Succinct Arguments from Subvector Commitments and Linear Map Commitments

Russell W. F. Lai and Giulio Malavolta

Friedrich-Alexander-Universität Erlangen-Nürnberg

Abstract. Succinct non-interactive arguments of knowledge (SNARK) allow a prover to convince a verifier of the validity of certain statements with a single short message. Revisiting the "CS proofs" paradigm [Micali, FOCS 1994], we leverage number-theoretic assumptions to construct arguments with significantly shorter proofs. With typical computational and statistical security parameters $\lambda \approx 100$ and $k \approx 80$ respectively, two main results in the random oracle model follow.

1. There exists a setup-free SNARK with proofs of size $\sim$4300 bits, under the adaptive root assumption over class groups of imaginary quadratic orders. This is the shortest setup-free SNARK at the time of writing.
2. There exists a pre-processing SNARK with proofs consisting of 2 group elements plus $\sim$250 bits (for a total of $\sim$750 bits), under the computational Diffie-Hellman (CDH) assumption over pairing groups. This matches the most efficient scheme in the same setting [Groth, EUROCRYPT 2016] but our verifier has to compute only 2 pairings, as opposed to 3.

The common ground of both constructions is a 4-message protocol that compiles any (possibly linear) probabilistic checkable proof (PCP) into an argument of knowledge. The protocol is made non-interactive using the Fiat-Shamir transform. Our main technical tool is a generalization of vector commitments called subvector commitments (SVC). The latter allows to open a committed vector at a set of positions, where the opening size is independent of the size of the set. We propose two constructions under variants of the root assumption and the CDH assumption respectively. We further generalize SVC to a notion called linear map commitments (LMC), which allows to open a committed vector to its images under linear maps. We propose two constructions under different assumptions over pairing groups. LMC allows one to construct pre-processing SNARKs but with a more efficient prover. Apart from the new application in constructing succinct arguments, SVC and LMC have numerous other applications which may be of independent interest.

1 Introduction

An argument system allows a prover, with a witness w, to convince a verifier that a certain statement x is in an NP language $\mathcal{L}$. In contrast with proof systems, argument systems are only required to be computationally sound. Due to this relaxation, it is possible that the interaction between the prover and the verifier is succinct, *i.e.*, the communication complexity is bounded by some polynomial $\mathsf{poly}(\lambda)$ in the (computational) security parameter and is independent of the size of w. One can also require an argument system to be zero-knowledge, meaning that the communication transcript can be efficiently simulated without knowing the witness. An argument is of knowledge if for any successful

prover there exists an extractor that can recover w. Finally, we say that an argument is non-interactive if it consists of a single message from the prover to the verifier.

1.1 The Quest of Constructing Ever Shorter Arguments

Recently, much progress has been made both in theory and practice to construct zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARK) for general NP languages. We distinguish between zk-SNARKs in the setup-free model and the pre-processing model. In the setup-free model, the prover and the verifier do not share any input other than the statement x to be proven. In the pre-processing model, they share a common reference string, generated by a trusted third party, which may depend on the language $\mathcal{L}$ and the statement x. In general, existing zk-SNARKs in the pre-processing model are more efficient (at least in terms of the computational complexity of verification) than those in the setup-free model. This reflects the intuition that pushing the majority of the verifier's workload to the offline pre-processing phase reduces its workload in the online phase. On the other hand, in some applications such as cryptocurrencies it is crucial to avoid a trusted setup. With the current landscape, one is forced to trade short proof size and verification time with the absence of a trusted setup.

In order to compare argument systems with drastically different constructions, throughout the paper we will consider instantiations with computational security parameter set to $\lambda \approx 100$ and statistical security parameter set to $k \approx 80$.

Pre-Processing SNARKs. In the pre-processing model, there exist plenty of zk-SNARKs constructed from linear interactive proofs (LIP) and pairings in the standard model [37]. The scheme with the shortest proofs, with 4 group elements, is due to Danezis *et al.* [29]. In the generic group model, Groth [40] proposed a scheme [57] with only 3 group elements, and showed that proofs constructed from LIP must consist of at least 2 group elements. These schemes can be instantiated over elliptic curves. A popular choice is "BN128" [7], in which a group element can be represented using 256 bits.

Setup-Free SNARKs. While it is known that setup-free non-interactive arguments for NP do not exists in the standard model [14], one can circumvent this impossibility by assuming the existence of a random oracle [8]. For a soundness parameter k, a proof in the scheme by Micali [52] consists of a λ-bit Merkle-tree commitment of a probabilistic checkable proof (PCP) string, $O(k)$ bits of the PCP string, and $O(k)$ openings of the commitment, each of size $\lambda \log |w|$ bits. For concreteness, assuming $|w| = 2^{30}$, $\lambda = 100$, $k = 80$, and the constant in $O(k)$ is 3, the proof size is around 88 KB. In Bulletproof [23], which improves upon Bootle *et al.* [18], a proof consists of $2 \log n + 13$ (group and field) elements, where n is the number of multiplication gates in the arithmetic circuit representation of the verification algorithm of $\mathcal{L}$. In an instantiation over the curve secp256k1, each of the group elements and integers can be represented by $\sim$256 bits, thus a proof consists of roughly $512 \log n + 3328$ bits. As n grows, the difference in proof size, when compared to those in the pre-processing model, can be quite substantial.

1.2 Our Results

Constructing argument systems, especially non-interactive and setup-free ones, with even smaller proof size is of both theoretical and practical importance. In this work, we present several families of constructions in various settings. We highlight two of them.

1. We construct a setup-free zk-SNARK in the random oracle model, under certain variants of the root assumption over class groups of imaginary quadratic orders. The proof size of a reasonable instantiation is 4336 bits, which is shorter than that of Bulletproof [23] for $n > 4$.
2. We construct a pre-processing zk-SNARK in the random oracle model, under the computational Diffie-Hellman (CDH) assumption over pairing groups. Instantiating it with BN128 yields a proof size of 752 bits, and verifying a proof requires 2 pairings plus some lightweight operations. In comparison, the scheme of Groth [40], which is sound in the generic group model, has roughly the same proof size, and requires 3 pairings plus some lightweight operations for verification.

Construction Overview. To explain our results in more detail, we recall the notion of probabilistic checkable proofs (PCP) [3] for NP. A PCP scheme allows the prover to efficiently compute a PCP string which encodes the witness of the statement to be proven. The verifier can then decide whether the statement is true by inspecting a short substring of the PCP string. Linear PCP [44] generalizes traditional PCP in the sense that the PCP string now encodes a linear function. The verifier, who is given oracle access to the function, can decide the veracity of the statement by making only a few queries.

The core of our results is a 4-move argument systems for NP, which follows the construction paradigm of Killian [46] and Micali [52]. For this explanation, it is useful to split the system into an offline setup phase and an online proving phase. In the setup phase, the verifier sets up the system and sends as the first move the public parameters (or the generating randomness in the setup-free case) to the prover. This public parameter can be reused for proving multiple statements. Next, in the proving phase, the prover first encodes its witness for the NP statement as a PCP string, and sends the commitment of the string to the verifier. The verifier then asks the prover to reveal some random positions of the string. The prover responds with the values of those positions and the corresponding openings to the commitment. Finally, the verifier checks the validity of the openings and the revealed values, and outputs a bit deciding the truth of the statement.

Typically, an argument system under this paradigm has public-coin verifiers (at least in the proving phase), and thus can be compiled into a non-interactive argument using the Fiat-Shamir transform [33]. If the underlying PCP is witness-extractable, then the argument system is of knowledge. Zero-knowledge can be obtained generically by using one of the following approaches: An option is to replace the PCP with an (honest-verifier) zero-knowledge PCP (*e.g.*, [48]). Another option is to first prove the knowledge of the witness using non-interactive zero-knowledge proof of knowledge (NIZKPoK) [31], and then prove the knowledge of the NIZKPoK proof using the argument system.

Being the main ingredients of the construction, the efficiency of the PCP and the commitment scheme determines the efficiency of the resulting argument system. The majority of the paper is dedicated to formalizing and constructing commitment schemes which are compatible with PCP schemes with varying efficiency.

Subvector Commitments. Previous schemes under Micali's paradigm use a Merkle-tree commitment to commit to the PCP string. For an encoding of size q, the commitment size is $O(\lambda)$ and the opening size is $O(\lambda \log q)$. Typically PCPs have only a constant soundness error, which means that one need to amplify the probability by running k instances of the verification. Our main idea is to replace the Merkle-tree commitment with other commitment schemes which allow batch opening to multiple positions. With this in mind, we study the notion of vector commitment (VC) [24].

A VC scheme [24] is a commitment scheme which allows a prover to commit to a vector $\boldsymbol{x}$ of q messages, such that it can later open the commitment at any position $i \in [q]$ of the vector, *i.e.*, reveal a message and show that it equals to the i-th committed message. A VC scheme is required to be position binding, meaning that no efficient algorithm can open a commitment at some position i to two distinct messages $x_i \neq x'_i$. Catalano and Fiore [24] constructed two VC schemes based on the CDH assumption over pairing groups and the RSA assumption, respectively. In both schemes, a commitment and an opening both consist of a single group element (in the respective groups). Furthermore, the scheme based on the RSA assumption has public parameters whose size is independent of the length of the vectors to be committed.

Inspired by this work, we modify both VC constructions of Catalano and Fiore so that the prover can open the commitment at any subset of positions $I \subseteq [q]$, with a single group element. Proving the security of the modified schemes requires a more complex manipulation of the exponents. We further generalize the RSA-based scheme to work over modules over Euclidean rings, where variants of the root assumption hold. This enables setup-free instantiations using class groups of imaginary quadratic orders.

Formalizing the idea, we generalize VC to what we call subvector commitments (SVC). Given a vector $\boldsymbol{x}$ of length q and an *ordered* index set $I \subseteq [q]$, we define the I-subvector of $\boldsymbol{x}$ as the vector formed by collecting the i-th component of $\boldsymbol{x}$ for all $i \in I$. It is important that an opening to an I-subvector has size sublinear in (or even independent of) $|I|$, otherwise it is not more efficient than opening a VC at $|I|$ positions separately. In addition to the functional differences, we define position binding without trusted setup to capture the security of plausible setup-free instantiations. This property says that, even if the public parameters are sampled using *public coins*, no efficient algorithm can open a commitment at some index sets I and J to some I-subvector $\boldsymbol{x}_I$ and and J-subvector $\boldsymbol{x}'_J$ respectively, such that there exists $i \in I \cap J$ with $x_i \neq x'_i$.

Linear Map Commitments. While argument systems constructed from (traditional) PCPs and SVC schemes have plausible setup-free instantiations, a main drawback is that PCP schemes typically have much less efficient provers than those in linear PCP schemes. This motivates us to study functional commitments (FC), which were introduced by Libert, Ramanna and Yung [49] as a generalization of VC.

Intuitively, an FC allows the prover to commit to a vector $\boldsymbol{x}$, and opens the commitment to function-value tuples (f, y) such that $y = f(\boldsymbol{x})$. Libert, Ramanna and Yung [49] formalized FC for linear forms $f : \mathbb{F}^q \to \mathbb{F}$ for some field $\mathbb{F}$. Generalizing position binding of VC, they define function binding for FC for linear forms, which means that no efficient algorithm can open a commitment to function-value tuples (f, y) and (f, y') where f is a linear form and $y \neq y'$. This notion aims to capture the intuition that

the prover cannot open a commitment inconsistently. Based on the q-Diffie-Hellman exponent (q-DHE) assumption over pairing groups, they gave a construction in which a commitment and an opening both consist of a single group element. Note that using their FC for linear forms and the linear PCP of Bitanski *et al.* [15], one can obtain an argument system (with trusted setup) where a proof consists of 4 group and 3 field elements.

To further reduce the proof size (*e.g.*, to 2 group and 3 field elements), we generalize FC for linear forms to FC for linear maps $f : \mathbb{F}^q \to \mathbb{F}^k$, similar in spirit to the generalization of VC to SVC. We call such class of FC *linear map commitments* (LMC). As in SVC, it is important to require an LMC to be compact, meaning that both the commitment and the openings are of size sublinear in, or even independent of, q and k. Note that an SVC can be viewed as an LMC restricted to the class of linear maps whose matrix representation has exactly one 1 in each row and 0 everywhere else.

Naively, one may attempt to generalize function binding for FC for linear forms to that of LMC by requiring that the prover cannot open a commitment to $(f, \boldsymbol{y})$ and $(f, \boldsymbol{y}')$ with $\boldsymbol{y} \neq \boldsymbol{y}'$, where f is a linear map and $\boldsymbol{y}, \boldsymbol{y}' \in \mathbb{F}^k$ are now vectors. Attempting to prove the soundness of an argument system from this definition of function binding suggests that the latter is insufficient. This is because the prover may be able to open to $(f, \boldsymbol{y})$ and $(f', \boldsymbol{y}')$ where $f \neq f'$ and $\boldsymbol{y} \neq \boldsymbol{y}'$ such that they form an inconsistent system of linear equations, yet the attack is not captured by the definition. We tackle this issue by defining a more general function binding notion (also applicable to FC for general functions) which requires that no efficient algorithm can produce openings for L function-value tuples $\{(f_\ell, \boldsymbol{y}_\ell)\}_{\ell \in [L]}$ for any $L \in \mathsf{poly}(\lambda)$, such that there does not exist $\boldsymbol{x}$ with $f_\ell(\boldsymbol{x}) = \boldsymbol{y}_\ell$ for all $\ell \in [L]$.

We then modify the construction of Libert, Ramanna and Yung [49] to support batch openings to linear forms, or equivalently opening to a linear map. Since the verification equation of their construction is linear, a natural way to support batch openings is to define the new verification equation as a random linear combination of previous ones. With this observation, our first construction embeds a secret linear combination in the public parameter, and is shown to be function binding in the generic group model (GGM). To avoid relying on the GGM, we give a second construction which samples random linear combinations using a random oracle. This construction achieves a weaker notion of function binding, in which the prover must first output a commitment and then provide openings to functions sampled by a sampler. We show that such a weaker notion is sufficient for proving the soundness of our argument system.

Summary of Results. The main result of this work is a compiler that turns PCPs into SNARKs. Our compiler works directly with both traditional and linear PCPs. Plugging in different instances of SVCs or LMCs, we obtain a wide range of argument systems in various settings. We summarize our results with a short list of constructions in the most interesting settings by means of informal theorems. Combining traditional PCPs with our SVCs (see Section 6) yields the following results.

Theorem 1. *(Informal.) If the strong RSA assumption holds in $\mathbb{Z}_N^*$ for some integer N, then there exist 4-move argument systems for NP with the following properties. The public parameter consists of N. In the proving phase, the verifier is public-coin, the*

communication from the verifier to the prover consists of λ bits, and the communication from the prover to the verifier consists of 2 $\mathbb{Z}_N^*$ elements and $3k$ bits.

Theorem 2. *(Informal.) If the adaptive root assumption holds in $Cl(\Delta)$, the class groups of imaginary quadratic order with discriminant Δ, and NIZKPoK exists in the common random string model, then there exist setup-free zk-SNARKs for NP in which a proof consists of 2 $Cl(\Delta)$ elements and $3k$ bits in the random oracle model.*

Theorem 3. *(Informal.) If the CDH assumption holds in a pairing group $\mathbb{G}$, and NIZKPoK exists in the common random string model, then there exist pre-processing zk-SNARKs for NP in which a proof consists of 2 $\mathbb{G}$ elements and $3k$ bits in the random oracle model. Furthermore, the verification requires 2 pairing operations.*

Combining linear PCPs with an LMC scheme (see Section 7) gives us the following.

Theorem 4. *(Informal.) If the q-DHE assumption holds in a pairing group $\mathbb{G}$ of prime order p, and NIZKPoK exists in the common random string model, then there exist pre-processing zk-SNARKs for NP in which a proof consists of 2 $\mathbb{G}$ elements and 3 $\mathbb{Z}_p$ elements in the random oracle model. The provers are potentially more efficient than the schemes obtained in Theorem 3, and the verification requires 3 pairing operations.*

1.3 Other Applications

Catalano and Fiore [24] suggested a list of applications of VC, including verifiable databases with efficient updates, updatable zero-knowledge elementary databases, and universal dynamic accumulators. In all of these applications, one can gain efficiency by replacing the VC scheme with an SVC scheme which allows for batch opening and updating. When instantiated with our first construction of SVC, one can further avoid the trusted setup, which is especially beneficial to database applications as trusted third parties are no longer required. For the applications of FCs for linear forms [49], similar efficiency improvements can be obtained by using LMCs instead.

1.4 Related Work

Succinct arguments were introduced by Kilian [46,47] and later improved, in terms of round complexity, by Lipmaa and Di Crescenzo [32]. Succinct non-interactive arguments, or computationally sound proofs, were first proposed by Micali [52]. These early approaches rely on PCP and have been recently extended [9] to handle interactive oracle proofs [12] (also known as probabilistic checkable interactive proofs [54]), in favor of a more efficient prover (but with the same asymptotics). A recent manuscript by Ben-Sasson *et al.* [10] improves the concrete efficiency of interactive oracle proofs. Ishai, Kushilevitz, and Ostrovsky [44] observed that linear PCP can be combined with a linearly homomorphic encryption to construct more efficient arguments, with pre-processing. Later, Groth [39] and Lipmaa [50] upgraded this approach to non-interactive proofs.

The first usage of knowledge assumptions to construct SNARKs appeared in the work of Mie [53]. Gennaro *et al.* [37] presented a very elegant linear PCP that gave

rise to a large body of work to improve the practical efficiency of non-interactive arguments [5,11,13,25,26,34,30]. All of these constructions assume a highly structured and honestly generated common reference string (of size proportional to the circuit to be evaluated) and rely on some variant of the knowledge of exponent assumption. Recently, Ames *et al.* [2] proposed an argument based on the MPC-in-the-head [45] paradigm to prove satisfiability of a circuit C with proofs of size $O(\lambda\sqrt{|C|})$. Zhang *et al.* [61] show how to combine interactive proofs and verifiable polynomial delegation schemes to construct succinct interactive arguments. The scheme requires a trusted pre-processing and the communication complexity is $O(\lambda \log |w|)$. A recent result by Whaby *et al.* [59] introduces a prover-efficient construction with proofs of size $O(\lambda\sqrt{|w|})$. Recent works [1,36] investigate on the resilience of SNARKs against a subverted setup.

Libert, Ramanna, and Yung [49] constructed an accumulator for subset queries. Although similar in spirit to SVC, the critical difference is that accumulators are not position binding, which is crucial for the soundness of our argument system.

2 Preliminaries

Throughout this work we denote by $\lambda \in \mathbb{N}$ the security parameter, and by $\mathsf{poly}(\lambda)$ and $\mathsf{negl}(\lambda)$ the sets of polynomials and negligible functions in λ, respectively. We say that a Turing machine is probabilistic polynomial time (PPT) if its running time is bounded by some polynomial function $\mathsf{poly}(\lambda)$. An interactive protocol Π between two machines A and B is referred to as $(A, B)_\Pi$. Given a set S, we denote sampling a random element from S as $s \leftarrow\!\!\$\, S$ and the output of an algorithm A on input x is written as $z \leftarrow A(x)$. Let $\ell \in \mathbb{N}$, the set $[\ell]$ is defined as $[\ell] := \{1, \ldots, \ell\}$. Vectors are written vertically.

2.1 Subvectors

We define the notion of subvectors. Roughly speaking, a subvector $(x_{i_1}, \ldots, x_{i_{|I|}})^T$ is an ordered subset (indexed by I) of the entries of a given vector $(x_1, \ldots, x_q)^T$.

Definition 1 (Subvectors). *Let $q \in \mathbb{N}$, $\mathcal{X}$ be a set, and $(x_1, \ldots, x_q)^T \in \mathcal{X}^q$ be a vector. Let $I = (i_1, \ldots, i_{|I|}) \subseteq [q]$ be an ordered index set. The I-subvector of $\boldsymbol{x}$ is defined as $\boldsymbol{x}_I := (x_{i_1}, \ldots, x_{i_{|I|}})^T$.*

2.2 Arguments of Knowledge

Let $\mathcal{R} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ be an NP-relation with corresponding NP-language $\mathcal{L} := \{x : \exists w \; s.t. \; \mathcal{R}(x, w) = 1\}$. We define arguments of knowledge [19] for interactive Turing machines [38]. To be as general as possible, we define an additional setup algorithm $\mathcal{S}$, which is executed once and for all by a trusted party. If the argument is secure without a setup, then such an algorithm can be omitted.

Definition 2 (Arguments of knowledge). *A tuple $(\mathcal{S}, (\mathcal{P}, \mathcal{V})_\Pi)$ is a (succinct) argument of knowledge for $\mathcal{R}$ if the following conditions hold.*

(Completeness) If $\mathcal{R}(x, w) = 1$ then $\underset{y \leftarrow \mathcal{S}(1^\lambda)}{\mathsf{Pr}}[(\mathcal{P}(x, w, y), \mathcal{V}(x, y))_\Pi = 1] = 1.$

(Argument of Knowledge) For any PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}$, such that for all $x, z \in \{0,1\}^*$, if there exists a constant c such that $\Pr_{y \leftarrow \mathcal{S}(1^\lambda)}[(\mathcal{A}(x,z,y), \mathcal{V}(x,y))_{\Pi} = 1] \geq \lambda^{-c}$, then there exists a constant d such that $\mathsf{Pr}[\mathcal{R}(x,w) = 1 | w \leftarrow \mathcal{E}^{\mathcal{A}(\cdot,\cdot,\cdot)}(x)] \geq \lambda^{-d}$.

(Succinctness) The communication between $\mathcal{P}$ and $\mathcal{V}$ is at most $\mathsf{poly}(\lambda)$.

An argument of knowledge satisfies the notion of zero-knowledge if the interaction between the prover and the verifier reveals nothing but the validity of the statement.

Definition 3 (Zero Knowledge). *A tuple $(\mathcal{S}, (\mathcal{P}, \mathcal{V})_{\Pi})$ is computationally (statistically, resp.) zero-knowledge if for all PPT $\mathcal{V}^*$ there exists a PPT $\mathcal{S}^*$ such that the following ensembles are computationally (statistically, resp.) indistinguishable*

$$\left\{y := \mathcal{S}(1^\lambda), (\mathcal{P}(x,w,y), \mathcal{V}^*(x,y))_{\Pi}\right\}_{\lambda \in \mathbb{N}, x \in \mathcal{L}, w \text{ s.t. } \mathcal{R}(x,w)=1} \approx \{\mathcal{S}^*(x)\}_{\lambda \in \mathbb{N}, x \in \mathcal{L}}.$$

2.3 Probabilistically Checkable Proofs

One of the principal tools in the construction of argument systems is probabilistic checkable proofs (PCP) [3]. The PCP theorem shows that any witness w for an NP-statement can be encoded into a PCP of length $|w|(\log |w|)^{O(1)}$ such that it is sufficient to probabilistically test $O(1)$ bits of the encoded witness.

Definition 4 (Probabilistically Checkable Proofs). *A pair of machines $(\mathcal{P}_{PCP}, \mathcal{V}_{PCP})$ is a PCP for an NP-relation $\mathcal{R}$ if the following conditions hold.*

(Completeness) If $\mathcal{R}(x,w) = 1$, then $\mathsf{Pr}\left[\mathcal{V}^{\boldsymbol{\pi}}_{PCP}(x) = 1 | \boldsymbol{\pi} \leftarrow \mathcal{P}_{PCP}(x,w)\right] = 1$.

(Soundness) For all $x \notin \mathcal{L}$ $\mathsf{Pr}\left[\mathcal{V}^{\boldsymbol{\pi}}_{PCP}(x) = 1\right] < \frac{1}{3}$.

The notation $\mathcal{V}^{\boldsymbol{\pi}}_{\text{PCP}}(x)$ means that $\mathcal{V}_{\text{PCP}}$ does not read the entire string $\boldsymbol{\pi}$ directly, but is given oracle access to the string. On input a position $i \in [|\boldsymbol{\pi}|]$, the oracle returns the value π_i. It is well known that one can diminish the soundness error to a negligible function by parallel repetition. We additionally require that the witness can be efficiently recovered from the encoding of the witness $\boldsymbol{\pi}$ [58].

Definition 5 (Witness-Extractability). *A PCP is witness-extractable if there exists a PPT algorithm $\mathcal{E}_{PCP}$ and a constant $\gamma \in (0,1)$ such that, given any strings x and $\boldsymbol{\pi}$ with $\mathsf{Pr}\left[\mathcal{V}^{\boldsymbol{\pi}}_{PCP}(x) = 1\right] \geq 1 - \gamma$, $\mathcal{E}_{PCP}$ extracts an NP witness w for x.*

Linear PCPs. Ishai et al. [44] considered the notion of *linear* PCP, where the string $\boldsymbol{\pi}$ is instead a vector in $\mathbb{F}^q$ for some finite field $\mathbb{F}$ (or in general a ring) and positive integer q. The oracle given to the verifier is modified, such that on input $\boldsymbol{f} \in \mathbb{F}^q$, it returns the inner product $\langle \boldsymbol{f}, \boldsymbol{\pi} \rangle$. Note that this generalizes the classical notion of PCP as one can recover the original definition by restricting the queries $\boldsymbol{f}$ to be unit vectors. Linear PCPs typically feature a more efficient computation of the encodings.

Unpredictability. To prove the soundness of one of our constructions, we are going to assume certain properties of the distribution of the queries of $\mathcal{V}^{\pi}_{\mathrm{PCP}}(x)$. Specifically, we need to assume that such queries are *unpredictable*, i.e., they are hard to guess for any PPT algorithm. In the following we formally define the notion of unpredictability for a function sampler FSamp.

Definition 6 (Unpredictability). *Let* FSamp *be a sampler for a function family* $\mathcal{F}$. FSamp *is said to be unpredictable, if for all PPT adversary* $\mathcal{A}$ *there exists a negligible function* $\epsilon(\lambda) \in \mathsf{negl}(\lambda)$ *such that*

$$\mathsf{Pr}\left[f = f' : \begin{array}{c}(\mathsf{seed}, f) \leftarrow \mathcal{A}(1^\lambda) \\ r \leftarrow\!\!\$\, \{0,1\}^\lambda \\ f' \leftarrow \mathsf{FSamp}(\mathsf{seed}; r)\end{array}\right] \leq \epsilon(\lambda).$$

We stress that virtually all known (linear) PCPs (*e.g.*, [44,15]) have unpredictable queries.

3 Mathematical Background and Assumptions.

To capture the minimal mathematical structure required for one of our constructions, we follow the module-based cryptography framework of Lipmaa [51].

Background. A (left) R-module R_D over the ring R (with identity) consists of an Abelian group $(D, +)$ and an operation $\circ : R \times D \to D$, denoted $r \circ A$ for $r \in R$ and $A \in D$, such that for all $r, s \in R$ and $A, B \in D$, we have

- $r \circ (A + B) = r \circ A + r \circ B$,
- $(r + s) \circ A = r \circ A + s \circ A$,
- $(r \cdot s) \circ A = r \circ (s \circ A)$, and
- $1_R \circ r = r$, where 1_R is the multiplicative identity of R.

Let $S = (s_1, \ldots, s_q) \subseteq \mathbb{N}$ be an ordered set, and $\boldsymbol{r} = (r_{s_1}, \ldots, r_{s_q})^T \in R^q$ and $\boldsymbol{A} = (A_{s_1}, \ldots, A_{s_q})^T \in D^q$ be vectors of ring and group elements respectively. For notational convenience, we denote $\sum_{i \in S} r_i \circ A_i$ by $\langle \boldsymbol{r}, \boldsymbol{A} \rangle$.

A commutative ring R with identity is called an *integral domain* if for all $r, s \in R$, $rs = 0_R$ implies $r = 0_R$ or $s = 0_R$, where 0_R is the additive identity of R. A ring R is *Euclidean* if it is an integral domain and there exists a function $\deg : R \to \mathbb{Z}^+$, called the Euclidean degree, such that i) if $r, s \in R$, then there exist $q, k \in R$ such that $r = qs + k$ with either $k = 0_R$, $k \neq 0_R$ and $\deg(k) < \deg(q)$, and ii) if $r, s \in R$ with $rs \neq 0_R$ and $r \neq 0_R$, then $\deg(r) < \deg(rs)$. The set of units $U(R) := \{u \in R : \exists v \; s.t. \; uv = vu = 1_R\}$ contains all invertible elements in R. An element $r \in R \setminus (\{0_R\} \cup U(R))$ is said to be *irreducible* if there are no elements $s, t \in R \setminus \{1_R\}$ such that $r = st$. The set of all irreducible elements of R is denoted by $\mathrm{IRR}(R)$. An element $r \in R \setminus (\{0_R\} \cup U(R))$ is said to be *prime* if for all $s, t \in R$, whenever r divides st, then r divides s or r divides t. If R is Euclidean, then an element is irreducible if and only if it is prime.

Adaptive Root. The adaptive root assumption (over unknown order groups, and in particular over class groups of imaginary quadratic orders) was introduced by Wesolowski [60] and re-formulated by Boneh *et al.* [17] to establish the security of the verifiable delay function scheme of Wesolowski [60]. Here we state the same assumption over modules in two variants – with and without trusted setup – similarly as above. Note that Wesolowski [60] and Boneh et al. [17] implicitly considered the *setup-free* variant.

Definition 7 (Adaptive Root (w/o Trusted Setup)). *Let I be some ordered set. Let $\mathcal{R}_{\mathcal{D}} = ((R_i)_{D_i})_{i \in I}$ be a family of modules. Let $\mathsf{MGen}(1^\lambda; \omega)$ be a deterministic algorithm which picks some $i \in I$ (hence some $R_D = (R_i)_{D_i} \in \mathcal{R}_{\mathcal{D}}$) and some element $A \in D$. For a ring R, let $\mathrm{IRR}_\lambda(R) \subseteq \mathrm{IRR}(R)$ be some set of prime elements in R of size 2^λ. The adaptive root assumption is said to hold over the family of modules $\mathcal{R}_{\mathcal{D}}$ with respect to IRR_λ, if for any PPT adversary $\mathcal{A}$ there exists $\epsilon(\lambda) \in \mathsf{negl}(\lambda)$ such that*

$$\mathsf{Pr}\left[e \circ Y = X \,\middle|\, \begin{array}{c} \omega \leftarrow_{\$} \{0,1\}^\lambda; (R_D, A) := \mathsf{MGen}(1^\lambda; \omega) \\ X \leftarrow \mathcal{A}(1^\lambda, R_D, A, \boxed{\omega}); e \leftarrow_{\$} \mathrm{IRR}_\lambda(R); Y \leftarrow \mathcal{A}(e) \end{array}\right] \leq \epsilon(\lambda),$$

where $\mathcal{A}$ is not *given ω (highlighted by the dashed box). If the inequality holds even if $\mathcal{A}$ is given ω, then we say that the assumption holds* without trusted setup.

Strong Distinct-Prime-Product Root. We define the following variant of the "strong root assumption" [27] over modules over Euclidean rings, which is a generalization of the strong RSA assumption. Let R_D be a module over some Euclidean ring R, and A be an element of D. The strong distinct-prime-product root problem with respect to A asks to find a set of distinct prime elements $\{e_i\}_{i \in S}$ in R and an element Y in D such that $\left(\prod_{i \in S} e_i\right) \circ Y = A$. We define the assumption in two variants depending on whether R_D and A are sampled with public coins.

Definition 8 (Strong Distinct-Prime-Product Root (w/o Trusted Setup)). *Let I be an ordered set, $\mathcal{R}_{\mathcal{D}} = ((R_i)_{D_i})_{i \in I}$ be a family of modules, and $\mathsf{MGen}(1^\lambda; \omega)$ be a deterministic algorithm which picks some $i \in I$ (hence some $R_D = (R_i)_{D_i} \in \mathcal{R}_{\mathcal{D}}$) and some element $A \in D$. The strong distinct-prime-product root assumption is said to hold over the family $\mathcal{R}_{\mathcal{D}}$, if for any PPT adversary $\mathcal{A}$ there exists $\epsilon(\lambda) \in \mathsf{negl}(\lambda)$ such that*

$$\mathsf{Pr}\left[\begin{array}{l} \left(\prod_{i \in S} e_i\right) \circ Y = A \\ \forall i \in S, e_i \in \mathrm{IRR}(R) \\ \forall i \neq j \in S, e_i \neq e_j \end{array} \,\middle|\, \begin{array}{c} \omega \leftarrow_{\$} \{0,1\}^\lambda \\ (R_D, A) := \mathsf{MGen}(1^\lambda; \omega) \\ (\{e_i\}_{i \in S}, Y) \leftarrow \mathcal{A}(1^\lambda, R_D, A, \boxed{\omega}) \end{array}\right] \leq \epsilon(\lambda),$$

where $\mathcal{A}$ is not *given ω (highlighted by the dashed box). If the inequality holds even if $\mathcal{A}$ is given ω, then we say that the assumption holds* without trusted setup.

Lipmaa [51] defined several variants of the (strong) root assumption with respect to a random element in D sampled with *private coin*, given the description of the module R_D sampled with *public coin*. Note that the strong distinct-prime-product root assumption (resp. without trusted setup) is weaker than the strong root assumption (resp. without trusted setup), where the latter requires the adversary to simply output (e, Y) such that $e \neq 1_R$ and $e \circ Y = A$. It is apparent that the strong distinct-prime-product root assumption over RSA groups is implied by the strong RSA assumption.

4 Functional Commitments

Functional commitments (FC) for linear functions, specifically for linear forms $f : \mathbb{F}^q \to \mathbb{F}$ for some field $\mathbb{F}$, were introduced by Libert, Ramanna and Yung [49] and is a generalization of vector commitments (VC) introduced by Catalano and Fiore [24]. Here we refine the notion to capture a more general class of function families, which allows the prover to open a commitment to the output of *multiple* functions or, equivalently, to the output of a *multi-output* function. We also extend the syntax of FC to include mechanisms for changing a subset of entries of the committed vector, while updating the commitment strings and openings accordingly.

Definition 9 (Functional Commitments (FC)). *Let $q, n \in \mathsf{poly}(\lambda)$ be positive integers, and $\mathcal{F} \subseteq \{f : \mathcal{X}^q \to \mathcal{Y}^n\}$ be a family of functions defined over some spaces $\mathcal{X}$ and $\mathcal{Y}$. A functional commitment scheme* FC *for a function family $\mathcal{F}$ consists of the following PPT algorithms* $(\mathsf{Setup}, \mathsf{Com}, \mathsf{Open}, \mathsf{Verify})$*:*

$\underline{\mathsf{Setup}(1^\lambda, \mathcal{F}; \omega)}$*: The* deterministic *setup algorithm inputs the security parameter 1^λ, the description of the function family $\mathcal{F}$, and a random tape ω. It outputs a public parameter* pp*. We assume that all other algorithms input* pp *which we omit.*

$\underline{\mathsf{Com}(\boldsymbol{x})}$*: The committing algorithm inputs a vector $\boldsymbol{x} \in \mathcal{X}^q$. It outputs a commitment string C and some auxiliary information* aux*.*

$\underline{\mathsf{Open}(f, \boldsymbol{y}, \mathsf{aux})}$*: The opening algorithm inputs a function $f \in \mathcal{F}$, an image $\boldsymbol{y} \in \mathcal{Y}^n$, and some auxiliary information* aux*. It outputs a proof Λ that $\boldsymbol{y} = f(\boldsymbol{x})$.*

$\underline{\mathsf{Verify}(C, f, \boldsymbol{y}, \Lambda)}$*: The verification algorithm inputs a commitment string C, a function $f \in \mathcal{F}$, an image $\boldsymbol{y}$, and a proof Λ. It accepts (i.e., it outputs 1) if and only if C is a commitment to $\boldsymbol{x}$ and $\boldsymbol{y} = f(\boldsymbol{x})$.*

Definition 10 (Correctness). *A functional commitment scheme* FC *for the function family $\mathcal{F}$ is said to be correct if, for any security parameter $\lambda \in \mathbb{N}$, random tape $\omega \in \{0,1\}^\lambda$, public parameters $\mathsf{pp} \in \mathsf{Setup}(1^\lambda, \mathcal{F}; \omega)$, $\boldsymbol{x} \in \mathcal{X}^q$, function $f \in \mathcal{F}$, $(C, \mathsf{aux}) \in \mathsf{Com}(\boldsymbol{x})$, $\Lambda \in \mathsf{Open}(f, f(\boldsymbol{x}), \mathsf{aux})$, there exists $\epsilon(\lambda) \in \mathsf{negl}(\lambda)$ such that*

$$\Pr\left[\mathsf{Verify}(C, f, f(\boldsymbol{x}), \Lambda) = 1\right] \geq 1 - \epsilon(\lambda).$$

We next generalize the notion of function binding for FC for general function families. For linear forms, as considered by Libert, Ramanna and Yung [49], it is required that it is hard to open a commitment to (f, y) and (f, y') where $y \neq y'$. When considering broader classes of functions, such as linear maps where the target space is multidimensional, each opening defines a system of equations. Note that in this case one might be able to generate an inconsistent system with just a single opening, or generate openings to (f, y) and (f', y') with $f \neq f'$ but the systems defined by the tuples are inconsistent. Therefore, our definition explicitly forbids the adversary to generate inconsistent equations.

Definition 11 (Function Binding (w/o Trusted Setup)). *A functional commitment* FC *for the function family $\mathcal{F}$ is function binding if for any PPT adversary $\mathcal{A}$ and $L \in$*

$\mathsf{poly}(\lambda)$*, there exists a negligible function* $\epsilon(\lambda) \in \mathsf{negl}(\lambda)$ *such that*

$$\Pr\left[\begin{array}{l}\forall \ell \in [L],\ \mathsf{Verify}(C, f_\ell, \boldsymbol{y}_\ell, \Lambda_\ell) = 1 \\ \nexists \boldsymbol{x} \in \mathcal{X}^q\ s.t.\ \forall \ell \in [L],\ f_\ell(\boldsymbol{x}) = \boldsymbol{y}_\ell\end{array}\middle|\begin{array}{c}\omega \leftarrow_{\$} \{0,1\}^\lambda \\ \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F}; \omega) \\ (C, \{(f_\ell, \boldsymbol{y}_\ell, \Lambda_\ell)\}_{\ell \in [L]}) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp}, \boxed{\omega})\end{array}\right] \leq \epsilon(\lambda)$$

where $\mathcal{A}$ *is* not *given* ω *(highlighted by the dashed box). If the inequality holds even if* $\mathcal{A}$ *is given* ω*, then we say that* FC *is function binding* without trusted setup.

For the purpose of constructing argument systems, it turns out that the above property is more than what is necessary. We therefore define a weaker variant below which is still sufficient for the purpose. In this variant, the adversary is split into two stages $\mathcal{A}_1$ and $\mathcal{A}_2$. In the first stage, $\mathcal{A}_1$ outputs a commitment string C. A set of functions is then sampled using some sampling algorithm FSamp and is given to $\mathcal{A}_2$. The latter must then produce openings of the commitment C with respect to these functions and their respective function values, such that all openings pass the verification, yet the function-value tuples are inconsistent. Apparently function binding implies weak function binding with respect to any function sampler.

Definition 12 (Weak Function Binding (w/o Trusted Setup)). *A functional commitment* FC *for the function family* $\mathcal{F}$ *is weakly function binding with respect to the function sampler* FSamp*, if for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *and* $L \in \mathsf{poly}(\lambda)$*, there exists a negligible function* $\epsilon(\lambda) \in \mathsf{negl}(\lambda)$ *such that*

$$\Pr\left[\begin{array}{l}\forall \ell \in [L],\ \mathsf{Verify}(C, f_\ell, \boldsymbol{y}_\ell, \Lambda_\ell) = 1 \\ \nexists \boldsymbol{x} \in \mathcal{X}^q\ s.t.\ \forall \ell \in [L],\ f_\ell(\boldsymbol{x}) = \boldsymbol{y}_\ell\end{array}\middle|\begin{array}{c}\omega \leftarrow_{\$} \{0,1\}^\lambda \\ \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F}; \omega) \\ (C, \mathsf{seed}, \mathsf{state}) \leftarrow \mathcal{A}_1(1^\lambda, \mathsf{pp}, \boxed{\omega}) \\ \forall \ell \in [L],\ r_\ell \leftarrow_{\$} \{0,1\}^\lambda \\ \forall \ell \in [L],\ f_\ell \leftarrow \mathsf{FSamp}(\mathsf{seed}; r_\ell) \\ \{(\boldsymbol{y}_\ell, \Lambda_\ell)\}_{\ell \in [L]} \leftarrow \mathcal{A}_2(\mathsf{state}, \{f_\ell\}_{\ell \in [L]})\end{array}\right] \leq \epsilon(\lambda)$$

where $\mathcal{A}$ *does* not *receive* ω *(highlighted by the dashed box) as an input. If the inequality holds even if* $\mathcal{A}$ *receives* ω *as an input, then we say that* FC *is weakly function binding with respect to* FSamp without trusted setup.

Theorem 5. *Let* FC *be a functional commitment for a function family* $\mathcal{F}$*. If* FC *is function binding (resp. without trusted setup), then for any PPT function sampler* FSamp *for* $\mathcal{F}$*,* FC *is weakly function binding with respect to* FSamp *(resp. without trusted setup).*

Since we are considering more general classes of function families, it is meaningful to consider a new property of FC that we call compactness, which requires that the size of the commitment strings and the openings are not only independent of the length of the committed message, but also independent of the description of the function. In particular, it does not depend on the length of outputs of the functions.

Definition 13 (Compactness). *A functional commitment* FC *for the function family* $\mathcal{F}$ *is compact if there exists a universal polynomial* $p \in \mathsf{poly}(\lambda)$ *(independent of* q *and* n*), such that for any random tape* $\omega \in \{0,1\}^\lambda$*, public parameters* $\mathsf{pp} \in \mathsf{Setup}(1^\lambda, \mathcal{F}; \omega)$*, vector* $\boldsymbol{x} \in \mathcal{X}^q$*, function* $f \in \mathcal{F}$*,* $(C, \mathsf{aux}) \in \mathsf{Com}(\boldsymbol{x})$*,* $\Lambda \in \mathsf{Open}(f, f(\boldsymbol{x}), \mathsf{aux})$*, it holds that* $|C| \leq p(\lambda)$ *and* $|\Lambda| \leq p(\lambda)$*.*

We remark that hiding properties, which require that the commitment strings do not leak information about the committed messages, was defined for FC [49] and VC [24] respectively. The definition of VC [24] also requires the commitment strings and the openings to be efficiently updatable. Since for the purpose of this paper, we do not require our FC schemes to be hiding nor updatable, we omit their definitions. Nevertheless, these properties can be defined and supported using existing techniques [49,24].

Finally, we remark that by letting $\mathcal{F}$ be the family of linear forms from $\mathcal{X}^q$ to $\mathcal{X}$, we recover the definition of functional commitments for linear forms [49]. By letting $\mathcal{F}$ to be the family of point functions from $\mathcal{X}^q$ to $\mathcal{X}$, which are functions that input $(x_1, \ldots, x_q) \in \mathcal{X}^q$ and output x_i for some $i \in [q]$, we recover the definition of VC [24].

4.1 Linear Map Commitments

In the remaining parts of the paper, we will focus on FC for linear maps. That is, we let $\mathcal{X} = \mathcal{Y} = R$ for some Euclidean ring R, q and n be positive integers, and $\mathcal{F}$ be the family of linear maps from R^q to R^n. We call such FC *linear map commitments* (LMC). Note that any linear map from R^q to R^n can be represented by a matrix $F \in R^{n \times q}$.

Subvector Commitments. We also consider a special case of linear maps where each map in $\mathcal{F}$ can be represented by an n-by-q binary matrix where each row contains exactly one 1 and 0 everywhere else, or equivalently by a tuple $(f_1, \ldots, f_n) \in [q]^n$ with possibly repeated entries. As illustrated by the following example, this essentially corresponds to the selection of a *subvector* of the original vector $\boldsymbol{x} \in R^q$.

$$\begin{bmatrix} 0,0,0,0,1,0,0 \\ 0,1,0,0,0,0,0 \\ 0,0,0,0,0,0,1 \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} x_5 \\ x_2 \\ x_7 \end{bmatrix} = \boldsymbol{x}_{(5,2,7)}.$$

We refer to this special class of FC as *subvector commitments* (SVC). Due to the simple functionality of SVC, its syntax can be slightly simplified. First, the function family $\mathcal{F}$ can be represented by q. Next, consider the verification algorithm $\mathsf{Verify}(C, F, \boldsymbol{y}, \Lambda)$ of an SVC, where $F = (f_1, \ldots, f_n) \in [q]^n$. Suppose that F contains repeated entries yet the corresponding entries in $\boldsymbol{y}$ are different. That is, there exists $i, j \in [n]$ with $i \neq j$, $f_i = f_j$, and $y_i \neq y_j$. As such a tuple $(F, \boldsymbol{y})$ is inconsistent when interpreted as a linear system of equations, the verification algorithm can simply output 0 without further computation. Therefore, without loss of generality, we assume that the verification algorithm takes as input an ordered index set $I \subseteq [q]$ and an I-subvector $\boldsymbol{x}_I$ instead of $F \in [q]^n$ and $\boldsymbol{y} \in \mathbb{F}^n$ respectively, and is written as $\mathsf{Verify}(C, I, \boldsymbol{x}_I, \Lambda)$. Correspondingly, the opening algorithm also takes as input I and $\boldsymbol{x}_I$, instead of F and $\boldsymbol{y}$ respectively.

We can also simplify the binding definition of SVC. Let $\{(I_\ell, \boldsymbol{x}_{I_\ell})\}_{\ell \in L}$ be a set of tuples of ordered index sets and subvectors. Suppose the set is inconsistent when interpreted as a system of linear equations, then there must exist $i \in I_\ell \cap I_{\ell'}$, for some ℓ and ℓ', such that $x_{\ell,i} \neq x_{\ell',i}$. Therefore it suffices to show that it is infeasible for an adversary to produce a commitment C and a tuple $(I, J, \boldsymbol{x}_I, \boldsymbol{x}'_J, \Lambda_I, \Lambda'_J)$ such that both Λ_I and Λ'_J are valid openings of C to $(I, \boldsymbol{x}_I)$ and $(J, \boldsymbol{x}'_J)$ respectively, but there exists $i \in I \cap J$ such that $x_i \neq x'_i$. Conversely, if no efficiently adversary can produce such a tuple, then no efficient adversary against function binding exists either. For this reason, we re-state the definition of function binding of SVC in the following simpler form. We call this equivalent property position binding, as it can be seen as a generalization of the position binding property of vector commitments [24].

Definition 14 (Position Binding for SVC (without Trusted Setup)). *A subvector commitment* SVC *is position binding if for any PPT adversary $\mathcal{A}$, there exists a negligible function $\epsilon(\lambda) \in \mathsf{negl}(\lambda)$ such that*

$$\Pr\left[\begin{array}{l|c} \mathsf{Verify}(C, I, \boldsymbol{x}_I, \Lambda_I) = 1 & \omega \leftarrow\!\!\$\, \{0,1\}^\lambda \\ \mathsf{Verify}(C, J, \boldsymbol{x}'_J, \Lambda'_J) = 1 & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, q; \omega) \\ \exists i \in I \cap J \text{ s.t. } x_i \neq x'_i & (C, I, J, \boldsymbol{x}_I, \boldsymbol{x}'_J, \Lambda_I, \Lambda'_J) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp}, \boxed{\omega}) \end{array}\right] \leq \epsilon(\lambda)$$

where $\mathcal{A}$ is not *given ω (highlighted by the dashed box). If the inequality holds even if $\mathcal{A}$ is given ω, then we say that* SVC *is function binding* without trusted setup.

Theorem 6. *Let* SVC *be a subvector commitment scheme.* SVC *is position binding (resp. without trusted setup) if and only if* SVC *is function binding (resp. without trusted setup).*

5 Succinct Arguments of Knowledge for NP

We construct interactive arguments of knowledge either from (traditional) PCPs and subvector commitments (Section 6), or from linear PCPs [44] and linear map commitments (Section 7). The constructions for both cases are in fact identical.

Let $(\mathcal{P}_{\text{PCP}}, \mathcal{V}_{\text{PCP}})$ be a witness extractable (linear) PCP over some field $\mathbb{F}$ for NP with q being a bound on the size of the encoded proof, r being a bound on the length of the random coins of the possibly adaptive verifier, h being a bound on the number of queries made by $\mathcal{V}_{\text{PCP}}$, and k being a statistical security parameter. We will set $k = O(\lambda)$ so that $\gamma^k \in \mathsf{negl}(\lambda)$, where $\gamma \in (0,1)$ is the soundness constant associated with the PCP (see Definition 5). Let $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{k \cdot r}$ be a pseudo-random generator and let $\mathsf{FC} := (\mathsf{Setup}, \mathsf{Com}, \mathsf{Open}, \mathsf{Verify})$ be a linear map commitment for some function family $\mathcal{F} \subseteq \{f : \mathbb{F}^q \to \mathbb{F}^{hk}\}$, possibly without trusted setup. We present a 4-move interactive argument of knowledge in Figure 1.

5.1 Protocol Description

We first describe some subroutines to be used in the protocol. We construct polynomial time algorithms Record, Reconstruct, and Decide which perform the following:

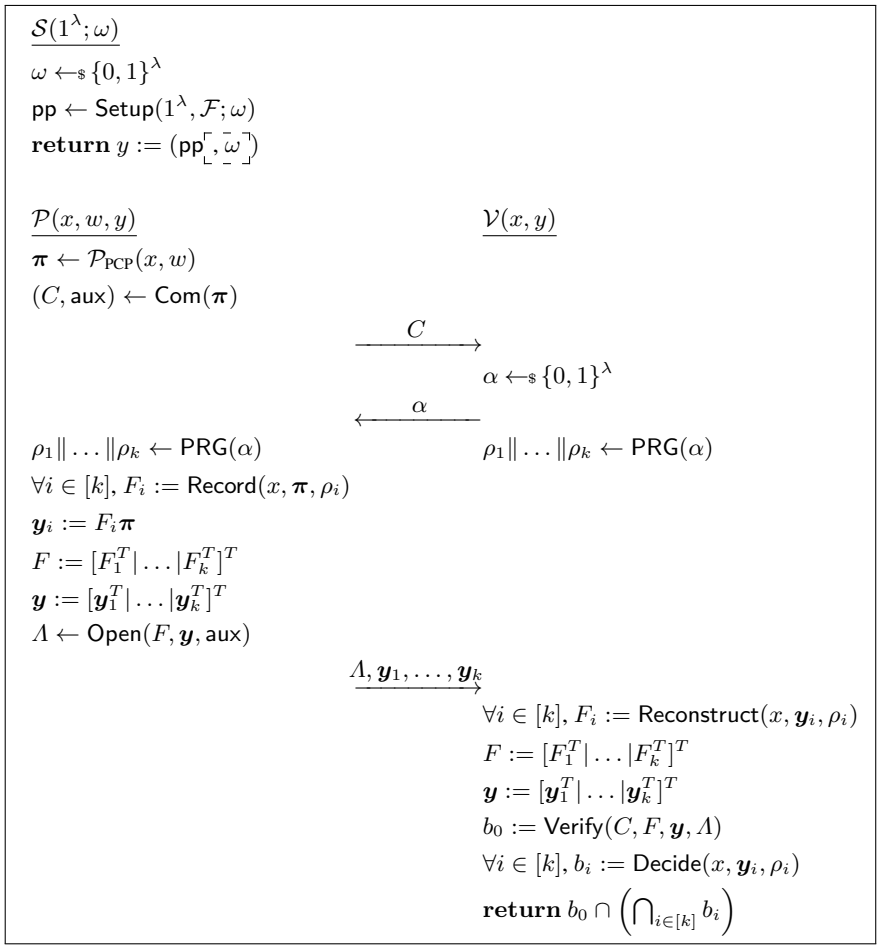

Fig. 1: Succinct Argument of Knowledge for NP

- Record: On input a statement x, a proof $\boldsymbol{\pi}$, a randomness ρ, it runs $\mathcal{V}_{\text{PCP}}^{\boldsymbol{\pi}}(x;\rho)$ and records the queries $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_h \in \mathbb{F}^q$ made by $\mathcal{V}_{\text{PCP}}$. It outputs a query matrix $F := [\boldsymbol{f}_1 | \ldots | \boldsymbol{f}_h]^T \in \mathbb{F}^{h \times q}$.
- Reconstruct: On input a statement x, a response vector $\boldsymbol{y} \in \mathbb{F}^h$, and a randomness ρ, it runs $\mathcal{V}_{\text{PCP}}^{\boldsymbol{\pi}}(x;\rho)$ by simulating the oracle $\boldsymbol{\pi}$ using the response vector $\boldsymbol{y}$. That is, when $\mathcal{V}_{\text{PCP}}$ makes the i-th query $\boldsymbol{f}_i$ for $i \in [h]$, it responds by returning the value y_i. It outputs a query matrix $F := [\boldsymbol{f}_1 | \ldots | \boldsymbol{f}_h]^T \in \mathbb{F}^{h \times q}$.
- Decide: On input a statement x, a response vector $\boldsymbol{y} \in \mathbb{F}^h$, it runs $\mathcal{V}_{\text{PCP}}^{\boldsymbol{\pi}}(x;\rho)$ by simulating the oracle $\boldsymbol{\pi}$ as in Reconstruct, and outputs whatever $\mathcal{V}_{\text{PCP}}^{\boldsymbol{\pi}}(x;\rho)$ outputs.

It is clear that for any strings x and $\boldsymbol{\pi}$ and randomness ρ, if $\boldsymbol{y}$ is formed in such a way that y_i is the response to the i-th query made by $\mathcal{V}_{\text{PCP}}^{\boldsymbol{\pi}}(x;\rho)$, then $\mathsf{Record}(x, \boldsymbol{\pi}, \rho) = \mathsf{Reconstruct}(x, \boldsymbol{y}, \rho)$, and $\mathsf{Decide}(x, \boldsymbol{y}, \rho) = \mathcal{V}_{\text{PCP}}^{\boldsymbol{\pi}}(x;\rho)$.

We now describe the protocol. The setup algorithm $\mathcal{S}$ samples a random string ω and computes the public parameters pp of FC using ω. It outputs pp if an FC with trusted setup is used, which results in an argument system with private-coin setup. Alternatively, if an FC without trusted setup is used, it outputs additionally ω (as highlighted in the dashed box). This results in a public-coin setup.

In the rest of the protocol, the verifier is entirely public-coin. On input the public parameter pp, the statement x and the witness w, the prover $\mathcal{P}$ produces $\boldsymbol{\pi}$ as the PCP encoding of the witness w, then it commits to $\boldsymbol{\pi}$ and sends its commitment C to the verifier $\mathcal{V}$. Upon receiving the commitment C, $\mathcal{V}$ responds with a random string α. The prover $\mathcal{P}$ stretches α with a PRG into $\rho_1 \| \ldots \| \rho_k$ and executes $\mathcal{V}_{\text{PCP}}$ on ρ_i for $i \in [k]$.

The prover $\mathcal{P}$ then records the sets of queries $F_i = \mathsf{Record}(x, \boldsymbol{\pi}, \rho_i)$ of $\mathcal{V}_{\text{PCP}}$ using randomness ρ_i to $\boldsymbol{\pi}$, and computes the responses $\boldsymbol{y}_i = F_i \boldsymbol{\pi}$. Let F and $\boldsymbol{y}$ be the vertical concatenation of F_i and $\boldsymbol{y}_i$ respectively for $i \in [k]$. It computes the opening Λ of the commitment C to the tuple $(F, \boldsymbol{y})$. The opening Λ along with the responses $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k$ are sent to the verifier $\mathcal{V}$.

The verifier $\mathcal{V}$ runs $\mathsf{Reconstruct}(x, \boldsymbol{y}_i, \rho_i)$ to reconstruct the query matrices F_i for each $i \in [k]$. It then reconstructs the tuple $(F, \boldsymbol{y})$ and checks if Λ is a valid opening of C to $(F, \boldsymbol{y})$. Finally, it checks if $\mathsf{Decide}(x, \boldsymbol{y}_i, \rho_i)$ returns 1 for all $i \in [k]$. If all checks are passed, it outputs 1. Otherwise, it outputs 0.

5.2 Analysis

Clearly, if $(\mathcal{P}_{\text{PCP}}, \mathcal{V}_{\text{PCP}})$ is a linear PCP, and FC is an LMC, then the argument system is complete. Alternatively, if $(\mathcal{P}_{\text{PCP}}, \mathcal{V}_{\text{PCP}})$ is a traditional PCP, and FC is an SVC, then the system is also complete. The succinctness of the system follows directly from the compactness of FC. Next, we show that the argument system is of knowledge by the following theorem. Due to space constraints, we refer to Section C.1 for a full proof.

Let FSamp be the following algorithm. It takes as input a statement x, vectors $(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k)$, and randomness α. It runs $\rho_1 \| \ldots \| \rho_k \leftarrow \mathsf{PRG}(\alpha)$, and $F_i \leftarrow \mathsf{Reconstruct}(x, \boldsymbol{y}_i, \rho_i)$ for $i \in [k]$. It then sets F to be the vertical concatenation of $F_1, \ldots, F_k$, and outputs F.

Theorem 7. *Let $(\mathcal{P}_{PCP}, \mathcal{V}_{PCP})$ be a witness extractable PCP for NP, PRG be a pseudo-random generator, and $\mathsf{FC} := (\mathsf{Setup}, \mathsf{Com}, \mathsf{Open}, \mathsf{Verify})$ be weakly function binding with respect to FSamp (resp. without trusted setup). Let $k = O(\lambda)$. Then the protocol in Figure 1 is a (resp. public-coin) argument of knowledge.*

By Theorem 5, a function binding FC is also weakly function binding with respect to any function sampler. We therefore have the following corollary.

Corollary 1. *Let $(\mathcal{P}_{PCP}, \mathcal{V}_{PCP})$ be a witness extractable PCP for NP, PRG be a pseudo-random generator, and $\mathsf{FC} := (\mathsf{Setup}, \mathsf{Com}, \mathsf{Open}, \mathsf{Verify})$ be function binding (resp. without trusted setup). Let $k = O(\lambda)$. Then the protocol in Figure 1 is a (resp. public-coin) argument of knowledge.*

6 Subvector Commitments

Subvector commitments (SVC) capture a specific class of functional commitments that allow the prover to open to subvectors of the committed vector. The proofs are compact in the sense that do not depend neither on the size of the committed vector, nor on the size of the output subvector. SVCs constitute the main cryptographic building block to instantiate succinct arguments using standard PCPs in Section 5.

We give two direct constructions of SVC, one from modules over Euclidean rings where certain variants of the root assumption hold, and one from pairing groups where the CDH assumption holds. Our constructions are inspired by the work of Catalano and Fiore [24] and extend the opening algorithms of their vector commitment schemes to simultaneously handle multiple positions. These modifications introduce several complications in the security proofs that require a careful manipulation of the exponents. It is worth noting that SVC can also be generically constructed from linear map commitments (*e.g.*, those constructed in Section 7), by restricting the class of linear maps. However, the resulting schemes would require pairing groups and somewhat stronger number-theoretic assumptions (see Section 7 for further details).

6.1 SVC from Modules over Euclidean Rings

Our first SVC scheme relies on modules over Euclidean rings where some variants of the root problem (the natural generalization of the RSA problem over composite order groups) is hard. Let $q \in \mathsf{poly}(\lambda)$ be a positive integer. Let MGen be an efficient module sampling algorithm as defined in Section 3 and let R be an Euclidean ring sampled by MGen. Let $\mathrm{IRR}_\lambda(R)$ be a set of prime elements in R of size 2^λ. Let $H : \{0,1\}^* \to \mathrm{IRR}_\lambda(R)^q$ be a prime-valued function which maps finite bit strings to tuples of q distinct elements in $\mathrm{IRR}_\lambda(R)$. That is, for all string $s \in \{0,1\}^*$, if $(e_1, \ldots, e_q) = H(s)$, then $e_i \neq e_j$ for all $i, j \in [q]$ where $i \neq j$. Let $\mathcal{X} := \{0_R, 1_R\}$[1] where 0_R and 1_R are the additive and multiplicative identity elements of R respectively. We construct our first subvector commitment scheme in Figure 2.

Note that in the opening algorithm, it is required to compute

$$\Lambda_I := \left(\prod_{i \in I} e_i\right)^{-1} \circ \langle \boldsymbol{x}_{[q]\setminus I}, \boldsymbol{S}_{[q]\setminus I} \rangle.$$

Although multiplicative inverses of ring elements do not exist in general, and if so, they may be hard to compute, the above are efficiently computable because, for all $i \in [q] \setminus I$ and hence for all $i \in J \setminus I$, we have

$$S_i := \left(\prod_{j \in [q]\setminus\{i\}} e_j\right) \circ X = \left(\prod_{j \in I} e_j \prod_{j \in [q]\setminus(I \cup \{i\})} e_j\right) \circ X.$$

The correctness of the construction follows straightforwardly by inspection. Depending on the instantiation of H, we can prove our scheme secure against different assumptions:

[1] In general, $\mathcal{X}$ can be set such that for all $x, x' \in \mathcal{X}$, $\gcd(x - x', e_i) = 1$ for all $i \in [q]$.

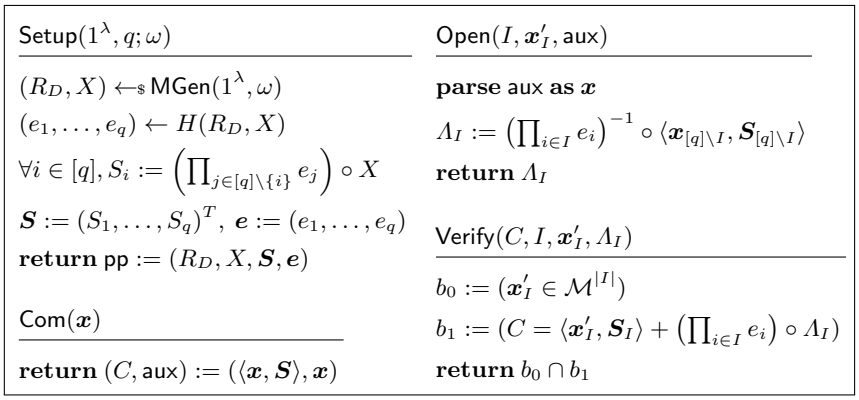

Fig. 2: SVC from variants of the root assumption.

- H is a (non-cryptographic) hash: Our construction is secure if the strong distinct-prime-product root assumption (introduced in Section 3) holds over the module family $\mathcal{R}_\mathcal{D}$. This is shown in Theorem 8.
- H is a random oracle: Our construction is secure if the adaptive root problem (introduced in [17]) is hard over the module family. This is shown in Theorem 9.

Theorem 8. *If the strong distinct-prime-product root assumption holds over the module family $\mathcal{R}_\mathcal{D}$ (resp. without trusted setup), then the scheme in Figure 2 is position binding (resp. without trusted setup).*

Proof. Suppose not, let $\mathcal{A}$ be a PPT adversary such that

$$\Pr\left[\begin{array}{l|c} \mathsf{Verify}(C, I, \boldsymbol{x}_I, \Lambda_I) = 1 & \omega \leftarrow_\$ \{0,1\}^\lambda \\ \mathsf{Verify}(C, J, \boldsymbol{x}'_J, \Lambda'_J) = 1 & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, q; \omega) \\ \exists i \in I \cap J \ s.t.\ x_i \neq x'_i & (C, I, J, \boldsymbol{x}_I, \boldsymbol{x}'_J, \Lambda_I, \Lambda'_J) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp}, \omega) \end{array}\right] > \frac{1}{f(\lambda)}$$

for some polynomial $f(\lambda) \in \mathsf{poly}(\lambda)$, where $\mathcal{A}$ gets ω as input (highlighted by the dashed box) only in the variant without trusted setup. We construct an algorithm $\mathcal{C}$ as follows, whose existence contracts the fact that $\mathcal{R}_\mathcal{D}$ is a strong distinct-prime-product root modules family (without trusted setup).

In the with-trusted-setup setting, $\mathcal{C}$ receives as input (R_D, A) generated by $\mathsf{MGen}(1^\lambda; \omega)$ for some $\omega \leftarrow_\$ \{0,1\}^\lambda$. It sets $X := A$, and computes $(e_1, \ldots, e_q) \leftarrow H(R_D, X)$. It then sets $S_i := \left(\prod_{j \in [q] \setminus \{i\}} e_j\right) \circ X$ for all $i \in [q]$, $\boldsymbol{S} := (S_1, \ldots, S_q)^T$, and $\boldsymbol{e} := (e_1, \ldots, e_q)$. It sets $\mathsf{pp} := (R_D, X, \boldsymbol{S}, \boldsymbol{e})$ and runs $\mathcal{A}$ on input $(1^\lambda, \mathsf{pp})$. In the without-trusted-setup setting, $\mathcal{C}$ receives additionally ω and runs $\mathcal{A}$ on $(1^\lambda, \mathsf{pp}, \omega)$ instead. In any case, it is clear that pp and ω obtained above distribute identically as

$$\{(\mathsf{pp}, \omega) : \omega \leftarrow_\$ \{0,1\}^\lambda; \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, q; \omega)\}_\lambda.$$

Hence, with probability at least $1/f(\lambda)$, $\mathcal{C}$ obtains $(C, I, J, \boldsymbol{x}_I, \boldsymbol{x}'_J, \Lambda_I, \Lambda'_J)$ such that

$$\langle \boldsymbol{x}_I, \boldsymbol{S}_I \rangle + \left(\prod_{i \in I} e_i\right) \circ \Lambda_I = \langle \boldsymbol{x}'_J, \boldsymbol{S}_J \rangle + \left(\prod_{i \in J} e_i\right) \circ \Lambda'_J$$

which implies

$$\begin{aligned}&\langle \boldsymbol{x}_{I\setminus J}, \boldsymbol{S}_{I\setminus J}\rangle - \langle \boldsymbol{x}'_{J\setminus I}, \boldsymbol{S}_{J\setminus I}\rangle + \langle \boldsymbol{x}_{I\cap J} - \boldsymbol{x}'_{I\cap J}, \boldsymbol{S}_{I\cap J}\rangle \\ =&\left(\prod_{i\in I\cap J} e_i\right)\left(\left(\prod_{i\in J\setminus I} e_i\right)\circ \Lambda'_J - \left(\prod_{i\in I\setminus J} e_i\right)\circ \Lambda_I\right).\end{aligned}$$

Recall that $S_i = \left(\prod_{j\in[q]\setminus\{i\}} e_j\right)\circ A$. Define $\delta_i := \begin{cases} x_i & i\in I\setminus J \\ -x'_i & i\in J\setminus I \\ x_i - x'_i & i\in I\cap J\end{cases}$ and $\Lambda := \left(\left(\prod_{i\in J\setminus I} e_i\right)\circ \Lambda'_J - \left(\prod_{i\in I\setminus J} e_i\right)\circ \Lambda_I\right)$. $\mathcal{C}$ obtains

$$\left(\sum_{i\in I\cup J}\delta_i \prod_{j\in[q]\setminus\{i\}} e_j\right)\circ A = \left(\prod_{i\in I\cap J} e_i\right)\circ \Lambda.$$

Let $K_0 := \{i\in I\cap J : \delta_i = 0_R\}$ and $K_1 := \{i\in I\cup J : \delta_i \neq 0_R\}$. Next, we show that $d := \gcd\left(\sum_{i\in I\cup J}\delta_i\prod_{j\in[q]\setminus\{i\}} e_j, \prod_{i\in I\cap J} e_i\right) = \prod_{j\in K_0} e_j$. Furthermore, suppose that this is the case, we have $(I\cap J)\setminus K_0 \neq \emptyset$ since there exists $i\in I\cap J$ such that $\delta_i = x_i - x'_i \neq 0_R$. To prove the above, we first note that

$$\sum_{i\in I\cup J}\delta_i \prod_{j\in[q]\setminus\{i\}} e_j = \sum_{i\in K_1}\delta_i \prod_{j\in[q]\setminus\{i\}} e_j = \prod_{j\in[q]\setminus(I\cup J)} e_j \left(\sum_{i\in K_1}\delta_i \prod_{j\in(I\cup J)\setminus\{i\}} e_j\right).$$

Hence

$$\begin{aligned}d &= \gcd\left(\sum_{i\in K_1}\delta_i \prod_{j\in(I\cup J)\setminus\{i\}} e_j, \prod_{i\in I\cap J} e_i\right) \\ &= \prod_{j\in K_0} e_j \cdot \gcd\left(\sum_{i\in K_1}\delta_i \prod_{j\in(I\cup J)\setminus(K_0\cup\{i\})} e_j, \prod_{i\in(I\cap J)\setminus K_0} e_i\right).\end{aligned}$$

It remains to show that $d' := \gcd\left(\sum_{i\in K_1}\delta_i\prod_{j\in(I\cup J)\setminus(K_0\cup\{i\})} e_j, \prod_{i\in(I\cap J)\setminus K_0} e_i\right) = 1_R$. Suppose not, let $d' = \prod_{i\in L} e_i$ for some $L\subseteq (I\cap J)\setminus K_0$. Suppose $\ell\in L\neq\emptyset$. This means $\delta_\ell \neq 0_R$ and hence $\ell\in K_1$. Then there exists $r\in R$ such that

$$\begin{aligned}e_\ell \cdot r =& \sum_{i\in K_1}\delta_i \prod_{j\in(I\cup J)\setminus(K_0\cup\{i\})} e_j \\ =&\delta_\ell \prod_{j\in(I\cup J)\setminus(K_0\cup\{\ell\})} e_j + e_\ell \sum_{i\in K_1\setminus\{\ell\}}\delta_i \prod_{j\in(I\cup J)\setminus(K_0\cup\{i\})} e_j.\end{aligned}$$

Let $r' := r - \sum_{i\in K_1\setminus\{\ell\}}\delta_i\prod_{j\in(I\cup J)\setminus(K_0\cup\{i\})} e_j$. We have

$$e_\ell\cdot r' = \delta_\ell \prod_{j\in(I\cup J)\setminus(K_0\cup\{\ell\})} e_j.$$

Since $\delta_\ell \neq 0_R$, *i.e.*, $\delta_\ell \in \{-1_R, 1_R\}$, the above contradicts the fact that e_ℓ is a prime element. Thus we must have $L = \emptyset$ and hence $d' = 1_R$.

Now that we have concluded $d = \gcd\left(\sum_{i\in I\cup J} \delta_i \prod_{j\in[q]\setminus\{i\}} e_j, \prod_{i\in I\cap J} e_i\right) = \prod_{j\in K_0} e_j$, $\mathcal{C}$ can use the extended Euclidean algorithm to find $a, b \in R$ such that

$$a \sum_{i\in I\cup J} \delta_i \prod_{j\in[q]\setminus\{i\}} e_j + b \prod_{i\in I\cap J} e_i = \prod_{j\in K_0} e_j.$$

Multiplying this to A, it gets

$$\begin{aligned}\left(\prod_{j\in K_0} e_j\right) \circ A &= \left(a \sum_{i\in I\cup J} \delta_i \prod_{j\in[q]\setminus\{i\}} e_j + b \prod_{i\in I\cap J} e_i = \prod_{j\in K_0} e_j\right) \circ A \\ &= \left(a \prod_{i\in I\cap J} e_i\right) \circ \Lambda + \left(b \prod_{i\in I\cap J} e_i\right) \circ A \\ &= \left(\prod_{i\in I\cap J} e_i\right)(a \circ \Lambda + b \circ A).\end{aligned}$$

Since $(I \cap J) \setminus K_0 \neq \emptyset$, $\mathcal{C}$ can set $S := (I \cap J) \setminus K_0$ and $Y := (a \circ \Lambda + b \circ A)$, and output $(\{e_i\}_{i\in S}, Y)$ as a solution to the strong distinct-prime-product root problem. □

Remark 1. Note that the proof of Theorem 8 does not rely on any properties of the function H other than that it outputs distinct primes. In particular, the primes output by H need not be (pseudo)random.

Theorem 9. *If the adaptive root assumption holds over the module family $\mathcal{R}_\mathcal{D}$ with respect to IRR_λ (resp. without trusted setup), then the scheme in Figure 2 is position binding (without trusted setup) in the random oracle model.*

Proof. The proof is similar to that of Theorem 8 except with a few changes which we highlight below. Let $H : \{0,1\}^* \to \mathrm{IRR}_\lambda(R)^q$ be modeled as a random oracle to which $\mathcal{A}$ has oracle access. Similar to the proof of Theorem 8, we will construct an algorithm $\mathcal{C}$ whose existence contracts the fact that the adaptive root assumption holds over $\mathcal{R}_\mathcal{D}$ (without trusted setup) in the random oracle model.

$\mathcal{C}$ simulates the public parameters pp slightly differently. In the with-trusted-setup setting, $\mathcal{C}$ receives as input (R_D, A) generated by $\mathsf{MGen}(1^\lambda; \omega)$ for some $\omega \leftarrow\!\!\$\ \{0,1\}^\lambda$. It sets $X := A$, and receives a random prime element $e \leftarrow\!\!\$\ \mathrm{IRR}_\lambda(R)$. $\mathcal{C}$ chooses a random index $i^* \leftarrow\!\!\$\ [q]$, and sets $e_{i^*} := e$. For the other indices $i \in [q]$ with $i \neq i^*$, it samples $e_i \leftarrow\!\!\$\ \mathrm{IRR}_\lambda(R)$ with the constraint that $e_i \neq e_j$ for all $i, j \in [q]$ where $i \neq j$. It then sets $S_i := \left(\prod_{j\in[q]\setminus\{i\}} e_j\right) \circ X$ for all $i \in [q]$, $\boldsymbol{S} := (S_1, \ldots, S_q)^T$, and $\boldsymbol{e} := (e_1, \ldots, e_q)$. It sets $\mathsf{pp} := (R_D, X, \boldsymbol{S}, \boldsymbol{e})$ and runs $\mathcal{A}^H$ on input $(1^\lambda, \mathsf{pp})$. $\mathcal{C}$ simulates the random oracle H for $\mathcal{A}$ by programming $H(R_D, X) := (e_1, \ldots, e_q)$, and answering all other queries by sampling random distinct primes from $\mathrm{IRR}(R)^q$. In the setup-free setting, $\mathcal{C}$ receives additionally ω and runs $\mathcal{A}^H$ on $(1^\lambda, \mathsf{pp}, \omega)$ instead.

$\mathsf{Setup}(1^\lambda, q; \omega)$

$(p, \mathbb{G}, \mathbb{G}_T, G, e) \leftarrow \mathsf{GGen}(1^\lambda; \omega)$

$\forall i \in [q],\ z_i \leftarrow\!\!\$\ \mathbb{Z}_p$

$\forall i, i' \in [q],\ G_i := G^{z_i},\ H_{i,i'} := G^{z_i z_{i'}}$

$\mathsf{pp} := \begin{pmatrix} p, \mathbb{G}, \mathbb{G}_T, G, \{G_i\}_{i\in[q]}, \\ \{H_{i,i'}\}_{i,i'\in[q], i\neq i'}, e \end{pmatrix}$

return pp

$\mathsf{Com}(\boldsymbol{x})$

return $(C, \mathsf{aux}) := \left(\prod_{i\in[q]} G_i^{x_i}, \boldsymbol{x} \right)$

$\mathsf{Open}(I, \boldsymbol{x}'_I, \mathsf{aux})$

parse aux **as** $\boldsymbol{x}$

return $\Lambda_I := \prod_{i\in I} \prod_{i'\notin I} H_{i,i'}^{x_{i'}}$

$\mathsf{Verify}(C, I, \boldsymbol{x}'_I, \Lambda_I)$

$b_0 := (\boldsymbol{x}'_I \in \mathcal{X}^{|I|})$

$b_1 := \left(e\left(\frac{C}{\prod_{i\in I} G_i^{x_i}}, \prod_{i\in I} G_i \right) = e(\Lambda_I, G) \right)$

return $b_0 \cap b_1$

Fig. 3: SVC from CDH.

Using the same argument as in the proof of Theorem 8, with probability at least $1/f(\lambda)$ for some $f \in \mathsf{poly}(\lambda)$, $\mathcal{C}$ obtains a tuple $(\{e_i\}_{i\in S}, Y)$ such that $\left(\prod_{i\in S} e_i\right) \circ Y = X$ (since $X = A$). Conditioned on this event, with probability at least $1/q$, it holds that $i^* \in S$. If that is the case, then $\mathcal{C}$ sets $Y' := \left(\prod_{i\in S\setminus\{i^*\}} e_i\right) \circ Y$ which satisfies $e \circ Y' = e_{i^*} \circ Y' = X$. It thus output Y' as a solution to the adaptive root problem. □

Optimizations. We observe that the values $S_1, \ldots, S_q$ do not depend on the committed vector and can be precomputed by both parties. Also, the two assumptions offer a tradeoff in terms of verifier efficiency: The main workload for the verifier is to compute the term $\left(\prod_{i\in I} e_i\right) \circ \Lambda_I$. Assuming $R = \mathbb{Z}$, and the term is computed by repeated squaring, the complexity of the computation depends on the bit-length of the primes e_i. In the adaptive root assumption, the primes $(e_1, \ldots, e_q)$ are sampled randomly from a set of primes of size 2^λ, therefore representing each prime requires at least λ bits. On the other hand, under the strong distinct-prime-product root assumption we can set $(e_1, \ldots, e_q)$ to be the smallest q primes. Since $q \in \mathsf{poly}(\lambda)$, each prime can be represented by $O(\log \lambda)$ bits. This greatly reduces the computational effort of the verifier.

6.2 SVC from the Computational Diffie-Hellman Assumption

Next we present our SVC construction from pairing groups. The public parameters consist of a set of random elements $\{G_i = G^{z_i}\}_{i\in[q]}$ and their pairwise "Diffie-Hellman products" $H_{i,i'} = G^{z_i z_{i'}}$ with $i \neq i'$. To commit to a vector $\boldsymbol{x}$ one computes $C := \prod_i G_i^{x_i}$. The opening of a subvector $\boldsymbol{x}_I$ is then $\prod_{i\in I} \prod_{i'\notin I} H_{i,i'}^{x_{i'}}$. Note that since $i \in I$ and $i' \notin I$, it is always true that $i \neq i'$. Therefore the product is efficiently computable for an honest prover. The relation can be easily checked using the pairing. A shortcoming of this scheme is that the public parameters grow quadratically with the vector size q.

Let GGen be an efficient bilinear group sampling algorithm (defined in Section B). Let $(p, \mathbb{G}, \mathbb{G}_T, G, e)$ be a group description output by GGen. Let $\mathcal{X} := \mathbb{Z}_p$. Our second subvector commitment scheme is shown in Figure 3. In the following we show that our SVC scheme is position binding. Since the public parameters are highly structured, we can only prove function binding in the presence of a trusted-setup.

Theorem 10. *If the computational Diffie-Hellman (CDH) assumption holds with respect to* GGen*, then the scheme in Figure 3 is position binding.*

Proof. Suppose not, let $\mathcal{A}$ be a PPT adversary such that

$$\mathsf{Pr}\left[\begin{array}{l}\mathsf{Verify}(C, I, \boldsymbol{x}_I, \Lambda_I) = 1 \\ \mathsf{Verify}(C, J, \boldsymbol{x}'_J, \Lambda'_J) = 1 \\ \exists i \in I \cap J \ s.t. \ x_i \neq x'_i\end{array}\middle|\begin{array}{c}\omega \leftarrow_{\$} \{0,1\}^\lambda \\ \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, q; \omega) \\ (C, I, J, \boldsymbol{x}_I, \boldsymbol{x}'_J, \Lambda_I, \Lambda'_J) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp})\end{array}\right] > \frac{1}{f(\lambda)}$$

for some $f(\lambda) \in \mathsf{poly}(\lambda)$. We construct a square-DH solver $\mathcal{C}$, which implies a CDH solver [6], as follows.

$\mathcal{C}$ receives as input $(p, \mathbb{G}, \mathbb{G}_T, G, H, e)$, where $(p, \mathbb{G}, \mathbb{G}_T, G, e) \leftarrow \mathsf{GGen}(1^\lambda)$ and $H = G^z$ for some random $z \leftarrow_{\$} \mathbb{Z}_p$, and must output G^{z^2}. It picks an index $i^* \leftarrow_{\$} [q]$ and set $G_{i^*} := H$. Symbolically, let $z_{i^*} := z$, which is not known by $\mathcal{C}$. For the other indices $i, i' \in [q] \setminus \{i^*\}$, it samples $z_i \leftarrow_{\$} \mathbb{Z}_p$ and sets $G_i := G^{z_i}$ and $H_{i,i'} := G^{z_i z_{i'}}$. It also sets $H_{i^*,i} = H_{i,i^*} = G^{z z_i}$ for each $i \in [q] \setminus \{i^*\}$. It then sets $\mathsf{pp} = (p, \mathbb{G}, \mathbb{G}_T, G, \{G_i\}_{i\in[q]}, \{H_{i,i'}\}_{i,i'\in[q], i\neq i'}, e)$, which is identically distributed as pp output by Setup. $\mathcal{C}$ runs $\mathcal{A}$ on input $(1^\lambda, \mathsf{pp})$. With probability at least $1/f(\lambda)$, it obtains $(C, I, J, \boldsymbol{x}_I, \boldsymbol{x}'_J, \Lambda_I, \Lambda'_J)$ such that $\mathsf{Verify}(C, I, \boldsymbol{x}_I, \Lambda_I) = 1$, $\mathsf{Verify}(C, J, \boldsymbol{x}'_J, \Lambda'_J) = 1$, and $\exists i \in I \cap J \ s.t. \ x_i \neq x'_i$. Conditioning on the above, with probability $1/q$, it holds that $i^* \in I \cap J$ and $x_{i^*} \neq x'_{i^*}$. By examining the verification equations, we have

$$\begin{aligned} e(\prod_{i\in I} G_i^{x_i}, \prod_{i\in I} G_i) \cdot e(\Lambda_I, G) &= e(\prod_{i\in J} G_i^{x'_i}, \prod_{i\in J} G_i) \cdot e(\Lambda_J, G) \\ e(\prod_{i\in J} G_i^{x'_i}, \prod_{i\in J} G_i) \cdot e(\prod_{i\in I} G_i^{-m_i}, \prod_{i\in I} G_i) &= e(\Lambda, G), \text{ where } \Lambda := \Lambda_I/\Lambda_J \\ (\sum_{i\in J} z_i x'_i)(\sum_{i\in J} z_i) - (\sum_{i\in I} z_i x_i)(\sum_{i\in I} z_i) &= \log_G \Lambda \\ \alpha z_{i^*}^2 + \beta z_{i^*} + \gamma &= \log_G \Lambda \end{aligned}$$

where

$$\alpha := (x'_{i^*} - x_{i^*}) \qquad \beta := \sum_{i\in J\setminus\{i^*\}} z_i(x'_i + x'_{i^*}) - \sum_{i\in I\setminus\{i^*\}} z_i(x_i + x_{i^*})$$

$$\gamma := (\sum_{i\in J\setminus\{i^*\}} z_i x'_i)(\sum_{i\in J\setminus\{i^*\}} z_i) - (\sum_{i\in I\setminus\{i^*\}} z_i x_i)(\sum_{i\in I\setminus\{i^*\}} z_i)$$

are computable by $\mathcal{C}$ since they do not depend on $z = z_{i^*}$. $\mathcal{C}$ then outputs $G^{z^2} = \left(\frac{\Lambda}{H^\beta G^\gamma}\right)^{1/\alpha}$ which is the solution to the square-DH instance. □

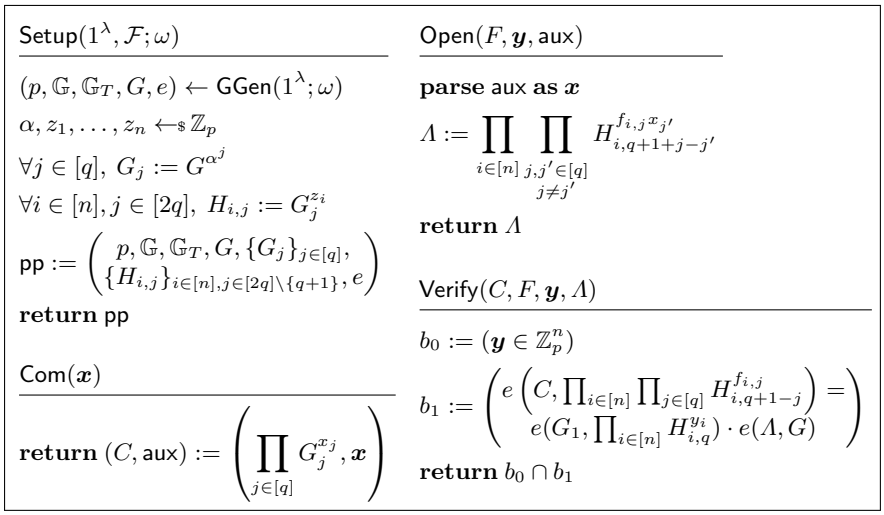

Fig. 4: Function Binding LMC in Generic Group Model.

6.3 Instantiations and Resulting SNARKs.

For our first construction, regardless of the assumption used, we can instantiate the construction over $Cl(\Delta)$, the class group of an imaginary quadratic order with discriminant $\Delta \in \mathbb{Z}$. With a 2048-bit Δ, which offers roughly 100 bits of security, each element in $Cl(\Delta)$ can be represented by at most 2048 bits. For more details, we refer to Section 8. Assuming a statistical security parameter $k = 80$, and a PCP which checks 3 bits per query, the resulting SNARK has proof size $2 \cdot 2048 + 3 \cdot 80 = 4336$ bits. For our second construction, we can instantiate the construction over the elliptic curve "BN128" [7], which offers also roughly 100 bits of security. In BN128, each group element can be represented by 256 bits. Therefore the resulting proof size is $2 \cdot 256 + 3 \cdot 80 = 752$ bits.

7 Linear Map Commitments

Linear map commitments (LMC) are functional commitments for linear maps. In contrast to functional commitments for linear forms [49], they allow the prover to open to a set of linear forms, or equivalently a linear map, computed over the committed vector. LMCs allow us to extend our argument system construction to be based on linear PCPs.

Inspired by the construction of FC for linear forms by Libert, Ramanna, and Yung [49], we give two constructions of LMC from pairing groups. The constructions are based on the following observations. First, when the vectors $\boldsymbol{x}, \boldsymbol{f} \in \mathbb{F}^q$ for some field $\mathbb{F}$ are encoded as the polynomials $\sum_{j\in[q]} x_j \alpha^j$ and $\sum_{j\in[q]} f_j \alpha^{q+1-i}$ with variable α respectively, their inner product is the coefficient of the monomial α^{q+1} in the polynomial product $\left(\sum_{j\in[q]} x_j \alpha^j\right)\left(\sum_{j\in[q]} f_j \alpha^{q+1-j}\right)$. Second, due to linearity of polynomial multiplication, if a matrix $F \in \mathbb{F}^{n\times q}$ is encoded in the polynomial

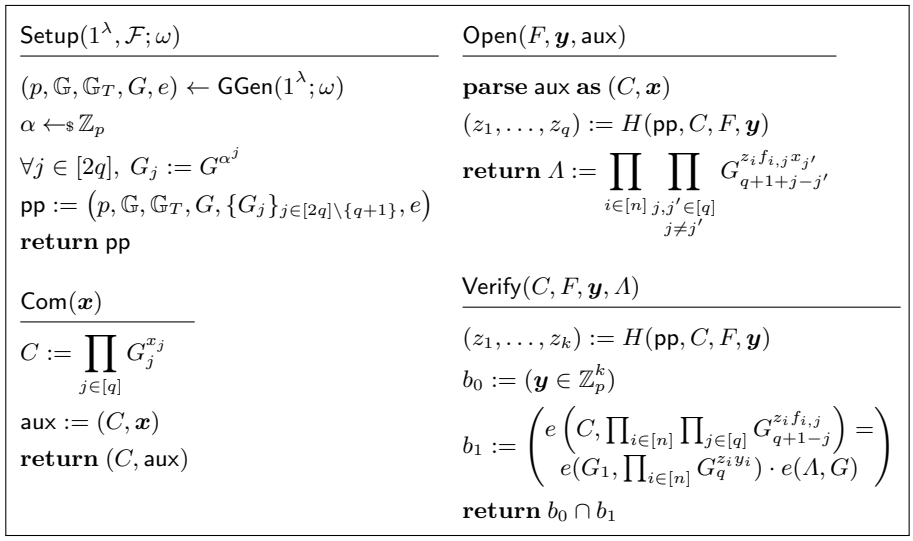

Fig. 5: Weakly Function Binding LMC from q-DHE in Random Oracle Model.

$\sum_{i \in [n], j \in [q]} f_{i,j} z_i \alpha^{q+1-i}$ with variables $(\alpha, z_1, \ldots, z_n)$, then the matrix-vector product $F\boldsymbol{x}$ is given in the coefficients of the monomials $z_i \alpha^{q+1}$ for $i \in [n]$ in the polynomial $\left(\sum_{j \in [q]} x_j \alpha^j\right) \left(\sum_{i \in [n], j \in [q]} f_{i,j} z_i \alpha^{q+1-j}\right)$.

The first construction can be proven function binding rather easily in the generic bilinear group model. The second construction is *weakly* function binding with respect to any unpredictable sampler under the q-DHE assumption in the random oracle model. It is worth mentioning that the public parameters in the second construction grow only linearly in the size of q whereas those in the first grow quadratically. Both constructions are compact by inspection and require a trusted setup.

Let GGen be an efficient bilinear group sampling algorithm. Let $(p, \mathbb{G}, \mathbb{G}_T, G, e)$ be a group description output by GGen. Let $\mathcal{X} := \mathbb{Z}_p$. Let $H : \{0,1\}^* \to \mathbb{Z}_p^q$ be a hash function to be modeled as a random oracle (only used in the second construction). Our two LMC constructions are given in Figure 4 and Figure 5, respectively. We show that our constructions are function binding and weakly function binding, respectively. Due to space constraints, we defer the formal argument to Section C.2 and Section C.3.

Theorem 11. *Let $q \in \mathsf{poly}(\lambda)$ and $1/p \in \mathsf{negl}(\lambda)$. The scheme in Figure 4 is function binding in the generic bilinear group model.*

Theorem 12. *Let* FSamp *be an unpredictable sampler for $\mathcal{F}$. If the q-DHE assumption holds with respect to* GGen*, then the scheme in Figure 5 is weakly function binding with respect to* FSamp *in the random oracle model.*

8 Candidate Module Families

In the following we suggest some candidate instantiations for modules (specifically groups) where the strong distinct-prime-root assumption and/or the adaptive root assumption are believed to hold.

8.1 Class Groups of Imaginary Quadratic Orders

The use of class groups in cryptography was first proposed by Buchmann and Williams [22]. We refer to, *e.g.*, [20,21], for more detailed discussions. We recall the basic properties of class groups necessary for our purpose. Let Δ be a negative integer such that $\Delta \equiv 0$ or $1 \pmod 4$. The ring $\mathcal{O}_\Delta := \mathbb{Z} + \frac{\Delta+\sqrt{\Delta}}{2}\mathbb{Z}$ is called an *imaginary quadratic order of discriminant* Δ. Its field of fractions is $\mathbb{Q}(\sqrt{\Delta})$. The discriminant is *fundamental* if $\Delta/4$ (resp. Δ) is square-free in the case of $\Delta \equiv 0 \pmod 4$ (resp. $\Delta \equiv 1 \pmod 4$). If Δ is fundamental, then $\mathcal{O}_\Delta$ is a *maximal* order. The *fractional ideals* of $\mathcal{O}_\Delta$ are of the form $q\left(a\mathbb{Z} + \frac{b+\sqrt{\Delta}}{2}\mathbb{Z}\right)$ with $q \in \mathbb{Q}$, $a \in \mathbb{Z}^+$, and $b \in \mathbb{Z}$, subject to the constraint that there exists $c \in \mathbb{Z}^+$ such that $\Delta = b^2 - 4ac$ and $\gcd(a, b, c) = 1$. A fractional ideal can therefore be represented by a tuple (q, a, b). If $q = 1$, then the ideal is called *integral* and can be represented by a tuple (a, b). An integral ideal (a, b) is *reduced* if it satisfies $-a < b \leq a \leq c$ and $b > 0$ if $a = c$. It is known that if an ideal (a, b) is reduced, then $a \leq \sqrt{|\Delta|/3}$. Two ideals $\mathfrak{a}, \mathfrak{b} \subseteq \mathcal{O}_\Delta$ are *equivalent* if there exists $0 \neq \alpha \in \mathbb{Q}(\sqrt{\Delta})$ such that $\mathfrak{b} = \alpha\mathfrak{a}$. It is known that, for each equivalence class of ideals, there exists exactly one reduced ideal which serves as the representative of the equivalence class. The set of equivalence classes of ideals equipped with ideal multiplication forms an Abelian group $Cl(\Delta)$ known as a *class group*.

Properties Useful in Cryptography. Since for all reduced ideals, $|b| \leq a \leq \sqrt{|\Delta|/3}$, $Cl(\Delta)$ is finite. For sufficiently large $|\Delta|$, no efficient algorithm is known for finding the cardinality of $Cl(\Delta)$, also known as the class number. Group operations can be performed efficiently, as there exist efficient algorithms for ideal multiplication and computing reduced ideals [20]. Assuming the extended Riemann hypothesis, $Cl(\Delta)$ is generated by the classes of all invertible prime ideals of norm smaller than $12(\log|\Delta|)^2$ [4], where the norm of a fractional ideal (q, a, b) is defined as q^2a ($= a$ for integral ideals). Since these ideals have norms logarithmic in $|\Delta|$, they can be found in polynomial time through exhaustive search. A random element can then be sampled by computing a power product of the elements in the generating set, with exponents randomly chosen from $[|\Delta|]$.

(Strong) Root Problem and its Variants in $Cl(\Delta)$. To recall, the strong root problem in $Cl(\Delta)$ is to find a prime $e \in \mathbb{Z}$ and a group element $Y \in Cl(\Delta)$ such that $Y^e = X$, for some given element $X \in Cl(\Delta)$. It is widely believed that root problems in $Cl(\Delta)$ for a large enough Δ are hard if the problem instances are sampled randomly with private coin [22]. Although the strong root problem in $Cl(\Delta)$ is not as well studied, it is shown to be hard for generic group algorithms [28]. The best attacks currently known are the ones for the root problem which runs in time proportional to $L_{|\Delta|}(\frac{1}{2}, 1)$ [41],

where $L_x(d, c) := \exp(c(\log x)^d(\log\log x)^{1-d})$. As discussed in [41], using a 2048-bit Δ offers approximately 100 bits of computational security.

The position binding property (resp. without trusted setup) of our first construction of SVC can be proven under either the strong distinct-prime-product root assumption (resp. without trusted setup) or the adaptive root assumption (resp. without trusted setup). Note that these two assumptions are somewhat "dual" to each other, in the sense that the former allows the adversary to choose which root it is going to compute, while the latter allows the adversary to choose the element whose root is to be found.

In the trusted-setup setting, it is clear that the strong distinct-prime-product root assumption is implied by the standard strong root assumption. In the setup-free setting, it is conjectured [60,17] that the adaptive root assumption holds in $Cl(\Delta)$. In the following, we first propose a simple candidate sampling algorithm MGen for sampling $Cl(\Delta)$ and random elements in $Cl(\Delta)$ with public coin, and then elaborate more about the strong distinct-prime-product root assumption with respect to MGen.

The sampling algorithm MGen first samples random integers of the appropriate length until it finds a fundamental discriminant Δ. Let $\{G_1, \ldots, G_k\}$ be a generating set of $Cl(\Delta)$. Our sampling algorithm samples random primes $c_1, \ldots, c_k \in [|\Delta|]$ subject to the constraint that the c_i's are pairwise coprime[2]. That is $\gcd(c_i, c_j) = 1$ for all $i, j \in [k]$ with $i \neq j$. The algorithm then outputs Δ along with $A = \prod_{i\in[k]} G_i^{c_i}$.

With the above restriction in place, it seems that the best strategy of finding an e-th root of A is to find an e-th root of G_i for all $i \in [k]$ simultaneously. On the other hand, the additional constraint seems necessary for the strong distinct-prime-product root problem with respect to A to be hard. Suppose that 1) there exists a subset $I = \{c_{i_1}, \ldots, c_{i_\ell}\} \subseteq [k]$ such that $\gcd(c_{i_1}, \ldots, c_{i_\ell}) = d \neq 1$; 2) d can be efficiently factorized into $\{e_i\}_{i\in S}$ such that $d = \prod_{i\in S} e_i$ for distinct primes $e_i \neq 1$; and 3) for all $j \in [k] \setminus I$, G_j can be efficiently represented as a product $G_j = \prod_{i\in I} G_i^{a_{i,j}}$ for some $a_{i,j}$. Then one can efficiently find a d-th root of A, say Y, and output $(\{e_i\}_{i\in S}, Y)$ as a solution to the strong distinct-prime-product root problem. Since it seems unreasonable to assume that d cannot be efficiently factorized into a product of distinct primes (see also the discussion of RSA-UFO below), nor is it sound to assume that none of the G_j can be represented with a power product of the G_i's where $i \neq j$, we impose the more reasonable restriction that the c_i's are pairwise coprime.

8.2 RSA Groups

RSA-based cryptosystems operate over $\mathbb{Z}_N^*$, the group of positive integers smaller and coprime with N, equipped with modular multiplication, where N is an integer with at least two distinct large prime factors. The security of these systems relies on the hardness of the (strong) root problem over $\mathbb{Z}_N^*$, known as the (strong) RSA assumption. Typically, N is chosen as a product of two secret distinct large primes p, q. However, the (strong) root problem over $\mathbb{Z}_N^*$ is easy if p and q are known. In other words, for N generated this way, the (strong) root assumption *without trusted setup* does not hold over $\mathbb{Z}_N^*$.

[2] This is assuming $k > 1$, else just set $c_1 = 1$.

RSA-UFOs. The problem of constructing RSA-based accumulators without trapdoors was considered by Sander [55], who proposed a way to generate (k, ϵ)-"generalized RSA moduli of unknown complete factionization (RSA-UFOs)" N which has at least two distinct k-bit prime factors with probability $1-\epsilon$, summarized as follows. Let $N_1, \ldots, N_r$ be random $3k$-bit integers with $r = O(\log 1/\epsilon)$. It is known that with constant probability N_i has at least two distinct k-bit prime factors [55]. It then follows that $N := \prod_{i\in[r]} N_i$ has at least two distinct k-bit prime factors. An important observation is that N can be generated with public coin, *e.g.*, using a random oracle. However, since N is a $3kr$-bit integer, any cryptosystem based on $\mathbb{Z}_N^*$ seems impractical. Nevertheless, one can show that strong RSA over RSA-UFO groups is implied by the standard strong RSA assumption in the presence of a random oracle. This result is implicitly shown by Sander [55] and a proof sketch is given in Section C.4.

References

1. Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, and Michal Zajkac. A subversion-resistant snark. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 3–33. Springer, 2017.
2. Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2087–2104. ACM, 2017.
3. Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of np. *Journal of the ACM (JACM)*, 45(1):70–122, 1998.
4. Eric Bach. Explicit bounds for primality testing and related problems. In *Mathematics of Computation*, volume 55 (191), pages 355–380, 1990.
5. Michael Backes, Manuel Barbosa, Dario Fiore, and Raphael M. Reischuk. ADSNARK: Nearly practical and privacy-preserving proofs on authenticated data. In IEEE S&P 2015 [42], pages 271–286.
6. Feng Bao, Robert H. Deng, and Huafei Zhu. Variations of Diffie-Hellman problem. In Sihan Qing, Dieter Gollmann, and Jianying Zhou, editors, *ICICS 03*, volume 2836 of *LNCS*, pages 301–312, Huhehaote, China, October 10–13, 2003. Springer, Heidelberg, Germany.
7. Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford Tavares, editors, *SAC 2005*, volume 3897 of *LNCS*, pages 319–331, Kingston, Ontario, Canada, August 11–12, 2006. Springer, Heidelberg, Germany.
8. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.
9. Eli Ben-Sasson, Iddo Bentov, Alessandro Chiesa, Ariel Gabizon, Daniel Genkin, Matan Hamilis, Evgenya Pergament, Michael Riabzev, Mark Silberstein, Eran Tromer, and Madars Virza. Computational integrity with a public random string from quasi-linear PCPs. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 551–579, Paris, France, May 8–12, 2017. Springer, Heidelberg, Germany.
10. Eli Ben-Sasson, Iddo Bentov, Ynon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 24, page 134, 2017.
11. Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In Ran Canetti and

Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 90–108, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.
12. Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60, Beijing, China, October 31 – November 3, 2016. Springer, Heidelberg, Germany.
13. Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 276–294, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.
14. Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the SNARK. *Journal of Cryptology*, 30(4):989–1066, October 2017.
15. Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 315–333, Tokyo, Japan, March 3–6, 2013. Springer, Heidelberg, Germany.
16. Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany.
17. Dan Boneh, Benedikt Bünz, and Ben Fisch. A survey of two verifiable delay functions. Technical report, Cryptology ePrint Archive, Report 2018/712, 2018. https://eprint. iacr. org/2018/712, 2018.
18. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Fischlin and Coron [35], pages 327–357.
19. Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.
20. Johannes Buchmann and Safuat Hamdy. A survey on iq-cryptography. In *Tech. Report TI-4/01, Technische Universitäat Darmstadt, Fachbereich Informatik*, 2000.
21. Johannes Buchmann, Tsuyoshi Takagi, and Ulrich Vollmer. Number field cryptography. In *High Primes and Misdemeanours: Lectures in Honour of the 60th Birthday of Hugh Cowie Williams*, volume 41, pages 111–125, 2004.
22. Johannes Buchmann and Hugh C. Williams. A key-exchange system based on imaginary quadratic fields. *Journal of Cryptology*, 1(2):107–118, 1988.
23. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more.
24. 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, Nara, Japan, February 26 – March 1, 2013. Springer, Heidelberg, Germany.
25. Alessandro Chiesa, Eran Tromer, and Madars Virza. Cluster computing in zero knowledge. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 371–403, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.
26. Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno, and Samee Zahur. Geppetto: Versatile verifiable computation. In IEEE S&P 2015 [42], pages 253–270.
27. 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, Queenstown, New Zealand, December 1–5, 2002. Springer, Heidelberg, Germany.

28. Ivan Damgård and Maciej Koprowski. Generic lower bounds for root extraction and signature schemes in general groups. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 256–271, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer, Heidelberg, Germany.
29. George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 532–550, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany.
30. George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct nizk arguments. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 532–550. Springer, 2014.
31. Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge proof systems. In Carl Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 52–72, Santa Barbara, CA, USA, August 16–20, 1988. Springer, Heidelberg, Germany.
32. Giovanni Di Crescenzo and Helger Lipmaa. Succinct np proofs from an extractability assumption. In *Conference on Computability in Europe*, pages 175–185. Springer, 2008.
33. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Heidelberg, Germany.
34. Dario Fiore, Rosario Gennaro, and Valerio Pastro. Efficiently verifiable computation on encrypted data. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 14*, pages 844–855, Scottsdale, AZ, USA, November 3–7, 2014. ACM Press.
35. Marc Fischlin and Jean-Sébastien Coron, editors. *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.
36. Georg Fuchsbauer. Subversion-zero-knowledge snarks. In *IACR International Workshop on Public Key Cryptography*, pages 315–347. Springer, 2018.
37. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany.
38. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.
39. Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340, Singapore, December 5–9, 2010. Springer, Heidelberg, Germany.
40. Jens Groth. On the size of pairing-based non-interactive arguments. In Fischlin and Coron [35], pages 305–326.
41. Safuat Hamdy and Bodo Möller. Security of cryptosystems based on class groups of imaginary quadratic orders. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 234–247, Kyoto, Japan, December 3–7, 2000. Springer, Heidelberg, Germany.
42. *2015 IEEE Symposium on Security and Privacy*, San Jose, CA, USA, May 17–21, 2015. IEEE Computer Society Press.
43. Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *21st ACM STOC*, pages 12–24, Seattle, WA, USA, May 15–17, 1989. ACM Press.
44. Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient arguments without short pcps. In *Computational Complexity, 2007. CCC'07. Twenty-Second Annual IEEE Conference on*, pages 278–291. IEEE, 2007.
45. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30, San Diego, CA, USA, June 11–13, 2007. ACM Press.

46. Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732, Victoria, British Columbia, Canada, May 4–6, 1992. ACM Press.
47. Joe Kilian. Improved efficient arguments (preliminary version). In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 311–324, Santa Barbara, CA, USA, August 27–31, 1995. Springer, Heidelberg, Germany.
48. Joe Kilian, Erez Petrank, and Gábor Tardos. Probabilistically checkable proofs with zero knowledge. In *29th ACM STOC*, pages 496–505, El Paso, TX, USA, May 4–6, 1997. ACM Press.
49. Benoît Libert, Somindu C. Ramanna, and Moti Yung. Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *ICALP 2016*, volume 55 of *LIPIcs*, pages 30:1–30:14, Rome, Italy, July 11–15, 2016. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.
50. Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189, Taormina, Sicily, Italy, March 19–21, 2012. Springer, Heidelberg, Germany.
51. 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, Singapore, June 26–29, 2012. Springer, Heidelberg, Germany.
52. Silvio Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453, Santa Fe, New Mexico, November 20–22, 1994. IEEE Computer Society Press.
53. Thilo Mie. Polylogarithmic two-round argument systems. *Journal of Mathematical Cryptology*, 2(4):343–363, 2008.
54. Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 49–62, Cambridge, MA, USA, June 18–21, 2016. ACM Press.
55. Tomas Sander. Efficient accumulators without trapdoor extended abstracts. In Vijay Varadharajan and Yi Mu, editors, *ICICS 99*, volume 1726 of *LNCS*, pages 252–262, Sydney, Australia, November 9–11, 1999. Springer, Heidelberg, Germany.
56. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266, Konstanz, Germany, May 11–15, 1997. Springer, Heidelberg, Germany.
57. Victor Shoup. OAEP reconsidered. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 239–259, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.
58. 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, San Francisco, CA, USA, March 19–21, 2008. Springer, Heidelberg, Germany.
59. Riad S Wahby, Ioanna Tzialla, Justin Thaler, and Michael Walfish. Doubly-efficient zksnarks without trusted setup.
60. Benjamin Wesolowski. Efficient verifiable delay functions. *IACR Cryptology ePrint Archive*, 2018:623, 2018.
61. Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. vSQL: Verifying arbitrary SQL queries over dynamic outsourced databases. In *2017 IEEE Symposium on Security and Privacy*, pages 863–880, San Jose, CA, USA, May 22–26, 2017. IEEE Computer Society Press.

A More Preliminaries

A.1 Hoeffding's Inequality

We recall a useful inequality by Hoeffding. Let $X_1, \ldots, X_n$ be independent random variables bounded by the interval $[0,1]$, let $\overline{X} := \frac{X_1+\ldots+X_n}{n}$, and let $0 < d < \overline{X} - E\left[\overline{X}\right]$, then it holds that

$$\mathsf{Pr}\left[\overline{X} - E\left[\overline{X}\right] \geq d\right] \leq e^{-2nd^2}.$$

A.2 Pseudo-Random Generators

A pseudorandom generator [43] stretches random strings into new random looking ones.

Definition 15 (Pseudo-Random Generator). *A function* $\mathsf{PRG} : \{0,1\}^n \to \{0,1\}^m$ *is a pseudo-random generator if* $m > n$ *and for all PPT adversaries* $\mathcal{A}$ *the following ensembles are computationally indistinguishable*

$$\{\mathsf{PRG}(s)\}_{s \leftarrow\$ \{0,1\}^n} \approx \{r\}_{r \leftarrow\$ \{0,1\}^m}.$$

B Pairing Groups

Let GGen be a probabilistic algorithm which inputs the security parameter 1^λ and outputs a tuple $(p, \mathbb{G}, \mathbb{G}_T, G, e)$, where p is a positive prime, $\mathbb{G}$ and $\mathbb{G}_T$ are (descriptions of) cyclic groups of order p (written multiplicatively) with identities $1_{\mathbb{G}}$ and $1_{\mathbb{G}_T}$ respectively, $G \in \mathbb{G}$ is a generator of $\mathbb{G}$, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a pairing satisfying the following:

- The map e is efficiently computable.
- The map e is non degenerate, i.e., $e(G,G) \neq 1_{\mathbb{G}_T}$.
- The map e is bilinear, i.e., $\forall (U,V) \in \mathbb{G}^2, \forall (a,b) \in \mathbb{Z}^2, e(U^a, V^b) = e(U,V)^{ab}$.

When the context is clear, we drop the subscripts and denote identity elements by 1.

Computational Diffie-Hellman. The computational Diffie-Hellman (CDH) assumption is one of the classical assumptions in cryptography. We re-state it over pairing groups.

Definition 16 (Computational Diffie-Hellman (CDH)). *The computational Diffie-Hellman assumption is said to hold with respect to* GGen *if for all* PPT *adversary* $\mathcal{A}$ *there exists* $\epsilon(\lambda) \in \mathsf{negl}(\lambda)$ *such that*

$$\mathsf{Pr}\left[Z = G^{xy} \middle| \begin{array}{c} (p, \mathbb{G}, \mathbb{G}_T, G, e) \leftarrow \mathsf{GGen}(1^\lambda); \\ (x,y) \leftarrow\$ \mathbb{Z}_p^2; Z \leftarrow \mathcal{A}(p, \mathbb{G}, \mathbb{G}_T, G, G^x, G^y, e) \end{array}\right] \leq \epsilon(\lambda).$$

For convenience, we define the following variant of the assumption that can be shown to be equivalent to the CDH assumption [6].

Definition 17 (Square Diffie-Hellman (square-DH)). *The square Diffie-Hellman assumption is said to hold with respect to* GGen *if for all* PPT *adversary* $\mathcal{A}$ *there exists* $\epsilon(\lambda) \in \mathsf{negl}(\lambda)$ *such that*

$$\mathsf{Pr}\left[Z = G^{x^2} \middle| \begin{array}{c} (p, \mathbb{G}, \mathbb{G}_T, G, e) \leftarrow \mathsf{GGen}(1^\lambda); \\ x \leftarrow\$ \mathbb{Z}_p; Z \leftarrow \mathcal{A}(p, \mathbb{G}, \mathbb{G}_T, G, G^x, e) \end{array}\right] \leq \epsilon(\lambda),$$

q-Diffie-Hellman Exponent. We are also going to rely on the q-Diffie-Hellman Exponent (q-DHE) assumption over bilinear groups. Loosely speaking, the problem is to compute G^{x^q} given all the powers $\left(G^x, \ldots, G^{x^{2q}}\right)$ except G^{x^q}.

Definition 18 (q-Diffie-Hellman Exponent (q-DHE)). *The q-Diffie-Hellman Exponent assumption is said to hold with respect to* GGen *if for all* PPT *adversary $\mathcal{A}$ there exists $\epsilon(\lambda) \in \mathsf{negl}(\lambda)$ such that*

$$\mathsf{Pr}\left[Z = G^{x^q} \,\middle|\, \begin{array}{c}(p, \mathbb{G}, \mathbb{G}_T, G, e) \leftarrow \mathsf{GGen}(1^\lambda); x \leftarrow_\$ \mathbb{Z}_p; \\ Z \leftarrow \mathcal{A}\left(p, \mathbb{G}, \mathbb{G}_T, G, G^x, \ldots, G^{x^{q-1}}, G^{x^{q+1}}, \ldots, G^{x^{2q}}, e\right)\end{array}\right] \leq \epsilon(\lambda),$$

C Analysis

We supplement the proofs which are omitted in the main text due to space constraints.

C.1 Proof of Theorem 7

Proof. Without loss of generality, let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a two-stage adversary. Consider the following extractor $\mathcal{E}^{\mathcal{A}}(x)$: On input a statement x, the extractor initializes a vector space $V = \mathbb{F}^q$, samples $\omega \leftarrow_\$ \{0,1\}^\lambda$, computes $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F}; \omega)$ and sends $y = \mathsf{pp}$ (or $y = (\mathsf{pp}, \omega)$ if using FC without trusted setup) to $\mathcal{A}_1$. The adversary $\mathcal{A}_1$ replies with a certain commitment C, and a state state which is passed to the second stage $\mathcal{A}_2$. In the following, we omit the input state to $\mathcal{A}_2$. The extractor enters into a loop. In the ℓ-th iteration of the loop, it performs the following:

1. Take an arbitrary vector $\boldsymbol{\pi}$ from the space V. Run $w \leftarrow \mathcal{E}_{\mathrm{PCP}}(\boldsymbol{\pi})$, if $\mathcal{R}(x, w) = 1$ then return w and terminate the execution.
2. Sample a random $\alpha_\ell \leftarrow_\$ \{0,1\}^\lambda$ and send it to $\mathcal{A}_2$. Set $\rho_{\ell,1} \| \ldots \| \rho_{\ell,k} := \mathsf{PRG}(\alpha_\ell)$.
3. $\mathcal{A}_2$ responds with $(\Lambda_\ell, \boldsymbol{y}_{\ell,1}, \ldots, \boldsymbol{y}_{\ell,k})$. Let $F^*_{\ell,i} = \mathsf{Reconstruct}(x, \boldsymbol{y}_{\ell,i}, \rho_{\ell,i})$ for $i \in [n]$, and let F_ℓ and $\boldsymbol{y}_\ell$ be the vertical concatenations of $F_{\ell,1}, \ldots, F_{\ell,k}$ and $\boldsymbol{y}_{\ell,1}, \ldots, \boldsymbol{y}_{\ell,k}$ respectively.
4. If $\mathsf{Verify}(C, F_\ell, \boldsymbol{y}_\ell, \Lambda_\ell) = 0$ or $\mathsf{Decide}(x, F_{\ell,i}, \boldsymbol{y}_{\ell,i}, \rho_{\ell,i}) = 0$ for some $i \in [n]$, then go to step 1 of the $(\ell+1)$-th iteration with fresh randomness.
5. Let $W = \{\boldsymbol{x} \in \mathbb{F}^q : F_\ell \boldsymbol{x} = \boldsymbol{y}_\ell\}$ be a subspace of $\mathbb{F}^q$ which satisfies the system of equations defined by $(F_\ell, \boldsymbol{y}_\ell)$. If $V \cap W = \emptyset$, then abort. Otherwise, set $V = V \cap W$.
6. Go to step 1 of the $(\ell+1)$-th iteration with fresh randomness

Note that step 1 and 5 are efficiently computable, *e.g.*, by Gaussian elimination. It is clear that whenever the extractor terminates without aborting, then the extraction is successful. We first argue that the extractor does not abort within polynomially-many steps except with negligible probability.

Lemma 1. *Let* FC *be weakly function binding with respect to* FSamp*. Then for all statements x, all auxiliary information z, all PPT adversary $\mathcal{A}$, and all polynomials $L \in \mathsf{poly}(\lambda)$ it holds that*

$$\mathsf{Pr}\left[\perp \leftarrow \mathcal{E}^{\mathcal{A}(x,z,y)}(x) \textit{ within } L \textit{ iterations}\right] \leq \mathsf{negl}(\lambda)\,.$$

Proof (Lemma 1). Let $L' \subseteq L$ be the set of iterations where the extractor reaches step 5. We observe that the extractor aborts if and only if the $\mathcal{A}_2$ successfully opens the commitment C (output by $\mathcal{A}_1$) to some function-value tuples $\{(F_\ell, \boldsymbol{y}_\ell)\}_{\ell \in L'}$, where $\boldsymbol{y}_\ell$ is the vertical concatenation of $\boldsymbol{y}_{\ell,1}, \ldots, \boldsymbol{y}_{\ell,k}$, such that there does not exist $\boldsymbol{x} \in \mathbb{F}^q$ so that $F_\ell \boldsymbol{x} = \boldsymbol{y}_\ell$ for all $\ell \in [L]$. This directly contradicts the assumption that FC is weakly function binding with respect to FSamp. □

The rest of the analysis establishes the probability that the extractor terminates. First we introduce the following helping lemma.

Lemma 2. *For all statements x, all $f(\lambda) \in \mathsf{poly}(\lambda)$, all $\boldsymbol{\pi} \in \mathbb{F}^q$, all constant $\gamma \in (0,1)$, all $k \geq \frac{-\log f(\lambda)}{\log(1-\gamma)}$ such that*

$$p_k := \mathsf{Pr}_{\forall i \in [n],\ \rho_i \leftarrow\!\$\ \{0,1\}^\lambda} \left[\forall i \in [n],\ \mathcal{V}^{\boldsymbol{\pi}}_{PCP}(x; \rho_i) = 1\right] \geq \frac{1}{f(\lambda)}$$

then $\mathsf{Pr}\left[\mathcal{R}(x, w) = 1 | w \leftarrow \mathcal{E}_{PCP}(\boldsymbol{\pi})\right] = 1$.

Proof (Lemma 2). Let $p_1 := \mathsf{Pr}_{\rho \leftarrow\!\$\ \{0,1\}^\lambda} \left[\mathcal{V}^{\boldsymbol{\pi}}_{\text{PCP}}(x; \rho) = 1\right]$. Since the random tapes ρ_i for $i \in [n]$ of the verifier are chosen uniformly we have that $p_k = (p_1)^k$ and therefore $p_1 = (p_k)^{\frac{1}{k}} \geq f(\lambda)^{\frac{-1}{k}}$. The following shows that $\frac{1}{f(\lambda)^k} \geq (1-\gamma)$:

$$\begin{aligned} k &\geq \frac{-\log f(\lambda)}{\log(1-\gamma)} \\ \frac{1}{k} \log\left(\frac{1}{f(\lambda)}\right) &\geq \log(1-\gamma) \\ \frac{1}{f(\lambda)^{\frac{1}{k}}} &\geq (1-\gamma). \end{aligned}$$

By Definition 5 it follows that the extractor $\mathcal{E}_{\text{PCP}}$ is successful with probability 1. □

Next we argue that for any given strings x and $\boldsymbol{\pi} \in \mathbb{F}^q$, running $\mathcal{V}^{\boldsymbol{\pi}}_{\text{PCP}}(x; \rho_i)$ over truly random coins ρ_i for $i \in [n]$ induces a distribution of outputs which is computationally indistinguishable from the distribution induced by $\mathcal{V}^{\boldsymbol{\pi}}_{\text{PCP}}(x; \rho_i)$, where $\rho_1 \| \ldots \| \rho_k \leftarrow \mathsf{PRG}(\alpha)$, executed over pseudo-random coins produced by $\mathsf{PRG}(\alpha)$. This is proven in the following lemma.

Lemma 3. *Let PRG be a pseudorandom generator. For all statements x, and all proof encodings $\boldsymbol{\pi} \in \mathbb{F}^q$, the ensembles*

$$\left\{(\mathcal{V}^{\boldsymbol{\pi}}_{PCP}(x; \rho_i))_{i \in [n]} : \rho_1 \| \ldots \| \rho_k := \mathsf{PRG}(\alpha)\right\}_{\alpha \leftarrow\!\$\ \{0,1\}^\lambda}$$

and

$$\left\{(\mathcal{V}^{\boldsymbol{\pi}}_{PCP}(x; \rho_i))_{i \in [n]}\right\}_{\forall i \in [n],\ \rho_i \leftarrow\!\$\ \{0,1\}^r}$$

are computationally indistinguishable.

Proof (Lemma 3). Assume the contrary, then we can construct the following distinguisher against PRG: On input a string ρ, it executes $b \leftarrow \mathcal{V}^{\boldsymbol{\pi}}_{\text{PCP}}(x;\rho)$ using ρ as the random tape. Then it outputs b. By initial assumption we have that the distributions of the output of the verifier are non-negligibly far depending on the random tape, consequently so are the distributions of the output of the distinguisher for the two cases. This contradicts the pseudo-randomness of PRG and shows the veracity of our proposition. □

Next we show that if the adversary convinces the verifier with non-negligible probability, then the extractor outputs w in polynomially many steps. This is shown in two steps. First, we show that if the adversary convinces the verifier with non-negligible probability, then the extractor produces a string $\boldsymbol{\pi}$ which is accepted by $\mathcal{V}_{\text{PCP}}$ with non-negligible probability. Second, given such a string $\boldsymbol{\pi}$, we show that the PCP extractor must succeed in extracting a witness w.

Concretely, for any statement x and for any auxiliary input z consider an adversary $\mathcal{A}$ such that

$$\epsilon_{\mathcal{A}} := \mathsf{Pr}\left[(\mathcal{A}(x,z,y),\mathcal{V}(x,y))_{\Pi} = 1\right] > \frac{1}{\lambda^c}$$

for some constant c, which is given by assumption. Set $t = \frac{\lambda}{\epsilon_{\mathcal{A}}} \leq \lambda^{c+1}$, and for $i \in [t]$, let Π_i be an independent execution of the protocol. Then we have that

$$\mathsf{Pr}\left[\forall i \in [t], (\mathcal{A}(x,z,y),\mathcal{V}(x,y))_{\Pi_i} = 0\right] = (1-\epsilon_{\mathcal{A}})^t \leq e^{-\epsilon_{\mathcal{A}} t} \leq e^{-\lambda}.$$

This means that with all but negligible probability the adversary is going to convince the verifier in at least one of these t executions. Let V be the subspace maintained by $\mathcal{E}$ after $t \cdot s$-many iterations, for $s := t\lambda = \frac{\lambda^2}{\epsilon_{\mathcal{A}}} \leq \lambda^{c+2} \in \mathsf{poly}(\lambda)$. Let $\boldsymbol{\pi}$ be an arbitrary vector picked from V. In the following we are going to show that, with overwhelming probability, there exists a constant d such that

$$\epsilon_{\mathcal{V}} := \mathsf{Pr}_{\alpha \leftarrow\$ \{0,1\}^\lambda}\left[\mathcal{V}^{\boldsymbol{\pi}}_{\text{PCP}}(x;\mathsf{PRG}(\alpha)) = 1\right] \geq \frac{1}{\lambda^d}. \tag{1}$$

Assume the contrary that, with non-negligible probability, there exists a constant c with $\epsilon \geq \frac{1}{\lambda^c}$ and $\epsilon_{\mathcal{V}} < \frac{1}{\lambda^d}$ for all constants d. We define a random variable X_ℓ which is equal to 1 if the extractor reaches step 5 in the ℓ-th iteration, and 0 otherwise. Note that if $X_\ell = 1$, then $\mathsf{Verify}(C, F_\ell, \boldsymbol{y}_\ell, \Lambda_\ell) = \mathsf{Decide}(x, F_{\ell,i}, \boldsymbol{y}_{\ell,i}, \rho_{\ell,i}) = 1$ for all $i \in [n]$. Let X^*_ℓ be another random variable defined like X_ℓ except that, instead of running $\mathsf{Decide}(x, F_{\ell,i}, \boldsymbol{y}_{\ell,i}, \rho_{\ell,i})$, the extractor runs $\mathcal{V}^{\boldsymbol{\pi}}_{\text{PCP}}(x;\rho_{\ell,i})$ and check if it equals 1 for all $i \in [n]$. By Lemma 1, with overwhelming probability $\mathcal{E}$ does not abort within ts steps.

Consider the set $L := \{\ell \in [t \cdot s] : X_\ell = 1\}$. By the definition of V, for any $\boldsymbol{x} \in V$, it holds that $F_{\ell,i}\boldsymbol{x} = \boldsymbol{y}_{\ell,i}$ for all $i \in [n]$ and $\ell \in L$. This implies that whenever $\mathsf{Decide}(x, F_{\ell,i}, \boldsymbol{y}_{\ell,i}, \rho_{\ell,i}) = 1$, then $\mathcal{V}^{\boldsymbol{\pi}}_{\text{PCP}}(x;\rho_{\ell,i}) = 1$. In other words, for all $\ell \in L$, if $X_\ell = 1$, then $X^*_\ell = 1$, which implies $X^*_\ell \geq X_\ell$.

The empirical mean $\overline{X}^*$ of $X^*_1, \ldots, X^*_{t\cdot s}$ can be computed as

$$\overline{X}^* := \frac{X^*_1 + \ldots + X^*_{t\cdot s}}{t \cdot s} \geq \frac{X_1 + \ldots + X_{t\cdot s}}{t \cdot s} \geq \frac{s}{t \cdot s} = \frac{1}{t}.$$

By assumption that Equation 1 is false, for all constants d,

$$E[\overline{X}^*] = \mathsf{Pr}_{\alpha \leftarrow\$ \{0,1\}^\lambda}\left[\mathcal{V}^{\boldsymbol{\pi}}_{\text{PCP}}(x;\mathsf{PRG}(\alpha)) = 1\right] \leq \frac{1}{\lambda^d}.$$

Let $d = c + 2$. By Hoeffding's inequality we have that

$$\begin{aligned}
&\mathsf{Pr}\left[\overline{X}^* - E[\overline{X}^*] \geq \frac{1}{t} - \frac{1}{\lambda^d}\right] \\
&\leq e^{-2ts\left(\frac{1}{t} - \frac{1}{\lambda^d}\right)^2} = e^{-2ts\left(\frac{1}{t^2} - \frac{2}{t\lambda^d} + \lambda^{-2d}\right)} = e^{-2\left(\lambda - \frac{2s}{\lambda^d} + \frac{s^2}{\lambda^{2d+1}}\right)} \\
&\leq e^{-2\left(\lambda - \frac{2s}{\lambda^d}\right)} \leq e^{-2\left(\lambda - \frac{2\lambda^{c+2}}{\lambda^d}\right)} \leq e^{-2(\lambda-2)} = \mathsf{negl}(\lambda)\,.
\end{aligned}$$

contradicting the assumption that this event happens with non-negligible probability.

To recap, we have shown that our extractor is able to produce a vector $\boldsymbol{\pi}$ such that

$$\epsilon_{\mathcal{V}} := \mathsf{Pr}_{\alpha \leftarrow\$ \{0,1\}^\lambda}\left[\mathcal{V}^{\boldsymbol{\pi}}_{\mathrm{PCP}}(x; \mathsf{PRG}(\alpha)) = 1\right] \geq \frac{1}{\lambda^{c+2}}.$$

By Lemma 3, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\begin{aligned}
&\mathsf{Pr}_{\rho \leftarrow\$ \{0,1\}^{rk}}\left[\mathcal{V}^{\boldsymbol{\pi}}_{\mathrm{PCP}}(x; \rho) = 1\right] \\
&\geq \mathsf{Pr}_{\alpha \leftarrow\$ \{0,1\}^\lambda}\left[\mathcal{V}^{\boldsymbol{\pi}}_{\mathrm{PCP}}(x; \mathsf{PRG}(\alpha)) = 1\right] - \mathsf{negl}(\lambda) \\
&\geq \frac{1}{\lambda^{c+2}} - \mathsf{negl}(\lambda)
\end{aligned}$$

Since $k = O(\lambda)$, with sufficiently large λ, by Lemma 2, we have that

$$\mathsf{Pr}\left[\mathcal{R}(x, w) = 1 | w \leftarrow \mathcal{E}_{\mathrm{PCP}}(\boldsymbol{\pi})\right] = 1.$$

This implies that the extractor terminates after $t \cdot s$ steps, except with negligible probability, and concludes the proof. □

C.2 Proof of Theorem 11

Proof. The proof uses the generic group model abstraction of Shoup [56] and we refer the reader to [16] for a comprehensive introduction to the bilinear group model. Here we state the central lemma useful for proving facts about generic attackers.

Lemma 4 (Schwartz-Zippel). *Let $F(X_1, \ldots, X_m)$ be a non-zero polynomial of degree $d \geq 0$ over a field $\mathbb{F}$. Then the probability that $F(x_1, \ldots, x_m) = 0$ for randomly chosen values $(x_1, \ldots, x_m)$ in $\mathbb{F}^n$ is bounded from above by $\frac{d}{|\mathbb{F}|}$.*

Fix $L \in \mathbb{N}$. Suppose there exists an adversary $\mathcal{A}$, who only performs generic bilinear group operations, such that there exists a polynomial $f \in \mathsf{poly}(\lambda)$ with

$$\mathsf{Pr}\left[\begin{array}{l} \forall \ell \in [L],\ \mathsf{Verify}(C, F_\ell, \boldsymbol{y}_\ell, \Lambda_\ell) = 1 \\ \nexists \boldsymbol{x} \in \mathbb{Z}_p^q \ s.t.\ \forall \ell \in [L],\ F_\ell(\boldsymbol{x}) = \boldsymbol{y}_\ell \end{array} \middle| \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (C, \{(F_\ell, \boldsymbol{y}_\ell, \Lambda_\ell)\}_{\ell \in [L]}) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp}) \end{array}\right] > \frac{1}{f(\lambda)}.$$

Since $\mathcal{A}$ is generic, and C and each of Λ_ℓ are $\mathbb{G}$ elements, we can write $\log_G C$ and each $\log_G \Lambda_\ell$ in the following form:

$$\log_G C = \gamma_0 + \sum_{j\in[q]} \gamma_j \alpha^j + \sum_{\substack{i\in[n]\\ j\in[2q]\setminus\{q+1\}}} \gamma_{i,j} z_i \alpha^j$$
$$\log_G \Lambda_\ell = \lambda_0 + \sum_{j\in[q]} \lambda_j \alpha^j + \sum_{\substack{i\in[n]\\ j\in[2q]\setminus\{q+1\}}} \lambda_{i,j} z_i \alpha^j$$

for some integer coefficients γ_j, $\gamma_{i,j}$, λ_j, and $\lambda_{i,j}$ for i and j in the appropriate ranges. Since for each $\ell \in [L]$, $\mathsf{Verify}(C, F_\ell, \boldsymbol{y}_\ell, \Lambda_\ell) = 1$, the following relations hold:

$$(\log_G C)(\sum_{i\in[n]}\sum_{j\in[q]} f_{\ell,i,j} z_i \alpha^{q+1-j}) = \sum_{i\in[n]} y_{\ell,i} z_i \alpha^{q+1} + \log_G \Lambda_\ell.$$

Note that the above defines a $(n+1)$-variate polynomial of degree $3q+2$ which evaluates to zero at a random point $(\alpha, z_1, \ldots, z_n)$. Suppose that the polynomial is non-zero. By the Schwartz-Zippel lemma, the probability that the above happens is bounded by $\frac{3q+2}{p}$ which is negligible as $q \in \mathsf{poly}(\lambda)$ and $1/p \in \mathsf{negl}(\lambda)$. We can therefore assume that the polynomial is always zero. In particular, the coefficients of the monomials $z_i \alpha^{q+1}$ are zero for all $i \in [q]$. Thus, we have the following relations for all $\ell \in [L]$ and $i \in [n]$:

$$\sum_{j\in[q]} f_{\ell,i,j}\gamma_j = y_{\ell,j}.$$

In other words, there exists $\boldsymbol{x} := (\gamma_1, \ldots, \gamma_q) \mod p \in \mathbb{Z}_p^q$ such that $F_\ell(\boldsymbol{x}) = \boldsymbol{y}_\ell$, for all $\ell \in [L]$, which contradicts the assumption about $\mathcal{A}$. We thus conclude that such adversaries exist only with negligible probability. Since the above holds for any $L \in \mathbb{N}$, we conclude that the construction is function binding. □

C.3 Proof of Theorem 12

Proof. Fix a positive integer $L \in \mathsf{poly}(\lambda)$. Suppose there exists an efficient adversary $\mathcal{A}$, who is given oracle access to H, and a polynomial $f \in \mathsf{poly}(\lambda)$ with

$$\mathsf{Pr}\left[\begin{array}{l|c} \forall \ell \in [L],\ \mathsf{Verify}(C, F_\ell, \boldsymbol{y}_\ell, \Lambda_\ell) = 1 & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F}) \\ \nexists \boldsymbol{x} \in \mathbb{Z}_p^q \ s.t.\ \forall \ell \in [L],\ F_\ell(\boldsymbol{x}) = \boldsymbol{y}_\ell & (C, \mathsf{seed}, \mathsf{state}) \leftarrow \mathcal{A}_1^H(1^\lambda, \mathsf{pp}) \\ & \forall \ell \in [L],\ r_\ell \leftarrow_\$ \{0,1\}^\lambda \\ & \forall \ell \in [L],\ F_\ell \leftarrow \mathsf{FSamp}(\mathsf{seed}; r_\ell) \\ & \{(\boldsymbol{y}_\ell, \Lambda_\ell)\}_{\ell\in[L]} \leftarrow \mathcal{A}_2^H(\mathsf{state}, \{F_\ell\}_{\ell\in[L]}) \end{array}\right] > \frac{1}{f(\lambda)}.$$

We construct a q-DHE solver $\mathcal{C}$ which runs $\mathcal{A}_1$ and $\mathcal{A}_2$ polynomially many times.

$\mathcal{C}$ receives as input a q-DHE instance $(p, \mathbb{G}, \mathbb{G}_T, G, \{G^{\alpha^j}\}_{j\in[2q]\setminus\{q+1\}})$. It sets pp to be equal to the q-DHE instance and runs $\mathcal{A}_1$ on pp. $\mathcal{C}$ simulates the random oracle H for $\mathcal{A}_1$ honestly. Let $(C, \mathsf{seed}, \mathsf{state})$ be the output of $\mathcal{A}_1$. $\mathcal{C}$ samples $r_\ell \leftarrow_\$ \{0,1\}^\lambda$, and

$F_\ell \leftarrow \mathsf{FSamp}(\mathsf{seed}; r_\ell)$, for $\ell \in [L]$. It then runs $\mathcal{A}_2$ on $(\mathsf{state}, \{F_\ell\}_{\ell\in[L]})$ who outputs $\{(\boldsymbol{y}_\ell, \Lambda_\ell)\}_{\ell\in[L]}$. With probability at least $\frac{1}{f(\lambda)}$, it holds that $\mathsf{Verify}(C, F_\ell, \boldsymbol{y}_\ell, \Lambda_\ell) = 1$ for all $\ell \in [L]$. This means that, conditioned on the above, $\mathcal{A}_1$ or $\mathcal{A}_2$ must have queried H on $(\mathsf{pp}, C, F_\ell, \boldsymbol{y}_\ell)$ for all $\ell \in [L]$ except with negligible probability. By the unpredictability of FSamp, all such queries must be made by $\mathcal{A}_2$ except with negligible probability, for otherwise $\mathcal{C}$ can extract a query $(\mathsf{pp}, C, F^*, \boldsymbol{y}^*)$ made by $\mathcal{A}_1$ and output (seed, F^*) with $F^* = \mathsf{FSamp}(\mathsf{seed}; r_\ell)$ for some $\ell \in [L]$. We can therefore assume that the view of $\mathcal{A}_1$, and in particular state, is independent of the values $H(\mathsf{pp}, C, F_\ell, \boldsymbol{y}_\ell)$ for all $\ell \in [L]$.

With the above analysis, $\mathcal{C}$ further runs $\mathcal{A}_2$ on $(\mathsf{state}, \{F_\ell\}_{\ell\in[L]})$ for additionally $n-1$ times, in such a way that at each instance $\mathcal{C}$ answer queries to H with independent randomness. Since the input $(\mathsf{state}, \{F_\ell\}_{\ell\in[L]})$ of $\mathcal{A}_2$ is independent of the values $H(\mathsf{pp}, C, F_\ell, \boldsymbol{y}_\ell)$ set during the first execution of $\mathcal{A}_2$, the view of $\mathcal{A}_2$ during all n executions is identical to that in the definition of weakly function binding. Let $H(\mathsf{pp}, C, F_\ell, \boldsymbol{y}_\ell^{(t)}) = (z_{\ell,1}^{(t)}, \ldots, z_{\ell,n}^{(t)})$ at the t-th execution of $\mathcal{A}_2$, and $\{(\boldsymbol{y}_\ell^{(t)}, \Lambda_\ell^{(t)})\}_{\ell\in[L]}$ be its output. We argue that $\boldsymbol{y}_\ell^{(t)} = \boldsymbol{y}_\ell^{(t')}$ for all $t, t' \in [n]$ and $\ell \in [L]$ except with negligible probability.

Suppose not, we tweak $\mathcal{C}$ slightly in the following way. It guesses a tuple (t, t', ℓ) such that the above event happens. $\mathcal{C}$ programs $H(\mathsf{pp}, C, F_\ell, \boldsymbol{y}_\ell^{(t)})$ and $H(\mathsf{pp}, C, F_\ell, \boldsymbol{y}_\ell^{(t')})$ such that $H(\mathsf{pp}, C, F_\ell, \boldsymbol{y}_\ell^{(t)}) = \mu \cdot H(\mathsf{pp}, C, F_\ell, \boldsymbol{y}_\ell^{(t')})$ for some $\mu \leftarrow_\$ \mathbb{Z}_p$. Since the t-th and t'-th execution of $\mathcal{A}_2$ are independent of each other, the distributions these answers are proper. Conditioning on the probability that the above event indeed happens, the probability that $\mathcal{C}$ guesses correctly is at least $\frac{1}{n^2 L}$ which is non-negligible as $n, L \in \mathsf{poly}(\lambda)$. Suppose that is the case, then it holds that

$$\begin{cases} e\left(C, \prod_{i\in[n]} \prod_{j\in[q]} G_{q+1-j}^{z_{\ell,i}^{(t)} f_{\ell,i,j}}\right) = e(G_1, \prod_{i\in[n]} G_q^{z_{\ell,i}^{(t)} y_{\ell,i}^{(t)}}) \cdot e(\Lambda_\ell^{(t)}, G) \\ e\left(C, \prod_{i\in[n]} \prod_{j\in[q]} G_{q+1-j}^{\mu z_{\ell,i}^{(t')} f_{\ell,i,j}}\right) = e(G_1, \prod_{i\in[n]} G_q^{\mu z_{\ell,i}^{(t')} y_{\ell,i}^{(t')}}) \cdot e(\Lambda_\ell^{(t')}, G) \end{cases}$$

which implies

$$e(G_1, \prod_{i\in[n]} G_q^{\mu z_{\ell,i}^{(t')} y_{\ell,i}^{(t)}}) \cdot e(\Lambda_\ell^{(t)}, G) = e(G_1, \prod_{i\in[n]} G_q^{\mu z_{\ell,i}^{(t')} y_{\ell,i}^{(t')}}) \cdot e(\Lambda_\ell^{(t')}, G)$$

$$\frac{\Lambda_\ell^{(t')}}{\Lambda_\ell^{(t)}} = G_{q+1}^{\sum_{i\in[n]} \mu z_{\ell,i}^{(t')} (y_{\ell,i}^{(t)} - y_{\ell,i}^{(t')})}.$$

Since $\boldsymbol{y}_\ell^{(t)} \neq \boldsymbol{y}_\ell^{(t')}$, and μ and $z_{\ell,i}^{(t')}$ are random in $\mathbb{Z}_p$ for $i \in [n]$, we have $\sum_{i\in[n]} \mu z_{\ell,i}^{(t')} (y_{\ell,i}^{(t)} - y_{\ell,i}^{(t')}) \neq 0$ except with negligible probability. $\mathcal{C}$ can thus output $G^{\alpha^{q+1}} = G_{q+1} = \left(\frac{\Lambda_\ell^{(t')}}{\Lambda_\ell^{(t)}}\right)^{\frac{1}{\sum_{i\in[n]} \mu z_{\ell,i}^{(t')} (y_{\ell,i}^{(t)} - y_{\ell,i}^{(t')})}}$ which is a solution to the q-DHE instance. We can thus assume that $\boldsymbol{y}_\ell^{(t)} = \boldsymbol{y}_\ell^{(t')}$ for all $t, t' \in [n]$ and $\ell \in [L]$, and denote $\boldsymbol{y}_\ell = \boldsymbol{y}_\ell^{(t)}$ for all $t \in [n]$ and $\ell \in [L]$.

We resume the analysis of $\mathcal{C}$ after running $\mathcal{A}_2$ n times (without the above change). Upon completion, $\mathcal{C}$ has collected the following relations

$$e\left(C, \prod_{i\in[n]}\prod_{j\in[q]} G_{q+1-j}^{z_{\ell,i}^{(t)} f_{\ell,i,j}}\right) = e(G_1, \prod_{i\in[n]} G_q^{z_{\ell,i}^{(t)} y_{\ell,i}}) \cdot e(\Lambda_\ell^{(t)}, G)$$

for all $t \in [n]$ and $\ell \in [L]$. Let $\boldsymbol{\alpha}$ be a vector such that $\alpha_j = \alpha^{q+1-j}$ for $j \in [q]$, $\boldsymbol{\Lambda}_\ell$ be a vector such that $\Lambda_{\ell,i} = \Lambda_\ell^{(i)}$ for all $i \in [n]$ and $\ell \in [L]$, and Z_ℓ be an n-by-n matrix such that $Z_{\ell,i,j} = z_{\ell,j}^{(i)}$ for all $i, j \in [n]$ and $\ell \in [L]$. For clarity we rewrite the relations collected by $\mathcal{C}$ as

$$(\log_G C) Z_\ell F_\ell \boldsymbol{\alpha} = \alpha^{q+1} Z_\ell \boldsymbol{y}_\ell + \log_G \boldsymbol{\Lambda}_\ell.$$

Since Z_ℓ is uniformly random, it is invertible except with negligible probability. $\mathcal{C}$ can therefore compute

$$\log_G \bar{\boldsymbol{\Lambda}}_\ell := Z_\ell^{-1} \log_G \boldsymbol{\Lambda}_\ell$$

such that $\log_G \bar{\boldsymbol{\Lambda}}_\ell$ satisfies

$$(\log_G C) F_\ell \boldsymbol{\alpha} = \alpha^{q+1} \boldsymbol{y}_\ell + \log_G \bar{\boldsymbol{\Lambda}}_\ell$$

for all $\ell \in [L]$. Next, we make use of the fact that the linear system of equations given by the tuples $\{(F_\ell, \boldsymbol{y}_\ell)\}_{\ell\in[L]}$ is inconsistent. Let F and $\boldsymbol{y}$ be the vertical concatenations of $F_1, \ldots, F_L$ and $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_L$ respectively. Consider the augmented matrix $A := [F|\boldsymbol{y}]$. Using Gaussian elimination, $\mathcal{C}$ can efficiently compute matrices U and $A' = [F'|\boldsymbol{y}']$ such that $A = UA$ and A' is in reduced row echelon form. By multiplying the relations obtained above by U, $\mathcal{C}$ obtains

$$(\log_G C) F' \boldsymbol{\alpha} = \alpha^{q+1} \boldsymbol{y}' + \log_G \boldsymbol{\Lambda}'$$

Since the system is inconsistent, there must be a row i^*, such that the i^*-th row of F', denoted F'_{i^*}, are all zero, and y'_{i^*} is non-zero. $\mathcal{C}$ can therefore obtain the relation

$$\alpha^{q+1} y'_{i^*} + \log_G \Lambda'_{i^*} = (\log_G C) F'_{i^*} \boldsymbol{\alpha} = 0.$$

Finally $\mathcal{C}$ outputs $G^{\alpha^{q+1}} = (\Lambda'_{i^*})^{\frac{1}{y'_{i^*}}}$ as a solution to the q-DHE instance. Since the above analysis holds for all $L \in \mathsf{poly}(\lambda)$, we conclude that the construction is weakly function binding. □

C.4 Proof that strong RSA implies strong root in RSA-UFO groups

Proof (Sketch). Let $\mathcal{A}$ be an adversary against the strong RSA assumption in RSA-UFO groups. Then, on input some (standard) RSA modulus N and a group element A, we can sample a set of primes $(p_1, \ldots, p_m)$ and set $N' = N \cdot \prod_{i\in[m]} p_i$ and $A' = A + qN$ for some random $q \in \left[\prod_{i\in[m]} p_i\right]$. Now we give (N', A') to $\mathcal{A}$. Note that the distribution

of prime factors of a random number is easy to simulate, and therefore N' is a correctly distributed RSA-UFO. Also note that A' is an element of $\mathbb{Z}_{N'}^*$ with high probability. When $\mathcal{A}$ returns some (e, Y) such that $e > 1$, $Y \in \mathbb{Z}_{N'}^*$ and $Y^e = A' \mod N'$, we have

$$Y^e = A' \mod N' \implies Y^e = A' \mod N$$

since N' is multiple of N. Furthermore,

$$A' \mod N = A + qN \mod N = A \mod N$$

and, for some integers $Z \in \mathbb{Z}_N^*$ and $h \in \mathbb{Z}$, we have

$$Y^e \mod N = (Z + hN)^e \mod N = Z^e \mod N$$

To conclude, we obtain (Z, e) with $e > 1$ and $Z^e = A \mod N$, which is a solution the strong RSA problem. □