-j-\nu}} = 2^{\nu-\mu}$, therefore for j we get a probability bounded by $\frac{\mathsf{poly}(\lambda)2^{\mu-\nu}}{1-\mathsf{negl}(\lambda)2^{\mu-\nu}} = \mathsf{negl}(\lambda)$ by assumption.

Finally, $Pr[\mathsf{CollisionFinding}(\mu, d, \mathbb{G}_q, n) = 1] \leq (d-1) \cdot \mathsf{negl}(\lambda) = \mathsf{negl}(\lambda)$. □

Remark 4.3. For the sake of generality, in $\mathsf{CollisionFinding}$ we do not specify how the random primes are generated. In practice in our scheme they are outputs of the hash function $\mathsf{H}_{\mathsf{prime}}$ that we model as a random oracle.

Now we are ready to give the proof of theorem 4.2:

Proof [of theorem 4.2] The proof is almost the same as the one of Theorem 4.1 except for the next-to-last paragraph, i.e. the justification of $e \in \Pi_{P,\{1\}}$. Since $d\mu + 2 > \nu$ we cannot use the same arguments to conclude to it. However, still $e \in \left(\Pi_{P,[1,\ldots,d]} \cup -\Pi_{P,[1,\ldots,d]}\right)$.

Let $e \in \left(\Pi_{P,[1,\ldots,d]} \cup -\Pi_{P,[1,\ldots,d]}\right)$, it is straightforward to reduce this case to the the collision finding problem. Assume that the adversary $\mathcal{P}^*$ made q_{H} random oracle queries to H and let Q_{H} be the set of answers she received. Further assume that exactly $q_{\mathsf{H}_{\mathsf{prime}}}$ of the them are primes and let $Q_{\mathsf{H}_{\mathsf{prime}}}$ be the set of them. We note that $P \subseteq Q_{\mathsf{H}_{\mathsf{prime}}}$, unless a collision happened in H.

Now let $Q_{\mathsf{H}_{\mathsf{prime}}}$ be the set of the $\mathsf{CollisionFinding}(\mu, d, \mathbb{G}_q, |Q_{\mathsf{H}_{\mathsf{prime}}}|)$ experiment. It satisfies all three conditions since each $e_i \in Q_{\mathsf{H}_{\mathsf{prime}}}$ is an output of $\mathsf{H}_{\mathsf{prime}}$. Therefore e_i is prime, $2^{\mu-1} < e_i < 2^{\mu}$ and since H is modeled as a random oracle the outputs of $\mathsf{H}_{\mathsf{prime}}$ are uniformly distributed in $\mathsf{Primes}\left(2^{\mu-1}, 2^{\mu}\right)$. Then for the extracted e, we know that $e = e_q \pmod q \in \left(2^{\mu-1}, 2^{\mu}\right)$ and from the assumption $e \in \left(\Pi_{P,[1,\ldots,d]} \cup -\Pi_{P,[1,\ldots,d]}\right)$, which (as noted above) means that $e \in \left(\Pi_{Q_{\mathsf{H}_{\mathsf{prime}}},[2,\ldots,d]} \cup -\Pi_{Q_{\mathsf{H}_{\mathsf{prime}}},[2,\ldots,d]}\right)$. So $\mathsf{CollisionFinding}(\mu, d, \mathbb{G}_q, |Q_{\mathsf{H}_{\mathsf{prime}}}|) = 1$. Since the adversary is PPT $|Q_{\mathsf{H}_{\mathsf{prime}}}| = \mathsf{poly}(\lambda)$. Also, $d = O(1)$ and $2^{\mu-\nu} \in \mathsf{negl}(\lambda)$ (from the assumptions of the theorem) so the previous happens with a negligible probability according to theorem 4.4. So we conclude that, unless with a negligible probability, $e \in \Pi_{P,\{1\}}$. □

4.3 Our CP-SNARK for Set Membership for Primes Sets

In this section we show a CP-SNARK for set membership $\mathsf{MemCP}_{\mathsf{RSAPrm}}$ that supports set elements that are prime numbers of exactly μ bits, i.e., $\mathcal{D}_{\mathsf{elm}} = \mathsf{Primes}(2^{\mu-1}, 2^{\mu})$, and $\mathcal{D}_{\mathsf{set}} = 2^{\mathcal{D}_{\mathsf{elm}}}$. $\mathsf{MemCP}_{\mathsf{RSAPrm}}$ works for a type-based commitment scheme Com_2 that is the canonical composition $\mathsf{SetCom}_{\mathsf{RSA}'} \bullet \mathsf{PedCom}$ where $\mathsf{SetCom}_{\mathsf{RSA}'}$ is in Fig. 6 (it is essentially a simplification of $\mathsf{SetCom}_{\mathsf{RSA}}$ since elements are already primes).

The scheme $\mathsf{MemCP}_{\mathsf{RSAPrm}}$ is described in figure 7. Its building blocks are the same as the ones for $\mathsf{MemCP}_{\mathsf{RSA}}$ except that instead of a CP-NIZK for proving correctness of a map-to-prime

$\mathsf{Setup}(1^\lambda, 1^\mu)$: Sample an RSA modulus $N \leftarrow \mathsf{GenSRSAmod}(1^\lambda)$, a generator $F \leftarrow\!\!\$\, \mathbb{Z}_N^*$, compute $G \leftarrow F^2 \bmod N \in \mathsf{QR}_N$. Return $\mathsf{ck} := (N, G)$.

$\mathsf{Commit}(\mathsf{ck}, \mathsf{t}_U, U)$: compute $\mathsf{Acc} \leftarrow G^{\mathsf{prod}\, P}$. Return $(c, o) := (\mathsf{Acc}, \varnothing)$.

$\mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_U, \mathsf{Acc}, U, \varnothing)$: Return 1 if for all $e_i \in P$ $e_i \in \mathsf{Primes}\left(2^{\mu-1}, 2^\mu\right)$ and $\mathsf{Acc} = G^{\mathsf{prod}\, P} \bmod N$, and 0 otherwise.

Fig. 6: $\mathsf{SetCom}_{\mathsf{RSA}'}$ Commitment to Sets.

$\mathsf{KeyGen}(\mathsf{ck}, R^\in)$: parse $\mathsf{ck} := ((N, G), (\mathbb{G}_q, g, h))$ as the commitment keys of $\mathsf{SetCom}_{\mathsf{RSA}'}$ and PedCom respectively. Sample a random generator H.
Generate $\mathsf{crs}_{\mathsf{Range}} \leftarrow\!\!\$\, \mathsf{CP}_{\mathsf{Range}}.\mathsf{KeyGen}((\mathbb{G}_q, g, h), R_{\mathsf{Range}})$, a crs for $\mathsf{CP}_{\mathsf{Range}}$.
Return $\mathsf{crs} := (N, G, H, \mathbb{G}_q, g, h, \mathsf{crs}_{\mathsf{Range}})$.
Given crs, one can define $\mathsf{crs}_{\mathsf{Root}} := (N, G, H)$, $\mathsf{crs}_{\mathsf{modEq}} := (N, G, H, \mathbb{G}_q, g, h)$.

$\mathsf{Prove}(\mathsf{crs}, (C_P, c_e), (P, e), (\varnothing, r_q))$:
$(C_e, r) \leftarrow \mathsf{IntCom}.\mathsf{Commit}((G, H), e)$
$W = G^{\prod_{e_i \in P \setminus \{e\}} e_i}$.
$\pi_{\mathsf{Root}} \leftarrow \mathsf{CP}_{\mathsf{Root}}.\mathsf{Prove}(\mathsf{crs}_{\mathsf{Root}}, (C_e, C_P, \mu), (e, r, W))$
$\pi_{\mathsf{modEq}} \leftarrow \mathsf{CP}_{\mathsf{modEq}}.\mathsf{Prove}(\mathsf{crs}_{\mathsf{modEq}}, (C_e, c_e), (e, e, r, r_q))$
$\pi_{\mathsf{Range}} \leftarrow \mathsf{CP}_{\mathsf{Range}}.\mathsf{Prove}(\mathsf{crs}_{\mathsf{Range}}, (2^{\mu-1}, 2^\mu), c_e, e, r_q)$
Return $\pi := (C_e, \pi_{\mathsf{Root}}, \pi_{\mathsf{modEq}}, \pi_{\mathsf{Range}})$.

$\mathsf{VerProof}(\mathsf{crs}, (C_P, c_e), \pi)$: Return 1 iff
$\mathsf{CP}_{\mathsf{Root}}.\mathsf{VerProof}(\mathsf{crs}_{\mathsf{Root}}, (C_e, C_P, \mu), \pi_{\mathsf{Root}}) = 1 \wedge \mathsf{CP}_{\mathsf{modEq}}.\mathsf{VerProof}(\mathsf{crs}_{\mathsf{modEq}}, (C_e, c_e), \pi_{\mathsf{modEq}}) = 1 \wedge$
$\mathsf{CP}_{\mathsf{Range}}.\mathsf{VerProof}(\mathsf{crs}_{\mathsf{Range}}, c_e, \pi_{\mathsf{Range}}) = 1$.

Fig. 7: $\mathsf{MemCP}_{\mathsf{RSAPrm}}$ CP-SNARK for set membership

computation, we use a CP-NIZK for range proofs. Namely, we let $\mathsf{CP}_{\mathsf{Range}}$ be a NIZK for the following relation on PedCom commitments c and two given integers $A < B$:

$$R_{\mathsf{Range}}\left((c_e, A, B), (e, r_q)\right) = 1 \text{ iff } c = g^e h^{r_q} \wedge A < e_q < B$$

The idea behind the security of the scheme is similar to the one of the $\mathsf{MemCP}_{\mathsf{RSA}}$ scheme. The main difference is that here we rely on the range proof π_{Range} in order to "connect" the Pedersen commitment c_e to the accumulator. In particular, in order to argue the absence of possible collisions here we assume that $d\mu + 2 \leq \nu$ holds, namely we argue security only for this setting of parameters. It is worth noting that in applications where $\mathsf{D}_{\mathsf{elm}}$ is randomly chosen subset of $\mathsf{Primes}\left(2^{\mu-1}, 2^{\mu}\right)$, we could argue security even when $d\mu + 2 > \nu$, in a way similar to Theorem 4.2. We omit the analysis of this case from the paper.

Theorem 4.5. *Let* PedCom, $\mathsf{SetCom}_{\mathsf{RSA}'}$ *and* IntCom *be computationally binding commitments,* $\mathsf{CP}_{\mathsf{Root}}$, $\mathsf{CP}_{\mathsf{modEq}}$ *and* $\mathsf{CP}_{\mathsf{Range}}$ *be knowledge-sound NIZK arguments, and assume that the Strong RSA assumption hold. If* $d\mu + 2 \leq \nu$, *then* $\mathsf{MemCP}_{\mathsf{RSAPrm}}$ *is knowledge-sound with partial opening of the set commitments* c_P. *Furthermore, if* PedCom, $\mathsf{SetCom}_{\mathsf{RSA}'}$ *and* IntCom *are statistically hiding commitments, and* $\mathsf{CP}_{\mathsf{Root}}$, $\mathsf{CP}_{\mathsf{modEq}}$ *and* $\mathsf{CP}_{\mathsf{Range}}$ *be zero-knowledge, then* $\mathsf{MemCP}_{\mathsf{RSAPrm}}$ *is zero-knowledge.*

Proof [of Theorem 4.5] Knowledge Soundness with Partial Opening of C_P: the proof is similar to the one of theorem 4.1 except for some minor parts.
Let a malicious prover $\mathcal{P}^*$, a PPT adversary of Knowledge Soundness with Partial Opening (see the definition in section 2.5) that on input $(\mathsf{ck}, R_{\mathsf{mem}}, \mathsf{crs}, \mathsf{aux}_R, \mathsf{aux}_Z)$ outputs (C_P, c_e, P, π) such that the verifier $\mathcal{V}$ accepts, i.e. $\mathsf{VerProof}(\mathsf{crs}, C_P, c_e), \pi) = 1$ and $\mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_U, C_P, P, \varnothing) = 1$ with non-negligible probability ϵ. We will construct a PPT extractor $\mathcal{E}$ that on the same input outputs a partial witness (e, r) such that $R_{\mathsf{mem}}(P, e) = 1 \wedge \mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_q, c_e, e, r) = 1$.

For this we rely on the Knowledge Soundness of $\mathsf{CP}_{\mathsf{Root}}, \mathsf{CP}_{\mathsf{modEq}}$ and $\mathsf{CP}_{\mathsf{Range}}$ protocols. $\mathcal{E}$ parses $\pi := (C_e, \pi_{\mathsf{Root}}, \pi_{\mathsf{modEq}}, \pi_{\mathsf{Range}})$ and $\mathsf{crs} := (N, G, H, \mathsf{H}_{\mathsf{prime}}, \mathbb{G}_q, g, h, \mathsf{crs}_{\mathsf{Range}})$, from which it computes the corresponding $\mathsf{crs}_{\mathsf{Root}} := (N, G, H)$ and $\mathsf{crs}_{\mathsf{modEq}} := (N, G, H, \mathbb{G}_q, g, h)$. Then constructs an adversary $\mathcal{A}_{\mathsf{Root}}$ for $\mathsf{CP}_{\mathsf{Root}}$ Knowledge Soundness that outputs $(C_e, C_P, \mu, \pi_{\mathsf{Root}})$. It is obvious that since $\mathcal{V}$ accepts π then it also accepts π_{Root}, i.e., $\mathsf{CP}_{\mathsf{Root}}.\mathsf{VerProof}(\mathsf{crs}_{\mathsf{Root}}, (C_e, C_P, \mu), \pi_{\mathsf{Root}}) = 1$. From Knowledge Soundness of $\mathsf{CP}_{\mathsf{Root}}$ we know that there is an extractor $\mathcal{E}_{\mathsf{Root}}$ that outputs (e, r, W) such that $C_e = \pm G^e H^r \pmod N \wedge W^e = C_P \pmod N \wedge e < 2^{\lambda_z + \lambda_s + \mu + 2}$. Similarly, $\mathcal{E}$ constructs adversaries $\mathcal{A}_{\mathsf{modEq}}$ and $\mathcal{A}_{\mathsf{Range}}$ of protocols $\mathsf{CP}_{\mathsf{modEq}}$ and $\mathsf{CP}_{\mathsf{Range}}$ respectively. And similarly there are extractors $\mathcal{E}_{\mathsf{modEq}}$ and $\mathcal{E}_{\mathsf{Range}}$ that output (e', e_q, r', r_q) such that $e' = e_q \pmod q \wedge C_{e'} = \pm G^{e'} H^{r'} \pmod N \wedge c_{e_q} = g^{e_q \bmod q} h^{r_q \bmod q}$ and (e'_q, r'_q) such that $c_e = g^{e'_q} h^{r'_q} \wedge 2^{\mu-1} < e'_q < 2^{\mu}$ respectively.

From the Binding property of the integer commitment scheme we get that $e = e'$ and $r = r'$ (over the integers), unless with a negligible probability. Similarly, from the Binding property of the Pedersen commitment scheme we get that $e_q = e'_q \pmod q$ and $r_q = r'_q \pmod q$, unless with a negligible probability. So if we put everything together the extracted values are (e, r, W, e_q, r_q) such that:

$$W^e = C_P \pmod N \wedge e < 2^{\lambda_z + \lambda_s + \mu + 2} \wedge e = e_q \pmod q \wedge 2^{\mu-1} < e_q < 2^{\mu}$$

and additionally

$$C_e = \pm G^e H^r \wedge c_e = g^{e_q \bmod q} h^{r_q \bmod q}$$

From $\mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_U, C_P, P, \varnothing) = 1$ we infer that $C_P = G^{\mathsf{prod}_P}$, where for each $e_i \in P$ it holds that $e \in \mathsf{Primes}\left(2^{\mu-1}, 2^{\mu}\right)$. From the strong RSA assumption since $W^e = C_P = G^{\mathsf{prod}_P} \pmod N$ we get $e \in \Pi_P$, unless with a negligible probability (see appendix A.1).

The rest of the analysis that justifies $e \in P$ is identical to the one of the proof of theorem 4.1. So $e \in P$ and as shown above $\mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_q, c_e, e_q, r_q) = 1$.

ZERO KNOWLEDGE: For the Zero Knowledge Property we rely on similar techniques with the ones of the proof of theorem 4.3 except for the use of $\mathcal{S}_{\mathsf{HashEq}}$. Here we use instead the simulator of the $\mathsf{CP}_{\mathsf{Range}}$ protocol, $\mathcal{S}_{\mathsf{Range}}$. □

4.4 Proposed Instantiations of Protocols for R_{Root} and R_{modEq}

Root protocol

On common reference string $\mathsf{crs} = (\mathbb{Z}_N^*, G, H)$

$\mathsf{Prove}(\mathsf{crs}, (C_e, Acc), (e, r, w))$:

1. samples $r_2, r_3 \leftarrow\$ \left(-\lfloor N/4 \rfloor, \lfloor N/4 \rfloor\right)$ and computes $C_W \leftarrow WH^{r_2}, C_r \leftarrow G^{r_2}H^{r_3}$.
2. Computes the non-interactive version of the above protocol
   $r_e \leftarrow\$ \left(-2^{\lambda_z+\lambda_s+\mu}, 2^{\lambda_z+\lambda_s+\mu}\right), r_r, r_{r_2}, r_{r_3} \leftarrow\$ \left(-\lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s}, \lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s}\right)$
   $r_\beta, r_\delta \leftarrow\$ \left(-\lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s+\mu}, \lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s+\mu}\right)$
   $\alpha_1 \leftarrow G^{r_e}H^{r_r}, \alpha_2 \leftarrow G^{r_{r_2}}H^{r_{r_3}}, \alpha_3 \leftarrow C_W^{r_e}\left(\frac{1}{H}\right)^{r_\beta}, \alpha_4 \leftarrow C_r^{r_e}\left(\frac{1}{H}\right)^{r_\delta}\left(\frac{1}{G}\right)^{r_\beta}$
   $c \leftarrow \mathsf{H}(\alpha_1, \alpha_2, \alpha_3, \alpha_4, C_e, Acc)$
   $s_e \leftarrow r_e - ce, s_r \leftarrow r_r - cr, s_{r_2} \leftarrow r_{r_2} - cr_2, s_{r_3} \leftarrow r_{r_3} - cr_{r_3}, s_\beta \leftarrow r_\beta - cer_2, s_\delta \leftarrow r_\delta - cer_3$

   Returns $\pi \leftarrow (C_W, C_r, \alpha_1, \alpha_2, \alpha_3, \alpha_4, s_e, s_r, s_{r_2}, s_{r_3}, s_\beta, s_\delta)$

$\mathsf{VerProof}(\mathsf{crs}, (C_e, Acc), \pi)$: returns 1 iff $\alpha_1 = C_e^c G^{s_e} H^{s_r} \wedge \alpha_2 = C_r^c G^{s_{r_2}} H^{s_{r_3}} \wedge \alpha_3 = Acc^c C_W^{s_e}\left(\frac{1}{H}\right)^{s_\beta} \wedge \alpha_4 = C_r^{s_e}\left(\frac{1}{H}\right)^{s_\delta}\left(\frac{1}{G}\right)^{s_\beta} \wedge s_e \in \left[-2^{\lambda+\lambda_s+\mu+1}, 2^{\lambda+\lambda_s+\mu+1}\right]$

Fig. 8

Protocol $\mathsf{CP}_{\mathsf{Root}}$. We first give a protocol $\mathsf{CP}_{\mathsf{Root}'}$ for a simpler version of the Root relation in which the upper bound on e is removed; let us call $R_{\mathsf{Root}'}$ this relation.

Below is an interactive ZK protocol for $R_{\mathsf{Root}'}$:

1. Prover computes a W such that $W^e = Acc$ and $C_W = WH^{r_2}, C_r = G^{r_2}H^{r_3}$ and sends to the verifier:
   $\underline{\mathcal{P} \rightarrow \mathcal{V}} : C_W, C_r$
2. Prover and Verifier perform a protocol for the relation:
   $R((C_e, C_r, C_W, Acc), (e, r, r_2, r_3, \beta, \delta)) = 1$ iff

$$C_e = G^e H^r \wedge C_r = G^{r_2}H^{r_3} \wedge Acc = C_W^e\left(\frac{1}{H}\right)^\beta \wedge 1 = C_r^e\left(\frac{1}{H}\right)^\delta\left(\frac{1}{G}\right)^\beta$$

   Let λ_s be the size of the challenge space, λ_z be the statistical security parameter and μ the size of e.

– Prover samples:

$$r_e \leftarrow\$ \left(-2^{\lambda_z+\lambda_s+\mu}, 2^{\lambda_z+\lambda_s+\mu}\right)$$
$$r_r, r_{r_2}, r_{r_3} \leftarrow\$ \left(-\lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s}, \lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s}\right)$$
$$r_\beta, r_\delta \leftarrow\$ \left(-\lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s+\mu}, \lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s+\mu}\right)$$

and computes:

$$\alpha_1 = G^{r_e} H^{r_r}, \quad \alpha_2 = G^{r_{r_2}} H^{r_{r_3}}, \quad \alpha_3 = C_W^{r_e} \left(\frac{1}{H}\right)^{r_\beta}, \quad \alpha_4 = C_r^{r_e} (\frac{1}{H})^{r_\delta} \left(\frac{1}{G}\right)^{r_\beta}$$

$\underline{\mathcal{P} \to \mathcal{V}} : (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$

– Verifier samples the challenge $c \leftarrow \{0,1\}^{\lambda_s}$
$\underline{\mathcal{V} \to \mathcal{P}} : c$

– Prover computes the response:

$$s_e = r_e - ce$$
$$s_r = r_r - cr, \quad s_{r_2} = r_{r_2} - cr_2, \quad s_{r_3} = r_{r_3} - cr_{r_3}$$
$$s_\beta = r_\beta - cer_2, \quad s_\delta = r_\delta - cer_3$$

$\underline{\mathcal{P} \to \mathcal{V}} : (s_e, s_r, s_{r_2}, s_{r_3}, s_\beta, s_\delta)$

– Verifier checks if:

$$\alpha_1 \stackrel{?}{=} C_e^c G^{s_e} H^{s_r}, \quad \alpha_2 \stackrel{?}{=} C_r^c G^{s_{r_2}} H^{s_{r_3}}, \quad \alpha_3 \stackrel{?}{=} Acc^c C_W^{s_e} \left(\frac{1}{H}\right)^{s_\beta}, \quad \alpha_4 \stackrel{?}{=} C_r^{s_e} \left(\frac{1}{H}\right)^{s_\delta} \left(\frac{1}{G}\right)^{s_\beta}$$

Theorem 4.6. *Let $\mathbb{Z}_N^*$ be an RSA group where strong-RSA assumption holds, then the above protocol is a correct, knowledge sound and honest-verifier zero knowledge protocol for $R_{\mathsf{Root}'}$.*

The proof of the above is similar to the one of [CL02] where the more specific protocol was introduced, but implicitly was including a protocol for $R_{\mathsf{Root}'}$. Before proceeding to the proof we recall some properties related to RSA groups. First we expose two standard arguments. The first is that obtaining a multiple of $\phi(N)$ is equivalent to factoring N. This directly allows us to argue that for any $G \in \mathbb{Z}_N^*$, if one is able to find an $x \in \mathbb{Z}$ such that $G^x = 1 \pmod N$ then under the factoring assumption $x = 0$, otherwise x is a multiple of $\phi(N)$. Secondly, finding any non-trivial solution of the equation $\mu^2 = 1 \pmod N$ in $\mathbb{Z}_N^*$ (non-trivial means $\mu \neq \pm 1$) is equivalent to factoring N.

Remark 4.4. In 2017 Couteau et al. proved that in fact knowledge soundness for the protocol of opening an integer commitment can be reduced to (plain) RSA problem [CPP17]. This could be inherited to our protocol too. However, the relation itself assumes strong RSA's hardness, otherwise finding a root would be computable in polynomial time. Additionally, in the reduction to (plain) RSA, the extractor's probability of success is cubic, while in the reduction to strong RSA linear, in the adversary's probability of success.

Proposition 4.1. *Let $\mathbb{Z}_N^*$ be an RSA group with a modulus N and QR_N the corresponding group of quadratic residues modulo N.*

1. *Let $G, H \leftarrow_{\$} \mathsf{QR}_N$ two random generators of QR_N and a PPT adversary $\mathcal{A}$ outputting $\alpha, \beta \in \mathbb{Z}_N^*$ such that $G^\alpha H^\beta = 1$ then under the assumption that DLOG problem is hard in QR_N it holds that $\alpha = \beta = 0$.*
2. *Let $A, B \in \mathbb{Z}_N^*$ and a PPT adversary $\mathcal{A}$ outputting $x, y \in \mathbb{Z}_N^*$ such that $A^y = B^x$ and $y \mid x$ then under the assumption that factoring of N is hard it holds that $A = \pm B^{\frac{x}{y}}$*

Proof

1. Since $G, H \in \mathsf{QR}_N$ there is an $x \in \mathbb{Z}_N^*$ such that $G = H^x \pmod N$ which leads to $H^{x\alpha+\beta} = 1$. As we discussed above under the assumption that factoring of N is hard, $x\alpha + \beta = 0$. If $\alpha \neq 0$ then $x \leftarrow -\frac{\beta}{\alpha}$ is a discrete logarithm of H, so assuming that DLOG is hard $\alpha = 0$. Similarly, there is an $y \in \mathbb{Z}_N^*$ such that $G^y = H \pmod N$ and with a similar argument we can conclude that $\beta = 0$.
2. We discern two cases, $y = \rho$ is odd or $y = 2^v\rho$ is even (for an odd ρ). In case y is odd then it is co-prime with $\phi(N) = p'q'$ (otherwise if $y = p'$ or $y = q'$ we would be able to factor N), so $y^{-1} \pmod{\phi(N)}$ exists and $A = B^{\frac{x}{y}}$. If $y = 2^v\rho$ then $\left(A^{-1}B^{\frac{x}{y}}\right)^y = 1 \Rightarrow \left(A^{-1}B^{\frac{x}{y}}\right)^{2^v\rho} = 1 \Rightarrow \left(A^{-1}B^{\frac{x}{y}}\right)^{2^v} = 1$. From the second fact that we discussed above under the factoring assumption $\left(A^{-1}B^{\frac{x}{y}}\right)^{2^{v-1}} = \pm 1$. However for $v > 1$ the left part of the equation is a quadratic residue so it cannot be -1, therefore $\left(A^{-1}B^{\frac{x}{y}}\right)^{2^{v-1}} = 1$. Using the same facts repeatedly we will eventually conclude that $\left(A^{-1}B^{\frac{x}{y}}\right)^2 = 1$, hence $A^{-1}B^{\frac{x}{y}} = \pm 1 \Rightarrow A = \pm B^{\frac{x}{y}}$.

□

Proof [proof of theorem 4.6] Correctness is straightforward. Honest-verifier zero knowledge can be shown with standard arguments used in Σ-protocols and the fact that the commitments to C_e, C_W, C_r are statistically hiding. That is the simulator $\mathcal{S}$ on input (C_e, Acc) samples $C_W^* \leftarrow_{\$} \mathbb{Z}_N^*$ and $C_r^* \leftarrow_{\$} \mathbb{Z}_N^*$. Then samples

$$s_e^* \leftarrow_{\$} \left(-2^{\lambda_z+\lambda_s+\mu} - 2^{\lambda_z+\mu}, 2^{\lambda_z+\lambda_s+\mu} + 2^{\lambda_z+\mu}\right),$$

$$s_r^*, s_{r_2}^*, s_{r_3}^* \leftarrow_{\$} \left(-\lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s} - \lfloor N/4 \rfloor 2^{\lambda_s}, \lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s} + \lfloor N/4 \rfloor 2^{\lambda_s}\right),$$

$$s_\beta^*, s_\delta^* \leftarrow_{\$} \left(-\lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s+\mu} - \lfloor N/4 \rfloor 2^{\lambda_s+\mu}, \lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s+\mu} + \lfloor N/4 \rfloor 2^{\lambda_s+\mu}\right).$$

Finally it samples $c^* \leftarrow_{\$} \{0,1\}^{\lambda_s}$. Then it sets $\alpha_1^* \leftarrow C_e^c G^{s_e} H^{s_r}$, $\alpha_2^* \leftarrow C_r^c G^{s_{r2}} H^{s_{r3}}$, $\alpha_3^* \leftarrow \mathsf{Acc}^c C_W^{s_e} \left(\frac{1}{H}\right)^{s_\beta}$ and $\alpha_4^* \stackrel{?}{=} C_r^{s_e} \left(\frac{1}{H}\right)^{s_\delta} \left(\frac{1}{G}\right)^{s_\beta}$. $\mathcal{S}$ outputs $\pi^* \leftarrow (C_W^*, C_r^*, \alpha_1^*, \alpha_2^*, \alpha_3^*, \alpha_4^*, c^*, s_e^*, s_r^*, s_{r_2}^*, s_{r_3}^*, s_\beta^*, s_\delta^*)$. The distribution of π^* is identical to the one of a real proof π.

For the knowledge soundness, let an adversary of the knowledge soundness $\mathcal{A}$ that is able to convince the verifier $\mathcal{V}$ with a probability at least ϵ. We will construct an extractor $\mathcal{E}$ that extracts the witness $(e, r, r_2, r_3, \beta, \delta)$. Using rewinding $\mathcal{E}$ gets two accepted transcripts

$$(C_W, C_r, \alpha_1, \alpha_2, \alpha_3, \alpha_4, c, s_e, s_r, s_{r_2}, s_{r_3}, s_\beta, s_\delta) \text{ and } (C_W, C_r, \alpha_1, \alpha_2, \alpha_3, \alpha_4, c', s_e', s_r', s_{r_2}', s_{r_3}', s_\beta', s_\delta')$$

on two different challenges c and c'. $\mathcal{E}$ aborts if it cannot get two such transcripts (abort1).

We denote $\Delta c \coloneqq c' - c, \Delta s_e \coloneqq s_e - s'_e, \Delta s_r \coloneqq s_r - s'_r, \Delta s_{r_2} \coloneqq s_{r_2} - s'_{r_2}, \Delta s_{r_3} \coloneqq s_{r_3} - s'_{r_3}, \Delta s_\beta \coloneqq s_\beta - s'_\beta, \Delta s_\delta \coloneqq s_\delta - s'_\delta$ then

$$C_e^{\Delta c} = G^{\Delta s_e} H^{\Delta s_r}, C_r^{\Delta c} = G^{\Delta s_{r_2}} H^{\Delta s_{r_3}}, \mathsf{Acc}^{\Delta c} = C_W^{\Delta s_e} \left(\frac{1}{H}\right)^{\Delta s_\beta}, 1 = C_r^{\Delta s_e} \left(\frac{1}{H}\right)^{\Delta s_\delta} \left(\frac{1}{G}\right)^{\Delta s_\beta}$$

Define the (possibly rational) numbers $\hat{e} \coloneqq \frac{\Delta s_e}{\Delta c}, \hat{r} \coloneqq \frac{\Delta s_r}{\Delta c}, \hat{r_2} \coloneqq \frac{\Delta s_{r_2}}{\Delta c}, \hat{r_3} \coloneqq \frac{\Delta s_{r_3}}{\Delta c}$. In case Δc doesn't divide Δs_e and Δs_r, $\mathcal{E}$ aborts ($\mathsf{abort}\,2a$). Similarly, in case Δc doesn't divide Δs_{r_2} and Δs_{r_3}, $\mathcal{E}$ aborts ($\mathsf{abort}\,2b$). Therefore, since the above aborts didn't happen and according to second point of proposition 4.1, $C_e = \pm G^{\hat{e}} H^{\hat{r}}$ and $C_r = \pm G^{\hat{r_2}} H^{\hat{r_3}}$.

Now if we replace C_r in the fourth equation we get $1 = (\pm 1)^{\Delta s_e} G^{\hat{r_2}\Delta s_e} H^{\hat{r_3}\Delta s_e} \left(\frac{1}{H}\right)^{\Delta s_\delta} \left(\frac{1}{G}\right)^{\Delta s_\beta}$ or $(\pm 1)^{\Delta s_e} G^{\hat{r_2}\Delta s_e - \Delta s_\beta} H^{\hat{r_3}\Delta s_e - \Delta s_\delta} = 1$. However, $(\pm 1)^{\Delta s_e} = 1$ otherwise if $(\pm 1)^{\Delta s_e} = -1$ then $-G^{\hat{r_2}\Delta s_e - \Delta s_\beta} H^{\hat{r_3}\Delta s_e - \Delta s_\delta}$ would be a non-quadratic residue (since G, H are both in QR_N and QR_N is closed under multiplication) equal to 1 which is a quadratic residue and this would be a contradiction, hence $G^{\hat{r_2}\Delta s_e - \Delta s_\beta} H^{\hat{r_3}\Delta s_e - \Delta s_\delta} = 1$. According to the first point of proposition 4.1, under the factoring assumption $\hat{r_2}\Delta s_e - \Delta s_\beta = \hat{r_3}\Delta s_e - \Delta s_\delta = 0$, so $\hat{r_2}\Delta s_e = \Delta s_\beta$.

Finally we replace Δs_β in the third equation and we get $Acc^{\Delta c} = C_W^{\Delta s_e} \left(\frac{1}{H}\right)^{\hat{r_2}\Delta s_e} \Rightarrow Acc^{\Delta c} = \left(\frac{C_w}{H^{\hat{r_2}}}\right)^{\Delta s_e}$. As stated above Δc divides Δs_e so according to the second point of proposition 4.1 $\mathsf{Acc} = \pm \left(\frac{C_W}{H^{\hat{r_2}}}\right)^{\frac{\Delta s_e}{\Delta c}} = \pm \left(\frac{C_W}{H^{\hat{r_2}}}\right)^{\hat{e}}$. We discern three cases:

- $\mathsf{Acc} = + \left(\frac{C_W}{H^{\hat{r_2}}}\right)^{\frac{\Delta s_e}{\Delta c}}$: Then $\mathcal{E}$ sets $\tilde{W} \leftarrow \frac{C_W}{H^{\hat{r_2}}}$ and $\tilde{e} \leftarrow \hat{e} \coloneqq \frac{\Delta s_e}{\Delta c}$ $\tilde{r} \leftarrow \hat{r} \coloneqq \frac{\Delta s_r}{\Delta c}$ as above. It is clear that $\mathsf{Acc} = \tilde{W}^{\tilde{e}}$ and as stated above $C_e = G^{\tilde{e}} H^{\tilde{r}}$.
- $\mathsf{Acc} = - \left(\frac{C_W}{H^{\hat{r_2}}}\right)^{\frac{\Delta s_e}{\Delta c}}$ and $\frac{\Delta s_e}{\Delta c}$ odd: Then $\mathcal{E}$ sets $\tilde{W} \leftarrow -\frac{C_W}{H^{\hat{r_2}}}$ and $\tilde{e} \leftarrow \hat{e} \coloneqq \frac{\Delta s_e}{\Delta c}$ $\tilde{r} \leftarrow \hat{r} \coloneqq \frac{\Delta s_r}{\Delta c}$ as above. It is clear that $\mathsf{Acc} = \tilde{W}^{\tilde{e}}$ and as stated above $C_e = G^{\tilde{e}} H^{\tilde{r}}$.
- $\mathsf{Acc} = - \left(\frac{C_W}{H^{\hat{r_2}}}\right)^{\frac{\Delta s_e}{\Delta c}}$ and $\frac{\Delta s_e}{\Delta c}$ even: this means that Acc is a non-quadratic residue, which is a contradiction since in the $R_{\mathsf{Root}'}$ relation we assume that $\mathsf{Acc} \in \mathsf{QR}_N$.

Finally the $\mathcal{E}$ outputs $(\tilde{e}, \tilde{r}, \tilde{W})$.

Now we show that the probability the extractor terminates with outputting a valid witness is $O(\epsilon)$. If the extractor does not abort then it clearly outputs a valid witness (under factoring assumption). For the first abort, with a standard argument it can be shown that the extractor is able to extract two accepting transcripts with probability $O(\epsilon)$ (for the probabilistic analysis we refer to [DF02]). Thus $Pr[\mathsf{abort1}] = 1 - O(\epsilon)$. For the second type of aborts ($\mathsf{abort}\,2a$ and $\mathsf{abort}\,2b$), they happen with negligible probability under the strong RSA assumption. For the details see lemma 4.2 below, which was proven in [DF02]. Putting them together the probability of success of $\mathcal{E}$ is at least $O(\epsilon) - \mathsf{negl}(\lambda_s)$.

Lemma 4.2 ([DF02]). *Given that* $\mathsf{abort}\,2a$ *occurs a PPT adversary* $\mathcal{B}$ *can solve the strong RSA problem with probability at least* $\frac{1}{2} - 2^{-\lambda_s}$.

From the above we get $Pr[\mathcal{B} \text{ solves } sRSA] \geq \left(\frac{1}{2} - 2^{-\lambda_s}\right) Pr[\mathsf{abort}\,2a]$, so we conclude to $Pr[\mathsf{abort}\,2a] \leq \frac{1}{\frac{1}{2} - 2^{-\lambda_s}} Pr[\mathcal{B} \text{ solves } sRSA] = \mathsf{negl}(\lambda_s)$. The same lemma holds for $\mathsf{abort}\,2b$. □

Notice in the above protocol that

$$-2^{\lambda_z+\lambda_s+\mu} - 2^{\lambda_s+\mu} \leq s_e \leq 2^{\lambda_z+\lambda_s+\mu} + 2^{\lambda_s+\mu} \Rightarrow$$
$$-2^{\lambda_z+\lambda_s+\mu+1} \leq s_e \leq 2^{\lambda_z+\lambda_s+\mu+1} \Rightarrow$$
$$-2^{\lambda_z+\lambda_s+\mu+2} \leq \Delta s_e \leq 2^{\lambda_z+\lambda_s+\mu+2} \Rightarrow$$
$$-2^{\lambda_z+\lambda_s+\mu+2} \leq \hat{e} \leq 2^{\lambda_z+\lambda_s+\mu+2}$$

so if we impose an additional verification check of honest s_e size, i.e., $s_e \in \left[-2^{\lambda_z+\lambda_s+\mu+1}, 2^{\lambda_z+\lambda_s+\mu+1}\right]$, we get that $|\hat{e}| \leq 2^{\lambda_z+\lambda_s+\mu+2}$. The verifier performs an extra range check $s_e \stackrel{?}{\in} \left[-2^{\lambda_z+\lambda_s+\mu+1}, 2^{\lambda_z+\lambda_s+\mu+1}\right]$ and the resulting protocol is the $\mathsf{CP}_{\mathsf{Root}}$ that except for proving of knowledge of an e-th root also provides a bound for the size of $|e|$:

$$R_{\mathsf{Root}}\left((C_e, Acc, \mu), (e, r, W)\right) = 1 \text{ iff } C_e = \pm G^e H^r \pmod N \wedge W^e = Acc \pmod N \wedge |e| < 2^{\lambda_z+\lambda_s+\mu+2}$$

Protocol $\mathsf{CP}_{\mathsf{modEq}}$. Below we describe the public-coin ZK protocol for R_{modEq}. In Figure 9 we summarize the corresponding NIZK obtained after applying the Fiat-Shamir transform to it.

1. Prover samples:
$$r_e \leftarrow \left(-2^{\lambda_z+\lambda_s+\mu}, 2^{\lambda_z+\lambda_s+\mu}\right)$$
$$r_r \leftarrow \left(-\lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s}, \lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s}\right)$$
$$r_{r_q} \leftarrow \mathbb{Z}_q$$
and computes:
$$\alpha_1 = G^{r_e} H^{r_r}, \quad \alpha_2 = g^{r_e} \pmod p h^{r_{r_q}}$$
$\underline{\mathcal{P} \rightarrow \mathcal{V}}$: (α_1, α_2)
2. Verifier samples the challenge $c \leftarrow \{0,1\}^{\lambda_s}$
$\underline{\mathcal{V} \rightarrow \mathcal{P}}$: c
3. Prover computes the response:
$$s_e = r_e - ce$$
$$s_r = r_r - cr$$
$$s_{r_q} = r_{r_q} - cr_q \pmod q$$
$\underline{\mathcal{P} \rightarrow \mathcal{V}}$: (s_e, s_r, s_{r_q})
4. Verifier checks if:
$$\alpha_1 \stackrel{?}{=} \pm C_e^c G^{s_e} H^{s_r} \pmod N, \alpha_2 \stackrel{?}{=} c_{e_q}^c g^{s_e} \pmod q h^{s_{r_q}}$$

Theorem 4.7. *Let $\mathbb{Z}_N^*$ be an RSA group where strong-RSA assumption holds and $\mathbb{G}$ be a prime order group where DLOG assumption holds then the above protocol is a correct, knowledge sound and honest-verifier zero knowledge protocol for R_{modEq}.*

The proof is quite simple and is omitted.

modEq protocol

On common reference string $\mathsf{crs} = (\mathbb{Z}_N^*, G, H, \mathbb{G}, g, h)$

$\mathsf{Prove}(\mathsf{crs}, (C_e, c_{e_q}), (e, e_q, r, r_q))$:
$r_e \leftarrow\$ \left(-2^{\lambda_z+\lambda_s+\mu}, 2^{\lambda_z+\lambda_s+\mu}\right), r_r \leftarrow\$ \left(-\lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s}, \lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s}\right), r_{r'} \leftarrow\$ \mathbb{Z}_q$
$\alpha_1 \leftarrow G^{r_e} H^{r_r}, \alpha_2 \leftarrow g^{r_e \pmod q} h^{r_{r'}}$
$c \leftarrow \mathsf{H}(\alpha_1, \alpha_2, C_e, c_{e_q})$
$s_e \leftarrow r_e - ce, s_r \leftarrow r_r - cr, s_{r'} \leftarrow r_{r_q} - cr_q \pmod q$
Returns $\pi \leftarrow (\alpha_1, \alpha_2, s_e, s_r, s_{r_q})$

$\mathsf{VerProof}(\mathsf{crs}, (C_e, c_{e_q}), \pi)$: returns 1 iff $\alpha_1 = \pm C_e^c G^{s_e} H^{s_r} \pmod N \wedge \alpha_2 = c_{e_q}^c g^{s_e \pmod q} h^{s_{r_q}}$

Fig. 9

4.5 Instantiations

We discuss the possible instantiations of our schemes $\mathsf{MemCP}_{\mathsf{RSA}}$ and $\mathsf{MemCP}_{\mathsf{RSAPrm}}$ that can be obtained by looking at applications' constraints and security parameters constraints.

Parameters for $d\mu + 2 \leq \nu$ and $\mu \leq \nu - 2$. First we analyze possible parameters that satisfy the conditions $d\mu + 2 \leq \nu \wedge \mu \leq \nu - 2$ that is used in Theorems 4.1 and 4.2; we recall $d = 1 + \lfloor \frac{\lambda_z+\lambda_s+2}{\mu} \rfloor$, where λ_z and λ_s are statistical security parameters for zero-knowledge and soundness respectively of $\mathsf{CP}_{\mathsf{Root}}$.

If the prime order group $\mathbb{G}_q$ is instantiated with (pairing-friendly) elliptic curves, then the bitsize ν of its order must be at least 2λ. And recall that for correctness we need $\mu < \nu$.

Considering these constraints, one way to satisfy $d\mu + 2 \leq \nu$ is to choose μ such that $\nu - 1 > \mu > \lambda_z + \lambda_s + 2$. More specifically, a choice that maximizes security is $\nu = 2\lambda$, $\mu = 2\lambda - 2$ and $\lambda_z = \lambda - 3, \lambda_s = \lambda - 2$. For the case of the $\mathsf{MemCP}_{\mathsf{RSA}}$ scheme, this choice yields an instantiation with nearly λ bits of security and where the function H does not necessarily need to be a random oracle (yet it must be collision resistant).

Because of the constraint $\mu > \lambda_z + \lambda_s + 2$, we the choice above implies the use of large primes. This would be anyway the case if one instantiates the scheme with a collision-resistant hash function H (e.g., SHA256 or SHA3), e.g., because set elements are quite arbitrary. If on the other hand, one could support more specific set elements, one could use instead a deterministic map-to-primes or even use our scheme $\mathsf{MemCP}_{\mathsf{RSAPrm}}$ in which set elements themselves are primes. In this case one may wonder if it is possible to choose values of μ smaller than 2λ; for example $\mu \approx 30, 60, 80$. The answer is positive although the characterization of such μ's require an involved analysis.

Let us fix $\nu = 2\lambda$, and say that the statistical security parameters λ_z, λ_s are such that $\lambda_z + \lambda_s + 2 = 2\lambda - 2 - c$ for some constant c (for example $c = 4$ if $\lambda_z = \lambda_s = \lambda - 4$). We are essentially looking for μ such that

$$\mu \leq 2\lambda - 2 - c \text{ and } \mu + \mu \left\lfloor \frac{2\lambda - 2}{\mu} - \frac{c}{\mu} \right\rfloor \leq 2\lambda - 2$$

$$\iff \mu \leq 2\lambda - 2 - c \text{ and } \left\lfloor \frac{2\lambda - 2}{\mu} - \frac{c}{\mu} \right\rfloor \leq \frac{2\lambda - 2}{\mu} - 1$$

From the fact $x \mod y = x - y\lfloor \frac{x}{y} \rfloor$, we can reduce the above inequality into

$$\mu \leq 2\lambda - 2 - c \text{ and } 2\lambda - 2 - c \bmod \mu \geq \mu - c$$

that can admit solutions for $c \geq 2$.

For instance, if $\lambda = 128$ and $c = 4$, then we get several options for μ, e.g., $\mu = 32, 42, 63, 84, 126, 127$.

Parameters for $d\mu + 2 > \nu$. This case concerns only $\mathsf{MemCP}_{\mathsf{RSA}}$ and Theorem 4.2 in particular. In this case, if one aims at maximizing security, say to get a scheme with λ-bits of security, then would have to set $\mu \approx 2\lambda$ for collision resistance, and consequently select the prime order group so that $\nu \geq 3\lambda$. This choice however is costly in terms of performance since the efficiency of all protocols that work in the prime order group degrades. Nevertheless, in our full paper we will analyze intermediate cases in which we can adjust the parameters in order to get some concrete security bounds that are still reasonable.

5 A CP-SNARK for Set Non-Membership with Short Parameters

Here we describe two CP-SNARKs for set non-membership that work in a setting identical to the one of section 4. Namely, the set is committed using an RSA accumulator, and the element (that one wants to prove not to belong to the set) is committed using a Pedersen commitment scheme. As in the previous section, we propose two protocols for non-membership, called $\mathsf{NonMemCP}_{\mathsf{RSA}}$ and $\mathsf{NonMemCP}_{\mathsf{RSAPrm}}$, in complete analogy to $\mathsf{MemCP}_{\mathsf{RSA}}$ and $\mathsf{MemCP}_{\mathsf{RSAPrm}}$. In the former, the elements of the set are arbitrary bit-strings of length η, $\mathcal{D}_{\mathsf{elm}} = \{0,1\}^\eta$, while in the latter the elements are primes of length μ. The schemes are fully described in figures 10 and 11.

An High-Level Overview of the Constructions. The main idea of $\mathsf{NonMemCP}_{\mathsf{RSA}}$ is similar to the one of the corresponding membership protocol, $\mathsf{MemCP}_{\mathsf{RSA}}$. It uses in the same modular way the modEq and HashEq protocols. The only difference lies in the third protocol: instead of using Root it uses a new protocol $\mathsf{Coprime}$. In a similar manner, $\mathsf{NonMemCP}_{\mathsf{RSAPrm}}$ uses modEq, Range and $\mathsf{Coprime}$.

Let us explain the need of the $\mathsf{Coprime}$ protocol and what it does. First, recall how a non-membership proof is computed in RSA Accumulators [LLX07]. Let P be a set of primes to be accumulated and prod the corresponding product. For any prime element $e \notin P$ it holds that $\gcd(e, \mathsf{prod}) = 1$, while for any member $e \in P$ it is $\gcd(e, \mathsf{prod}) = e \neq 1$. Thus, proving that $\gcd(e, \mathsf{prod}) = 1$ would exhibit non-membership of e in P. Recall, also, that using the extended Euclidean algorithm one can efficiently compute coefficients (a, b) such that $a \cdot e + b \cdot \mathsf{prod} = \gcd(e, \mathsf{prod})$. A non-membership proof for an element e w.r.t. an accumulator $\mathsf{Acc} = G^{\mathsf{prod}}$ consists of a pair $(D = G^a, b)$, where a, b are such that $a \cdot e + b \cdot \mathsf{prod} = 1$. The verification is $D^e \mathsf{Acc}^b = G$, which ensures that e and prod are coprime, i.e. $\gcd(e, \mathsf{prod}) = 1$. Therefore, the goal of the $\mathsf{Coprime}$ protocol is to prove knowledge of an element e committed in an integer commitment C_e that satisfies this relation. A more formal definition of $\mathsf{Coprime}$ is given below and an instantiation of this protocol is in Section 5.1.

Argument of Knowledge for a coprime element. We make use of a non-interactive argument of knowledge of a non-membership witness of an element such that the verification equation explained above holds. More formally $\mathsf{CP}_{\mathsf{Coprime}}$, is a NIZK for the relation: $R_{\mathsf{Coprime}} : (\mathbb{Z}_N^* \times \mathsf{QR}_N) \times (\mathbb{Z} \times \mathbb{Z} \times \mathsf{QR}_N \times \mathbb{Z})$ defined as

$\mathsf{KeyGen}(\mathsf{ck}, R^{\in})$: parse $\mathsf{ck} := ((N, G, \mathsf{H}_{\mathsf{prime}}), (\mathbb{G}_q, g, h))$ as the commitment keys of $\mathsf{SetCom}_{\mathsf{RSA}}$ and PedCom respectively. Sample a random generator H.
Generate $\mathsf{crs}_{\mathsf{HashEq}} \leftarrow\$ \mathsf{CP}_{\mathsf{HashEq}}.\mathsf{KeyGen}((\mathbb{G}_q, g, h), R_{\mathsf{HashEq}})$, a crs for $\mathsf{CP}_{\mathsf{HashEq}}$.
Return $\mathsf{crs} := (N, G, H, \mathsf{H}_{\mathsf{prime}}, \mathbb{G}_q, g, h, \mathsf{crs}_{\mathsf{HashEq}})$.
Given crs, one can define $\mathsf{crs}_{\mathsf{Coprime}} := (N, G, H)$, $\mathsf{crs}_{\mathsf{modEq}} := (N, G, H, \mathbb{G}_q, g, h)$.

$\mathsf{Prove}(\mathsf{crs}, (C_U, c_u), (U, u), (\varnothing, r_u))$: $e \leftarrow \mathsf{H}_{\mathsf{prime}}(u) = (1|\mathsf{H}(u, j))$, $(c_e, r_q) \leftarrow \mathsf{Com}_1.\mathsf{Commit}(\mathsf{ck}, \mathsf{t}_q, e)$.
$(C_e, r) \leftarrow \mathsf{IntCom}.\mathsf{Commit}((G, H), e)$; $P \leftarrow \{\mathsf{H}_{\mathsf{prime}}(u) : u \in U\}$, compute a, b s.t. $a \cdot e + b \cdot \prod_{e_i \in P} e_i = 1$ and set $D = G^a$.
$\pi_{\mathsf{Coprime}} \leftarrow \mathsf{CP}_{\mathsf{Coprime}}.\mathsf{Prove}(\mathsf{crs}_{\mathsf{Coprime}}, (C_e, C_U, \mu), (e, r, D, b))$
$\pi_{\mathsf{modEq}} \leftarrow \mathsf{CP}_{\mathsf{modEq}}.\mathsf{Prove}(\mathsf{crs}_{\mathsf{modEq}}, (C_e, c_e), (e, e, r, r_q))$
$\pi_{\mathsf{HashEq}} \leftarrow \mathsf{CP}_{\mathsf{HashEq}}.\mathsf{Prove}(\mathsf{crs}_{\mathsf{HashEq}}, (c_e, c_u), (e, u), (r_q, r_u), j)$
Return $\pi := (C_e, c_e, \pi_{\mathsf{Root}}, \pi_{\mathsf{modEq}}, \pi_{\mathsf{HashEq}})$.

$\mathsf{VerProof}(\mathsf{crs}, (C_U, c_u), \pi)$: Return 1 iff $\mathsf{CP}_{\mathsf{Root}}.\mathsf{VerProof}(\mathsf{crs}_{\mathsf{Coprime}}, (C_e, C_U, \mu), \pi_{\mathsf{Coprime}}) = 1 \wedge$
$\mathsf{CP}_{\mathsf{modEq}}.\mathsf{VerProof}(\mathsf{crs}_{\mathsf{modEq}}, (C_e, c_e), \pi_{\mathsf{modEq}}) = 1 \wedge \mathsf{CP}_{\mathsf{HashEq}}.\mathsf{VerProof}(\mathsf{crs}_{\mathsf{HashEq}}, (c_e, c_u), \pi_{\mathsf{HashEq}}) = 1$.

Fig. 10: $\mathsf{NonMemCP}_{\mathsf{RSA}}$ CP-SNARK for set non-membership

$\mathsf{KeyGen}(\mathsf{ck}, R^{\in})$: parse $\mathsf{ck} := ((N, G, \mathsf{H}_{\mathsf{prime}}), (\mathbb{G}_q, g, h))$ as the commitment keys of $\mathsf{SetCom}_{\mathsf{RSA}'}$ and PedCom respectively. Sample a random generator H.
Generate $\mathsf{crs}_{\mathsf{Range}} \leftarrow\$ \mathsf{CP}_{\mathsf{Range}}.\mathsf{KeyGen}((\mathbb{G}_q, g, h), R_{\mathsf{Range}})$, a crs for $\mathsf{CP}_{\mathsf{Range}}$.
Return $\mathsf{crs} := (N, G, H, \mathsf{H}_{\mathsf{prime}}, \mathbb{G}_q, g, h, \mathsf{crs}_{\mathsf{Range}})$.
Given crs, one can define $\mathsf{crs}_{\mathsf{Coprime}} := (N, G, H)$, $\mathsf{crs}_{\mathsf{modEq}} := (N, G, H, \mathbb{G}_q, g, h)$.

$\mathsf{Prove}(\mathsf{crs}, (C_P, c_e), (P, e), (\varnothing, r_e))$: $(C_e, r) \leftarrow \mathsf{IntCom}.\mathsf{Commit}((G, H), e)$; , compute a, b s.t. $a \cdot e + b \cdot \prod_{e_i \in P} e_i = 1$ and set $D = G^a$.
$\pi_{\mathsf{Coprime}} \leftarrow \mathsf{CP}_{\mathsf{Coprime}}.\mathsf{Prove}(\mathsf{crs}_{\mathsf{Coprime}}, (C_e, C_P, \mu), (e, r, D, b))$
$\pi_{\mathsf{modEq}} \leftarrow \mathsf{CP}_{\mathsf{modEq}}.\mathsf{Prove}(\mathsf{crs}_{\mathsf{modEq}}, (C_e, c_e), (e, e, r, r_q))$
$\pi_{\mathsf{Range}} \leftarrow \mathsf{CP}_{\mathsf{Range}}.\mathsf{Prove}(\mathsf{crs}_{\mathsf{Range}}, (2^{\mu-1}, 2^{\mu}), c_e, e, r_q)$
Return $\pi := (C_e, c_e, \pi_{\mathsf{Coprime}}, \pi_{\mathsf{modEq}}, \pi_{\mathsf{Range}})$.

$\mathsf{VerProof}(\mathsf{crs}, (C_P, c_e), \pi)$: Return 1 iff $\mathsf{CP}_{\mathsf{Coprime}}.\mathsf{VerProof}(\mathsf{crs}_{\mathsf{Coprime}}, (C_e, C_P, \mu), \pi_{\mathsf{Coprime}}) = 1 \wedge$
$\mathsf{CP}_{\mathsf{modEq}}.\mathsf{VerProof}(\mathsf{crs}_{\mathsf{modEq}}, (C_e, c_e), \pi_{\mathsf{modEq}}) = 1 \wedge \mathsf{CP}_{\mathsf{Range}}.\mathsf{VerProof}(\mathsf{crs}_{\mathsf{Range}}, c_e, \pi_{\mathsf{Range}}) = 1$.

Fig. 11: $\mathsf{NonMemCP}_{\mathsf{RSAPrm}}$ CP-SNARK for set non-membership

$R_{\mathsf{Coprime}}((C_e, \mathsf{Acc}), (e, r, D, b)) = 1$ iff

$$C_e = \pm G^e H^r \bmod N \ \wedge \ D^e \mathsf{Acc}^b = G \wedge |e| < 2^{\lambda_z + \lambda_s + \mu + 2}$$

We propose an instantiation of a protocol for the above relation in the Section 5.1.

Our Constructions of $\mathsf{NonMemCP}_{\mathsf{RSA}}$ and $\mathsf{NonMemCP}_{\mathsf{RSAPrm}}$. In Figures 10 and 11 we give a full description of the schemes.

The security of these schemes follow very closely the one of the corresponding membership schemes given in Section 4. Below we give the Theorems that state their security. The proofs are omitted since they are almost identical to the corresponding proofs for the membership schemes.

Theorem 5.1. *Let* PedCom, $\mathsf{SetCom}_{\mathsf{RSA}}$ *and* IntCom *be computationally binding commitments,* $\mathsf{CP}_{\mathsf{Coprime}}$, $\mathsf{CP}_{\mathsf{modEq}}$ *and* $\mathsf{CP}_{\mathsf{HashEq}}$ *be knowledge-sound NIZK arguments, and assume that the Strong*

RSA assumption hold, and that H *is collision resistant. If* $d\mu+2 \leq \nu$, $\lambda_s+1 < \mu$ *and* $\lambda_s < \log(N)/2$ *then* $\mathsf{NonMemCP}_{\mathsf{RSA}}$ *is knowledge-sound with partial opening of the set commitments* C_U.

Theorem 5.2. *Let* PedCom, $\mathsf{SetCom}_{\mathsf{RSA}}$ *and* IntCom *be computationally binding commitments,* $\mathsf{CP}_{\mathsf{Coprime}}$, $\mathsf{CP}_{\mathsf{modEq}}$ *and* $\mathsf{CP}_{\mathsf{HashEq}}$ *be knowledge-sound NIZK arguments, and assume that the Strong RSA assumption hold, and that* H *is collision resistant. If* $d\mu+2 > \nu$, $\lambda_s+1 < \mu$, $\lambda_s < \log(N)/2$, $d = O(1)$ *is a small constant,* $2^{\mu-\nu} \in \mathsf{negl}(\lambda)$ *and* H *is modeled as a random oracle, then* $\mathsf{NonMemCP}_{\mathsf{RSA}}$ *is knowledge-sound with partial opening of the set commitments* C_U.

Theorem 5.3. *Let* PedCom, $\mathsf{SetCom}_{\mathsf{RSA}'}$ *and* IntCom *be computationally binding commitments,* $\mathsf{CP}_{\mathsf{Coprime}}$, $\mathsf{CP}_{\mathsf{modEq}}$ *and* $\mathsf{CP}_{\mathsf{Range}}$ *be knowledge-sound NIZK arguments, and assume that the Strong RSA assumption hold. If* $d\mu + 2 \leq \nu$, $\lambda_s + 1 < \mu$ *and* $\lambda_s < \log(N)/2$ *then* $\mathsf{NonMemCP}_{\mathsf{RSAPrm}}$ *is knowledge-sound with partial opening of the set commitments* c_P. *Furthermore, if* PedCom, $\mathsf{SetCom}_{\mathsf{RSA}'}$ *and* IntCom *are statistically hiding commitments, and* $\mathsf{CP}_{\mathsf{Coprime}}$, $\mathsf{CP}_{\mathsf{modEq}}$ *and* $\mathsf{CP}_{\mathsf{Range}}$ *be zero-knowledge, then* $\mathsf{NonMemCP}_{\mathsf{RSAPrm}}$ *is zero-knowledge.*

5.1 Proposed Instantiations of Protocol for R_{Coprime}

Below we propose an interactive ZK protocol for R_{Coprime}. As the relation indicates, we need to prove knowledge of (D, b) such that $D^e\mathsf{Acc}^b = G$, for a committed e. Proving opening of C_e to e is straightforward, so the main challenge is to prove the non-membership equation. For this the prover should send D and Acc^b to the verifier so that she can check that $D^e\mathsf{Acc}^b = G$ herself. Of course, there are two caveats. The first one is that D and Acc^b cannot be sent in the plain as we require zero-knowledge; we solve this by sending them in a hiding manner, i.e., $C_a = DH^{r_a}$ and $C_B = \mathsf{Acc}^b H^{\rho_B}$ for random values r_a, ρ_B. Consequently, the verification now should work with the hiding elements. Secondly, the verifier should be ensured that Acc^b is indeed an exponentiation of Acc with a known (to the prover) value b, otherwise soundness can be broken. More specifically we require extraction of b, ρ_B such that $C_B = \mathsf{Acc}^b H^{\rho_B}$. This is done using the partial opening of Acc to the set represented by prod, i.e., the protocol assumes that $\mathsf{Acc} = G^{\mathsf{prod}}$ is a common knowledge.

Below we present our protocol in full details.

1. Prover computes $C_a = DH^{r_a}, C_{r_a} = G^{r_a}H^{r'_a}, C_B = Acc^b H^{\rho_B}, C_{\rho_B} = G^{\rho_B}H^{\rho'_B}$ and sends to the verifier:
   $\underline{\mathcal{P} \rightarrow \mathcal{V}} : C_a, C_{r_a}, C_B, C_{\rho_B}$
2. Prover and Verifier perform a protocol for the relation:
   $R((\mathsf{Acc}, C_e, C_a, C_{r_a}, C_B, C_{\rho_B}), (e, b, r, r_a, r'_a, \rho_B, \rho'_B, \beta, \delta)) = 1$ iff

$$C_B = Acc^b H^{\rho_B} \wedge C_e = G^e H^r \wedge C_{r_a} = G^{r_a} H^{r'_a}$$

$$\wedge C_{\rho_B} = G^{\rho_B} H^{\rho'_B} \wedge C_a^e C_B = GH^{\beta} \wedge C_{r_a}^e C_{\rho_B} = G^{\beta} H^{\delta}$$

   Let λ_s be the size of the challenge space, λ_z be the statistical security parameter and μ the size of e.
   – Prover samples:

$$r_b, r_e \leftarrow\!\!\$ \left(-2^{\lambda_z+\lambda_s+\mu}, 2^{\lambda_z+\lambda_s+\mu}\right)$$

$$r_{\rho_B}, r_r, r_{r_a}, r_{r'_a}, r_{\rho'_B} \leftarrow\!\!\$ \left(-\lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s}, \lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s}\right)$$

$$r_\beta, r_\delta \leftarrow\!\!\$ \left(-\lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s+\mu}, \lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s+\mu}\right)$$

and computes:

$$\alpha_2 = \mathsf{Acc}^{r_b} H^{r_{\rho_B}}, \quad \alpha_3 = G^{r_e} H^{r_r}, \quad \alpha_4 = G^{r_{r_a}} H^{r_{r'_a}},$$
$$\alpha_5 = C_a^{r_e} H^{r_\beta}, \quad \alpha_6 = C_{r_a}^{r_e} G^{r_\beta} H^{r_\delta}, \quad \alpha_7 = G^{r_{\rho_B}} H^{r_{\rho'_B}}$$

$\underline{\mathcal{P} \to \mathcal{V}} : (\alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7)$

– Verifier samples the challenge $c \leftarrow \{0,1\}^{\lambda_s}$
$\underline{\mathcal{V} \to \mathcal{P}} : c$

– Prover computes the response:

$$s_b = r_b - cb, s_e = r_e - ce$$
$$s_{\rho_B} = r_{\rho_B} - c\rho_B, s_r = r_r - cr, s_{r_a} = r_{r_a} - cr_a, s_{r'_a} = r_{r'_a} - cr'_a, s_{\rho'_B} = r_{\rho'_B} - c\rho'_B$$
$$s_\beta = r_\beta + c(er_a + \rho_B), \quad s_\delta = r_\delta + c(er'_a + \rho'_B)$$

$\underline{\mathcal{P} \to \mathcal{V}} : (s_b, s_e, s_{\rho_B}, s_r, s_{r_a}, s_{r'_a}, s_{\rho'_B}, s_\beta, s_\delta)$

– Verifier checks if:

$$\alpha_2 \stackrel{?}{=} C_B^c \mathsf{Acc}^{s_b} H^{s_{\rho_B}}, \quad \alpha_3 \stackrel{?}{=} C_e^c G^{s_e} H^{s_r}, \quad \alpha_4 \stackrel{?}{=} C_{r_a}^c G^{s_{r_a}} H^{s_{r'_a}},$$
$$\alpha_5 \stackrel{?}{=} C_a^{s_e} H^{s_\beta} G^c C_B^{-c}, \quad \alpha_6 \stackrel{?}{=} C_{r_a}^{s_e} H^{s_\delta} G^{s_\beta} C_{\rho_B}^{-c}, \quad \alpha_7 \stackrel{?}{=} C_{\rho_B}^c G^{s_{\rho_B}} H^{s_{\rho'_B}},$$
$$s_e \stackrel{?}{\in} \left[-2^{\lambda_z+\lambda_s+\mu+1}, 2^{\lambda_z+\lambda_s+\mu+1}\right]$$

Coprime protocol

On common reference string $\mathsf{crs} = (\mathbb{Z}_N^*, G, H)$

$\mathsf{Prove}(\mathsf{crs}, (C_e, \mathsf{Acc}), (e, r, (D, b)))$:

1. samples $r_a, r_{a'}, \rho_B, \rho_{B'} \leftarrow\$ (-\lfloor N/4 \rfloor, \lfloor N/4 \rfloor)$ and computes $C_a = DH^{r_a}, C_{r_a} = G^{r_a} H^{r'_a}, C_B = Acc^b H^{\rho_B}, C_{\rho_B} = G^{\rho_B} H^{\rho'_B}$.
2. Computes the non-interactive version of the above protocol
   $r_b, r_e \leftarrow\$ \left(-2^{\lambda_z+\lambda_s+\mu}, 2^{\lambda_z+\lambda_s+\mu}\right)$
   $r_{\rho_B}, r_r, r_{r_a}, r_{r'_a}, r_{\rho'_B} \leftarrow\$ \left(-\lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s}, \lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s}\right)$
   $r_\beta, r_\delta \leftarrow\$ \left(-\lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s+\mu}, \lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s+\mu}\right)$
   $\alpha_2 = \mathsf{Acc}^{r_b} H^{r_{\rho_B}}, \alpha_3 = G^{r_e} H^{r_r}, \alpha_4 = G^{r_{r_a}} H^{r_{r'_a}}, \alpha_5 = C_a^{r_e} H^{r_\beta}, \alpha_6 = C_{r_a}^{r_e} G^{r_\beta} H^{r_\delta}, \alpha_7 = G^{r_{\rho_B}} H^{r_{\rho'_B}}$
   $c \leftarrow \mathsf{H}(\alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7, C_e, Acc)$
   $s_b = r_b - cb, s_e = r_e - ce, s_{\rho_B} = r_{\rho_B} - c\rho_B, s_r = r_r - cr, s_{r_a} = r_{r_a} - cr_a, s_{r'_a} = r_{r'_a} - cr_{r_a}, s_{\rho'_B} = r_{\rho'_B} - c\rho'_B, s_\beta = r_\beta + c(er_a + \rho_B), s_\delta = r_\delta + c(er'_a + \rho'_B)$

Returns $\pi \leftarrow (C_a, C_{r_a}, C_B, C_{\rho_B}, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7, s_b, s_e, s_{r_b}, s_{\rho_B}, s_r, s_{r_a}, s_{r'_a}, s_{\rho'_B}, s_\beta, s_\delta)$

$\mathsf{VerProof}(\mathsf{crs}, (C_e, \mathsf{Acc}), \pi)$: returns 1 iff $\alpha_2 = C_B^c \mathsf{Acc}^{s_b} H^{s_{\rho_B}} \wedge \alpha_3 = C_e^c G^{s_e} H^{s_r} \wedge \alpha_4 = C_{r_a}^c G^{s_{r_a}} H^{s_{r'_a}} \wedge \alpha_5 = C_a^{s_e} H^{s_\beta} G^c C_B^{-c} \wedge \alpha_6 = C_{r_a}^{s_e} H^{s_\delta} G^{s_\beta} C_{\rho_B}^{-c} \wedge \alpha_7 = C_{\rho_B}^c G^{s_{\rho_B}} H^{s_{\rho'_B}} \wedge \wedge s_e \in \left[-2^{\lambda+\lambda_s+\mu+1}, 2^{\lambda+\lambda_s+\mu+1}\right]$

Fig. 12

Correctness. Here we show the correctness of the protocol.

$$\begin{aligned}
\alpha_2 &= \mathsf{Acc}^{r_b} H^{r_{\rho_B}} = \mathsf{Acc}^{s_b+cb} H^{s_{\rho_B}+c\rho_B} = \mathsf{Acc}^{s_{r_b}} H^{s_{\rho_B}} (\mathsf{Acc}^b H^{\rho_B})^c \\
&= \mathsf{Acc}^{s_b} H^{s_{\rho_B}} C_B^c \\
\alpha_3 &= G^{r_e} H^{r_r} = G^{s_e+ce} H^{s_r+cr} = G^{s_e} H^{s_r} (G^e H^r)^c \\
&= G^{s_e} H^{s_r} C_e^c \\
\alpha_4 &= G^{r_{r_a}} H^{r_{r'_a}} = G^{s_{r_a}+cr_a} H^{s_{r'_a}+cr'_a} = G^{s_{r_a}} H^{s_{r'_a}} (G^{r_a} H^{r'_a})^c \\
&= G^{s_{r_a}} H^{s_{r'_a}} C_{r_a}^c \\
\alpha_5 &= C_a^{r_e} H^{r_\beta} = C_a^{s_e+ce} H^{s_\beta - c(er_a+\rho_B)} = C_a^{s_e} H^{s_\beta} (D^e H^{er_a})^c H^{-c(er_a+\rho_B)} \\
&= C_a^{s_e} H^{s_\beta} (D^e H^{-\rho_B})^c = C_a^{s_e} H^{s_\beta} (GAcc^{-b} H^{-\rho_B})^c = \\
&= C_a^{s_e} H^{s_\beta} G^c C_B^{-c} \\
\alpha_6 &= C_{r_a}^{r_e} G^{r_\beta} H^{r_\delta} = C_{r_a}^{s_e+ce} G^{s_\beta - c(er_a+\rho_B)} H^{s_\delta - c(er'_a+\rho'_B)} \\
&= C_{r_a}^{s_e} G^{s_\beta} H^{s_\delta} (G^{r_a} H^{r'_a})^{ce} G^{-c(er_a+\rho_B)} H^{-c(er'_a+\rho'_B)} = C_{r_a}^{s_e} G^{s_\beta} H^{s_\delta} G^{-c\rho_B} H^{-c\rho'_B} \\
&= C_{r_a}^{s_e} G^{s_\beta} H^{s_\delta} C_{\rho_B}^{-c} \\
\alpha_7 &= G^{r_{\rho_B}} H^{r_{\rho'_B}} = G^{s_{\rho_B}+c\rho_B} H^{s_{\rho'_B}+c\rho'_B} = G^{s_{\rho_B}} H^{s_{\rho'_B}} (G^{\rho_B} H^{\rho'_B})^c \\
&= G^{s_{\rho_B}} H^{s_{\rho'_B}} C_{\rho_B}^c
\end{aligned}$$

Security. Security of our scheme holds with the partial opening of Acc, i.e., when it is ensured outside the protocol that Acc is a valid commitment of the set. The proof is similar to the one of theorem 4.6. The main technical difference is in the extraction of the opening of C_B, because Acc is not a random generator sampled at the setup phase. However, from partial opening we know that it is $\mathsf{Acc} = G^{\mathsf{prod}}$ for a random generator G. This will allow us to state an alternative to lemma 4.2 to justify the extraction of the opening of C_B.

Theorem 5.4. *Let $\mathbb{Z}_N^*$ be an RSA group where strong-RSA assumption holds, then the above protocol is honest-verifier zero knowledge protocol and, also, if $\lambda_s + 1 < \mu$ and $\lambda_s < \log(N)/2$, is knowledge sound with partial opening of Acc for R_{Coprime}.*

Proof Zero-Knowledge can be proven with standard techniques, similar to the ones in the proof of theorem 4.6 and is therefore omitted.

For the knowledge soundness, let an adversary of the knowledge soundness $\mathcal{A}$ that is able to convince the verifier $\mathcal{V}$ with a probability at least ϵ. We will construct an extractor $\mathcal{E}$ that extracts the witness $(e, r, r_2, r_3, \beta, \delta)$. Using rewinding $\mathcal{E}$ gets two accepted transcripts

$$(C_a, C_{r_a}, C_B, C_{\rho_B}, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7, c, s_b, s_e, s_{\rho_B}, s_r, s_{r_a}, s_{r'_a}, s_{\rho'_B}, s_\beta, s_\delta)$$

$$(C_a, C_{r_a}, C_B, C_{\rho_B}, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7, c', s'_b, s'_e, s'_{\rho_B}, s'_r, s'_{r_a}, s'_{r'_a}, s'_{\rho'_B}, s'_\beta, s'_\delta)$$

on two different challenges c and c'. $\mathcal{E}$ aborts if it cannot get two such transcripts ($\mathsf{abort1}$).

We denote $\Delta c \coloneqq c' - c, \Delta s_b \coloneqq s_b - s'_b, \Delta s_e \coloneqq s_e - s'_e, \Delta s_{\rho_B} \coloneqq s_{\rho_B} - s'_{\rho_B}, \Delta s_r \coloneqq s_r - s'_r, \Delta s_{r_a} \coloneqq s_{r_a} - s'_{r_a}, \Delta s_{r'_a} \coloneqq s_{r'_a} - s'_{r'_a}, \Delta s_{\rho'_B} \coloneqq s_{\rho'_B} - s'_{\rho'_B}, \Delta s_\beta \coloneqq s_\beta - s'_\beta, \Delta s_\delta \coloneqq s_\delta - s'_\delta$ then

$$C_B^{\Delta c} = \mathsf{Acc}^{\Delta s_b} H^{\Delta s_{\rho_B}} \Rightarrow C_B = \pm \mathsf{Acc}^{\hat{b}} H^{\hat{\rho_B}} \tag{1}$$

$$C_e^{\Delta c} = G^{\Delta s_e} H^{\Delta s_r} \Rightarrow C_e = \pm G^{\hat{e}} H^{\hat{r}} \quad (2)$$

$$C_{r_a}^{\Delta c} = G^{\Delta s_{r_a}} H^{\Delta s_{r'_a}} \Rightarrow C_{r_a} = \pm G^{\hat{r_a}} H^{\hat{r'_a}} \quad (3)$$

$$1 = C_a^{\Delta s_e} H^{\Delta s_\beta} G^{-\Delta c} C_B^{\Delta c} \quad (4)$$

$$1 = C_{r_a}^{\Delta s_e} H^{\Delta s_\delta} G^{\Delta s_\beta} C_{\rho_B}^{\Delta c} \quad (5)$$

$$C_{\rho_B}^{\Delta c} = G^{\Delta s_{\rho_B}} H^{\Delta s_{\rho'_B}} \Rightarrow C_{\rho_B} = \pm G^{\hat{\rho_B}} H^{\hat{\rho'_B}} \quad (6)$$

define the (possibly rational) numbers $\hat{b} := \frac{\Delta s_b}{\Delta c}$, $\hat{e} := \frac{\Delta s_e}{\Delta c}$, $\hat{r} := \frac{\Delta s_r}{\Delta c}$, $\hat{r_a} := \frac{\Delta s_{r_a}}{\Delta c}$, $\hat{r'_a} := \frac{\Delta s_{r'_a}}{\Delta c}$, $\hat{\rho_B} := \frac{\Delta s_{\rho_B}}{\Delta c}$, $\hat{\rho'_B} := \frac{\Delta s_{\rho'_B}}{\Delta c}$.

$\mathcal{E}$ aborts in case Δc doesn't divide: Δs_e and Δs_r($\mathsf{abort}\,2a$), Δs_{r_a} and $\Delta s_{r'_a}$($\mathsf{abort}\,2b$), Δs_{ρ_B} and $\Delta s_{\rho'_B}$($\mathsf{abort}\,2c$). And finally, $\mathcal{E}$ aborts if Δc doesn't divide Δs_b and Δs_{ρ_B} ($\mathsf{abort}\,2d$). Therefore, after these aborts didn't happen we can infer the equivalent equalities on the right of equations 2,3,6 and 1.

If we replace equations 3 and 6 in equation 5 we get $1 = \left(\pm G^{\hat{r_a}} H^{\hat{r'_a}}\right)^{\Delta s_e} H^{\Delta s_\beta} G^{\Delta s_\beta} \left(\pm G^{\hat{\rho_B}} H^{\hat{\rho'_B}}\right)^{\Delta c}$ or $1 = (\pm 1)^{\Delta s_e} (\pm 1)^{\Delta c} G^{\hat{r_a}\Delta s_e + \hat{\rho_B}\Delta c + \Delta s_\beta} H^{\hat{r'_a}\Delta s_e + \hat{\rho'_B}\Delta c + \Delta s_\beta}$. Since $G, H, 1$ are quadratic residues then $(\pm 1)^{\Delta s_e}(\pm 1)^{\Delta c} = 1$, hence $1 = G^{\hat{r_a}\Delta s_e + \hat{\rho_B}\Delta c + \Delta s_\beta} H^{\hat{r'_a}\Delta s_e + \hat{\rho'_B}\Delta c + \Delta s_\beta}$. Then under the DLOG assumption $\hat{r_a}\Delta s_e + \hat{\rho_B}\Delta c + \Delta s_\beta = 0 = \hat{r'_a}\Delta s_e + \hat{\rho'_B}\Delta c + \Delta s_\beta$, which gives us that

$$\Delta s_\beta = -\hat{r_a}\Delta s_e - \hat{\rho_B}\Delta c \quad (7)$$

Finally, we replace equations 1 and 7 in equation 4 we get $1 = C_a^{\Delta s_e} H^{-\hat{r_a}\Delta s_e - \hat{\rho_B}\Delta c} G^{-\Delta c} \left(\pm \mathsf{Acc}^{\hat{b}} H^{\hat{\rho_B}}\right)^{\Delta c}$ or $1 = (\pm 1)^{\Delta c} C_a^{\Delta s_e} \mathsf{Acc}^{\hat{b}\Delta c} G^{-\Delta c} H^{-\hat{r_a}\Delta s_e}$ or $\left(\pm \mathsf{Acc}^{\hat{b}} G^{-1}\right)^{\Delta c} = \left(C_a^{-1} H^{r_a}\right)^{\Delta s_e}$. But as noted above Δc divides Δs_e so $\pm \mathsf{Acc}^{\hat{b}} G^{-1} = \pm \left(C_a^{-1} H^{r_a}\right)^{\hat{e}} \Rightarrow \mathsf{Acc}^{\hat{b}} G^{-1} = \pm \left(C_a^{-1} H^{\hat{r_a}}\right)^{\hat{e}} \Rightarrow \left(\frac{C_a}{H^{\hat{r_a}}}\right)^{\hat{e}} \mathsf{Acc}^{\hat{b}} = \pm G$. We discern two cases:

- $\underline{\left(\frac{C_a}{H^{\hat{r_a}}}\right)^{\hat{e}} \mathsf{Acc}^{\hat{b}} = +G}$: Then $\mathcal{E}$ sets $\tilde{D} \leftarrow \frac{C_a}{H^{\hat{r_a}}}$, $\tilde{e} \leftarrow \hat{e} := \frac{\Delta s_e}{\Delta c}$, $\tilde{r} \leftarrow \hat{r} := \frac{\Delta s_r}{\Delta c}$ and $\tilde{b} \leftarrow \hat{b} := \frac{\Delta s_b}{\Delta c}$
- $\underline{\left(\frac{C_a}{H^{\hat{r_a}}}\right)^{\hat{e}} \mathsf{Acc}^{\hat{b}} = -G}$: Then $\hat{e}$ should be odd otherwise if $\hat{e} = 2\rho$ then $G = -\left(\frac{C_a}{H^{\hat{r_a}}}\right)^{2\rho} \mathsf{Acc}^{\hat{b}}$ would be a non-quadratic residue. So $\mathcal{E}$ sets $\tilde{D} \leftarrow -\frac{C_a}{H^{\hat{r_a}}}$, $\tilde{e} \leftarrow \hat{e} := \frac{\Delta s_e}{\Delta c}$, $\tilde{r} \leftarrow \hat{r} := \frac{\Delta s_r}{\Delta c}$ and $\tilde{b} \leftarrow \hat{b} := \frac{\Delta s_b}{\Delta c}$. It is clear that $\tilde{D}^{\tilde{e}} \mathsf{Acc}^{\tilde{b}} = G$.

Finally the $\mathcal{E}$ outputs $(\tilde{e}, \tilde{r}, \tilde{D}, \tilde{b})$.

Now we show that the probability the extractor terminates with outputting a valid witness is $O(\epsilon)$. If the extractor does not abort then it clearly outputs a valid witness (under the factoring assumption). For the first abort, with a standard argument it can be shown that the extractor is able to extract two accepting transcripts with probability $O(\epsilon)$ (for the probabilistic analysis we refer to [DF02]). Thus $Pr[\mathsf{abort1}] = 1 - O(\epsilon)$. For the aborts $\mathsf{abort}\,2a$, $\mathsf{abort}\,2b$ and $\mathsf{abort}\,2c$ they happen with negligible probability ($\leq \frac{2}{1-2^{-\lambda_s+1}} Pr[\mathcal{B}$ solves $sRSA]$ each, for any PPT adversary $\mathcal{B}$) under the strong RSA assumption according to lemma 4.2. For $\mathsf{abort}\,2d$ we cannot directly use the same lemma as Acc is not a random generator that is part of the crs. However, with a similar argument and using partial extractability we show below that the probability for this abort is the same. Putting them together the probability of success of $\mathcal{E}$ is at least $O(\epsilon) - \frac{8}{1-2^{-\lambda_s+1}} Pr[\mathcal{B}$ solves $sRSA] = O(\epsilon) - \mathsf{negl}(\lambda_s)$.

For equation 1, we get from partial opening that $\mathsf{Acc} = G^{\mathsf{prod}_P}$, where $P := \{\mathsf{H}_{\mathsf{prime}}(u) \mid u \in U\}$, so

$$C_B^{\Delta c} = G^{\prod_{u \in U} \mathsf{H}_{\mathsf{prime}}(u) \cdot \Delta s_b} H^{\Delta s_{\rho_B}}$$

We use a similar to [DF02] argument to prove that Δc divides Δs_b and Δs_{ρ_B} under the strong RSA assumption, given that $\lambda_s + 1 < \mu$. Then

$$C_B = \pm \mathsf{Acc}^{\hat{b}} H^{\hat{\rho_B}} \tag{8}$$

Lemma 5.1. *Let $\lambda_s + 1 < \mu$ and $\lambda_s < \log(N)/2$ then Δc divides Δs_b and Δs_{ρ_B} under the strong RSA assumption.*

Proof An adversary against the strong RSA assumption receives $H \in \mathsf{QR}_N$ and does the following: sets $G = H^\tau$ for $\tau \leftarrow_\$ [0, 2^{\lambda_s} N^2]$ and sends (G, H) to the adversary $\mathcal{A}$ which outputs a proof π_{Coprime}. Then we rewind to get another successful proof π'_{Coprime} and we use the extractor as above to get $C_B^{\Delta c} = G^{\prod_{u \in U} \mathsf{H}_{\mathsf{prime}}(u) \cdot \Delta s_b} H^{\Delta s_{\rho_B}}$ or

$$C_B^{\Delta c} = H^{\tau \prod_{u \in U} \mathsf{H}_{\mathsf{prime}}(u) \cdot \Delta s_b + \Delta s_{\rho_B}}$$

We can exclude the case that Δc divides $\prod_{u \in U} \mathsf{H}_{\mathsf{prime}}(u)$, since Δc is smaller than the domain of the hash function $\mathsf{H}_{\mathsf{prime}}$, i.e. $\Delta c < \mathsf{H}_{\mathsf{prime}}(u)$ for each $u \in U$, which comes from $\lambda_s + 1 < \mu$. Assume that $\Delta c \nmid \Delta s_b \vee \Delta c \nmid \Delta s_{\rho_B}$. we discern two cases:

- Δc doesn't divide $\tau \prod_{u \in U} \mathsf{H}_{\mathsf{prime}}(u) \cdot \Delta s_b + \Delta s_{\rho_B}$: then $\gcd(\Delta c, \tau \prod_{u \in U} \mathsf{H}_{\mathsf{prime}}(u) \cdot \Delta s_b + \Delta s_{\rho_B}) = g$ and there are χ, ψ such that $\chi \cdot \Delta c + \psi \cdot \left(\tau \prod_{u \in U} \mathsf{H}_{\mathsf{prime}}(u) \cdot \Delta s_b + \Delta s_{\rho_B}\right) = g$. Thus

  $$H^g = H^{\chi \cdot \Delta c + \psi \cdot \left(\tau \prod_{u \in U} \mathsf{H}_{\mathsf{prime}}(u) \cdot \Delta s_b + \Delta s_{\rho_B}\right)} = H^{\chi \Delta c} \cdot C_B^{\psi \Delta c} = \left(H^\chi \cdot C_B^\psi\right)^{\Delta c}$$

  Since g divides Δc we get $H = \pm \left(H^\chi \cdot C_B^\psi\right)^{\frac{\Delta c}{g}}$. However H is a quadratic residue (thus C_B is so), meaning that $H = \left(H^\chi \cdot C_B^\psi\right)^{\frac{\Delta c}{g}}$, thus $(H^\chi \cdot C_B^\psi, \frac{\Delta c}{g})$ is a solution to the strong RSA problem.
- Δc divides $\tau \prod_{u \in U} \mathsf{H}_{\mathsf{prime}}(u) \cdot \Delta s_b + \Delta s_{\rho_B}$: let q^ℓ be the maximal q-power that divides Δc (i.e. q^ℓ is a factor of Δ) and doesn't divide at least one of Δs_b and Δs_{ρ_B}, where q is prime. Such a q^ℓ should exist otherwise Δc would divide both Δs_b and Δs_{ρ_B}, which we assumed it doesn't. Notice that if q^ℓ divided Δs_b then it would also divide Δs_{ρ_B}, as q^ℓ divides $\tau \prod_{u \in U} \mathsf{H}_{\mathsf{prime}}(u) \cdot \Delta s_b + \Delta s_{\rho_B}$ (from assumption), so $q^\ell \nmid \Delta s_b$.

  $$q^\ell \mid \left(\tau \prod_{u \in U} \mathsf{H}_{\mathsf{prime}}(u) \cdot \Delta s_b + \Delta s_{\rho_B}\right) \Rightarrow \tau \prod_{u \in U} \mathsf{H}_{\mathsf{prime}}(u) \cdot \Delta s_b + \Delta s_{\rho_B} = 0 \pmod{q^\ell}$$

  We can write $\tau := \tau_1 + \tau_2 \, \mathsf{ord}(H)$. Notice that τ_2 is information theoretically hidden to the adversary and thus is uniformly random in $[0, 2^{\lambda_s} N^2 / \mathsf{ord}(H)] \supset [0, 2^{\lambda_s} N]$ in its view.

  $$\Rightarrow \tau_1 \prod_{u \in U} \mathsf{H}_{\mathsf{prime}}(u) \cdot \Delta s_b + \tau_2 \mathsf{ord}(H) \prod_{u \in U} \mathsf{H}_{\mathsf{prime}}(u) \cdot \Delta s_b + \Delta s_{\rho_B} = 0 \pmod{q^\ell}$$

$$\Rightarrow \tau_2 \cdot \Delta s_b = \left(-\tau_1 \prod_{u \in U} \mathsf{H}_{\mathsf{prime}}(u) \cdot \Delta s_b - \Delta s_{\rho_B}\right) \cdot \left(\prod_{u \in U} \mathsf{H}_{\mathsf{prime}}(u)\right)^{-1} \cdot (\mathsf{ord}(H))^{-1} \pmod{q^\ell}$$

To see that $\prod_{u \in U} \mathsf{H}_{\mathsf{prime}}(u)$ has an inverse modulo q^ℓ note that since $\Delta c < \mathsf{H}_{\mathsf{prime}}(u)$ implies $q^\ell < \mathsf{H}_{\mathsf{prime}}(u)$, so $\gcd(\prod_{u \in U} \mathsf{H}_{\mathsf{prime}}(u), q^\ell) = 1$. For the inverse of $\mathsf{ord}(H)$ note that $H \in \mathsf{QR}_N$ so $\mathsf{ord}(H) \in \{q_1, q_2, q_1 q_2\}$, where $N = (2q_1 + 1)(2q_2 + 1)$ is the RSA modulus. Then from $\lambda_s < \log(N)/2$ we get $\Delta c < q_1, q_2$ and thus $\gcd(\mathsf{ord}(H), q^\ell) = 1$.
As noted above, τ_2 is uniformly random in a superset of $[0, 2^{\lambda_s} N]$. But $q^\ell < \Delta c < N$, so $2^{\lambda_s} N$ is at least 2^{λ_s} larger than q^ℓ. Thus τ_2 is statistically close to uniform in $\{0, 1, \ldots, q^\ell - 1\}$ (with $2^{-\lambda_s}$ error), $Pr_{\tau_2}[\tau_2 = C \pmod{q^\ell}] \approx \frac{1}{q^\ell}$. Furthermore, for any Δs_b, $Pr_{\tau_2}[\tau_2 \cdot \Delta s_b = C \pmod{q^\ell}] \approx \frac{1}{q^\ell} \cdot \gcd(q^\ell, \Delta s_b) \leq \frac{1}{q^\ell} \cdot q^{\ell-1}$ (since q^ℓ doesn't divide Δs_b). This is because for variable τ_2, the equation $\tau_2 \Delta s_b = C \pmod{q^\ell}$ has $\gcd(q^\ell, \Delta s_b)$ solutions.
In conclusion, the probability that the above equation holds is at most $\frac{1}{q} + 2^{-\lambda_s} \leq \frac{1}{2} + 2^{-\lambda_s}$.

To summarize we showed that the probability to fall in the second case is at most $\frac{1}{2} + 2^{-\lambda_s}$. So with probability to fall in the first case, and thus solve the strong RSA problem, is at least $\frac{1}{2} - 2^{-\lambda_s}$. □

By a simple argument identical to the one of section 4.4, we can also conclude about the range of the extracted $\tilde{e}$: $s_e \stackrel{?}{\in} \left[-2^{\lambda_z + \lambda_s + \mu + 1}, 2^{\lambda_z + \lambda_s + \mu + 1}\right]$ implies $-2^{\lambda_z + \lambda_s + \mu + 2} \leq \hat{e} \leq 2^{\lambda_z + \lambda_s + \mu + 2}$. □

6 A CP-SNARK for Set Membership in Bilinear Groups

In this section we propose another CP-SNARK, called $\mathsf{MemCP}_{\mathsf{VC}}$, for the set membership relation that works in bilinear groups. Unlike the schemes of Section 4, the CP-SNARK given in this section does not have short parameters; specifically it has a CRS linear in the size of the sets to be committed. On the other hand, it enjoys other features that are not satisfied by our previous schemes (nor by other schemes in the literature): first, it works solely in Bilinear Groups without having to deal with RSA groups; second, it allows to commit the set in an hiding manner and, for the sake of soundness, does not need to be opened by the adversary. This is possible thanks to the fact that the set is committed in a way that (under a knowledge assumption) guarantees that the prover knows the set.

More in detail, $\mathsf{MemCP}_{\mathsf{VC}}$ is a CP-SNARK for set membership where set elements are elements from the large field $\mathbb{F} = \mathbb{Z}_q$ where q is the order of bilinear groups. So $\mathcal{D}_{\mathsf{elm}} = \mathbb{F}$. In terms of set it supports all the subsets of $2^{\mathcal{D}_{\mathsf{elm}}}$ of cardinality bounded by n, $\mathcal{D}_{\mathsf{set}} = \{U \in 2^{\mathcal{D}_{\mathsf{elm}}} : \#U \leq n\}$, which we denote by $\mathcal{S}_n$, # symbol denotes the cardinality of a set. So U has elements in $\mathbb{F}$ and is a subset of $\mathcal{S}_n$.

6.1 Preliminaries and Building Blocks

Bilinear Groups. A *bilinear group generator* $\mathcal{BG}(1^\lambda)$ outputs $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are additive groups of prime order q, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable, non-degenerate, bilinear map. For ease of exposition we present our results with Type-1 groups where we assume that $\mathbb{G}_1 = \mathbb{G}_2$. Our results are under the $(\ell + 1)d$-Strong Diffie Hellman and

the (d, ℓ)-Extended Power Knowledge of Exponent assumptions, for which we refer the reader to [ZGK+17].

A Polynomial-Pedersen Type-Based Commitment Scheme. First we present $\mathsf{PolyCom}$, a type-based commitment scheme which was introduced in [CFQ19] extracted from the verifiable polynomial delegation scheme of [ZGK+17]. The scheme has two types: one for ℓ-variate polynomials $f : \mathbb{F}^\ell \to \mathbb{F}$ over $\mathbb{F}$ of variable degree at most d, and one which is a standard Pedersen commitment for field elements. Let $\mathcal{W}_{\ell,d}$ be the set of all multisets of $\{1, \ldots, \ell\}$ where the cardinality of each element is at most d. The scheme is described in figure 13.

$\mathsf{Setup}(1^\lambda, \ell, d)$: samples a bilinear group of order q, $\mathsf{bp} := (q, g, \mathbb{G}_1, \mathbb{G}_T, e) \leftarrow \mathsf{BilGen}(1^\lambda)$, samples $\alpha, \beta, s_1, \ldots, s_\ell \leftarrow \mathbb{F}$. Computes $\mathsf{prk} \leftarrow \left\{ g^{\prod_{i \in W} s_i} : W \in \mathcal{W}_{\ell,d} \right\}$ and $\mathsf{prk}^\alpha \leftarrow \left\{ g^{\alpha \cdot \prod_{i \in W} s_i} : W \in \mathcal{W}_{\ell,d} \right\}$. Finally samples an $s_{\ell+1} \leftarrow\!\$\, \mathbb{F}$ and computes $h \leftarrow g^{s_{\ell+1}}$ and h^α.
Return $\mathsf{ck} \leftarrow (\mathsf{bp}, \mathsf{prk}, \mathsf{prk}^\alpha, g^\alpha, g^\beta, h, h^\alpha, h^\beta)$

$\mathsf{Commit}(\mathsf{ck}, \mathsf{t}_{\mathbb{F}[\boldsymbol{s}]}, f)$: parses $\mathsf{ck} := (\mathsf{bp}, \mathsf{prk}, \mathsf{prk}^\alpha, g^\alpha, g^\beta, h, h^\alpha, h^\beta)$ and uses $\mathsf{prk} := \left\{ g^{\prod_{i \in W} s_i} : W \in \mathcal{W}_{\ell,d} \right\}$ and $\mathsf{prk}^\alpha := \left\{ g^{\alpha \cdot \prod_{i \in W} s_i} : W \in \mathcal{W}_{\ell,d} \right\}$ to compute $g^{f(\boldsymbol{s})}$ and $g^{\alpha \cdot f(\boldsymbol{s})}$ respectively. Then samples a random $r_f \leftarrow\!\$\, \mathbb{F}$ and computes $c_{f,1} \leftarrow g^{f(\boldsymbol{s})} h^{r_f}$ and $c_{f,2} \leftarrow g^{\alpha \cdot f(\boldsymbol{s})} (h^\alpha)^{r_f}$
Return $(c, o) \leftarrow ((c_{f,1}, c_{f,2}), r_f)$

$\mathsf{Commit}(\mathsf{ck}, \mathsf{t}_q, y)$: parses $\mathsf{ck} := (\mathsf{bp}, \mathsf{prk}, \mathsf{prk}^\alpha, g^\alpha, g^\beta, h, h^\alpha, h^\beta)$ and samples $r \leftarrow\!\$\, \mathbb{F}$. Computes $c_{y,1} \leftarrow g^y h^r$ and $c_{y,2} \leftarrow (g^\beta)^y (h^\beta)^r$ and return $(c, o) := ((c_{y,1}, c_{y,2}), r)$.

$\mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_{\mathbb{F}[\boldsymbol{s}]}, c, f, o)$: parses $\mathsf{ck} := (\mathsf{bp}, \mathsf{prk}, \mathsf{prk}^\alpha, g^\alpha, g^\beta, h, h^\alpha, h^\beta)$ and uses $\mathsf{prk} := \left\{ g^{\prod_{i \in W} s_i} : W \in \mathcal{W}_{\ell,d} \right\}$ to compute $g^{f(\boldsymbol{s})}$. Parses $c := (c_{f,1}, c_{f,2})$. Output 1 iff $c_{f,1} = g^{f(\boldsymbol{s})} h^o \wedge e(c_{f,1}, g^\alpha) = e(c_{f,2}, g)$.

$\mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_q, c, y, o)$: parses $\mathsf{ck} := (\mathsf{bp}, \mathsf{prk}, \mathsf{prk}^\alpha, g^\alpha, g^\beta, h, h^\alpha, h^\beta)$. Parses $c := (c_{y,1}, c_{y,2})$ Output 1 iff $c_{y,1} = g^y h^r \wedge e(c_{y,1}, g^\beta) = e(c_{y,2}, g)$.

Fig. 13: $\mathsf{PolyCom}$ Commitment Scheme

Theorem 6.1. *Under the $(\ell+1)d$-Strong Diffie Hellman and the (d, ℓ)-Extended Power Knowledge of Exponent assumptions* $\mathsf{PolyCom}$ *is an extractable trapdoor commitment scheme.*

For the proof we refer to [CFQ19, ZGK+17].

Input-Hiding CP-SNARK for Polynomial Evaluation The main building block of our main protocol is a CP-SNARK $\mathsf{CP}_{\mathsf{PolyEval}}$ for the type-based commitment $\mathsf{PolyCom}$. Loosely speaking the idea is to commit to the input $\boldsymbol{t}$ and the output y of a polynomial (with a Pedersen commitment), further commit to the polynomial f itself (with a polynomial commitment) and then prove that the opening of the committed polynomial evaluated on the opening of the committed input gives the committed output. The relation of the protocol is $R_{\mathsf{PolyEval}}((t_k)_{k \in [\ell]}, f, y)) = 1$ iff $f(t_1, \ldots, t_\ell) = y$:

$\mathbf{R} = (\mathsf{ck}, R_{\mathsf{PolyEval}})$ where $\mathbf{R}$ is over

$$(\boldsymbol{x}, \boldsymbol{w}) = ((x, c), (u, o, \omega)) = \left((\varnothing, (c_y, (c_{t_k})_{k \in [\ell]}, c_f)) \,,\, ((y, (t_k)_{k \in [\ell]}, f), (r_y, (r_{t_k})_{k \in [\ell]}, r_f), \varnothing) \right)$$

We will present a CP-SNARK for this relation, $\mathsf{CP}_{\mathsf{PolyEval}}$, in section 6.3. $\mathsf{CP}_{\mathsf{PolyEval}}$ is based on a similar protocol for polynomial evaluation given in [CFQ19] which was in turn based on the

verifiable polynomial delegation scheme of zk-vSQL [ZGK+17]. In those protocols, however, the input $\mathbf{t}$ is public whereas in ours we can keep it private and committed.

Range Proof CP-NIZK. We make use of $\mathsf{CP}_{\mathsf{Range}}$, a CP-NIZK for the following relation on PedCom commitments c and two given integers $A < B$:

$$R_{\mathsf{Range}}\left((c_e, A, B), (e, r_q)\right) = 1 \text{ iff } c = g^e h^{r_q} \wedge A < e_q < B$$

$\mathsf{CP}_{\mathsf{Range}}$ can have various instantiations such as Bulletproofs [BBB+18].

Multilinear Extensions of vectors. Let $\mathbb{F}$ be a field and $n = 2^\ell$. The multilinear extension of a vector $\boldsymbol{a} = (a_0, \ldots, a_{n-1})$ in $\mathbb{F}$ is a polynomial $f_{\boldsymbol{a}} : \mathbb{F}^\ell \to \mathbb{F}$ with variables $x_1, \ldots, x_\ell$ defined as

$$f_{\boldsymbol{a}}(x_1, \ldots, x_\ell) = \sum_{i=0}^{n-1} a_i \cdot \prod_{k=1}^{\ell} \mathsf{select}_{i_k}(x_k)$$

where $i_\ell i_{\ell-1} \ldots i_2 i_1$ is the bit representation of i and $\mathsf{select}_{i_k}(x_k) = \begin{cases} x_k, & \text{if } i_k = 1 \\ 1 - x_k, & \text{if } i_k = 0 \end{cases}$

A property of Multilinear extension of $\boldsymbol{a}$ is that $f_{\boldsymbol{a}}(i_1, \ldots, i_\ell) = a_i$ for each $i \in [n]$.

The type-based commitment scheme of $\mathsf{MemCP}_{\mathsf{VC}}$. We define the type-based commitment $\mathsf{C}_{EdraxPed}$ for our CP-SNARK $\mathsf{MemCP}_{\mathsf{VC}}$. We recall we need a commitment that allows one to commit to both elements and sets. We build this based on an hiding variant of EDRAX Vector Commitment [CPZ18], which in turn relies on a polynomial commitment. Therefore, we use a special case of $\mathsf{PolyCom}$ for polynomials of maximum variable degree $d = 1$. Let $\ell := \lceil \log(n) \rceil$ and $2^{[\ell]}$ be the powerset of $[\ell] = \{1, ..., \ell\}$ then $\mathcal{W}_{\ell,1} = 2^{[\ell]}$. Furthermore, for any $n' \leq n$ let $L : \mathcal{S}_{n'} \to \mathbb{F}^{n'}$ be a function that maps a set of cardinality n' to its corresponding vector according to an ordering. The description of the scheme can be found in figure 14. Essentially the idea is to take the set, fix some ordering so that we can encode it with a vector, and then commit to such vector using the vector commitment of [CPZ18], which in turn commits to a vector by committing to its multilinear extension polynomial.

$\mathsf{Setup}(1^\lambda, \ell)$: executes $\mathsf{ck} \leftarrow \mathsf{PolyCom.Setup}(1^\lambda, \ell, 1)$
$\mathsf{Commit}(\mathsf{ck}, \mathsf{t}_U, U)$: computes $\vec{U} \leftarrow L(U)$ and then the corresponding multilinear extension of $\vec{U}$, $f_{\vec{U}}$. Returns $(c, o) \leftarrow \mathsf{PolyCom.Commit}(\mathsf{ck}, \mathsf{t}_{\mathbb{F}[\boldsymbol{s}]}, f_{\vec{S}})$.
$\mathsf{Commit}(\mathsf{ck}, \mathsf{t}_q, y)$: returns $(c, o) \leftarrow \mathsf{PolyCom.Commit}(\mathsf{ck}, \mathsf{t}_q, y)$
$\mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_U, c, U, o)$: computes $\vec{U} \leftarrow L(U)$ and then the corresponding multilinear extension of $\vec{U}$, $f_{\vec{U}}$. Outputs $\mathsf{PolyCom.VerCommit}(\mathsf{ck}, \mathsf{t}_{\mathbb{F}[\boldsymbol{s}]}, c, f_{\vec{S}}, o)$.
$\mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_q, c, y, o)$: outputs $\mathsf{PolyCom.VerCommit}(\mathsf{ck}, \mathsf{t}_q, c, y, o)$.

Fig. 14: $\mathsf{C}_{EdraxPed}$

6.2 CP-SNARK for Set membership using EDRAX Vector Commitment

Here we present a CP-SNARK for set membership that uses a Vector Commitment - an EDRAX [CPZ18] variant - to commit to a set. The idea is to transform a set to a vector (using for example

lexicographical order) and then commit to the vector with a vector commitment. Then the set membership is proven with a zero knowledge proof of opening of the corresponding position of the vector. However to preserve zero knowledge we additionally need to hide the position of the element. For this we construct a zero knowledge proof of knowledge of an opening of a position that does not give out the position. Finally, since the position is hidden we additionally need to ensure that the prover is not cheating by providing a proof for a position that exceeds the length of the vector. For this we, also, need a proof of range for the position, i.e. that $i < n$.

In this section the domain of the elements is a field, $\mathcal{D}_{\mathsf{elm}} := \mathbb{F}$, and the domain of the set is all the subsets of $2^{\mathcal{D}_{\mathsf{elm}}}$ of cardinality bounded by n, $\mathcal{D}_{\mathsf{set}} = \{U \in 2^{\mathcal{D}_{\mathsf{elm}}} : \#U \leq n\}$, which we denote by $\mathcal{S}_n$ (the # symbol denotes the cardinality of a set). So U has elements in $\mathbb{F}$ and is a subset of $\mathcal{S}_n$.

The type-based commitment of our scheme is $\mathsf{C}_{EdraxPed}$ (fig. 14) that is presented in the previous section, and the relation is

$\mathbf{R} = (\mathsf{ck}, R_{\mathsf{VCmem}})$ where $\mathbf{R}$ is over

$$(\boldsymbol{x}, \boldsymbol{w}) = ((x, c), (u, o, \omega)) = \big(\big(\#U, (c_y, (c_{i_k})_{k\in[\ell]}, c_U)\big), \big((y, (i_k)_{k\in[\ell]}, U), (r_y, (r_{i_k})_{k\in[\ell]}, r_U), \varnothing\big)\big)$$

$R_{\mathsf{VCmem}}(\#U, \big(y, (i_k)_{k\in[\ell]}, U\big)) = 1$ iff $y = L(U)[i] \wedge i < \#U \wedge i = \sum_{k=1}^{\ell} i_k 2^{k-1}$

Note that in the above the prover should normally give exactly $\ell = \lceil \log(\#U) \rceil$ commitments. In case $\ell < \lceil \log(\#U) \rceil$ the position is not fully hiding since it is implicit that $i < 2^{\ell-1}$ so the verifier gets a partial information about the position.

For this we will compose a CP-SNARK $\mathsf{CP}_{\mathsf{PolyEval}}$ and a CP-NIZK $\mathsf{CP}_{\mathsf{Range}}$ for the relations $R_{\mathsf{PolyEval}}((i_k)_{k\in[\ell]}, f, y)) = 1$ iff $f(i_1, \ldots, i_\ell) = y$ and $R_{\mathsf{Range}}(T, (i_k)_{k\in[\ell]}) = 1$ iff $i < T$ respectively and the commitment scheme $\mathsf{C}_{EdraxPed}$. So $\mathsf{CP}_{\mathsf{VCmem}}$ is a conjuction of the former, where the common commitments are $(c_{i_k})_{k\in[\ell]}$.

CP-SNARK for R_{VCmem}

$\mathsf{KeyGen}(\mathsf{ck}, R_{\mathsf{VCmem}})$: computes $(\mathsf{ek}_1, \mathsf{vk}_1) \leftarrow \mathsf{CP}_{\mathsf{PolyEval}}.\mathsf{KeyGen}(\mathsf{ck}, R_{\mathsf{PolyEval}})$ and $(\mathsf{ek}_2, \mathsf{vk}_2) \leftarrow \mathsf{CP}_{\mathsf{Range}}.\mathsf{KeyGen}(\mathsf{ck}, R_{\mathsf{Range}})$
Return $(\mathsf{ek}, \mathsf{vk}) \leftarrow ((\mathsf{ek}_1, \mathsf{ek}_2), (\mathsf{vk}_1, \mathsf{vk}_2))$

$\mathsf{Prove}(\mathsf{ek}, \#U, (c_y, (c_{i_k})_{k\in[\ell]}, c_U), (y, (i_k)_{k\in[\ell]}, U), (r_y, (r_{i_k})_{k\in[\ell]}, r_U), \varnothing)$: parses $\mathsf{ek} := (\mathsf{ek}_1, \mathsf{ek}_2)$ and computes $\pi_1 \leftarrow \mathsf{CP}_{\mathsf{PolyEval}}.\mathsf{Prove}\big(\mathsf{ek}_1, \varnothing, (c_y, (c_{i_k})_{k\in[\ell]}), c_U), (y, (i_k)_{k\in[\ell]}, U), (r_y, (r_{i_k})_{k\in[\ell]}, r_U), \varnothing\big)$. Parses $\mathsf{ck} := (\mathsf{bp}, \mathsf{prk}, \mathsf{prk}^\alpha, g^\alpha, g^\beta, h, h^\alpha, h^\beta)$ and further $\mathsf{bp} := (q, g, \mathbb{G}_1, \mathbb{G}_T, e)$ to get (g, h), then computes $i \leftarrow \sum_{k=1}^{\ell} i_k 2^{k-1}$ and $r_i \leftarrow \sum_{k=1}^{\ell} r_{i_k} 2^{k-1}$ and the corresponding commitment $c_i \leftarrow g^i h^{r_i}$. Notice that c_i is a commitment to i with $o = r_i$. Computes $\pi_2 \leftarrow \mathsf{CP}_{\mathsf{Range}}.\mathsf{Prove}\,(\mathsf{ek}_1, (1, \#U), c_i, i, r_i, \varnothing)$
Return $\pi = (\pi_1, \pi_2)$

$\mathsf{VerProof}(\mathsf{vk}, \#U, (c_y, (c_{i_k})_{k\in[\ell]}, c_U), \pi)$: parses $\mathsf{vk} := (\mathsf{vk}_1, \mathsf{vk}_2)$ and $\pi := (\pi_1, \pi_2)$. Then computes homomorhically $c_{i,1} \leftarrow \prod_{k=1}^{\ell} (c_{i_k,1})^{2^{k-1}}$ and $c_{i,2} \leftarrow \prod_{k=1}^{\ell} (c_{i_k,2})^{2^{k-1}}$.
Return $b \leftarrow \mathsf{CP}_{\mathsf{PolyEval}}.\mathsf{VerProof}(\mathsf{vk}_1, \varnothing, (c_y, (c_{i_k})_{k\in[\ell]}), c_U), \pi_1) \wedge \mathsf{CP}_{\mathsf{Range}}.\mathsf{VerProof}(\mathsf{vk}_2, (1, \#U), c_i, \varnothing)$

Fig. 15: $\mathsf{MemCP}_{\mathsf{VC}}$

Theorem 6.2. *Let* $\mathsf{CP}_{\mathsf{PolyEval}}$ *and* $\mathsf{CP}_{\mathsf{Range}}$ *be zero knowledge CP-SNARKs for the relations* R_{PolyEval} *and* R_{Range} *respectively under the commitment scheme* PolyCom *then the above scheme is a zero*

knowledge CP-SNARK for the relation R_{VCmem} and the commitment scheme $\mathsf{C}_{EdraxPed}$. Further it is a CP-SNARK for R_{mem} under the same commitment scheme.

Proof Zero Knowledge comes directly from the zero knowledge of $\mathsf{CP}_{\mathsf{PolyEval}}$ and $\mathsf{CP}_{\mathsf{PolyEval}}$.

For Knowledge Soundness, let an adversary $\mathcal{A}(\mathbf{R}, \mathsf{crs}, \mathsf{aux}_R, \mathsf{aux}_Z)$ outputting $(x, c) := \big(\#U, (c_y, (c_{i_k})_{k\in[\ell]}, c_U)\big)$ and π such that $\mathsf{VerProof}(\mathsf{vk}, \#U, (c_y, (c_{i_k})_{k\in[\ell]}, c_U), \pi) = 1$. We will construct an extractor $\mathcal{E}$ that on input $(\mathbf{R}, \mathsf{crs}, \mathsf{aux}_R, \mathsf{aux}_Z)$ outputs a valid witness $w := \big((y, (i_k)_{k\in[\ell]}, U), (r_y, (r_{i_k})_{k\in[\ell]}, r_U), \varnothing\big)$.

$\mathcal{E}$ uses the extractors of $\mathcal{E}_{\mathsf{PolyEval}}$, $\mathcal{E}_{\mathsf{Range}}$ of $\mathsf{CP}_{\mathsf{PolyEval}}$ and $\mathsf{CP}_{\mathsf{Range}}$. $\mathcal{E}_{\mathsf{PolyEval}}$ outputs $(y, (i_k)_{k\in[\ell]}, f)$, $(r_y, (r_{i_k})_{k\in[\ell]}, r_f)$ such that $f(i_1, \ldots, i_\ell) = y \wedge \mathsf{PolyCom.VerCommit}(\mathsf{ck}, \mathsf{t}_{\mathbb{F}[\boldsymbol{s}]}, c_U, f, r_f) = 1 \wedge$ $\mathsf{PolyCom.VerCommit}(\mathsf{ck}, \mathsf{t}_q, c_y, y, r_y) = 1 \bigwedge_{k=1}^{\ell} \mathsf{PolyCom.VerCommit}(\mathsf{ck}, \mathsf{t}_q, c_{i_k}, i_k, r_{i_k}) = 1$. Further, from the Extended Power Knowledge of Exponent assumption we know that f is an ℓ-variate polynomial of maximum variable degree 1. Therefore it corresponds to a multilinear extension of a unique vector $\vec{U}$, which is efficiently computable. The extractor computes the vector $\vec{U}$ from f and the corresponding set U. It is clear that, since f is the multilinear extension of the U and $\mathsf{PolyCom.VerCommit}(\mathsf{ck}, \mathsf{t}_{\mathbb{F}[\boldsymbol{s}]}, c_U, f, r_f) = 1$, $\mathsf{C}_{EdraxPed}.\mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_U, c_U, U, r_f) = 1$. $\mathsf{C}_{EdraxPed}.\mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_q, c_y, y, r_y) = 1 \bigwedge_{k=1}^{\ell} \mathsf{C}_{EdraxPed}.\mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_q, c_{i_k}, i_k, r_{i_k}) = 1$ is straightforward from the definition of the $\mathsf{C}_{EdraxPed}$ commitment scheme for field elements type.

$\mathcal{E}$ uses the extractor of the commitment scheme $\mathsf{PolyCom}$, $\mathcal{E}_{\mathsf{PolyCom}}$, that outputs for each $k = 1, \ldots, \ell$ i_k, r_{i_k} such that $c_{i_k,1} = g^{i_k} h^{r_{i_k}} \wedge e(c_{i_k,1}, g^\beta) = e(c_{i_k,2}, g)$ or $\mathsf{C}_{EdraxPed}.\mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_q, c_{i_k}, r_{i_k}) = 1$. $\mathcal{E}_{\mathsf{Range}}$ outputs (i, r_i) such that $i < \#U \wedge \mathsf{PolyCom.VerCommit}(\mathsf{ck}, \mathsf{t}_q, c_i, i, r_i) = 1$ which means that $c_{i,1} = g^i h^{r_i}$. Since the proof π is verified then $c_{i,1} = \prod_{k=1}^{\ell} (c_{i_k,1})^{2^{k-1}}$ or $g^i h^{r_i} = g^{\sum_{k=1}^{\ell} i_k 2^{k-1}} h^{\sum_{k=1}^{\ell} r_{i_k} 2^{k-1}}$. From the binding property of the Pedersen commitment we get that $i = \sum_{k=1}^{\ell} i_k 2^{k-1}$ and $r_i = \sum_{k=1}^{\ell} r_{i_k} 2^{k-1}$.

Putting them together the extractor outputs $\big((y, (i_k)_{k\in[\ell]}, U), (r_y, (r_{i_k})_{k\in[\ell]}, r_f), \varnothing\big)$ such that $\mathsf{C}_{EdraxPed}.\mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_q, c_y, r_y) = 1 \bigwedge_{i=1}^{\ell} \mathsf{C}_{EdraxPed}.\mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_q, c_{i_k}, r_{i_k}) = 1 \wedge$ $\mathsf{C}_{EdraxPed}.\mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_U, c_f, U, r_f) = 1$ and further $y = L(U)[i] \wedge i < \#U \wedge i = \sum_{k=1}^{\ell} i_k 2^{k-1}$. It is straightforward that $y = L(U)[i] \wedge i < \#U$ means that $y \in U$ which leads to $R_{\mathsf{mem}}(y, U) = 1$. □

6.3 Input-hiding CP-SNARKs for Polynomial Evaluation

Here, we present an instantiation of a a zero knowledge CP-SNARK for the relation R_{PolyEval} presented in section 6.1.

To give an intuition of the protocol we recall that zk-vSQL uses lemma 6.1 to prove the correct evaluation of the polynomial, that we recall below.

Lemma 6.1 ([PST13]). *Let $f : \mathbb{F}^\ell \to \mathbb{F}$ be a polynomial of variable degree d. For all $\boldsymbol{t} := (t_1, \ldots, t_\ell) \in \mathbb{F}^\ell$ there exist efficiently computable polynomials $q_1, \ldots, q_\ell$ such that: $f(\boldsymbol{z}) - f(\boldsymbol{t}) = \sum_{i=1}^{\ell} (z_i - t_i) q_i(\boldsymbol{z})$.*

With this one can verify in time linear in the number of variables that $f(\boldsymbol{t}) = y$ by checking iff $g^{f(\boldsymbol{t})} g^{-y} = \prod_{i=1}^{\ell} e(g^{s_i}, w_i)$, given the values $g^{f(\boldsymbol{s})}, \{g^{s_i}\}_{i=1}^{\ell}, \{w_i = g^{q_i(\boldsymbol{s})}\}_{i=1}^{\ell}$ We are interested in the committed values of $f, y = f(\boldsymbol{t})$ and $\boldsymbol{t}$, c_f, c_y, c_t respectively, that hide them. For this we will use instead the below equation for verification:

$$
\begin{aligned}
(f(\boldsymbol{z}) + r_f z_{\ell+1}) - (f(\boldsymbol{t}) + r_y z_{\ell+1}) = \\
&\sum_{k=1}^{\ell}(z_k - t_k)q_k(\boldsymbol{z}) + z_{\ell+1}(r_f - r_y) = \\
&\sum_{k=1}^{\ell}(z_k - t_k)(q_k(\boldsymbol{z}) + r_k z_{\ell+1}) + z_{\ell+1}\left(r_f - r_y - \sum_{k=1}^{\ell} r_k(z_k - t_k)\right) = \\
&\sum_{k=1}^{\ell}\left[z_k - (t_k + r_{t_k} z_{\ell+1})\right] \cdot \left[q_k(\boldsymbol{z}) + r_k z_{\ell+1}\right] + \\
&\quad z_{\ell+1}\left(r_f - r_y - \sum_{k=1}^{\ell} r_k(z_k - t_k) + \sum_{k=1}^{\ell} r_{t_k}\left[q_k(\boldsymbol{z}) + r_k z_{\ell+1}\right]\right)
\end{aligned}
$$

The equation indicates us how to construct the protocol which we present in figure 16.

Theorem 6.3. *Under the $(\ell+1)d$-Strong Diffie Hellmann and the (d,ℓ)-Extended Power Knowledge of Exponent assumptions, $\mathsf{CP}_{\mathsf{PolyEval}}$ is a Knowledge Extractable CP-SNARK for the relation R_{PolyEval} and the commitment scheme $\mathsf{PolyCom}$.*

Proof Below is a proof sketch, which however is quite similar to the one of $\mathsf{CP}_{\mathsf{poly}}$ in [CFQ19].

Knowledge Soundness. The proof comes directly from Evaluation Extractability of vSQL (see [ZGK+17]) with the difference that here t_k for each $k \in [\ell]$ should also be extracted. However, its extraction is straightforward from the extractability of the commitment scheme.

Zero-Knowledge. Consider the following proof simulator algorithm $\mathcal{S}_{\mathsf{prv}}(\mathsf{td}, c_f, (c_{t_k})_{k\in[\ell]}, c_y)$:

- Use td to get α.
- for $k = 1$ to ℓ, sample $w_k \leftarrow_{\$} \mathbb{G}_1$.
- compute $w_{\ell+1}$ such that $e\left(c_{f,1} \cdot c_{y,1}^{-1}, g\right) = \prod_{k=1}^{\ell} e\left(g^{s_k} c_{t_k,1}^{-1}, w_k\right) \cdot e\left(g^{s_{\ell+1}}, w_{\ell+1}\right)$ holds
- Use α to compute $w'_k = w_k^a$ for all $k \in [\ell+1]$
- Return $\{w_1, ..., w_\ell, w_{\ell+1}, w'_1, \ldots, w'_\ell, w'_{\ell+1}\}$

It is straightforward to check that proofs created by $\mathcal{S}_{\mathsf{prv}}$ are identically distributed to the ones returned by $\mathsf{CP}_{\mathsf{PolyEval}}.\mathsf{Prove}$. $(w_k)_{k\in[\ell]}$'s are uniformely distributed in both cases. For $w_{\ell+1}$ there is a function W such that $w_{\ell+1} = W(c_{f,1}, c_{y,1}, \mathsf{vk}, (c_{t_k,1})_{k\in[\ell]}, (w_k)_{k\in[\ell]})$ in both cases. Since the inputs are either identical or identically distributed, the outputs $w_{\ell+1}$ are also identically distributed in the case of of $\mathcal{S}_{\mathsf{prv}}$ and $\mathsf{CP}_{\mathsf{PolyEval}}.\mathsf{Prove}$. □

7 Experimental Evaluation

We implemented all our RSA-based CP-SNARKs for set-membership and non-membership as a Rust library **cpsnarks-set**.[18] Our library is implemented in a modular fashion such that any

[18] The library will be made open-source and is available to the reviewers upon request.

CP-SNARK for R_{PolyEval}

$\mathsf{KeyGen}(\mathsf{ck}, R_{\mathsf{PolyEval}})$: parses $\mathsf{ck} := (\mathsf{bp}, \mathsf{prk}, \mathsf{prk}^{\alpha}, g^{\alpha}, g^{\beta}, h, h^{\alpha}, h^{\beta})$ and computes $\mathsf{vrk} \leftarrow \{g^{s_1}, \ldots, g^{s_\ell}\}$
Return $(\mathsf{ek}, \mathsf{vk}) \leftarrow \left(\left(\mathsf{bp}, \mathsf{prk}, \mathsf{prk}^{\alpha}, g^{\alpha}, g^{\beta}, h, h^{\alpha}, h^{\beta}\right) , \left(\mathsf{bp}, \mathsf{vrk}, g^{\alpha}, g^{\beta}, h\right) \right)$

$\mathsf{Prove}(\mathsf{ek}, \varnothing, (c_y, (c_{t_k})_{k\in[\ell]}, c_f), (y, (t_k)_{k\in[\ell]}, f), (r_y, (r_{t_k})_{k\in[\ell]}, r_f), \varnothing)$: let $\mathsf{ck} := (\mathsf{bp}, \mathsf{prk}, \mathsf{prk}^{\alpha}, g^{\alpha}, g^{\beta}, h, h^{\alpha}, h^{\beta}) := \left((q, g, \mathbb{G}_1, \mathbb{G}_T, e), \left\{g^{\prod_{i\in W} s_i} : W \in \mathcal{W}_{\ell,d}\right\}, \left\{g^{\alpha\cdot\prod_{i\in W} s_i} : W \in \mathcal{W}_{\ell,d}\right\}, g^{\alpha}, g^{\beta}, g^{s_{\ell+1}}, g^{\alpha s_{\ell+1}}, g^{\beta s_{\ell+1}}\right)$ and

1. Sample $r_1, ..., r_\ell \leftarrow \mathbb{F}$ and compute $q_1, \ldots, q_\ell$ such that

$$(f(\boldsymbol{z}) + r_f z_{\ell+1}) - (f(\boldsymbol{t}) + r_y z_{\ell+1}) =$$
$$\sum_{k=1}^{\ell} [z_k - (t_k + r_{t_k} z_{\ell+1})] \cdot [q_k(\boldsymbol{z}) + r_k z_{\ell+1}] + z_{\ell+1} \left(r_f - r_y - \sum_{k=1}^{\ell} r_k (z_k - t_k) + \sum_{k=1}^{\ell} r_{t_k} \left[q_k(\boldsymbol{z}) + r_k z_{\ell+1}\right] \right)$$

   By using $\mathsf{prk} := \left\{g^{\prod_{i\in W} s_i} : W \in \mathcal{W}_{\ell,d}\right\}$ and h compute $w_k = g^{q_k(\boldsymbol{s}) + r_k s_{\ell+1}}$ for each $k = 1, \ldots, \ell$ and $w_{\ell+1} = g^{r_f - r_y - \sum_{k=1}^{\ell} r_k (s_k - t_k) + \sum_{k=1}^{\ell} r_{t_k} \left[q_k(\boldsymbol{s}) + r_k s_{\ell+1}\right]}$
2. By using $\mathsf{prk}^{\alpha} := \left\{g^{\alpha\cdot\prod_{i\in W} s_i} : W \in \mathcal{W}_{\ell,d}\right\}$ and h^{α} compute $w'_k = g^{\alpha\cdot(q_k(\boldsymbol{s}) + r_k s_{\ell+1})}$ for each $k = 1, \ldots, \ell$ and $w'_{\ell+1} = g^{\alpha\cdot\left(r_f - r_y - \sum_{k=1}^{\ell} r_k (s_k - t_k) + \sum_{k=1}^{\ell} r_{t_k} \left[q_k(\boldsymbol{s}) + r_k s_{\ell+1}\right]\right)}$

Return $\pi = \{w_1, ..., w_\ell, w_{\ell+1}, w'_1, ..., w'_\ell, w'_{\ell+1}\}$

$\mathsf{VerProof}(\mathsf{vk}, \varnothing, (c_y, (c_{t_k})_{k\in[\ell]}, c_f), \pi)$: parse $\pi := \{w_1, ..., w_\ell, w_{\ell+1}, w'_1, ..., w'_\ell, w'_{\ell+1}\}$, $\mathsf{vk} := \left(\mathsf{bp}, \mathsf{vrk}, g^{\alpha}, g^{\beta}, h\right)$ and $c_y := (c_{y,1}, c_{y,2})$, $c_{t_k} := (c_{t_k,1}, c_{t_k,2})$ for each $k = 1, \ldots, \ell]$ and $c_f := (c_{f,1}, c_{f,2})$
Return 1 iff

1. $e(c_{y,1}, g^{\beta}) = e(c_{y,2}, g)$
2. $e(c_{f,1}, g^{\alpha}) = e(c_{f,2}, g)$
3. $e(c_{t_k,1}, g^{\beta}) = e(c_{t_k,2}, g)$ for all $k = 1, ..., \ell$
4. $e(w_k, g^{\alpha}) = e(w'_k, g)$ for all $k = 1, ..., \ell, \ell+1$
5. $e\left(c_f \cdot c_y^{-1}, g\right) = \prod_{k=1}^{\ell} e\left(g^{s_k} c_{t_k}^{-1}, w_k\right) \cdot e\left(g^{s_{\ell+1}}, w_{\ell+1}\right)$

Fig. 16

elliptic curve from **libzexe** [SCI] and Ristretto from **curve25519-dalek** [LdV] can be used. In particular, this means that our CP-SNARKs can be easily (and efficiently) used in combination with other CP-SNARKs implemented over these elliptic curves, such as Bulletproofs [BBB+18] and LegoGroth16[19] [CFQ19].

In this section, we provide details on the implementation, we present experimental results to validate the concrete efficiency of our solutions and we compare with existing approaches.

7.1 Implementation of cpsnarks-set

Our **cpsnarks-set** library includes implementations of the schemes $\mathsf{MemCP}_{\mathsf{RSA}}$, $\mathsf{MemCP}_{\mathsf{RSAPrm}}$, $\mathsf{NonMemCP}_{\mathsf{RSA}}$, and $\mathsf{NonMemCP}_{\mathsf{RSAPrm}}$. In all the schemes, the RSA accumulator implementation is a modification of **accumulator** [Cam19], and the internal protocols are implemented as interactive and are made non-interactive with the use of **Merlin** [Hen19]. For $\mathsf{MemCP}_{\mathsf{RSA}}$ and $\mathsf{NonMemCP}_{\mathsf{RSA}}$ – where we recall set elements can be binary strings and the protocol encodes them into primes – we used our implementation of LegoGroth16 on top of **libzexe** to provide efficient instantiations of $\mathsf{CP}_{\mathsf{HashEq}}$. For $\mathsf{MemCP}_{\mathsf{RSAPrm}}$ and $\mathsf{NonMemCP}_{\mathsf{RSAPrm}}$ – where set elements are already primes and one needs to verify a claim about ranges – we implemented two instantiations of $\mathsf{CP}_{\mathsf{Range}}$: one based on LegoGroth16 and one based on Bulletproofs.

Each of the protocols $\mathsf{Root}, \mathsf{Coprime}, \mathsf{modEq}, \mathsf{HashEq}$ and the different instantiations of Range are implemented individually and are further composed into the higher level membership and non-membership protocols. The higher level protocols are modular: they can use any hash-to-prime proof—or range proof in the prime elements case—as long as it implements the appropriate interface.

We benchmark the implementation on a 2019 MacBook Pro, having a 2.8 Ghz Quad-Core Intel Core i7 processor and 16GB RAM (the benchmark code is available upon request).

7.2 CP-SNARKs for Set Membership

For the problem of set membership, we tested the following instantiations of our solutions using the RSA-2048 [rsa] modulus: 1. $\mathsf{MemCP}_{\mathsf{RSA}}$ with LegoGroth16 for $\mathsf{CP}_{\mathsf{HashEq}}$ and a Blake2s-based hash-to-prime mapping to 252-bit primes ($\mathsf{MemCP}^{\mathsf{LG}}_{\mathsf{RSA}}$);
2. $\mathsf{MemCP}_{\mathsf{RSAPrm}}$ with LegoGroth16 on the BLS12-381 curve for $\mathsf{CP}_{\mathsf{Range}}$ ($\mathsf{MemCP}^{\mathsf{LG}}_{\mathsf{RSAPrm}}$), and: (a) 252-bit primes, (b) 63-bit primes; 3. $\mathsf{MemCP}_{\mathsf{RSAPrm}}$ with Bulletproofs on the Ristretto curve for $\mathsf{CP}_{\mathsf{Range}}$ ($\mathsf{MemCP}^{\mathsf{BP}}_{\mathsf{RSAPrm}}$), and: (a) 250-bit primes; (b) 62-bit primes. The results of our experiments are summarized in Figure 17.

Comparison with Merkle-tree approach. We compare our solutions against one based on proving a valid opening of a Merkle Tree in a SNARK. Specifically, we ran experiments for Merkle trees with maximum capacities of $\{2^8, 2^{16}, 2^{32}, 2^{64}\}$ elements, using the Groth16 SNARK [Gro16] over the BLS12-381 curve, with the following hash functions: 1. Pedersen Hash over the Jubjub curve, a curve defined over the scalar field of the BLS12-381 $\mathbb{G}_1$ group. [20] 2. SHA256. The Merkle tree benchmark code is based on the production Zcash code from [Zca]. The results of the experiments are in Fig. 18. We recall that proofs in this solution are of 192 bytes.

As one can see from the results, our solutions are highly attractive in terms of proving time and CRS size. For instance, compared to an optimized solution based on a Pedersen-Hash-based

[19] We implemented this scheme in Rust on top of **libzexe** as part of this work.
[20] This is the Bowe-Hopwood variant of a Pedersen hash, as described in [HBHW16].

Fig. 17: Set Membership benchmarks – Our RSA schemes ($|x|$: size of set elements)

Solution	$\|x\|$	P_{time}	V_{time}	$\|\text{crs}\|$	$\|\Pi\|$	P_{memory}
$\mathsf{MemCP}^{\mathsf{LG}}_{\mathsf{RSA}}$	252	479.50	37.14	6852	3.6	45
$\mathsf{MemCP}^{\mathsf{LG}}_{\mathsf{RSAPrm}}$	252	54.51	33.78	86	3.6	5
$\mathsf{MemCP}^{\mathsf{LG}}_{\mathsf{RSAPrm}}$	63	54.11	32.26	86	3.6	5
$\mathsf{MemCP}^{\mathsf{BP}}_{\mathsf{RSAPrm}}$	250	66.24	48.53	8	0.9	5
$\mathsf{MemCP}^{\mathsf{BP}}_{\mathsf{RSAPrm}}$	62	39.93	24.84	2	0.8	5
	bits	ms	ms	KB	KB	MB

Fig. 18: Set membership benchmarks – Merkle trees through [Gro16] zkSNARK

Depth	Hash	P_{time}	V_{time}	$\|\text{crs}\|$	$\|\Pi\|$	P_{memory}
8	Pedersen	380	3.7	2512	0.192	22
16	Pedersen	720	3.7	5023	0.192	35
32	Pedersen	1120	3.7	10047	0.192	49
64	Pedersen	2280	3.7	20094	0.192	79
8	SHA256	2390	3.7	41276	0.192	93
16	SHA256	4530	3.7	82430	0.192	196
32	SHA256	11330	3.7	164737	0.192	423
64	SHA256	21620	3.7	329352	0.192	913
		ms	ms	KB	KB	MB

Merkle tree containing up to 2^{32} elements, our $\mathsf{MemCP}_{\mathsf{RSA}}$ scheme for arbitrary elements enjoys a sub-second proof generation on a commodity laptop, it is more than twice faster and requires a shorter CRS. A price to pay in our solution is a larger proof size (3.6 kilobytes vs. 192 bytes) and higher verification time (37ms vs. 3.7ms). Nevertheless, these values stay within practical reach. When comparing to less optimized solutions based on Merkle trees (e.g., using SHA256, something common in lack of specialized elliptic curves), we achieve up to 40× faster proving time and a 48× shorter CRS.

In addition to the aforementioned gains in prover efficiency, our solutions can benefit from the use of RSA accumulators to succinctly represent sets in comparison to using Merkle trees. In particular, the algebraic properties of RSA accumulators yield simple and efficient methods to add (resp. delete) elements to (resp. from) the set.

For instance, we can insert an element in an RSA accumulator in $O(1)$ time and space, and with the same complexity we can update each existing membership and non-membership witness. This means that, once having an updated witness, our zero-knowledge proofs can also be recomputed in $O(1)$ time and space. With respect to deleting elements, this can also be done in constant time and space by a party who holds a valid membership witness.

Insertion and deletion in ordinary Merkle Trees may require $O(n)$ time by rebuilding the tree from scratch from the whole set (thus also requiring $O(n)$ storage). A more efficient method for insertion requires clients to store a "frontier" of size $\Theta(\mathsf{log}(n))$ of internal hashes which lowers the time complexity to $O(\mathsf{log}(n))$. One can also lower deletion times to $O(\mathsf{log}(n))$ by using other techniques, e.g., [Ray19], but at the expense of keeping $O(n)$ storage. Updating a Sparse Merkle Trees requires $O(n)$ time and space during updates. Inserting and deleting elements in Interval Merkle trees requires keeping the elements contiguous and sorted. This brings the time/storage complexity to $O(n)$ for insertion and deletion, since we may need to rebuild substantial portions of the tree from scratch.

Fig. 19: Set non-membership benchmarks – Our RSA schemes ($|x|$: size of set elements)

Solution	$\|x\|$	P_{time}	V_{time}	$\|\text{crs}\|$	$\|\Pi\|$	P_{memory}
$\mathsf{NMem}^{\mathsf{LG}}_{\mathsf{RSA}}$	252	508.53	45.98	6852	5.4	45
$\mathsf{NMem}^{\mathsf{LG}}_{\mathsf{RSAPrm}}$	252	83.37	44.44	86	5.4	5
$\mathsf{NMem}^{\mathsf{BP}}_{\mathsf{RSAPrm}}$	250	81.32	39.72	8	0.8	5
	bits	ms	ms	KB	KB	MB

7.3 CP-SNARKs for Set Non-Membership

For set non-membership, we tested the following instantiations of our solutions using the RSA-2048 [rsa] modulus: 1. $\mathsf{NonMemCP}_{\mathsf{RSA}}$ with LegoGroth16 for $\mathsf{CP}_{\mathsf{HashEq}}$ and a Blake2s-based hash–to-prime mapping yielding primes of 252 bits; 2. $\mathsf{NonMemCP}_{\mathsf{RSAPrm}}$ with LegoGroth16 on the BLS12-381 curve for $\mathsf{CP}_{\mathsf{Range}}$, and 252-bit primes; 3. $\mathsf{NonMemCP}_{\mathsf{RSAPrm}}$ with Bulletproofs on the Ristretto curve for $\mathsf{CP}_{\mathsf{Range}}$, and 250-bit primes.

The results of our experiments are summarized in Figure 19.

Comparison to other approaches for non-membership Non-membership proofs are usually a more computationally intensive task in SNARKs. There are two common approaches to deal with this problem using Merkle trees: *sparse Merkle trees* and *interval Merkle trees*. We did not test these solutions experimentally. However, as we detail below, creating a zero-knowledge proof for one of these solutions would not be more efficient than proving one Merkle tree path. Therefore, our solutions for non-membership achieve at least the same improvement as in the previous section.

Sparse Merkle trees for a set S are built through an ordinary Merkle Tree T on the *universe* $\mathbb{U}$ of elements (we assume there is some conventional way to index the elements). For each element x not in the set S we store a dummy element in T corresponding to the index of x. For each element in the S we store that particular element at the corresponding index. In order to prove that $x \notin S$ we provide an opening path of a Merkle tree whose leaf is a dummy value at the right index. Although there are efficient techniques to build or update a sparse Merkle Tree [LK12, DPP16], the main drawback with this technique is the opening size, which is $\Theta(\mathsf{log}(|\mathbb{U}|))$ instead of $\Theta(\mathsf{log}(|S|))$. If we perform the opening inside a SNARK, we have to pay a higher proving time. For example, consider if we use SHA256 to index elements in a set with a roughly 32 bit-representations. This would require a tree of size 2^{256} which typically implies at least a $256/32 = 8\times$ slowdown.

Interval Merkle trees work by sorting the leaves on each insertion and storing a pair of adjacent elements in each leaf, signifying intervals that don't contain elements in the set. The depth of an Interval Merkle Tree is the same as in an ordinary Merkle Tree. Nonetheless it has the following performance overheads: *(i) opening* requires two opening paths instead of only one (typically doubling the proving time); *(ii) insertion* requires sorting all leaves, which may be computationally demanding if the set is large.

Unlike either of the approaches above, the size of the set does not impact proving time in our constructions. Moreover, both insertions and non-membership witness updates are efficient to compute.

8 Applications

In this section, we discuss applications of our solutions for proving set (non-)membership in a succinct and modular way.

As one can note, in our solutions the set of committed elements is public and not hidden to the verifier. Nevertheless, our solutions can still capture some applications in which the "actual" data in the set is kept private. This is for example the case of anonymous cryptocurrencies like Zerocash. In this scenario, the public set of elements to be accumulated, U, is derived from creating a commitment to the underlying data, X, e.g., $u = COMM(x)$. To support this setting, we can use our solutions for arbitrary elements (so supporting virtually any commitment scheme). Interestingly, though, we can also use our (more efficient) solution for sets of primes if commitments are prime numbers. This can be done by using for example the *hash-to-prime* method described in Section 4.1 or another method for Pedersen commitments that we explain below in the context of Zerocash.

We now discuss concrete applications for which our constructions are suitable, both for set-membership and set non-membership. In particular these are applications in which: (1) the prover time must be fast; (2) the size of the state (i.e.: the accumulator value and commitments) must be small (potentially constant); (3) the verifier time should be small; and (4) the time to update the accumulator—adding or deleting an element—should be fast. As we discuss below, our RSA-based constructions are suitable candidate for settings with these constraints.

ZEROCASH. Zerocash [BCG+14] is a UTXO-type (Unspent Transaction Output) crytptocurrency protocol which extends Bitcoin with privacy-preserving (*shielded*) transactions. When performing a shielded transaction users need to prove they are spending an output note from a token they had previously received. Users concerned with privacy should not reveal which note they are spending, else their new transaction could be linked to the original note that contained the note commitment. This would reveal information *both to the public and the sender of the initial transaction, and hence partially revealing the transaction graph.* In order to keep transactions unlinkable, the protocol uses zkSNARKs to prove a set membership relation, namely that a note commitment is in a publicly known set of "usable" note commitments.

Zcash is a full-fledged digital currency using Zerocash as the underlying protocol. In its current deployment, Sapling [HBHW16], it employs Pedersen commitments of the notes and makes a zero-knowledge set membership proof of these commitments using a Pedersen-Hash-based Merkle tree approach. This is the part of the protocol that can be replaced by one of our RSA-based solutions in order to obtain a speedup in proving time. In particular, we could slightly modify the note commitments in order to enable the use of our scheme $\mathsf{MemCP}_{\mathsf{RSAPrm}}$ for sets of prime numbers, which gives the best efficiency. We can proceed as follows. Let us recall that the note commitments are represented by their x coordinates in the underlying elliptic curve group. We can then modify them so that the sender chooses a blinding factor such that the commitment representation of a note is a prime number, and we can add a consensus rule that enforces this check. With this change, we can achieve a solution that is significantly more efficient than that currently used in Zcash. Currently Zcash uses a Merkle Tree whose depth is 32. In this setting, *we would be able to reduce proving time of set-membership from* $1.12s$ *to* $54.51ms$, trading it for larger proof sizes. We note that in this application, the set-membership proof about $u \in S$ is accompanied by another predicate $P(u)$. In the proof statement of the Zcash protocol, proving that $P(u)$ is satisfied takes considerably less time than the membership proof, hence this is why our solution would improve the overall proving time considerably, albeit the proof having more components. Another interesting comment is that our solution significantly reduces the size of the circuit, hence the need of a succinct proof system is reduced and one may even consider instantiations with other proof systems, such as Bulletproofs, that would offer transparency at the price of larger proofs and verification time.

Asset Governance. In the context of *blockchain-based asset transfers protocols*, a governance system must be established to determine who can create new assets. In many cases these assets must be publicly traceable (i.e., their total supply must be public), yet in others, where the assets can be issued privately, validators still need to verify that the assets were issued by an authorized issuer. Specifically, there may be a public set of rules, X (where a $\mathsf{rule} = (pk, [a, b])$), defining which entities (public keys) are allowed to issue which assets (defined by a range of *asset types*), forming an "issuance whitelist". When one of those issuers wants to issue a new asset, they need to prove (in zero knowledge) that their public key belongs to the issuance whitelist, which entails set membership, as well as prove that the asset type they issued is within the allowed range of asset types (as defined in the original rule). In this case, the accumulated set of rules is public to all, and this public information may also include a mapping between rules and prime numbers. Our RSA-based scheme for sets of primes (Section 4.3) can suit this scenario.

Anonymous Broadcast. In a peer-to-peer setting, anonymous broadcast allows users in a group to broadcast a message without revealing their identity. They can only broadcast *once* on each topic. One approach described in [sem19] works by asking users to put down a deposit which they will lose if they try and broadcast multiple messages on the same topic. In this approach users joining a group deposit their collateral in a smart contract. Whoever has the private key used by the client for the deposit can claim the sum. The approach in [sem19] makes sure that the key is leaked if one broadcasts more than one message. To enforce this leakage we require that at broadcast time users *(i)* derive an encryption key K that depends on their private key and the topic, and *(ii)* compute an encryption of the private key by the newly derived K. Then the users publish both the ciphertext and a secret share of the encryption key K, and prove (in zero-knowledge) their public key is part of the group and that *(i)* and *(ii)* were performed correctly. Which specific share needs to be revealed depends on the broadcasted message, thus making it likely two different shares will be leaked for two different messages.

This way, broadcasting multiple messages on the same topic reveals the user's private key, allowing other users to remove them from the group by calling a function in the smart contract and receive part of the deposit.

A particularly interesting use case for anonymous broadcast is that in which the group is comprised of validators participating in a consensus algorithm, who would like to broadcast messages without exposing their node's identity and thus prevent targeted DoS attacks. This setting requires proofs to be computed extremely fast while verification performance requirements are less strict. Our $\mathsf{MemCP}_{\mathsf{RSAPrm}}$ can satisfy these performance requirements trading for a modest increase in proof size.

Financial Identities. In the financial world, regulations establish that financial organizations must *know* who their costumers are [FIN]. This is called a KYC check and allows to reduce the risk of fraud. Some common practices for KYC often undermines user privacy as they involve collecting a lot of personal information on them. Zero-knowledge proofs allow for an alternative approach. In modern systems, one can expect that individuals or companies will be able to prove that they belong to a set of accepted or legitimate identities. A privacy-preserving KYC check would then be reduced to generating a set-membership proof in zero-knowledge. Often some further information is required, e.g. the credit score of the individual. In such cases our CP-SNARK for set membership can be combined with one proving an additional predicate $P(\mathsf{id})$ on the identity in a modular fashion.

Regarding applications of non-membership proofs, we expand on the well-known concept of "blacklists", where identities (or credentials) must be shown to *not* belong to a certain set of identities (or credentials). As an example, in the context of financial identities, anti-money laundering regulations (AML) [SC18] require customers not to be in a list of fraudulent identities. Here one can use our non-membership construction to generate a proof that the customer does not belong to the set of money launderers (or those thought to be). Because, as in the set-membership case, a user may have to prove additional information about their identity, here we can also benefit from a modular framework. Furthermore, modularity allows us to cheaply prove both membership and non-membership (at the same time) for the same identity id together with some additional information $P(\mathsf{id})$: holding commitment $c(\mathsf{id})$ one can produce the following tuple of proofs: (1) a membership proof ($\mathsf{id} \in S$); (2) a non-membership proof ($\mathsf{id} \notin S'$); (3) a CP-SNARK proof that includes the statement to be proven on that identity ($P(\mathsf{id})$).

We note that in some cases, a central authority, who controls the white and black lists, is trusted to ensure the integrity of the lists. This means that the identities can be added or removed from the lists, which means that our RSA-based construction is ideal given the comparatively reduced cost of updating the dynamic accumulator.

Zerocoin Vulnerability. Another specific application of our RSA-based constructions is that of solving the security vulnerability of the implementation of the Zerocoin protocol [MGGR13] used in the Zcoin cryptocurrency [Yap19]. The vulnerability in a nutshell: when proving equality of values committed under the RSA commitment and the prime-order group commitment, the equality may not hold over the integers, and hence one could easily produce collisions in the prime order group. Our work can provide different ways to solve this problem by generating a proof of equality over the integers.

Acknowledgements

Research leading to these results has been partially supported by the Spanish Government under projects SCUM (ref. RTI2018-102043-B-I00), CRYPTOEPIC (refs. ERC2018-092822, EUR2019-103816), and SECURITAS (ref. RED2018-102321-T), by the Madrid Regional Government under project BLOQUES (ref. S2018/TCS-4339), and by research gifts from Protocol Labs.

References

AGM18. Shashank Agrawal, Chaya Ganesh, and Payman Mohassel. Non-interactive zero-knowledge proofs for composite statements. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 643–673. Springer, Heidelberg, August 2018.

BBB+18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.

BBF18a. Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching techniques for accumulators with applications to iops and stateless blockchains. *IACR Cryptology ePrint Archive*, 2018:1188, 2018.

BBF18b. Dan Boneh, Benedikt Bünz, and Ben Fisch. A survey of two verifiable delay functions. Cryptology ePrint Archive, Report 2018/712, 2018. `https://eprint.iacr.org/2018/712`.

BCF19. Daniel Benarroch, Matteo Campanelli, and Dario Fiore. Community standards proposal for commit-and-prove zero-knowledge proof systems, 2019. https://www.binarywhales.com/assets/misc/zkproof-cp-standards.pdf.

BCG+14. Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014.

Bd94. Josh Cohen Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In Tor Helleseth, editor, *EUROCRYPT'93*, volume 765 of *LNCS*, pages 274–285. Springer, Heidelberg, May 1994.

BH01. Johannes Buchmann and Safuat Hamdy. A survey on {IQ} cryptography, 2001. `http://tubiblio.ulb.tu-darmstadt.de/100933/`.

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

Cam19. Cambrian Tech. Cryptographic accumulators in rust., 2019. `https://github.com/cambrian/accumulator`.

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

CFQ19. Matteo Campanelli, Dario Fiore, and Anaïs Querol. Legosnark: Modular design and composition of succinct zero-knowledge proofs. *To appear at ACM CCS 2019. IACR Cryptology ePrint Archive*, 2019, 2019.

Cha85. David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, October 1985.

CKS09. Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 481–500. Springer, Heidelberg, March 2009.

CL02. Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 61–76. Springer, Heidelberg, August 2002.

CLOS02. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002.

CMS99. Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 402–414. Springer, Heidelberg, May 1999.

CPP17. Geoffroy Couteau, Thomas Peters, and David Pointcheval. Removing the strong RSA assumption from arguments over the integers. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 321–350. Springer, Heidelberg, April / May 2017.

CPZ18. Alexander Chepurnoy, Charalampos Papamanthou, and Yupeng Zhang. Edrax: A cryptocurrency with stateless transaction validation. 2018.

CS99. Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. In Juzar Motiwalla and Gene Tsudik, editors, *ACM CCS 99*, pages 46–51. ACM Press, November 1999.

DF02. Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 125–142. Springer, Heidelberg, December 2002.

DG20. Samuel Dobson and Steven D. Galbraith. Trustless groups of unknown order with hyperelliptic curves. Cryptology ePrint Archive, Report 2020/196, 2020. `https://eprint.iacr.org/2020/196`.

DPP16. Rasmus Dahlberg, Tobias Pulls, and Roel Peeters. Efficient sparse merkle trees: Caching strategies and secure (non-)membership proofs. Cryptology ePrint Archive, Report 2016/683, 2016. `https://eprint.iacr.org/2016/683`.

DT08. Ivan Damgård and Nikos Triandopoulos. Supporting non-membership proofs with bilinear-map accumulators. Cryptology ePrint Archive, Report 2008/538, 2008. `http://eprint.iacr.org/2008/538`.

EG14. Alex Escala and Jens Groth. Fine-tuning Groth-Sahai proofs. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 630–649. Springer, Heidelberg, March 2014.

FFG+16. Dario Fiore, Cédric Fournet, Esha Ghosh, Markulf Kohlweiss, Olga Ohrimenko, and Bryan Parno. Hash first, argue later: Adaptive verifiable computations on outsourced data. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1304–1316. ACM Press, October 2016.

FIN. FINRA. `https://www.finra.org/rules-guidance/rulebooks/finra-rules/2090#the-rule`.

FN02. Nelly Fazio and Antonio Nicolosi. Cryptographic accumulators: Definitions, constructions and applications. *Paper written for course at New York University: www. cs. nyu. edu/nicolosi/papers/accumulators. pdf*, 2002.

FO97. Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 16–30. Springer, Heidelberg, August 1997.

FT14. Pierre-Alain Fouque and Mehdi Tibouchi. Close to uniform prime number generation with fewer random bits. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *ICALP 2014, Part I*, volume 8572 of *LNCS*, pages 991–1002. Springer, Heidelberg, July 2014.

GHR99. Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 123–139. Springer, Heidelberg, May 1999.

GMR89. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008.

GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011.

HBHW16. Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash protocol specification. *Tech. rep. 2016–1.10. Zerocoin Electric Coin Company, Tech. Rep.*, 2016. `https://github.com/zcash/zips/blob/master/protocol/sapling.pdf`.

Hen19. Henry de Valence. Merlin: composable proof transcripts for public-coin arguments of knowledge, 2019. `https://github.com/dalek-cryptography/merlin`.

LdV. Isis Agora Lovecruft and Henry de Valence. curve25519-dalek: A pure-rust implementation of group operations on ristretto and curve25519. `https://github.com/dalek-cryptography/curve25519-dalek`.

Lee20. Jonathan Lee. The security of groups of unknown order based on jacobians of hyperelliptic curves. Cryptology ePrint Archive, Report 2020/289, 2020. `https://eprint.iacr.org/2020/289`.

Lip12. Helger Lipmaa. Secure accumulators from euclidean rings without trusted setup. In Feng Bao, Pierangela Samarati, and Jianying Zhou, editors, *ACNS 12*, volume 7341 of *LNCS*, pages 224–240. Springer, Heidelberg, June 2012.

LK12. Ben Laurie and Emilia Kasper. Revocation transparency. *Google Research, September*, page 33, 2012.

LLNW16. Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 1–31. Springer, Heidelberg, May 2016.

LLX07. Jiangtao Li, Ninghui Li, and Rui Xue. Universal accumulators with efficient nonmembership proofs. In Jonathan Katz and Moti Yung, editors, *ACNS 07*, volume 4521 of *LNCS*, pages 253–269. Springer, Heidelberg, June 2007.

LY10. Benoît Libert and Moti Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 499–517. Springer, Heidelberg, February 2010.

Mer88. Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 369–378. Springer, Heidelberg, August 1988.

MGGR13. Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed E-cash from Bitcoin. In *2013 IEEE Symposium on Security and Privacy*, pages 397–411. IEEE Computer Society Press, May 2013.

Ngu05. Lan Nguyen. Accumulators from bilinear pairings and applications. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 275–292. Springer, Heidelberg, February 2005.

OWWB19. Alex Ozdemir, Riad S. Wahby, Barry Whitehat, and Dan Boneh. Scaling verifiable computation using efficient set accumulators. Cryptology ePrint Archive, Report 2019/1494, 2019. `https://eprint.iacr.org/2019/1494`.

PHGR13. Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013.

PST13. Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. Signatures of correct computation. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 222–242. Springer, Heidelberg, March 2013.

PSTY13. Charalampos Papamanthou, Elaine Shi, Roberto Tamassia, and Ke Yi. Streaming authenticated data structures. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 353–370. Springer, Heidelberg, May 2013.

Ray19. James Ray. Patricia tree, 2019. https://github.com/ethereum/wiki/wiki/Patricia-Tree.

rsa. Rsa-2048. `https://en.wikipedia.org/wiki/RSA_numbers#RSA-2048`.

SC18. U.S. Securities and Exchange Commission. Anti-money laundering (aml) source tool for broker-dealers, October 2018. `https://www.sec.gov/about/offices/ocie/amlsourcetool.htm`.

SCI. SCIPR Lab. Zexe (zero knowledge execution). `https://github.com/scipr-lab/zexe`.

sem19. Semaphore rln, rate limiting nullifier for spam prevention in anonymous p2p setting, February 2019. `https://ethresear.ch/t/semaphore-rln-rate-limiting-nullifier-for-spam-prevention-in-anonymous-p2p-setti 5009`.

Val08. Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 1–18. Springer, Heidelberg, March 2008.

Wes18. Benjamin Wesolowski. Efficient verifiable delay functions. Cryptology ePrint Archive, Report 2018/623, 2018. `https://eprint.iacr.org/2018/623`.

Yap19. Reuben Yap. Cryptographic description of zerocoin attack, 2019. https://zcoin.io/cryptographic-description-of-zerocoin-attack/.

Zca. Zcash. Zcash rust crates. `https://github.com/zcash/librustzcash`.

ZGK+17. Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. A zero-knowledge version of vSQL. Cryptology ePrint Archive, Report 2017/1146, 2017. `https://eprint.iacr.org/2017/1146`.

ZKP17. Y. Zhang, J. Katz, and C. Papamanthou. An expressive (zero-knowledge) set accumulator. In *2017 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 158–173, April 2017.

A Accumulator Definitions

Below is the definition of Accumulators, following the definition of [FN02]. We insist on public key accumulators, meaning that after the key generation phase no party has access to the secret key.

Definition A.1 (Accumulators). *A static (non-Universal) Accumulator with domain $\mathbb{X}$ is a tuple of 4-algorithms,* $\mathsf{Acc} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Witness}, \mathsf{VerWit})$

- $\mathsf{Gen}(1^\lambda, t) \to (\mathsf{sk}, \mathsf{ek}, \mathsf{vk})$ *is a (probabilistic) algorithm that takes the security parameter λ and a parameter t for the upper bound of the number of elements to be accumulated. If $t = \infty$ there is no upper bound. Returns a secret key* sk*, an evaluation key* ek *and a verification key* vk*.*
- $\mathsf{Eval}(\mathsf{ek}, \mathcal{X}) \to (acc_\mathcal{X}, \mathsf{aux})$ *takes the evaluation key and a set $\mathcal{X}$ and in case $\mathcal{X} \subseteq \mathbb{X}$ outputs the accumulated value $acc_\mathcal{X}$ and some auxiliary information* aux*. If $\mathcal{X} \nsubseteq \mathbb{X}$ outputs $\perp$.*
- $\mathsf{Witness}(\mathsf{ek}, x, \mathsf{aux}) \to wit_x$ *takes the evaluation key* ek*, the value x and the auxiliary information* aux *and outputs either a witness wit_x of $x \in \mathcal{X}$ or $\perp$ if $x \notin \mathcal{X}$.*
- $\mathsf{VerWit}(\mathsf{vk}, acc_\mathcal{X}, x, w) \to b$ *takes the verification key* vk*, the accumulation value $acc_\mathcal{X}$, a value x and a witness w and outputs 1 if wit_x is a witness of $x \in \mathcal{X}$ and 0 otherwise.*

Further, we give the definition of Dynamic Accumulators, a notion that was introduced by Camenisch and Lysyanskaya [CL02]. Dynamic Accumulators are Accumulators that additionally provide the ability to update the accumulated value and the witnesses when the set is updated, either on addition of a new element or on deletion.

Definition A.2 (Dynamic Accumulators). *A Dynamic Accumulator* Acc *with domain $\mathbb{X}$ is a static Accumulator that additionally provides three algorithms* $(\mathsf{Add}, \mathsf{Delete}, \mathsf{WitUpdate})$*.*

$\mathsf{Add}(\mathsf{ek}, acc_{\mathcal{X}}, y, \mathsf{aux}) \to (acc_{\mathcal{X}'}, \mathsf{aux}')$ *takes the evaluation key* ek*, the accumulated value* $acc_{\mathcal{X}}$*, the value to be added to the set* y *and the auxiliary information* aux*. If* $y \notin \mathcal{X} \wedge y \in \mathbb{X}$ *outputs the new accumulation value for* $\mathcal{X}' = \mathcal{X} \cup \{y\}$*,* $acc_{\mathcal{X}'}$ *and a new auxiliary information* aux'*. In case* $y \in \mathcal{X}$ *or* $y \notin \mathbb{X}$ *outputs* $\perp$*.*

$\mathsf{Delete}(\mathsf{ek}, acc_{\mathcal{X}}, y, \mathsf{aux}) \to (acc_{\mathcal{X}'}, \mathsf{aux}')$ *takes the evaluation key* ek*, the accumulated value* $acc_{\mathcal{X}}$*, the value to be deleted from the set* y *and the auxiliary information* aux*. If* $y \in \mathcal{X} \wedge y \in \mathbb{X}$ *outputs the new accumulation value for* $\mathcal{X}' = \mathcal{X} \setminus \{y\}$*,* $acc_{\mathcal{X}'}$ *and a new auxiliary information* aux'*. In case* $y \notin \mathcal{X}$ *or* $y \notin \mathbb{X}$ *outputs* $\perp$*.*

$\mathsf{WitUpdate}(\mathsf{ek}, wit_x, y, \mathsf{aux}) \to wit'_x$ *takes the evaluation key* ek*, a witness* wit_x *to be updated, the value* y *that was either added or deleted from* $\mathcal{X}$ *and the auxiliary information. In case* $x \in \mathcal{X}'$ *outputs the updated witness* wit'_x*, otherwise outputs* $\perp$*.*

Normally, we demand that update algorithms, Add and Delete are more efficient than recomputing the accumulation value from scratch with Eval. However in the publicly updatable setting this is not always possible, while it may be possible when the party holds the secret key. Still in this work we treat public key accumulators.

Security Correctness. For every $t = \mathsf{poly}(\lambda)$ and $|\mathcal{X}| \leq t$:

$$Pr\left[\begin{array}{l}(\mathsf{sk}, \mathsf{ek}, \mathsf{vk}) \leftarrow \mathsf{GenAcc}(1^\lambda, t); \\ \quad acc_{\mathcal{X}} \leftarrow \mathsf{EvalAcc}(\mathsf{ek}, \mathcal{X}); \quad : \mathsf{VerWit}(\mathsf{vk}, acc_{\mathcal{X}}, x, w) \\ \quad\quad w \leftarrow \mathsf{Witness}(\mathsf{ek}, \mathcal{X}, x)\end{array}\right] = 1$$

Soundness. A cryptographic accumulator is sound if for all $t = \mathsf{poly}(\lambda)$ and for all PPT adversaries $\mathcal{A}$ there is a negligible function $\mathsf{negl}(\cdot)$ such that:

$$Pr\left[\begin{array}{l}(\mathsf{sk}, \mathsf{ek}, \mathsf{vk}) \leftarrow \mathsf{GenAcc}(1^\lambda, t); (y^*, wit^*_y, \mathcal{X}^*) \leftarrow \mathcal{A}(1^\lambda, \mathsf{ek}, \mathsf{vk}); \\ acc_{\mathcal{X}^*} \leftarrow \mathsf{EvalAcc}(\mathsf{ek}, \mathcal{X}^*) : \mathsf{VerWit}(\mathsf{vk}, acc_{\mathcal{X}^*}, y^*, wit^*_y) = 1 \wedge y^* \in \mathcal{X}^*\end{array}\right] \leq \mathsf{negl}(\lambda)$$

A.1 Dynamic Strong RSA Accumulators

We formally define Dynamic Strong RSA Accumulators [Bd94, BP97, CL02] described in section 4.1. It has domain $\mathbb{X} = \mathsf{Primes}$.

$\mathsf{Gen}(1^\lambda, \infty) \to (\mathsf{sk}, \mathsf{ek}, \mathsf{vk})$ samples an RSA modulus $(N, (q_1, q_2)) \leftarrow \mathsf{GenSRSAmod}(1^\lambda)$ and a generator $F \leftarrow_\$ \mathbb{Z}^*_N$ and computes a quadratic residue $G \leftarrow F^2 \pmod N$.
Return $(\mathsf{sk}, \mathsf{ek}, \mathsf{vk}) \leftarrow ((q_1, q_2), (N, G), (N, G))$

$\mathsf{Eval}(\mathsf{ek}, \mathcal{X}) \to (acc_{\mathcal{X}}, \mathsf{aux})$ parses $\mathsf{ek} := (N, G)$. If $\mathcal{X} \nsubseteq \mathsf{Primes}$ return $\perp$, otherwise computes $\mathsf{prod}_{\mathcal{X}} := \prod_{x_i \in \mathcal{X}} x_i$ and
Return $(acc_{\mathcal{X}}, \mathsf{aux}) \leftarrow (G^{\mathsf{prod}_{\mathcal{X}}} \mod N, \mathcal{X})$

$\mathsf{Witness}(\mathsf{ek}, x, \mathsf{aux}) \to wit_x$ parses $\mathsf{ek} := (N, G)$, $\mathcal{X} := \mathsf{aux}$ and computes $\mathsf{prod}_{\mathcal{X}\setminus\{x\}} := \prod_{x_i \in \mathcal{X}\setminus\{x\}} x_i$
Return $wit_x \leftarrow G^{\mathsf{prod}_{\mathcal{X}\setminus\{x\}}} \mod N$

$\mathsf{VerWit}(\mathsf{vk}, acc_{\mathcal{X}}, x, w) \to b$ parses $\mathsf{vk} := (N, G)$
Return $b \leftarrow (w^x = acc_{\mathcal{X}} \pmod N)$

Security of strong RSA Accumulator and Batch-Verification Collision Freeness of the above Accumulator comes directly from strong RSA assumption. What is more interesting is that the same scheme allows for many memberships to be verified at the same time, what is called batch-verification. That is, given $x_1, \ldots, x_m \subseteq \mathsf{Primes}$ one can compute a batch-witness $W = G^{\mathsf{prod}_{\mathcal{X}\setminus\{x_1,\ldots,x_m\}}}$ and the verification will be $b \leftarrow (W^{x_1 \ldots x_m} = acc_{\mathcal{X}})$. Again the security of the batch-verification comes from strong RSA assumption and it allows us argue that for any W, x if $W^x = acc_{\mathcal{X}} := G^{\mathsf{prod}_{\mathcal{X}}}$ then $x \in \Pi_{\mathcal{X}}$, meaning that x is a product of primes of the set $\mathcal{X}$.

B Generic CP-SNARK for set membership from accumulators with proof of knowledge

We show here that any accumulator Acc scheme together with a zero knowledge proof of knowledge that a committed value is accumulated, with a commitment scheme Com, can generically construct a CP-SNARK for set membership. Let $\mathsf{CP}_{\mathsf{AccWit}}$ be a zero knowledge proof for the relation $R_{\mathsf{AccWit}}((Acc, c_u), (wit, u, o)) = 1$ iff $\mathsf{VerCommit}(\mathsf{ck}, c_u, u, o) = 1 \wedge \mathsf{VerWit}(\mathsf{vk}, Acc, u, wit) = 1$. Consider a type commitment scheme that takes one type for sets that can be accumulated by Acc and one for elements of the domain of Com. So it is the canonical composition of $\mathsf{Com}_{\mathsf{Acc}} \bullet \mathsf{Com}$, where $\mathsf{Com}_{\mathsf{Acc}}$ is decribed in figure 20.

$\mathsf{Setup}((1^\lambda, t))$: computes $(\mathsf{sk}, \mathsf{ek}, \mathsf{vk}) \leftarrow \mathsf{Acc.Gen}(1^\lambda, t)$ and returns $\mathsf{ck} := (\mathsf{ek}, \mathsf{vk})$.
$\mathsf{Commit}(\mathsf{ck}, \mathsf{t}_U, U)$: parses $\mathsf{ck} := (\mathsf{ek}, \mathsf{vk})$, computes $(Acc, \mathsf{aux}) \leftarrow \mathsf{Eval}(\mathsf{ek}, U)$ and returns $(c, o) := (Acc, \varnothing)$.
$\mathsf{VerCommit}(\mathsf{ck}, \mathsf{t}_U, c, U, \varnothing)$: parses $\mathsf{ck} := (\mathsf{ek}, \mathsf{vk})$, computes $(Acc, \mathsf{aux}) \leftarrow \mathsf{Eval}(\mathsf{ek}, U)$ and return 1 iff $c = Acc$.

Fig. 20: $\mathsf{Com}_{\mathsf{Acc}}$

Finally the generic CP-SNARK can be seen in figure 21.

$\mathsf{KeyGen}(\mathsf{ck}, R^{\in})$: Generate the $\mathsf{crs}_{\mathsf{AccWit}} \leftarrow\$ \mathsf{CP}_{\mathsf{AccWit}}.\mathsf{KeyGen}(R_{\mathsf{AccWit}})$
Return $\mathsf{crs} := \mathsf{crs}_{\mathsf{AccWit}}$.
$\mathsf{Prove}(\mathsf{crs}, (c_U, c_u), (U, u), (\varnothing, o))$: parse $\mathsf{ck} := ((\mathsf{ek}_{\mathsf{Acc}}, \mathsf{vk}_{\mathsf{Acc}}), \mathsf{ck}_{\mathsf{Com}})$ and compute $wit_u \leftarrow \mathsf{Acc.Witness}(\mathsf{ek}, u, U)$.
Then compute $\pi_{\mathsf{AccWit}} \leftarrow \mathsf{CP}_{\mathsf{AccWit}}.\mathsf{Prove}(\mathsf{crs}, (Acc, c_u), (wit, u, o))$
Return $\pi := \pi_{\mathsf{AccWit}}$
$\mathsf{VerProof}(\mathsf{crs}, (c_U, c_u), \pi)$: Return 1 iff $\mathsf{CP}_{\mathsf{AccWit}}.\mathsf{VerProof}(\mathsf{crs}_{\mathsf{AccWit}}, (c_e, c_U), \pi_{\mathsf{Root}}) = 1$.

Fig. 21: $\mathsf{MemCP}_{\mathsf{Acc}}$ CP-SNARK for set membership

Theorem B.1. *Let* Com *be a computationally binding commitment scheme,* Acc *a sound Accumulator scheme and* $\mathsf{CP}_{\mathsf{AccWit}}$ *be a knowledge sound proof then* $\mathsf{MemCP}_{\mathsf{Acc}}$ *is a knowledge-sound with partial opening of the set commitments* c_U *for the* R_{mem} *relation and the* $\mathsf{Com}_{\mathsf{Acc}}$ *commitment*

scheme. Furthermore, if Com*,is statistically hiding commitments and* $\mathsf{CP}_{\mathsf{AccWit}}$ *is zero-knowledge, then* $\mathsf{MemCP}_{\mathsf{Acc}}$ *is zero-knowledge.*

C Vector Commitments

A vector commitment (VC) [LY10, CF13] is a primitive that allows one to commit to a vector $\boldsymbol{v}$ of length n in such a way that it can later open the commitment at any position $i \in [n]$. In terms of security, a VC should be *position binding* in the sense that it is not possible to open a commitment to two different values at the same position. Also, what makes VC an interesting primitive is *conciseness*, which requires commitment and openings to be of fixed size, independent of the vector's length. Furthermore, a vector commitment can also support updates, meaning that updates in the underlying vector allow efficient updates of the commitment and the opening proofs. We note that in this case position binding should also hold with respect to updates.

C.1 Definition

We follow the definition of a Vector Commitment Scheme and its security with respect to updates as defined in [CPZ18].

Definition C.1. *A Vector Commitment Scheme is tuple of PPT algorithms,* $\mathsf{VC} = (\mathsf{KeyGen}, \mathsf{Com}, \mathsf{Prove}, \mathsf{Ver}, \mathsf{Updat}$

$\mathsf{KeyGen}(1^\lambda, n) \to (\mathsf{prk}, \mathsf{vrk}, \mathsf{upk}_0, \ldots, \mathsf{upk}_{n-1})$: *given the security parameter* λ *and the size* n *of the committed vector it outputs a prover key* prk*, a verifier key* vrk *and update keys* $\mathsf{upk}_0, \ldots, \mathsf{upk}_{n-1}$*.*

$\mathsf{Com}(\mathsf{prk}, a_0, \ldots, a_{n-1}) \to \mathsf{dig}_{\boldsymbol{a}}$: *given prover key* prk *and vector* $\boldsymbol{a} = (a_0, ..., a_{n-1})$*, it outputs a digest* $\mathsf{dig}_{\boldsymbol{a}}$ *of vector* $\boldsymbol{a}$*.*

$\mathsf{Prove}(\mathsf{prk}, i, \boldsymbol{a}) \to (a_i, \pi_i)$: *given prover key* prk*, a vector* $\boldsymbol{a} = (a_0, ..., a_{n-1})$ *and an index* i*, it outputs the element* a_i *in the* i*-th position of the vector and a proof* π_i*.*

$\mathsf{Ver}(\mathsf{vrk}, \mathsf{dig}, i, a, \pi) \to b$: *given the verifier key* prk*, a digest* dig*, an index* i*, a value* a *and a proof* π *it outputs* 1 *iff* π *is a valid proof that* a *is in the* i*-th position of the vector that is committed in* dig*.*

$\mathsf{UpdateCom}(\mathsf{dig}, i, \delta, \mathsf{upk}_i) \to \mathsf{dig}'$: *given a digest* dig*, an index* i*, an update* δ *and an update key of* i*-th position it outputs an updated digest* dig' *of a vector the same as before but with value* $a + \delta$ *(instead of* a*) in the* i*-th position.*

$\mathsf{UpdateProof}(\pi, i, \delta, \mathsf{upk}_i) \to \pi'$: *given a digest* dig*, an index* i*, an update* δ *and an update key of* i*-th position it outputs an updated proof that* $a + \delta$ *(instead of* a*) is in the* i*-th position of the vector.*

Soundness A Vector Commitment Scheme VC is sound if for all PPT adversaries $\mathcal{A}$ the below probability is $\mathsf{negl}(\lambda)$

$$\Pr\left[\begin{array}{l}\mathsf{Ver}(\mathsf{vrk},\mathsf{dig},i,a,\pi)=1\\ \wedge\, a\neq a_i\end{array} : \begin{array}{r}(n,\mathsf{state})\leftarrow\mathcal{A}\\ (\mathsf{prk},\mathsf{vrk},\mathsf{upk}_0,\ldots,\mathsf{upk}_{n-1})\leftarrow\mathsf{KeyGen}(1^\lambda,n)\\ \boldsymbol{a}\leftarrow\mathcal{A}\\ \mathsf{dig}\leftarrow\mathsf{Com}(\mathsf{prk},\boldsymbol{a})\\ \textbf{for } k=1,\ldots,t=\mathsf{poly}(\lambda)\\ (j,\delta)\leftarrow\mathcal{A}\\ \mathsf{dig}\leftarrow\mathsf{UpdateCom}(\mathsf{dig},j,\delta,\mathsf{upk}_j)\\ \textbf{endfor}\\ (i,a,\pi)\leftarrow\mathcal{A}\end{array}\right]=\mathsf{negl}(\lambda)$$

C.2 EDRAX - A Vector Commitment from multilinear extensions

Multilinear Extension of vectors Let $\mathbb{F}$ be a field and $n=2^\ell$. Multilinear Extension of a vector a $\boldsymbol{a}=(a_0,\ldots,a_{n-1})$ in $\mathbb{F}$ is a polynomial $f_{\boldsymbol{a}}:\mathbb{F}^\ell\to\mathbb{F}$ with variables $x_1,\ldots,x_\ell$

$$f_{\boldsymbol{a}}(x_1,\ldots,x_\ell)=\sum_{i=0}^{n-1}a_i\cdot\prod_{k=1}^{\ell}\mathsf{select}_{i_k}(x_k)$$

where $i_\ell i_{\ell-1}\ldots i_2 i_1$ is the bit representation of i and $\mathsf{select}_{i_k}(x_k)=\begin{cases}x_k, & \text{if } i_k=1\\ 1-x_k, & \text{if } i_k=0\end{cases}$

A property of Multilinear extension of $\boldsymbol{a}$ is that $f_{\boldsymbol{a}}(i_1,\ldots,i_\ell)=a_i$ for each $i\in[n]$.

Vector Commitment Scheme We describe the EDRAX Vector Commitment:

Definition C.2. *Let a bilinear group* $\mathsf{bp}=(q,g,\mathbb{G}_1,\mathbb{G}_T,e)\leftarrow\mathcal{RG}(1^\lambda)$ *generated by a group generator. Let* $n=2^\ell$ *be the length of the vector and* $2^{[\ell]}$ *be the powerset of* $[\ell]=\{1,\ldots,\ell\}$

$\mathsf{KeyGen}(1^\lambda,n)\to(\mathsf{prk},\mathsf{vrk},\mathsf{upk}_0,\ldots,\mathsf{upk}_{n-1})$: *samples random* $s_1,\ldots,s_\ell\leftarrow_\$\mathbb{F}$ *and computes* $\mathsf{prk}\leftarrow\left\{g^{\prod_{i\in S}s_i}:S\in 2^{[\ell]}\right\}$ *and* $\mathsf{vrk}\leftarrow\{g^{s_1},\ldots,g^{s_\ell}\}$. *For each* $i=0,\ldots,n-1$ *computes the update key* $\mathsf{upk}_i\leftarrow\left\{g^{\prod_{k=1}^{t}\mathsf{select}_{i_k}(s_k)}:t=1,\ldots,\ell\right\}:=\{\mathsf{upk}_{i,t}:t=1,\ldots,\ell\}$.

$\mathsf{Com}(\mathsf{prk},a_0,\ldots,a_{n-1})\to\mathsf{dig}_{\boldsymbol{a}}$: *let* $\boldsymbol{a}:=(a_0,\ldots,a_{n-1})$. *Computes* $\mathsf{dig}_{\boldsymbol{a}}\leftarrow g^{f_{\boldsymbol{a}}(s_1,\ldots,s_\ell)}$ *where* $f_{\boldsymbol{a}}$ *is the multilinear extension of vector* $\boldsymbol{a}$ *as described above.*

$\mathsf{Prove}(\mathsf{prk},i,\boldsymbol{a})\to(a_i,\pi_i)$: *let* $\boldsymbol{x}=(x_1,\ldots,x_\ell)$ *be an* ℓ*-variable. Compute* $q_1,\ldots,q_\ell$ *such that* $f_{\boldsymbol{a}}(\boldsymbol{x})-f_{\boldsymbol{a}}(i_1,\ldots,i_\ell)=\sum_{k=1}^{\ell}(x_k-i_k)q_k(\boldsymbol{x})$ *and* $\pi_i\leftarrow\left\{g^{q_1(\boldsymbol{s})},\ldots,g^{q_\ell(\boldsymbol{s})}\right\}$ *(where* $g^{q_i(\boldsymbol{s})}$ *is evaluated by using* $\mathsf{prk}:=\left\{g^{\prod_{i\in S}s_i}:S\in 2^{[\ell]}\right\}$ *without* $\boldsymbol{s}$*).*

$\mathsf{Ver}(\mathsf{vrk},\mathsf{dig},i,a,\pi)\to b$: *parse* $\pi:=(w_1,\ldots,w_\ell)$ *and outputs* 1 *iff* $e(\mathsf{dig}/g^a,g)=\prod_{k=1}^{\ell}e(g^{s_k-i_k},w_k)$

$\mathsf{UpdateCom}(\mathsf{dig},i,\delta,\mathsf{upk}_i)\to\mathsf{dig}'$: *computes* $dig'\leftarrow dig\cdot\left[g^{\prod_{k=1}^{\ell}\mathsf{select}_{i_k}(s_k)}\right]^\delta:=\mathsf{dig}\cdot\left[\mathsf{upk}_{i,\ell}\right]^\delta=g^{(a_i+\delta)\cdot\prod_{k=1}^{\ell}\mathsf{select}_{i_k}(s_k)+\sum_{j=0,j\neq i}^{n-1}a_j\cdot\prod_{k=1}^{\ell}\mathsf{select}_{j_k}(s_k)}$

$\mathsf{UpdateCom}(\pi,i,a',\mathsf{upk}_i)\to\pi'$: *Parses* $\pi:=(w_1,\ldots,w_\ell)$ *and computes* $w'_k\leftarrow w_k\cdot g^{\Delta_i(\boldsymbol{s})}$ *for each* $k=1,\ldots,\ell$*, where* $\Delta_k(\boldsymbol{x})$ *are the delta polynomials computed by the* DELTAPOLYNOMIALS *algorithm (for more details about the algorithm and its correctness we refer to [CPZ18]).*

The above scheme is proven in [CPZ18] to satisfy the Soundness property under the q-Strong Bilinear Diffie-Hellman assumption.

D Another Instantiations of Protocol for R_{Coprime}

Below we propose another interactive ZK protocol for R_{Coprime}. The difference with the above is that it doesn't have the limitation of $\lambda_s + 1 < \mu$ and $\lambda_s < \log(N)/2$. Also, partial opening of Acc isn't needed. This comes with a cost of 2 more group elements in the proof size, 4 more exponentiations for the prover and 2 more for the verifier.

1. Prover computes $C_a = DH^{r_a}, C_{r_a} = G^{r_a}H^{r'_a}, C_b = G^bH^{\rho_b}, C_B = Acc^bH^{\rho_B}, C_{\rho_B} = G^{\rho_B}H^{\rho'_B}$ and sends to the verifier:
   $\underline{\mathcal{P} \to \mathcal{V}} : C_a, C_b, C_{r_a}, C_B, C_{\rho_B}$
2. Prover and Verifier perform a protocol for the relation:
   $R((\mathsf{Acc}, C_e, C_a, C_{r_a}, C_b, C_B, C_{\rho_B}), (e, r, r_a, r'_a, b, \rho_b, \rho_B, \rho'_B, D, B, \beta, \delta)) = 1$ iff

$$C_b = G^bH^{\rho_b} \wedge C_B = Acc^bH^{\rho_B} \wedge C_e = G^eH^r \wedge C_{r_a} = G^{r_a}H^{r'_a}$$

$$\wedge C_{\rho_B} = G^{\rho_B}H^{\rho'_B} \wedge C_a^eC_B = GH^\beta \wedge C_{r_a}^eC_{\rho_B} = G^\beta H^\delta$$

   Let λ_s be the size of the challenge space, λ_z be the statistical security parameter and μ the size of e.
   - Prover samples:

$$r_b, r_e \leftarrow\!\!\$ \left(-2^{\lambda_z+\lambda_s+\mu}, 2^{\lambda_z+\lambda_s+\mu}\right)$$
$$r_{\rho_b}, r_{\rho_B}, r_r, r_{r_a}, r_{r'_a}, r_{\rho'_B} \leftarrow\!\!\$ \left(-\lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s}, \lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s}\right)$$
$$r_\beta, r_\delta \leftarrow\!\!\$ \left(-\lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s+\mu}, \lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s+\mu}\right)$$

     and computes:

$$\alpha_1 = G^{r_b}H^{r_{\rho_b}}, \quad \alpha_2 = \mathsf{Acc}^{r_b}H^{r_{\rho_B}}, \quad \alpha_3 = G^{r_e}H^{r_r}, \quad \alpha_4 = G^{r_{r_a}}H^{r_{r'_a}},$$
$$\alpha_5 = C_a^{r_e}H^{r_\beta}, \quad \alpha_6 = C_{r_a}^{r_e}G^{r_\beta}H^{r_\delta}, \quad \alpha_7 = G^{r_{\rho_B}}H^{r_{\rho'_B}}$$

     $\underline{\mathcal{P} \to \mathcal{V}} : (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7)$
   - Verifier samples the challenge $c \leftarrow \{0,1\}^{\lambda_s}$
     $\underline{\mathcal{V} \to \mathcal{P}} : c$
   - Prover computes the response:

     $s_b = r_b - cb, \quad s_e = r_e - ce$
     $s_{\rho_b} = r_{\rho_b} - c\rho_b, s_{\rho_B} = r_{\rho_B} - c\rho_B, s_r = r_r - cr, s_{r_a} = r_{r_a} - cr_a, s_{r'_a} = r_{r'_a} - cr'_a, s_{\rho'_B} = r_{\rho'_B} - c\rho'_B$
     $s_\beta = r_\beta + c(er_a + \rho_B), \quad s_\delta = r_\delta + c(er'_a + \rho'_B)$

     $\underline{\mathcal{P} \to \mathcal{V}} : (s_b, s_e, s_{\rho_b}, s_{\rho_B}, s_r, s_{r_a}, s_{r'_a}, s_{\rho'_B}, s_\beta, s_\delta)$

– Verifier checks if:

$$\alpha_1 \stackrel{?}{=} C_b^c G^{s_b} H^{s_{\rho_b}}, \quad \alpha_2 \stackrel{?}{=} C_B^c \mathsf{Acc}^{s_b} H^{s_{\rho_B}}, \quad \alpha_3 \stackrel{?}{=} C_e^c G^{s_e} H^{s_r}, \quad \alpha_4 \stackrel{?}{=} C_{r_a}^c G^{s_{r_a}} H^{s_{r'_a}},$$

$$\alpha_5 \stackrel{?}{=} C_a^{s_e} H^{s_\beta} G^c C_B^{-c}, \quad \alpha_6 \stackrel{?}{=} C_{r_a}^{s_e} H^{s_\delta} G^{s_\beta} C_{\rho_B}^{-c}, \quad \alpha_7 \stackrel{?}{=} C_{\rho_B}^c G^{s_{\rho_B}} H^{s_{\rho'_B}},$$

$$s_e \stackrel{?}{\in} \left[-2^{\lambda_z+\lambda_s+\mu+1}, 2^{\lambda_z+\lambda_s+\mu+1}\right]$$

Coprime2 protocol

On common reference string $\mathsf{crs} = (\mathbb{Z}_N^*, G, H)$

$\mathsf{Prove}(\mathsf{crs}, (C_e, \mathsf{Acc}), (e, r, (D, b)))$:

1. samples $r_a, r_{a'}, \rho_b, \rho_B, \rho_{B'} \leftarrow\$ (-\lfloor N/4 \rfloor, \lfloor N/4 \rfloor)$ and computes $C_a = DH^{r_a}, C_{r_a} = G^{r_a} H^{r'_a}, C_b = G^b H^{\rho_b}, C_B = Acc^b H^{\rho_B}, C_{\rho_B} = G^{\rho_B} H^{\rho'_B}$.
2. Computes the non-interactive version of the above protocol
   $r_b, r_e \leftarrow\$ \left(-2^{\lambda_z+\lambda_s+\mu}, 2^{\lambda_z+\lambda_s+\mu}\right)$
   $r_{\rho_b}, r_{\rho_B}, r_r, r_{r_a}, r_{r'_a}, r_{\rho'_B} \leftarrow\$ \left(-\lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s}, \lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s}\right)$
   $r_\beta, r_\delta \leftarrow\$ \left(-\lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s+\mu}, \lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s+\mu}\right)$
   $\alpha_1 = G^{r_b} H^{r_{\rho_b}}, \alpha_2 = \mathsf{Acc}^{r_b} H^{r_{\rho_B}}, \alpha_3 = G^{r_e} H^{r_r}, \alpha_4 = G^{r_{r_a}} H^{r_{r'_a}}, \alpha_5 = C_a^{r_e} H^{r_\beta}, \alpha_6 = C_{r_a}^{r_e} G^{r_\beta} H^{r_\delta}, \alpha_7 = G^{r_{\rho_B}} H^{r_{\rho'_B}}$
   $c \leftarrow \mathsf{H}(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7, C_e, Acc)$
   $s_b = r_b - cb, s_e = r_e - ce, s_{\rho_b} = r_{\rho_b} - c\rho_b, s_{\rho_B} = r_{\rho_B} - c\rho_B, s_r = r_r - cr, s_{r_a} = r_{r_a} - cr_a, s_{r'_a} = r_{r'_a} - cr_{r_a}, s_{\rho'_B} = r_{\rho'_B} - c\rho'_B, s_\beta = r_\beta + c(er_a + \rho_B), s_\delta = r_\delta + c(er'_a + \rho'_B)$

   Returns $\pi \leftarrow (C_a, C_{r_a}, C_b, C_B, C_{\rho_B}, \alpha_1, \alpha_2, \alpha_3, \alpha_4, cs_b, s_e, s_{\rho_b}, s_{\rho_B}, s_r, s_{r_a}, s_{r'_a}, s_{\rho'_B}, s_\beta, s_\delta)$

$\mathsf{VerProof}(\mathsf{crs}, (C_e, Acc), \pi)$: returns 1 iff $\alpha_1 = C_b^c G^{s_b} H^{s_{\rho_b}} \wedge \alpha_2 = C_B^c \mathsf{Acc}^{s_b} H^{s_{\rho_B}} \wedge \alpha_3 = C_e^c G^{s_e} H^{s_r} \wedge \alpha_4 = C_{r_a}^c G^{s_{r_a}} H^{s_{r'_a}} \wedge \alpha_5 = C_a^{s_e} H^{s_\beta} G^c C_B^{-c} \wedge \alpha_6 = C_{r_a}^{s_e} H^{s_\delta} G^{s_\beta} C_{\rho_B}^{-c} \wedge \alpha_7 = C_{\rho_B}^c G^{s_{\rho_B}} H^{s_{\rho'_B}} \wedge \wedge s_e \in \left[-2^{\lambda+\lambda_s+\mu+1}, 2^{\lambda+\lambda_s+\mu+1}\right]$

Fig. 22

Correctness. Here we show the correctness of the protocol.

$$\begin{aligned}
\alpha_1 &= G^{r_b} H^{r_{\rho_b}} = G^{s_b+cb} H^{s_{\rho_b}+cr_b} = G^{s_b} H^{s_{\rho_b}} (G^b H^{\rho_b})^c \\
&= G^{s_b} H^{s_{\rho_b}} C_b^c \\
\alpha_2 &= \mathsf{Acc}^{r_b} H^{r_{\rho_B}} = \mathsf{Acc}^{s_b+cb} H^{s_{\rho_B}+c\rho_B} = \mathsf{Acc}^{s_{r_b}} H^{s_{\rho_B}} (\mathsf{Acc}^b H^{\rho_B})^c \\
&= \mathsf{Acc}^{s_b} H^{s_{\rho_B}} C_B^c \\
\alpha_3 &= G^{r_e} H^{r_r} = G^{s_e+ce} H^{s_r+cr} = G^{s_e} H^{s_r} (G^e H^r)^c \\
&= G^{s_e} H^{s_r} C_e^c \\
\alpha_4 &= G^{r_{r_a}} H^{r_{r'_a}} = G^{s_{r_a}+cr_a} H^{s_{r'_a}+cr'_a} = G^{s_{r_a}} H^{s_{r'_a}} (G^{r_a} H^{r'_a})^c \\
&= G^{s_{r_a}} H^{s_{r'_a}} C_{r_a}^c \\
\alpha_5 &= C_a^{r_e} H^{r_\beta} = C_a^{s_e+ce} H^{s_\beta - c(er_a+\rho_B)} = C_a^{s_e} H^{s_\beta} (D^e H^{er_a})^c H^{-c(er_a+\rho_B)} \\
&= C_a^{s_e} H^{s_\beta} (D^e H^{-\rho_B})^c = C_a^{s_e} H^{s_\beta} (GAcc^{-b} H^{-\rho_B})^c = \\
&= C_a^{s_e} H^{s_\beta} G^c C_B^{-c} \\
\alpha_6 &= C_{r_a}^{r_e} G^{r_\beta} H^{r_\delta} = C_{r_a}^{s_e+ce} G^{s_\beta - c(er_a+\rho_B)} H^{s_\delta - c(er'_a+\rho'_B)} \\
&= C_{r_a}^{s_e} G^{s_\beta} H^{s_\delta} (G^{r_a} H^{r'_a})^{ce} G^{-c(er_a+\rho_B)} H^{-c(er'_a+\rho'_B)} = C_{r_a}^{s_e} G^{s_\beta} H^{s_\delta} G^{-c\rho_B} H^{-c\rho'_B} \\
&= C_{r_a}^{s_e} G^{s_\beta} H^{s_\delta} C_{\rho_B}^{-c} \\
\alpha_7 &= G^{r_{\rho_B}} H^{r_{\rho'_B}} = G^{s_{\rho_B}+c\rho_B} H^{s_{\rho'_B}+c\rho'_B} = G^{s_{\rho_B}} H^{s_{\rho'_B}} (G^{\rho_B} H^{\rho'_B})^c \\
&= G^{s_{\rho_B}} H^{s_{\rho'_B}} C_{\rho_B}^c
\end{aligned}$$

Security.

Theorem D.1. *Let $\mathbb{Z}_N^*$ be an RSA group where strong-RSA assumption holds, then the above protocol is an honest-verifier zero knowledge and knowledge sound protocol for R_{Coprime}.*

Proof Zero-Knowledge can be proven with standard techniques, similar to the ones in the proof of theorem 4.6 and is therefore omitted.

For the knowledge soundness, let an adversary of the knowledge soundness $\mathcal{A}$ that is able to convince the verifier $\mathcal{V}$ with a probability at least ϵ. We will construct an extractor $\mathcal{E}$ that extracts the witness $(e, r, r_2, r_3, \beta, \delta)$. Using rewinding $\mathcal{E}$ gets two accepted transcripts

$$(C_a, C_b, C_{r_a}, C_B, C_{\rho_B}, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7, c, s_b, s_e, s_{\rho_b}, s_{\rho_B}, s_r, s_{r_a}, s_{r'_a}, s_{\rho'_B}, s_\beta, s_\delta)$$

$$(C_a, C_b, C_{r_a}, C_B, C_{\rho_B}, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7, c', s'_b, s'_e, s'_{\rho_b}, s'_{\rho_B}, s'_r, s'_{r_a}, s'_{r'_a}, s'_{\rho'_B}, s'_\beta, s'_\delta)$$

on two different challenges c and c'. $\mathcal{E}$ aborts if it cannot get two such transcripts ($\mathsf{abort1}$).

We denote $\Delta c := c' - c, \Delta s_b := s_b - s'_b, \Delta s_e := s_e - s'_e, \Delta s_{\rho_b} := s_{\rho_b} - s'_{\rho_b}, \Delta s_{\rho_B} := s_{\rho_B} - s'_{\rho_B}, \Delta s_r := s_r - s'_r, \Delta s_{r_a} := s_{r_a} - s'_{r_a}, \Delta s_{r'_a} := s_{r'_a} - s'_{r'_a}, \Delta s_{\rho'_B} := s_{\rho'_B} - s'_{\rho'_B}, \Delta s_\beta := s_\beta - s'_\beta, \Delta s_\delta := s_\delta - s'_\delta$ then

$$C_b^{\Delta c} = G^{\Delta s_b} H^{\Delta s_{\rho_b}} \Rightarrow C_b = \pm G^{\hat{b}} H^{\hat{\rho_b}} \tag{9}$$

$$C_B^{\Delta c} = \mathsf{Acc}^{\Delta s_b} H^{\Delta s_{\rho_B}} \Rightarrow C_B = \pm \mathsf{Acc}^{\hat{b}} H^{\hat{\rho_B}} \tag{10}$$

$$C_e^{\Delta c} = G^{\Delta s_e} H^{\Delta s_r} \Rightarrow C_e = \pm G^{\hat{e}} H^{\hat{r}} \tag{11}$$

$$C_{r_a}^{\Delta c} = G^{\Delta s_{r_a}} H^{\Delta s_{r'_a}} \Rightarrow C_{r_a} = \pm G^{\hat{r_a}} H^{\hat{r'_a}} \tag{12}$$

$$1 = C_a^{\Delta s_e} H^{\Delta s_\beta} G^{-\Delta c} C_B^{\Delta c} \tag{13}$$

$$1 = C_{r_a}^{\Delta s_e} H^{\Delta s_\delta} G^{\Delta s_\beta} C_{\rho_B}^{\Delta c} \tag{14}$$

$$C_{\rho_B}^{\Delta c} = G^{\Delta s_{\rho_B}} H^{\Delta s_{\rho'_B}} \Rightarrow C_{\rho_B} = \pm G^{\hat{\rho_B}} H^{\hat{\rho'_B}} \tag{15}$$

define the (possibly rational) numbers $\hat{b} := \frac{\Delta s_b}{\Delta c}$, $\hat{\rho_b} := \frac{\Delta s_{\rho_b}}{\Delta c}$, $\hat{e} := \frac{\Delta s_e}{\Delta c}$, $\hat{r} := \frac{\Delta s_r}{\Delta c}$, $\hat{r_a} := \frac{\Delta s_{r_a}}{\Delta c}$, $\hat{r'_a} := \frac{\Delta s_{r'_a}}{\Delta c}$, $\hat{\rho_B} := \frac{\Delta s_{\rho_B}}{\Delta c}$, $\hat{\rho'_B} := \frac{\Delta s_{\rho'_B}}{\Delta c}$.

$\mathcal{E}$ aborts in case Δc doesn't divide: Δs_b and Δs_{ρ_b}(abort $2a$), Δs_e and Δs_r(abort $2b$), Δs_{r_a} and $\Delta s_{r'_a}$(abort $2c$), Δs_{ρ_B} and $\Delta s_{\rho'_B}$(abort $2d$). And finally, $\mathcal{E}$ aborts if Δc doesn't divide Δs_{ρ_B} (abort $2e$). Therefore, after these aborts didn't happen we can infer the equivalent equalities on the right of equations 9,11,12,15 and 10.

If we replace equations 12 and 15 in equation 14 we get $1 = \left(\pm G^{\hat{r_a}} H^{\hat{r'_a}}\right)^{\Delta s_e} H^{\Delta s_\beta} G^{\Delta s_\beta} \left(\pm G^{\hat{\rho_B}} H^{\hat{\rho'_B}}\right)^{\Delta c}$ or $1 = (\pm 1)^{\Delta s_e} (\pm 1)^{\Delta c} G^{\hat{r_a}\Delta s_e + \hat{\rho_B}\Delta c + \Delta s_\beta} H^{\hat{r'_a}\Delta s_e + \hat{\rho'_B}\Delta c + \Delta s_\beta}$. Since $G, H, 1$ are quadratic residues then $(\pm 1)^{\Delta s_e}(\pm 1)^{\Delta c} = 1$, hence $1 = G^{\hat{r_a}\Delta s_e + \hat{\rho_B}\Delta c + \Delta s_\beta} H^{\hat{r'_a}\Delta s_e + \hat{\rho'_B}\Delta c + \Delta s_\beta}$. Then under the DLOG assumption $\hat{r_a}\Delta s_e + \hat{\rho_B}\Delta c + \Delta s_\beta = 0 = \hat{r'_a}\Delta s_e + \hat{\rho'_B}\Delta c + \Delta s_\beta$, which gives us that

$$\Delta s_\beta = -\hat{r_a}\Delta s_e - \hat{\rho_B}\Delta c \tag{16}$$

Finally, we replace equations 10 and 16 in equation 13 we get $1 = C_a^{\Delta s_e} H^{-\hat{r_a}\Delta s_e - \hat{\rho_B}\Delta c} G^{-\Delta c} \left(\pm \mathsf{Acc}^{\hat{b}} H^{\hat{\rho_B}}\right)^{\Delta c}$ or $1 = (\pm 1)^{\Delta c} C_a^{\Delta s_e} \mathsf{Acc}^{\hat{b}\Delta c} G^{-\Delta c} H^{-\hat{r_a}\Delta s_e}$ or $\left(\pm \mathsf{Acc}^{\hat{b}} G^{-1}\right)^{\Delta c} = \left(C_a^{-1} H^{r_a}\right)^{\Delta s_e}$. But as noted above Δc divides Δs_e so $\pm \mathsf{Acc}^{\hat{b}} G^{-1} = \pm \left(C_a^{-1} H^{r_a}\right)^{\hat{e}} \Rightarrow \mathsf{Acc}^{\hat{b}} G^{-1} = \pm \left(C_a^{-1} H^{\hat{r_a}}\right)^{\hat{e}} \Rightarrow \left(\frac{C_a}{H^{\hat{r_a}}}\right)^{\hat{e}} \mathsf{Acc}^{\hat{b}} = \pm G$. We discern two cases:

- $\underline{\left(\frac{C_a}{H^{\hat{r_a}}}\right)^{\hat{e}} \mathsf{Acc}^{\hat{b}} = +G}$: Then $\mathcal{E}$ sets $\tilde{D} \leftarrow \frac{C_a}{H^{\hat{r_a}}}$, $\tilde{e} \leftarrow \hat{e} := \frac{\Delta s_e}{\Delta c}$, $\tilde{r} \leftarrow \hat{r} := \frac{\Delta s_r}{\Delta c}$ and $\tilde{b} \leftarrow \hat{b} := \frac{\Delta s_b}{\Delta c}$
- $\underline{\left(\frac{C_a}{H^{\hat{r_a}}}\right)^{\hat{e}} \mathsf{Acc}^{\hat{b}} = -G}$: Then $\hat{e}$ should be odd otherwise if $\hat{e} = 2\rho$ then $G = -\left(\frac{C_a}{H^{\hat{r_a}}}\right)^{2\rho} \mathsf{Acc}^{\hat{b}}$ would be a non-quadratic residue. So $\mathcal{E}$ sets $\tilde{D} \leftarrow -\frac{C_a}{H^{\hat{r_a}}}$, $\tilde{e} \leftarrow \hat{e} := \frac{\Delta s_e}{\Delta c}$, $\tilde{r} \leftarrow \hat{r} := \frac{\Delta s_r}{\Delta c}$ and $\tilde{b} \leftarrow \hat{b} := \frac{\Delta s_b}{\Delta c}$. It is clear that $\tilde{D}^{\tilde{e}} \mathsf{Acc}^{\tilde{b}} = G$.

Finally the $\mathcal{E}$ outputs $(\tilde{e}, \tilde{r}, \tilde{D}, \tilde{b})$.

Now we show that the probability the extractor terminates with outputting a valid witness is $O(\epsilon)$. If the extractor does not abort then it clearly outputs a valid witness (under factoring assumption). For the first abort, with a standard argument it can be shown that the extractor is able to extract two accepting transcripts with probability $O(\epsilon)$ (for the probabilistic analysis we refer to [DF02]). Thus $Pr[\mathsf{abort1}] = 1 - O(\epsilon)$. For the aborts abort $2a$, abort $2b$, abort $2c$ and abort $2d$ they happen with negligible probability ($\leq \frac{2}{1-2^{-\lambda_s+1}} Pr[\mathcal{B}$ solves $sRSA]$ each, for any PPT adversary $\mathcal{B}$) under the strong RSA assumption according to lemma 4.2. For abort $2e$ we show in the lemma below that in case it happens an adversary can solve the strong RSA problem. Putting them together the probability of success of $\mathcal{E}$ is at least $O(\epsilon) - \left(\frac{8}{1-2^{-\lambda_s+1}} + 1\right) Pr[\mathcal{B}$ solves $sRSA] = O(\epsilon) - \mathsf{negl}(\lambda_s)$.

Lemma D.1. *If Δc divides Δs_b then it also divides $\Delta \rho_B$ under the strong RSA assumption.*

Proof An adversary to the strong RSA assumption receives $H \in \mathsf{QR}_N$ and does the following: set $G = H^\tau$ for $\tau \leftarrow_\$ [0, 2^{\lambda_s} N^2]$ and send (G, H) to the adversary $\mathcal{A}$ which outputs a proof π_{Coprime2}. Then we rewind to get another successful proof π'_{Coprime2} and we use the extractor as above to get $C_B^{\Delta c} = \mathsf{Acc}^{\Delta s_b} H^{\Delta s_{\rho_B}}$.

Assume that $\Delta c \nmid \Delta\rho_B$. Since Δc divides Δs_b then there is a k such that $k \cdot \Delta c = \Delta s_b$. Then $C_B^{\Delta c} = \mathsf{Acc}^{k \cdot \Delta c} H^{\Delta s_{\rho_B}} \Rightarrow \left(C_B \mathsf{Acc}^{-k}\right)^{\Delta c} = H^{\Delta s_{\rho_B}}$. From assumption Δc doesn't divide $\Delta\rho_B$, so $\gcd(\Delta c, \Delta\rho_B) = g$ for a $g \neq \Delta c, \Delta\rho_B$. Hence, there are there are χ, ψ such that $\chi \cdot \Delta c + \psi \cdot \Delta\rho_B = g$. Thus, $H^g = H^{\chi \cdot \Delta c + \psi \cdot \Delta\rho_B} = H^{\chi \Delta c} \left(C_B \mathsf{Acc}^{-k}\right)^{\psi \Delta c} = \left(H^\chi C_B^\psi \mathsf{Acc}^{-\psi k}\right)^{\Delta c}$ so $H = \pm \left(H^\chi C_B^\psi \mathsf{Acc}^{-\psi k}\right)^{\frac{\Delta c}{g}}$. Now since H and Acc are quadratic residues (and so is C_B) we get that $H = \left(H^\chi C_B^\psi \mathsf{Acc}^{-\psi k}\right)^{\frac{\Delta c}{g}}$ and thus $\left(H^\chi C_B^\psi \mathsf{Acc}^{-\psi k}, \frac{\Delta c}{g}\right)$ is a solution to the strong RSA problem. □

By a simple argument identical to the one of section 4.4, we can also conclude about the range of the extracted $\tilde{e}$: $s_e \overset{?}{\in} \left[-2^{\lambda_z + \lambda_s + \mu + 1}, 2^{\lambda_z + \lambda_s + \mu + 1}\right]$ implies $-2^{\lambda_z + \lambda_s + \mu + 2} \leq \hat{e} \leq 2^{\lambda_z + \lambda_s + \mu + 2}$. □

E Instantiation over Hidden Order Groups

In sections 4, 5 we construct zero knowledge protocols for set membership/non-membership, where the sets are committed using an RSA accumulator. The integer commitment scheme IntCom, the RSA accumulator-based commitments to sets $\mathsf{SetCom}_{\mathsf{RSA}}$, $\mathsf{SetCom}_{\mathsf{RSA}'}$, the proof of equality modEq, the argument of knowledge of a root Root and the argument of knowledge of coprime element $\mathsf{Coprime}$ are all working over RSA groups.

Although in our work above we specify the group to be an RSA group, we note that our protocols can also work over any Hidden Order Group. For example Class Groups [BH01] or the recently proposed groups from Hyperelliptic Curves [DG20, Lee20].

Here we describe the (slight) modifications, in the protocols and the assumptions under which they would be secure, that are necessary to switch to (general) Hidden Order Groups.

Let $\mathsf{Ggen}(1^\lambda)$ be a probabilistic algorithm that generates such a group $\mathbb{G}$ with order in a specific range $[\mathsf{ord}_{min}, \mathsf{ord}_{max}]$ such that $\frac{1}{\mathsf{ord}_{min}}, \frac{1}{\mathsf{ord}_{max}}, \frac{1}{\mathsf{ord}_{max} - \mathsf{ord}_{min}} \in \mathsf{negl}(\lambda)$.

The additional assumption that we need to make is that it is hard to find any group element in $\mathbb{G}$ of low (poly-size) order. This is the Low Order Assumption [BBF18b]], which is formally defined below:

Definition E.1 (Low Order Assumption [BBF18b]). *We say that the* low order assumption *holds for a* Ggen *if for any PPT adversary* $\mathcal{A}$*:*

$$\Pr\left[\begin{array}{l} u^\ell = 1 \\ \wedge u \neq 1 \\ \wedge 1 < \ell < 2^{\mathsf{poly}(\lambda)} \end{array} : \begin{array}{l} \mathbb{G} \leftarrow \mathsf{Ggen}(\lambda) \\ (u, \ell) \leftarrow \mathcal{A}(\mathbb{G}) \end{array}\right] = \mathsf{negl}(\lambda)$$

We note that specifically for RSA groups, for Low Order assumption to hold, we have to work in the quotient group $\mathbb{Z}_N^*/\{1, -1\}$ [Wes18], since otherwise -1 would trivially break the assumption. So

$\mathbb{Z}_N^*/\{1,-1\}$ would be an instantiation of a Hidden Order Group where the Low Order assumption holds.

In terms of constructions, one difference regards the upper bound on the order of $\mathbb{G}$ that is used in the protocols. More precisely, throughout the main core of our work we use N as an upper bound for the order of the group $\mathbb{Z}_N^*$ and $N/2$ as an upper bound for the order of the quadratic residues subgroup QR_N. Similarly, in a Hidden Order Group $\mathbb{G}$ generated by Ggen, although the order of the group is unknown, a range in which the order lies is known $[\mathsf{ord}_{min}, \mathsf{ord}_{max}]$. So the maximum order ord_{max} can be used, instead of N, as an upper bound. In many cases these values are used either to securely sample a random value or to bound the size of a value needed for a security proof. For example a random value that is sampled from $\left(-\lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s}, \lfloor N/4 \rfloor 2^{\lambda_z+\lambda_s}\right)$ in the RSA group instantiation will be sampled from $\left(-\frac{\mathsf{ord}_{max}}{2} 2^{\lambda_z+\lambda_s}, \frac{\mathsf{ord}_{max}}{2} 2^{\lambda_z+\lambda_s}\right)$ in the case of hidden order groups.

Here we give other specific changes that need to be made to instantiate our protocols in general hidden order groups. For IntCom, the verification equation becomes $C = G^x H^r$ (without the $\pm$). Then the argument of knowledge of opening of such a commitment would be secure under the strong RSA and low order assumptions. The set commitments $\mathsf{SetCom}_{\mathsf{RSA}}$, $\mathsf{SetCom}_{\mathsf{RSA}'}$ remain the same and are binding under the strong RSA assumption for Ggen (and collision resistance of $\mathsf{H}_{\mathsf{prime}}$ for the case of $\mathsf{SetCom}_{\mathsf{RSA}}$). For modEq, the same difference as for the AoK of an opening of an IntCom commitment is inherited. For Root and $\mathsf{Coprime}$, the proposition 4.1 needs to be slightly modified: $A = B^{\frac{x}{y}}$ can be without $\pm$, and can be proven under the low order assumption instead. Finally, in the proof of security of protocol $\mathsf{Coprime}$, in lemma 5.1 the assumption $\lambda_s < \log(N)/2$ is not needed as long as the low order assumption holds (an adversary that can find $H, \Delta c$ such that $\gcd(\mathsf{ord}(H), q^\ell) = 1$ can be used to break low order assumption).