Subvector Commitments with Application to Succinct Arguments

Russell W. F. Lai[1] and Giulio Malavolta[2*]

[1] Friedrich-Alexander-Universität Erlangen-Nürnberg
[2] Carnegie Mellon University

Abstract. We put forward the notion of subvector commitments (SVC): An SVC allows one to open a committed vector at a set of positions, where the opening size is independent of length of the committed vector and the number of positions to be opened. 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 one to open a committed vector to its images under linear maps with a single short message, and propose a construction over pairing groups.

Equipped with these newly developed tools, we revisit the "CS proofs" paradigm [Micali, FOCS 1994] which turns any arguments with public-coin verifiers into non-interactive arguments using the Fiat-Shamir transform in the random oracle model. We propose a compiler that turns any (linear, resp.) PCP into a non-interactive argument, using exclusively SVCs (LMCs, resp.). For an approximate 80 bits of soundness, we highlight the following new implications:

1. There exists a succinct non-interactive argument of knowledge (SNARK) with public-coin setup with proofs of size 5360 bits, under the adaptive root assumption over class groups of imaginary quadratic orders against adversaries with runtime 2^{128}. At the time of writing, this is the shortest SNARK with public-coin setup.
2. There exists a non-interactive argument with private-coin setup, where proofs consist of 2 group elements and 3 field elements, in the generic bilinear group model.

1 Introduction

Commitment schemes are one of the fundamental building blocks and one of the most well-studied primitives in cryptography. Due to their pivotal importance in the design of cryptographic protocols, even small efficiency improvements have magnified repercussions in the field. In a recent work, Catalano and Fiore [27] put forth the notion of Vector Commitments (VC): A VC allows a prover to commit to a vector $\boldsymbol{x}$ of ℓ messages, such that it can later open the commitment at any position $i \in [\ell]$ of the vector, *i.e.*, reveal a message and show that it equals to the i-th committed message. The distinguishing feature of VCs is that the size of the commitments and openings is independent of ℓ. 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

[*] Part of the work done while at Friedrich-Alexander-Universität Erlangen-Nürnberg.

and Fiore [27] 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.

This concept was later generalized by Libert *et al.* [48], who formalized the notion of functional commitment (FC). Intuitively, an FC allows the prover to commit to a vector $\boldsymbol{x}$, and to open the commitment to function-value tuples (f, y) such that $y = f(\boldsymbol{x})$. Libert *et al.* [48] proposed a construction for *linear forms*[3] based on the Diffie-Hellman exponent assumption over pairing groups, where a commitment and an opening both consist of a single group element. VCs and FCs for linear forms are very versatile tools and turned out to be useful for a variety of applications, such a zero-knowledge sets [54], polynomial commitments [44], accumulators, and credentials, to mention a few.

While a short commitment is certainly an appealing feature, there are contexts where there is still a lot to be desired. For example, in case the prover wants to reveal multiple locations of the committed vector (resp. multiple function outputs) the best known solution is to repeat the above protocol in parallel. This means that the size of the openings grows linearly with the amount of revealed locations (resp. function outputs).

1.1 Commitments with Even Shorter Openings

We introduce the notion of *subvector commitments* (SVCs). An SVC allows one to commit to a vector $\boldsymbol{x}$ of length ℓ and later open to a *subvector* of an arbitrary length $\leq \ell$. Given an *ordered* index set $I \subseteq [\ell]$, we define the I-subvector of $\boldsymbol{x}$ as the vector formed by collecting the i-th component of $\boldsymbol{x}$ for all $i \in I$. While a VC is required to be succinct, namely the commitment size and the size of the proof of the opening are independent of the length of the committed vector, an SVC has a stronger compactness[4] property which additionally requires that these sizes do not depend on the length of the subvector to be opened. This difference is going to be critical for our applications (explained later). Improving upon the VC constructions of Catalano and Fiore [27], we propose two constructions of SVCs based on the CDH assumption over pairing groups and the RSA assumption, respectively. We further generalize the RSA-based scheme to work over modules over Euclidean rings [51], where variants of the root assumption are conjectured to hold. Loosely speaking, the root assumption states that it is hard to find the e-th root of a random ring element, for any non-trivial e. In these settings we obtain public-coin-setup instantiations of SVCs using class groups of imaginary quadratic orders.

We then generalize the notion of SVCs to allow the prover to reveal arbitrary *linear maps* $f : \mathbb{F}^\ell \to \mathbb{F}^q$ computed over the committed vector. We call such class of schemes

[3] A linear form is a linear map from a vector space to its field of scalars. Libert *et al.* [48] used the more general term linear functions to refer to linear forms.

[4] The term "compactness" is borrowed from the literature of randomized encodings (RE) and functional encryption, and not to be confused with the compactness notion of homomorphic encryption. For example, a compact RE of a computation with n outputs should have size independent of n [49].

Scheme	$\lvert\mathsf{pp}\rvert$	$\lvert C\rvert$	$\lvert\Lambda\rvert$	$\mathsf{time}(\mathsf{Com})$	$\mathsf{time}(\mathsf{Open})$	$\mathsf{time}(\mathsf{Verify})$	Setup	Assumption
Merkle Tree [52]	1	λ	$\lambda q \log \ell$	$\lambda\ell$	$\lambda q \log \ell$	$\lambda q \log \ell$	Pub	CRH
VC (RSA) [27]	$\lambda^3\ell$	λ^3	$\lambda^3 q$	$\lambda^3\ell$	$\lambda^3 q\ell^2$	$\lambda^3 q$	Pri	RSA
VC (CDH) [27]	$\lambda\ell^2$	λ	λq	$\lambda\ell$	$\lambda q\ell$	λq	Pri	CDH
SVC (Class Group)	$\lambda^2\ell$	λ^2	λ^2	$\lambda^2\ell$	$\lambda^2(\ell - q^2)$	$\lambda^2 q$	Pub	Root
SVC (CDH)	$\lambda\ell^2$	λ	λ	$\lambda\ell$	$\lambda q\ell$	λq	Pri	CDH
FC (linear form) [48]	$\lambda^3\ell$	λ^3	$\lambda^3 q$	$\lambda^3\ell$	$\lambda^3 q\ell$	$\lambda^3 q\ell$	Pri	SD
LMC	$\lambda q\ell$	λ	λ	$\lambda\ell$	$\lambda q\ell^2$	$\lambda q\ell$	Pri	GGM

Table 1: Comparison of subvector and linear map commitments for messages of length ℓ, with binding against adversaries of runtime 2^λ. All constants are omitted. pp: public parameters, C: commitment, Λ: proof, Pub: public-coin, Pri: private-coin, CRH: collision-resistant hash, Root: strong or adaptive root, SD: subgroup decision, GGM: generic bilinear group model.

linear map commitments (LMC). As in SVC, it is important to require an LMC to be compact, meaning that both the commitment and the proofs are of size independent of ℓ and q, whereas succinctness only requires their size to be independent of ℓ. 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 position binding for 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. This turns out to be insufficient for our applications: 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 which requires that no efficient algorithm can produce openings for Q function-value tuples $\{(f_k, \boldsymbol{y}_k)\}_{k\in[Q]}$ for any $Q \in \mathsf{poly}(\lambda)$, such that there does not exist $\boldsymbol{x}$ with $f_k(\boldsymbol{x}) = \boldsymbol{y}_k$ for all $k \in [Q]$.

We then modify the construction of Libert *et al.* [48] 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, we embed a secret linear combination in the public parameters, and show that the resulting construction is function binding in the generic bilinear group model. In Table 1 we compare our SVC and LMC constructions with existing schemes.

1.2 The Quest of Constructing Ever Shorter Arguments

In addition to enabling batching in the original applications of VCs and FCs for linear forms mentioned above, the compactness of SVCs and LMCs opens the new possibilities of application in constructing succinct argument systems.

Background. An argument system for an NP language $\mathcal{L}$ allows a prover, with a witness w, to convince a verifier that a certain statement x is in $\mathcal{L}$. 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 security parameter and is independent of the size of w. Other desirable properties of an argument system are:

- "of knowledge": a successful prover implies an extractor that can recover the witness;
- non-interactive: the protocol consists of a single message from the prover;
- (verifier) public-coin: messages from the verifier are sampled from public domains.

Recently, much progress has been made both in theory and practice to construct succinct non-interactive arguments of knowledge (SNARK) for general NP languages. We distinguish between SNARKs in the public-coin-setup model and the pre-processing model. In the public-coin-setup 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 SNARKs in the pre-processing model are more efficient, in terms of both communication and computation, than those in the public-coin-setup 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 have a public-coin setup, which can be publicly initialized via, *e.g.*, a random oracle [8].

Public-Coin-Setup SNARKs. While it is known that public-coin-setup non-interactive arguments for NP do not exists in the standard model [15], one can circumvent this impossibility by working in the random oracle model [8]. A common way to obtain public-coin-setup SNARKs is through the "CS proofs" paradigm [45,53] based on probabilistically checkable proofs (PCP) [3]. To recall, a q-query $2^{-\sigma}$-soundness 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 with probability close to $1 - 2^{-\sigma}$ by inspecting q entries of the PCP string. Given a PCP, a SNARK under the CS proofs paradigm are constructed in two steps. First, the PCP is turned into an interactive argument system: The prover first commits to the PCP string, typically using a Merkle-tree commitment. The verifier then sends the indices of the entries to be inspected. Next, the prover opens the commitment at these entries. Finally, by inspecting the revealed entries, the verifier can decide whether the statement is valid. Typically, an argument system constructed this way has a public-coin verifier and can be made non-interactive using the Fiat-Shamir transform [35].

Under the CS proofs paradigm, a proof (*e.g.*, in the scheme by Micali [53]) consists of a λ-bit Merkle-tree commitment of a ℓ-bit PCP string, q bits of the PCP string, and q openings of the commitment, each of size $\lambda \log \ell$ bits. For concreteness, assuming a 3-query PCP and $\ell = 2^{30}$, for 2^{-80}-soundness against a 2^{128}-time adversary, the proof size is around 113 KB. Despite having linear verification time (hence not being a SNARK) Bulletproof [21,26] is arguably the most practically efficient non-interactive argument to date. A proof in [26] 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 their 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.

Pre-Processing SNARKs. In the pre-processing model, there exist plenty of SNARK constructions originated by [37] based on pairings and linear interactive proofs (LIP), where the latter can be constructed from linear PCPs. To recall, linear PCPs [42] generalizes traditional PCPs in the sense that the PCP string now encodes a linear form. In a q-query linear PCP, the verifier, who is given oracle access to the linear form, can decide the veracity of the statement with overwhelming probability by making only q queries. SNARK constructions in this category typically have a computationally expensive stetement-dependent pre-processing phase, meaning that one set of public parameters has to be generated per statement to be proven.

In this setting, the scheme with the shortest proofs (4 group elements) in the standard model is due to Danezis *et al.* [32]. In the generic bilinear group model, Groth [40] proposed a scheme [60] 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 pairing-friendly elliptic curves. A popular choice is the 256-bit Barreto-Naehrig curve [7], in which a group element can be represented using 256 bits.

Our Approach. Equipped with our newly developed tools, we revisit the CS proofs paradigm. In previous schemes following this paradigm, the proof size is dominated by the factor $q \log \ell$ due to the q Merkle-tree commitment openings. Moreover, due to the lack of structure of a Merkle-tree commitment, prior schemes do not work with linear PCPs. The main idea is thus to replace the Merkle-tree commitment with an SVC / LMC, so that the q openings can be compressed into a single one which has size independent of ℓ and q. By doing so, we obtain a compiler which compiles any (resp. linear) PCP into an interactive argument using an SVC (resp. LMC).

We highlight two interesting instantiations of our construction. The first instantiation is with classical PCPs and our public-coin-setup SVC based on $Cl(\Delta)$, the class group of imaginary quadratic order with discriminant Δ.

Instantiation 1 *If the adaptive root assumption holds in $Cl(\Delta)$, then there exist public-coin-setup SNARKs for NP with soundness error $2^{-\sigma}$ in which a proof consists of 2 $Cl(\Delta)$ elements and q bits in the random oracle model, using* any *q-query $2^{-\sigma}$-soundness PCP.*

If one aims for an extremely short proof and is willing to accept expensive prover computation, then a 3-query 2^{-1}-soundness PCP can be amplified into a 3σ-query $2^{-\sigma}$-soundness PCP and gives the shortest SNARK. Based on the best known attacks on the root problem in class groups [41], for a soundness error of 2^{-80} against a 2^{128}-time adversary, we obtain a proof size of 5360 bits, which is shorter than that of Bulletproof [26] for $n > 16$, *i.e.*, the verification circuit has more than 16 multiplication gates. We view this instantiation as a feasibility for extremely succinct proofs and a step forward towards optimal ($O(\lambda)$-sized) public-coin-setup SNARKs. Next we turn our attention to the instantiation with linear PCPs and our pairing-based LMCs.

Scheme	$\lvert\mathsf{pp}\rvert$	$\lvert\pi\rvert$	Setup	Assumption
CS Proof (Merkle Tree Compiler) [45,53]	1	$\lambda^2 \log n$	Pub	ROM
Bulletproof [21,26]	λn	$\lambda \log n$	Pub	DLog, ROM
Aurora [12]	1	$\lambda \log^2 n$	Pub	ROM
SVC Compiler (Class Group)	$\lambda^2 \ell_{\mathsf{PCP}}$	λ^2	Pub	Root, ROM
Groth [40]	λn	λ	Pre-Proc	GGM
SVC Compiler (CDH)	$\lambda \ell_{\mathsf{LPCP}}^2$	λ	Pri	CDH, ROM
LMC Compiler	$\lambda \ell_{\mathsf{LPCP}}$	λ	Pri	GGM, ROM

Table 2: Comparison of SNARKs with $2^{-\lambda}$-soundness against adversaries of runtime 2^{128}. All constants are omitted. pp: public parameters, π: proof, n: size of circuit, ℓ_{PCP}: length of PCP proof, ℓ_{LPCP}: length of linear PCP proof, Pub: public-coin, Pri: private-coin, Pre-Proc: pre-processing, Root: strong or adaptive root assumption, GGM: generic group model.

Instantiation 2 *In the generic blinear group and random oracle model, there exist preprocessing non-interactive arguments for NP in which a proof consists of 2 $\mathbb{G}$ elements and q field elements, using* any *q-query linear PCP.*

Using a 3-query linear PCP (*e.g.* [17]) and instantiating the pairing group over the 256-bit Barreto-Naehrig curve yields a proof consisting of 5 elements or 1280 bits. Compared to other pairing-based compilers from linear PCPs to preprocessing SNARKs (*e.g.*, [40]), our compiler has the advantages that it supports *any* linear PCPs, but not only those where the verifier is restricted to only evaluate quadratic polynomials. Moreover the setup phase is independent of the statements to be proven, and thus the same public parameters can be reused for proving many statements.

A comparison with the shortest succicnt arguments from the literature is given in Table 2. To summarize, our approach yields extremely short proofs in exchange for a higher prover complexity and the usage of public-key cryptography. We also stress that our compiler is compatible with a broader class of PCPs, when compared with schemes under the CS proofs paradigm and pairing-based schemes. Being a very active area of research, we expect significant advancements in the design of more efficient PCPs, which are going to benefit from the generality of our approach.

Other Applications. Catalano and Fiore [27] suggested a number 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 private-coin setup, which is especially beneficial to database applications as trusted third parties are no longer required.

The notion of SVC has already attracted the attention of the community. A follow up work by Boneh *et al.* [18] shows how SVCs can be used as a drop-in replacement for Merkle-trees in SNARKs based on interactive oracle proofs (IOPs) which generalizes PCPs. They leverage the structure of class group-based SVCs to reduce the proof size to

$(r+1)$ group elements and r integers, where r is the number of iterations of the underlying IOP. They also propose a technique to improve the efficiency of the verification algorithm and they estimate a decrease in verification time of $\sim 80\%$. Finally, they discuss how to use SVCs to improve the current design of blockchain-based transaction ledger in such a way that no user has to store the entire state of the ledger in memory.

1.3 Related Work

Succinct arguments were introduced by Kilian [45,46] and later improved, in terms of round complexity, by Lipmaa and Di Crescenzo [34]. Succinct non-interactive arguments, or computationally sound proofs, were first proposed by Micali [53]. These early approaches rely on PCP and have been recently extended [9] to handle interactive oracle proofs [13] (also known as probabilistic checkable interactive proofs [57]), largely improving the efficiency of the prover. A recent manuscript by Ben-Sasson *et al.* [10] improves the concrete efficiency of interactive oracle proofs. The first usage of knowledge assumptions to construct SNARKs appeared in the work of Mie [55]. Later, Groth [39] and Lipmaa [50] upgraded this approach to non-interactive proofs.

Ishai, Kushilevitz, and Ostrovsky [42] observed that linear PCPs can be combined with a linearly homomorphic encryption to construct more efficient arguments, with pre-processing. The also introduced a new (interactive) commitment scheme with private-coin verifier for linear functions. However, in contrast with LMC, their binding definition does not esure that the committed function is actually linear. 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 [5,11,14,28,29,33]. 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 [43] paradigm to prove satisfiability of a circuit C with proofs of size $O(\lambda\sqrt{|C|})$. Zhang *et al.* [64] show how to combine interactive proofs and verifiable polynomial delegation schemes to construct succinct interactive arguments. The scheme requires a private-coin pre-processing and the communication complexity is $O(\lambda \log |w|)$. A recent result by Whaby *et al.* [62] 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 [48] 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 $[\ell] := \{1, \dots, \ell\}$. 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 $(x_1, \ldots, x_\ell)^T$.

Definition 1 (Subvectors). *Let $\ell \in \mathbb{N}$, $\mathcal{X}$ be a set, and $(x_1, \ldots, x_\ell)^T \in \mathcal{X}^\ell$ 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 [22] 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 possibly 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 $2^{-\sigma}$-sound (succinct) argument of knowledge for $\mathcal{R}$ if the following conditions hold.*

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

(Soundness) For any PPT adversary $\mathcal{A}$, all $x \notin \mathcal{L}$, and all $z \in \{0,1\}^*$, $\Pr_{y \leftarrow \mathcal{S}(1^\lambda)}[(\mathcal{A}(x,z,y), \mathcal{V}(x,y))_\Pi = 1] < 2^{-\sigma}$.

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

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

2.3 Probabilistically Checkable Proofs

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

Definition 3 (Probabilistically Checkable Proofs). *A pair of machines $(\mathcal{P}_{PCP}, \mathcal{V}_{PCP})$ is a ℓ-long q-query $2^{-\sigma}$-sound PCP for an NP-relation $\mathcal{R}$ if the following hold.*

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

(Soundness) For all $x \notin \mathcal{L}$, $\Pr[\mathcal{V}_{PCP}^{\boldsymbol{\pi}}(x) = 1 | \boldsymbol{\pi} \leftarrow \mathcal{P}_{PCP}(x,w)] < 2^{-\sigma}$.

(Proof Length) If $\mathcal{R}(x,w) = 1$, then for all $\boldsymbol{\pi} \in \mathcal{P}_{PCP}(x,w)$, $|\boldsymbol{\pi}| \leq \ell$.

(Query Complexity) For all $x, \boldsymbol{\pi} \in \{0,1\}^*$, $\mathcal{V}_{PCP}^{\boldsymbol{\pi}}(x)$ queries at most q locations of $\boldsymbol{\pi}$.

The notation $\mathcal{V}_{\mathrm{PCP}}^{\boldsymbol{\pi}}(x)$ means that $\mathcal{V}_{\mathrm{PCP}}$ 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 repetition. We additionally require that the witness can be efficiently recovered from the encoding of the witness $\boldsymbol{\pi}$ [61].

Definition 4 (Proof of Knowledge). *A PCP is of knowledge if there exists a PPT algorithm $\mathcal{E}_{PCP}$ such that, given any strings x and $\boldsymbol{\pi}$ with $\Pr\left[\mathcal{V}_{PCP}^{\boldsymbol{\pi}}(x) = 1\right] > \mathsf{negl}(\lambda)$, $\mathcal{E}_{PCP}^{\boldsymbol{\pi}}(x)$ extracts an NP witness w for x.*

Linear PCPs. Ishai et al. [42] considered the notion of *linear* PCP, where the string $\boldsymbol{\pi}$ is instead a vector in $\mathbb{F}^\ell$ for some finite field $\mathbb{F}$ (or in general a ring) and positive integer ℓ. The oracle given to the verifier is modified, such that on input $\boldsymbol{f} \in \mathbb{F}^\ell$, 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. In this paper we are interested in the notion of linear PCP where soundness is only guaranteed to hold against linear functions (same as considered in [17]).

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 $S = (s_1, \ldots, s_\ell) \subseteq \mathbb{N}$ be an ordered set, and $\boldsymbol{r} = (r_{s_1}, \ldots, r_{s_\ell})^T \in R^\ell$ and $\boldsymbol{A} = (A_{s_1}, \ldots, A_{s_\ell})^T \in D^\ell$ 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 [63] and re-formulated by Boneh *et al.* [20] to establish the security of the verifiable delay function scheme of Wesolowski [63]. Here we state the same assumption over modules in two variants – with private and public coins. Note that Wesolowski [63] and Boneh et al. [20] implicitly considered the *public-coin-setup* variant.

Definition 5 ((Public-Coin) Adaptive Root). *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 $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists $\epsilon(\lambda) \in \mathsf{negl}(\lambda)$ such that*

$$\Pr\left[e \circ Y = X \,\middle|\, \begin{array}{c} \omega \leftarrow_\$ \{0,1\}^\lambda; (R_D, A) := \mathsf{MGen}(1^\lambda; \omega) \\ X \leftarrow \mathcal{A}_1(R_D, A\boxed{, \omega}); e \leftarrow_\$ \mathrm{IRR}_\lambda(R); Y \leftarrow \mathcal{A}_2(e) \end{array} \right] \leq \epsilon(\lambda),$$

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 is* public-coin.

Strong Distinct-Prime-Product Root. We define the following variant of the "strong root assumption" [30] 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 6 ((Public-Coin) Strong Distinct-Prime-Product Root). *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*

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

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 is* public-coin.

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 (resp. public-coin) strong distinct-prime-product root assumption is weaker than the (resp. public-coin) strong root assumption, 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 Subvector Commitments

In the following we define the main object of interest for our work. Subvector commitments are a generalization of vector commitments [27], where the opening is performed with respect to subvectors.

Definition 7 (Subvector Commitments (SVC)). *A subvector commitment scheme* SVC *over* $\mathcal{X}$ *consists of the following* PPT *algorithms* $(\mathsf{Setup}, \mathsf{Com}, \mathsf{Open}, \mathsf{Verify})$*:*

$\underline{\mathsf{Setup}(1^\lambda, 1^\ell; \omega)}$*: The* deterministic *setup algorithm inputs the security parameter* 1^λ*, the vector size* 1^ℓ*, 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* $\boldsymbol{x} \in \mathcal{X}^\ell$*. It outputs a commitment string* C *and some auxiliary information* aux*.*

$\underline{\mathsf{Open}(I, \boldsymbol{x}'_I, \mathsf{aux})}$*: The opening algorithm inputs an index set* I*, an* I*-subvector* $\boldsymbol{x}'_I$*, and some auxiliary information* aux*. It outputs a proof* Λ_I *that* $\boldsymbol{x}'_I$ *is the* I*-subvector of the committed vector.*

$\underline{\mathsf{Verify}(C, I, \boldsymbol{x}'_I, \Lambda_I)}$*: The verification algorithm inputs a commitment string* C*, an index set* I*, an* I*-subvector* $\boldsymbol{x}'_I$*, and a proof* Λ_I*. It accepts (i.e., it outputs* 1*) if and only if* C *is a commitment to* $\boldsymbol{x}$ *and* $\boldsymbol{x}'_I$ *is the* I*-subvector of* $\boldsymbol{x}$*.*

The definition of correctness is given as follows.

Definition 8 (Correctness). *A subvector commotment* SVC *over* $\mathcal{X}$ *is said to be correct if, for any security parameter* $\lambda, \ell \in \mathbb{N}$*, random tape* $\omega \in \{0,1\}^\lambda$*, public parameters* $\mathsf{pp} \in \mathsf{Setup}(1^\lambda, 1^\ell; \omega)$*,* $\boldsymbol{x} \in \mathcal{X}^\ell$*, index set* $I \in [\ell]$*,* $(C, \mathsf{aux}) \in \mathsf{Com}(\boldsymbol{x})$*,* $\Lambda_I \in \mathsf{Open}(I, \boldsymbol{x}_I, \mathsf{aux})$*, there exists* $\epsilon(\lambda) \in \mathsf{negl}(\lambda)$ *such that*

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

The distinguishing property for SVCs is compactness. Loosely speaking it says that the size of the commitment strings C and the proofs Λ_I are not only independent of the length of the committed vector $\boldsymbol{x}$, but also that of $\boldsymbol{x}_I$.

Definition 9 (Compactness). *A subvector commitment* SVC *over* $\mathcal{X}$ *is compact if there exists a universal polynomial* $p \in \mathsf{poly}(\lambda)$ *such that for any* $\ell \in \mathsf{poly}(\lambda)$*, random tape* $\omega \in \{0,1\}^\lambda$*, public parameters* $\mathsf{pp} \in \mathsf{Setup}(1^\lambda, 1^\ell; \omega)$*, vector* $\boldsymbol{x} \in \mathcal{X}^\ell$*, index set* $I \in [\ell]$*,* $(C, \mathsf{aux}) \in \mathsf{Com}(\boldsymbol{x})$*,* $\Lambda_I \in \mathsf{Open}(I, \boldsymbol{x}_I, \mathsf{aux})$*, it holds that* $|C| \leq p(\lambda)$ *and* $|\Lambda_I| \leq p(\lambda)$*.*

We consider the notion of position binding for subvector commitments with public-coin setup. Recall that position binding for vector commitments requires that it is infeasible to open a commitment with respect to some position i to two distinct messages $\boldsymbol{x}_i$ and $\boldsymbol{x}'_i$. We extend this notion to subvector commitments, by requiring that it is infeasible to open a commitment with respect to some index sets I and J to subvectors $\boldsymbol{x}_I$ and $\boldsymbol{x}'_J$, respectively, such that there exists an index $i \in I \cap J$ where $x_i \neq x'_i$. Furthermore, we require this property to hold even if the setup algorithm is public coin.

Definition 10 ((Public-Coin) Position Binding). *A subvector commitment* SVC *over* $\mathcal{X}$ *is position binding if for any* PPT *adversary* $\mathcal{A}$*, there exists a negligible function* $\epsilon(\lambda) \in \mathsf{negl}(\lambda)$ *such that*

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

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 with* public coins.

We do not define hiding as it is not needed for our purpose. However, as discussed in [27], one can construct a hiding VC generically by committing to (normal) commitments using VC. This naturally extends to SVC as well.

4.1 Linear Map Commitments

Functional commitments for linear functions, specifically for linear forms $f : \mathbb{F}^\ell \to \mathbb{F}$ for some field $\mathbb{F}$, were introduced by Libert, Ramanna and Yung [48] and is a generalization of vector commitments (VC) introduced by Catalano and Fiore [27]. 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* linear forms or, equivalently, to the output of a *linear map* $f : \mathbb{F}^\ell \to \mathbb{F}^q$. Note that any linear map from $\mathbb{F}^\ell$ to $\mathbb{F}^q$ can be represented by a matrix $F \in \mathbb{F}^{q \times \ell}$.

Definition 11 (Linear Map Commitments (LMC)). *A linear map commitment scheme* LMC *over* $\mathbb{F}$ *consists of the following* PPT *algorithms* $(\mathsf{Setup}, \mathsf{Com}, \mathsf{Open}, \mathsf{Verify})$*:*

$\underline{\mathsf{Setup}(1^\lambda, \mathcal{F}; \omega)}$*: Let* $\ell, q \in \mathsf{poly}(\lambda)$ *be positive integers, and* $\mathcal{F} \subseteq \{f : \mathbb{F}^\ell \to \mathbb{F}^q\}$ *be a family of linear maps. The* deterministic *setup algorithm inputs the security parameter* 1^λ*, the description of the 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* $\boldsymbol{x} \in \mathbb{F}^\ell$*. It outputs a commitment string* C *and some auxiliary information* aux.

$\underline{\mathsf{Open}(f, \boldsymbol{y}, \mathsf{aux})}$*: The opening algorithm inputs an* $f \in \mathcal{F}$*, an image* $\boldsymbol{y} \in \mathbb{F}^q$*, 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*, an* $f \in \mathcal{F}$*, 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* $\boldsymbol{y} = f(\boldsymbol{x})$.

In the following we define correctness and compactness for LMCs.

Definition 12 (Correctness). *A linear map commitment scheme* LMC *over* $\mathbb{F}$ *is said to be correct if, for any security parameter and length* $\lambda, \ell, q \in \mathbb{N}$*, random tape* $\omega \in \{0,1\}^\lambda$*, linear map family* $\mathcal{F} \subseteq \left\{f : \mathbb{F}^\ell \to \mathbb{F}^q\right\}$*, public parameters* $\mathsf{pp} \in \mathsf{Setup}(1^\lambda, \mathcal{F}; \omega)$*,* $\boldsymbol{x} \in \mathbb{F}^\ell$*, linear map* $f \in \mathcal{F}$*,* $(C, \mathsf{aux}) \in \mathsf{Com}(\boldsymbol{x})$*,* $\Lambda \in \mathsf{Open}(f, f(\boldsymbol{x}), \mathsf{aux})$*, there exists* $\epsilon(\lambda) \in \mathsf{negl}(\lambda)$ *such that*

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

Definition 13 (Compactness). *A linear map commitment* LMC *over* $\mathbb{F}$ *is compact if there exists a universal polynomial* $p \in \mathsf{poly}(\lambda)$*, such that for any* $\ell, q \in \mathsf{poly}(\lambda)$*, family of linear maps* $\mathcal{F} \subseteq \{f : \mathbb{F}^\ell \to \mathbb{F}^q\}$*, random tape* $\omega \in \{0,1\}^\lambda$*, public parameters* $\mathsf{pp} \in \mathsf{Setup}(1^\lambda, \mathcal{F}; \omega)$*, vector* $\boldsymbol{x} \in \mathbb{F}^\ell$*, linear map* $f \in \mathcal{F}$*,* $(C, \mathsf{aux}) \in \mathsf{Com}(\boldsymbol{x})$*,* $\Lambda \in \mathsf{Open}(f, f(\boldsymbol{x}), \mathsf{aux})$*, it holds that* $|C| \leq p(\lambda)$ *and* $|\Lambda| \leq p(\lambda)$*.*

We next generalize the notion of function binding for linear maps. The original definition, as considered by Libert, Ramanna and Yung [48], requires 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 14 ((Public-Coin) Function Binding). *A linear map commitment* LMC *over* $\mathbb{F}$ *is function binding if for any* PPT *adversary* $\mathcal{A}$*, positive integers* $Q, \ell, q \in \mathsf{poly}(\lambda)$*, and family of linear maps* $\mathcal{F} \subseteq \{f : \mathbb{F}^\ell \to \mathbb{F}^q\}$*, there exists a negligible function* $\epsilon(\lambda) \in \mathsf{negl}(\lambda)$ *such that*

$$\Pr\left[\begin{array}{c|c} \forall k \in [Q], \begin{array}{c} f_k \in \mathcal{F} \wedge \boldsymbol{y}_k \in \mathbb{F}^q \wedge \\ \mathsf{Verify}(C, f_k, \boldsymbol{y}_k, \Lambda_k) = 1 \end{array} & \begin{array}{c} \omega \leftarrow_{\$} \{0,1\}^\lambda \\ \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F}; \omega) \end{array} \\ \nexists \boldsymbol{x} \in \mathcal{X}^\ell \ s.t. \ \forall k \in [Q], \ f_k(\boldsymbol{x}) = \boldsymbol{y}_k & (C, \{(f_k, \boldsymbol{y}_k, \Lambda_k)\}_{k \in [Q]}) \leftarrow \mathcal{A}(\mathsf{pp}\boxed{, \omega}) \end{array}\right] \leq \epsilon(\lambda)$$

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* LMC *is function binding with* public coins.

As for SVC, we omit the hiding definition as it is not needed for our purpose.

5 Constructions for SVCs

We propose 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. Both schemes allow one to commit to binary strings (i.e., we consider the field $\mathcal{X} = \mathbb{F}_2$). Our constructions are inspired by the work of Catalano and Fiore [27] 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.

5.1 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) is hard. Let $\ell \in \mathsf{poly}(\lambda)$ 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

```
$\mathsf{Setup}(1^\lambda, 1^\ell; \omega)$
    $(R_D, X) \leftarrow_\$ \mathsf{MGen}(1^\lambda, \omega)$
    $(e_1, \dots, e_\ell) \leftarrow H(R_D, X)$
    $\forall i \in [\ell], S_i := \left(\prod_{j \in [\ell] \setminus \{i\}} e_j\right) \circ X$
    $\boldsymbol{S} := (S_1, \dots, S_\ell)^T, \ \boldsymbol{e} := (e_1, \dots, e_\ell)^T$
    return $\mathsf{pp} := (R_D, X, \boldsymbol{S}, \boldsymbol{e})$

$\mathsf{Com}(\boldsymbol{x})$
    return $(C, \mathsf{aux}) := (\langle \boldsymbol{x}, \boldsymbol{S} \rangle, \boldsymbol{x})$

$\mathsf{Open}(I, \boldsymbol{x}'_I, \mathsf{aux})$
    parse $\mathsf{aux}$ as $\boldsymbol{x}$
    $\Lambda_I := \left(\prod_{i \in I} e_i\right)^{-1} \circ \langle \boldsymbol{x}_{[\ell] \setminus I}, \boldsymbol{S}_{[\ell] \setminus I} \rangle$
    return $\Lambda_I$

$\mathsf{Verify}(C, I, \boldsymbol{x}'_I, \Lambda_I)$
    $b_0 := (\boldsymbol{x}'_I \in \mathcal{M}^{|I|})$
    $b_1 := (C = \langle \boldsymbol{x}'_I, \boldsymbol{S}_I \rangle + \left(\prod_{i \in I} e_i\right) \circ \Lambda_I)$
    return $b_0 \cap b_1$
```

Fig. 1: SVC from the Root Assumption.

of prime elements in R of size 2^λ. Let $H : \{0,1\}^* \to \mathrm{IRR}_\lambda(R)^\ell$ be a prime-valued function which maps finite bit strings to tuples of ℓ distinct elements in $\mathrm{IRR}_\lambda(R)$. That is, for all string $s \in \{0,1\}^*$, if $(e_1, \dots, e_\ell) = H(s)$, then $e_i \neq e_j$ for all $i, j \in [q]$ where $i \neq j$. Let $\mathcal{X} := \{0_R, 1_R\}$[5] 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 1. Note that in the opening algorithm, it is required to compute

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

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 $i \in [\ell] \setminus I$ and hence for all $i \in J \setminus I$, we have

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

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

- 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 1.
- H is a random oracle: Our construction is secure if the adaptive root problem (introduced in [20]) is hard over the module family. This is shown in Theorem 2.

Theorem 1. *If the (resp. public-coin) strong distinct-prime-product root assumption holds over the module family $\mathcal{R}_\mathcal{D}$, then the scheme in Figure 1 is (resp. public-coin) position binding.*

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

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

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

for some polynomial $f(\lambda) \in \mathsf{poly}(\lambda)$, where $\mathcal{A}$ gets ω as input (highlighted by the dashed box) only in the public-coin variant. We construct an algorithm $\mathcal{C}$ as follows, whose existence contracts the fact that $\mathcal{R}_{\mathcal{D}}$ is a (public-coin) strong distinct-prime-product root modules family.

In the pivate-coin 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 $(e_1, \ldots, e_\ell) \leftarrow H(R_D, X)$. It then sets $S_i := \left(\prod_{j \in [\ell] \setminus \{i\}} e_j\right) \circ X$ for all $i \in [\ell]$, $\boldsymbol{S} := (S_1, \ldots, S_q)^T$, and $\boldsymbol{e} := (e_1, \ldots, e_\ell)$. It sets $\mathsf{pp} := (R_D, X, \boldsymbol{S}, \boldsymbol{e})$ and runs $\mathcal{A}$ on input $(1^\lambda, \mathsf{pp})$. In the public-coin 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

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

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 $S_i = \left(\prod_{j \in [\ell] \setminus \{i\}} e_j\right) \circ A$. 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

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

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 $d := \gcd\left(\sum_{i \in I \cup J} \delta_i \prod_{j \in [\ell] \setminus \{i\}} e_j, \prod_{i \in I \cap J} e_i\right) = \prod_{j \in K_0} e_j$. 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

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

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

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

Multiplying this to A, it gets

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

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. □

Theorem 2. *If the (resp. public-coin) adaptive root assumption holds over the module family $\mathcal{R}_{\mathcal{D}}$ with respect to IRR_λ, then the scheme in Figure 1 is (resp. public-coin) position binding in the random oracle model.*

Due to space constraints, we refer to [47] for a full proof.

Efficiency and Optimizations. Our construction admits two complementary instantiations, discussed in the following.

- Efficient Verifier (assuming random access to public parameters): The vectors $\boldsymbol{S}$ and $\boldsymbol{e}$ are explicitly included in the public parameters (as it is currently described). In this case, and suppose the verifier has random access to each e_i and S_i, the computational effort of the verifier is only proportional to $|I|$, the size of the subvector. The shortcoming of this scheme is that the size of the public parameters is linear in ℓ, which can be very large depending on the application.
- Short Public Parameters: One can reduce the size of the public parameters to a constant by including only the module description (R_D, X) and letting each algorithm recompute the terms of $\boldsymbol{S}$ needed for the computations. This however increases the computational complexity of the verifier, since the computation needed for each element of $\boldsymbol{S}$ is linear in the vector length ℓ. This can be partiallly amortized by observing that the values $(S_1, \ldots, S_\ell)$ do not depend on the committed vector and can be precomputed by both parties.

Another possible tradeoff is given by the assumption that one is willing to rely on: Note that the main workload for the verifier (in the verifier-optimized variant) 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 $(e_1, \ldots, e_\ell)$ 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 $(e_1, \ldots, e_\ell)$ to be the smallest ℓ primes. Since $\ell \in \mathsf{poly}(\lambda)$, each prime can be represented by $O(\log \lambda)$ bits. This greatly reduces the computational effort of the verifier.

5.2 SVC from the Computational Diffie-Hellman Assumption

Next we present our SVC construction from pairing groups. In favor of a simpler presentation and a more general result we describe our scheme assuming symmetric pairings. However, we stress that the scheme can be easily adapted to work over the more efficient asymmetric (type III) bilinear groups without affecting computational efficiency nor opening size by, e.g., replicating all public parameters in both source 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. Assuming that the verifier has random access to each G_i in the public parameters, it can check the relation by accessing $|I|$ entries in

```
$\mathsf{Setup}(1^\lambda, 1^\ell; \omega)$
    $(p, \mathbb{G}, \mathbb{G}_T, G, e) \leftarrow \mathsf{GGen}(1^\lambda; \omega)$
    $\forall i \in [\ell],\ z_i \leftarrow_{\$} \mathbb{Z}_p$
    $\forall i, i' \in [\ell],\ G_i := G^{z_i},\ H_{i,i'} := G^{z_i z_{i'}}$
    $\mathsf{pp} := \begin{pmatrix} p, \mathbb{G}, \mathbb{G}_T, G, \{G_i\}_{i \in [\ell]}, \\ \{H_{i,i'}\}_{i,i' \in [\ell], i \neq i'}, e \end{pmatrix}$
    return $\mathsf{pp}$

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

$\mathsf{Open}(I, \boldsymbol{x}'_I, \mathsf{aux})$
    parse $\mathsf{aux}$ 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. 2: SVC from CDH.

the public parameters, and computing $2 \cdot |I|$ group operations and 2 pairings (which are independent of ℓ). Since the public parameters are highly structured, this scheme does not admit an instantiation with short public parameters, which grow quadratically with the vector size ℓ.

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 $\mathcal{X} := \mathbb{Z}_p$. Our second subvector commitment scheme is shown in Figure 2. In the following we show that our SVC scheme is position binding with a private-coin setup.

Theorem 3. *If the computational Diffie-Hellman (CDH) assumption holds with respect to* GGen*, then the scheme in Figure 2 is position binding.*

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

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

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 $i^* \leftarrow_{\$} [\ell]$ and set $G_{i^*} := H$. Symbolically, let $z_{i^*} := z$, which is not known by $\mathcal{C}$. For the other indices $i, i' \in [\ell] \setminus \{i^*\}$, 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 $i \in [\ell] \setminus \{i^*\}$. It then sets $\mathsf{pp} = (p, \mathbb{G}, \mathbb{G}_T, G, \{G_i\}_{i \in [\ell]}, \{H_{i,i'}\}_{i,i' \in [\ell], i \neq i'}, e)$, 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 $1/\ell$, it holds that $i^* \in I \cap J$ and $x_{i^*} \neq x'_{i^*}$. By examining the verification equations, we have

$$e\left(\prod_{i\in I} G_i^{x_i}, \prod_{i\in I} G_i\right) \cdot e(\Lambda_I, G) = e\left(\prod_{i\in J} G_i^{x'_i}, \prod_{i\in J} G_i\right) \cdot e(\Lambda_J, G)$$

$$e\left(\prod_{i\in J} G_i^{x'_i}, \prod_{i\in J} G_i\right) \cdot e\left(\prod_{i\in I} G_i^{-m_i}, \prod_{i\in I} G_i\right) = e(\Lambda, G), \text{ where } \Lambda := \Lambda_I/\Lambda_J$$

$$\left(\sum_{i\in J} z_i x'_i\right)\left(\sum_{i\in J} z_i\right) - \left(\sum_{i\in I} z_i x_i\right)\left(\sum_{i\in I} z_i\right) = \log_G \Lambda$$

$$\alpha z_{i^*}^2 + \beta z_{i^*} + \gamma = \log_G \Lambda$$

where

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

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

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. □

6 Construction for LMC

Our LMC construction is inspired by the scheme presented in [48] and it is based upon the following observations. First, when the vectors $\boldsymbol{x}, \boldsymbol{f} \in \mathbb{F}^\ell$ for some field $\mathbb{F}$ are encoded as the polynomials $p_{\boldsymbol{f}}(\alpha) := \sum_{j\in[\ell]} f_j \alpha^{\ell+1-j}$ and $p_{\boldsymbol{x}}(\alpha) := \sum_{j\in[\ell]} x_j \alpha^j$ with variable α respectively, their inner product is the coefficient of the monomial $\alpha^{\ell+1}$ in the polynomial product $p_{\boldsymbol{f}}(\alpha) p_{\boldsymbol{x}}(\alpha)$. Second, due to linearity of polynomial multiplication, if a matrix $F \in \mathbb{F}^{q\times\ell}$ is encoded in the polynomial $p_F(\alpha) := \sum_{i\in[q],j\in[\ell]} f_{i,j} z_i \alpha^{\ell+1-j}$ with variables $(\alpha, z_1, \ldots, z_q)$, then the matrix-vector product $F\boldsymbol{x}$ is given in the coefficients of the monomials $z_i \alpha^{\ell+1}$ for $i \in [q]$ in the polynomial $p_F(\alpha) p_{\boldsymbol{x}}(\alpha)$.

With the above observations, we give an overview of our construction. We let the commitment C to $\boldsymbol{x}$ be $G^{p_{\boldsymbol{x}}(\alpha)}$, which is computable by combining elements of the form G^{α^j} given in the public parameters. Given $(F, \boldsymbol{y})$, to verify that $F\boldsymbol{x} = \boldsymbol{y}$, the verifier computes via pairing $e(G^{p_F(\alpha, z_1, \ldots, z_q)}, G^{p_{\boldsymbol{x}}(\alpha)})$, where the left-input is computable by combining elements of the form $G^{z_i \alpha^j}$ given in the public parameters. If the relation $F\boldsymbol{x} = \boldsymbol{y}$ indeed holds, then the coefficients of $\boldsymbol{y}$ must be encoded as the coefficients of the (lifted) monomials $G^{z_i \alpha^{\ell+1}}$. To convince the verifier that this is the case, it suffices for the prover to provide the remaining terms of the product polynomial.

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 $\mathbb{F} = \mathbb{Z}_p$, $\ell, q \in \mathbb{N}$, and $\mathcal{F}$ be the set of all

```
$\mathsf{Setup}(1^\lambda, \mathcal{F}; \omega)$
    $(p, \mathbb{G}, \mathbb{G}_T, G, e) \leftarrow \mathsf{GGen}(1^\lambda; \omega)$
    $\alpha, z_1, \ldots, z_q \leftarrow_\$ \mathbb{Z}_p$
    $\forall j \in [\ell],\ G_j := G^{\alpha^j}$
    $\forall i \in [q], j \in [2\ell],\ H_{i,j} := G^{z_i \alpha^j}$
    $\mathsf{pp} := \begin{pmatrix} p, \mathbb{G}, \mathbb{G}_T, G, \{G_j\}_{j \in [\ell]}, \\ \{H_{i,j}\}_{i \in [q], j \in [2\ell] \setminus \{\ell+1\}}, e \end{pmatrix}$
    return pp

$\mathsf{Com}(\boldsymbol{x})$
    return $(C, \mathsf{aux}) := \left( \prod_{j \in [\ell]} G_j^{x_j}, \boldsymbol{x} \right)$

$\mathsf{Open}(F, \boldsymbol{y}, \mathsf{aux})$
    parse aux as $\boldsymbol{x}$
    $\Lambda := \prod_{i \in [q]} \prod_{j \in [\ell]} \prod_{j' \in [\ell] \setminus \{j\}} H_{i,\ell+1+j-j'}^{f_{i,j} x_{j'}}$
    return $\Lambda$

$\mathsf{Verify}(C, F, \boldsymbol{y}, \Lambda)$
    $b_0 := (\boldsymbol{y} \in \mathbb{Z}_p^q)$
    $b_1 := \begin{pmatrix} e\left(C, \prod_{i \in [q]} \prod_{j \in [\ell]} H_{i,\ell+1-j}^{f_{i,j}}\right) = \\ e(G_1, \prod_{i \in [q]} H_{i,\ell}^{y_i}) \cdot e(\Lambda, G) \end{pmatrix}$
    return $b_0 \cap b_1$
```

Fig. 3: LMC from Bilinear Pairings.

linear maps from $\mathbb{Z}_p^\ell$ to $\mathbb{Z}_p^q$. Our LMC for $\mathbb{Z}_p$ is given in Figure 3. For full generality we present the construction over symmetric pairings, however one can easily convert it to the more efficient asymmetric pairing groups via standard techniques, without affecting the size of the openings. Although we do not aim to achieve the hiding property, our construction can be easily modified to be hiding, by introducing randomness similar to that in Pedersen commitment [56]. Indeed this is how the FC of [48] achieves hiding. We show that our construction is function binding (in the generic bilinear group model) in the following.

Theorem 4. *Let $\ell, q \in \mathsf{poly}(\lambda)$ and $1/p \in \mathsf{negl}(\lambda)$. The scheme in Figure 3 is function binding in the generic bilinear group model.*

Proof. The proof uses the generic group model abstraction of Shoup [59] and we refer the reader to [19] for a comprehensive introduction to the bilinear group model. Here we state the central lemma useful for proving facts about generic attackers.

Lemma 1 (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 $Q \in \mathbb{N}$. 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

$$\Pr\left[\begin{array}{c|c} \forall k \in [Q], \begin{array}{c} F_k \in \mathbb{Z}_p^{q \times \ell} \wedge \boldsymbol{y}_k \in \mathbb{Z}_p^q \wedge \\ \mathsf{Verify}(C, f_k, \boldsymbol{y}_k, \Lambda_k) = 1 \end{array} & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F}) \\ \nexists \boldsymbol{x} \in \mathbb{Z}_p^\ell\ s.t.\ \forall k \in [Q],\ F_k(\boldsymbol{x}) = \boldsymbol{y}_k & (C, \{(F_k, \boldsymbol{y}_k, \Lambda_k)\}_{k \in [Q]}) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp}) \end{array}\right] > \frac{1}{f(\lambda)}.$$

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

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

$$\log_G \varLambda_k = \lambda_{k,0} + \sum_{j\in[\ell]} \lambda_{k,j} \alpha^j + \sum_{\substack{i\in[q] \\ j\in[2\ell]\setminus\{\ell+1\}}} \lambda_{k,i,j} z_i \alpha^j$$

for some integer coefficients γ_j, $\gamma_{i,j}$, $\lambda_{k,j}$, and $\lambda_{k,i,j}$ for i, j, and k in the appropriate ranges. Since for each $k \in [Q]$, $\mathsf{Verify}(C, F_k, \boldsymbol{y}_k, \varLambda_k) = 1$, the following relations hold:

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

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

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

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

7 Succinct Arguments of Knowledge from SVC / LMC

We present our compiler for constructing interactive arguments of knowledge either from traditional PCPs and subvector commitments (Section 5), or from linear PCPs [42] and linear map commitments (Section 6). The constructions for both cases are in fact identical and we present only the latter since it is strictly more general (an traditional PCP can be seen as a linear PCP where queries are restricted to unit vectors).

Let $(\mathcal{P}_{\text{PCP}}, \mathcal{V}_{\text{PCP}})$ be an ℓ-long q-query (linear) PCP over some field $\mathbb{F}$ for NP with r being the length of the random coins of the possibly adaptive verifier. Let $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^r$ be a pseudo-random generator and let $\mathsf{LMC} := (\mathsf{Setup}, \mathsf{Com}, \mathsf{Open}, \mathsf{Verify})$ be a linear map commitment for the set of all linear maps $\mathcal{F}$ from $\mathbb{F}^\ell$ to $\mathbb{F}^q$, possibly with public-coin setup. We present a 4-move interactive argument of knowledge in Figure 4.

```
$\mathcal{S}(1^\lambda;\omega)$
$\omega \leftarrow_\$ \{0,1\}^\lambda$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F}; \omega)$
$\mathbf{return}\ y := (\mathsf{pp}, \omega)$

$\mathcal{P}(x, w, y)$                                   $\mathcal{V}(x, y)$
$\boldsymbol{\pi} \leftarrow \mathcal{P}_{\text{PCP}}(x, w)$
$(C, \mathsf{aux}) \leftarrow \mathsf{Com}(\boldsymbol{\pi})$
                          $\xrightarrow{C}$
                                                         $\alpha \leftarrow_\$ \{0,1\}^\lambda$
                          $\xleftarrow{\alpha}$
$\rho \leftarrow \mathsf{PRG}(\alpha)$                   $\rho \leftarrow \mathsf{PRG}(\alpha)$
$F := \mathsf{Record}(x, \boldsymbol{\pi}, \rho)$
$\boldsymbol{y} := F\boldsymbol{\pi}$
$\Lambda \leftarrow \mathsf{Open}(F, \boldsymbol{y}, \mathsf{aux})$
                          $\xrightarrow{\Lambda, \boldsymbol{y}}$
                                                         $F := \mathsf{Reconstruct}(x, \boldsymbol{y}, \rho)$
                                                         $b_0 := \mathsf{Verify}(C, F, \boldsymbol{y}, \Lambda)$
                                                         $b_1 := \mathsf{Decide}(x, \boldsymbol{y}, \rho)$
                                                         $\mathbf{return}\ b_0 \cap b_1$
```

Fig. 4: Succinct Argument of Knowledge for NP from SVC / LMC

7.1 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:

- 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 $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_q \in \mathbb{F}^q$ made by $\mathcal{V}_{\text{PCP}}$. It outputs a query matrix $F := [\boldsymbol{f}_1|\ldots|\boldsymbol{f}_q]^T \in \mathbb{F}^{q\times\ell}$.
- Reconstruct: On input a statement x, a response vector $\boldsymbol{y} \in \mathbb{F}^q$, 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 $i \in [q]$, it responds by returning the value y_i. It outputs a query matrix $F := [\boldsymbol{f}_1|\ldots|\boldsymbol{f}_q]^T \in \mathbb{F}^{q\times\ell}$.
- Decide: On input a statement x, a response vector $\boldsymbol{y} \in \mathbb{F}^q$, 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 LMC using ω. It outputs pp if an LMC with private-coin setup is used, which results in an argument system with private-coin setup. Alternatively, if an LMC with public-coin 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 ρ and executes $\mathcal{V}_{\text{PCP}}$ on ρ. Here the PRG is used to compress the (possibly large) randomness of the verifier, which is strictly needed only for linear PCPs (standard PCPs typically have low randomness complexity and therefore the random coins can be sent in plain).

The prover $\mathcal{P}$ then records the sets of queries $F = \mathsf{Record}(x, \boldsymbol{\pi}, \rho)$ of $\mathcal{V}_{\text{PCP}}$ using randomness ρ to $\boldsymbol{\pi}$, and computes the responses $\boldsymbol{y} = F\boldsymbol{\pi}$. Next, it computes the opening Λ of the commitment C to the tuple $(F, \boldsymbol{y})$. The opening Λ along with the response $\boldsymbol{y}$ are sent to the verifier $\mathcal{V}$. The verifier $\mathcal{V}$ runs $\mathsf{Reconstruct}(x, \boldsymbol{y}, \rho)$ to reconstruct the query matrix F. It then checks if Λ is a valid opening of C to $(F, \boldsymbol{y})$. Finally, it checks if $\mathsf{Decide}(x, \boldsymbol{y}, \rho)$ returns 1. If all checks are passed, it outputs 1. Otherwise, it outputs 0.

7.2 Analysis

Clearly, if $(\mathcal{P}_{\text{PCP}}, \mathcal{V}_{\text{PCP}})$ is a complete linear PCP, and LMC is a correct LMC, then the argument system is complete. Alternatively, if $(\mathcal{P}_{\text{PCP}}, \mathcal{V}_{\text{PCP}})$ is a complete (traditional) PCP, and LMC is a correct SVC, then the system is also complete. The succinctness of the system follows directly from the compactness of LMC. Next, we show that the argument system is of knowledge by the following theorem. Due to space constraints, we refer to [47] for a full proof.

Theorem 5. *Let $(\mathcal{P}_{PCP}, \mathcal{V}_{PCP})$ be a $2^{-\sigma}$-sound linear PCP of knowledge for NP,* PRG *be a pseudo-random generator, and* $\mathsf{LMC} := (\mathsf{Setup}, \mathsf{Com}, \mathsf{Open}, \mathsf{Verify})$ *be (resp. public-coin) function binding. Then the protocol in Figure 4 is a $2^{-\sigma}$-sound (resp. public-coin) argument of knowledge.*

7.3 Instantiations and Efficiency.

Since our argument system has a public-coin verifier, we can apply the Fiat-Shamir transformation to turn it into a non-interactive argument and sometimes a SNARK.[6] We highlight some interesting instantiations of our compiler: Regardless of the specific root assumption used, we can instantiate our first SVC construction over $Cl(\Delta)$, the class group of an imaginary quadratic order with discriminant Δ. Considering the current best attacks, we can assume that root problems for a $O(\lambda^2)$-bit Δ are hard for a 2^{λ}-time adversary. Concretely, with a 2560-bit Δ, which roughly offers security against a 2^{128}-time adversary, each element in $Cl(\Delta)$ can be represented by at most 2560 bits

[6] In the original definition of Bitansky *et al.* [16], a SNARK verifier is a Turing machine with runtime logarithmic in that of the corresponding NP verifier. We consider a relaxed definition where the SNARK verifier is a random access machine.

(see Section 8 for more details). Using a 240-query 2^{-80}-sound PCP, the resulting proof size is $2 \cdot 2560 + 240 = 5360$ bits. When using the verifier-optimized SVC (see Section 5.1) the workload of the verifier is dominated by 240 exponentiations, regardless of the witness size. However the public parameters grow linearly with the length of the PCP encoding. One can reduce the size of the public parameters to constant at the cost of having an inefficient verifier. We stress that class groups of imaginary quadratic orders have a public-coin setup and so does the resulting SNARK.

Alternatively, we can use our second SVC construction over the pairing-friendly 256-bit Barreto-Naehrig curve [7], which roughly offers security against 2^{128}-time adversaries. In such a curve, each group element can be represented by 256 bits. Therefore the resulting proof size is $2 \cdot 256 + 240 = 752$ bits. This marginally improves over the shortest proofs known [40]. A shortcoming of this approach is that the public parameters of the resulting SNARK grow quadratically in the length of the PCP proof.

An unsatisfactory aspect of the instantiations above is that PCPs with such short queries have typically a very high prover complexity and are therefore very expensive to compute, which means that our arguments described above have a high prover complexity. One approach to address this issue is to leverage the large body of work on linear PCPs [42,17], which significantly improve the complexity of the prover. Any of these schemes can be used in combination with an LMC (such as the construction of Section 6) to obtain a non-interactive argument with slightly larger proofs (by a constant factor) but with a more efficient prover. We stress that our compiler supports *any* linear PCP, whereas existing compilers only support those with a verifier who only evaluates quadratic polynomials. Moreover, although our pairing-based instantiations inherit the private-coin setup from underlying SVC / LMC, the setup is statement-independent. In contrast, the setup in existing pairing-based schemes such as [40] depends on the statement to be proven. We shall mention however that our LMC has a linear verifier complexity and therefore it yields an argument with verifier computation linear in the lenght of the PCP.

For the efficiency of the verifier, there are several techniques to reduce its computational overhead: As an example, one could compose our scheme with a verifier-optimized SNARK to prove the validity of the verification equation, instead of having the verifier computing it. Very recently, Boneh et al. [18] presented a special-purpose proof of knowledge of co-prime roots (PoKCR) that drastically reduces the running time of the verifier in class group-based SVCs (see Section 5) by trading group operations for modular multiplications and additions, which are orders of magnitude more efficient. We refer the reader to [18] for a detailed analysis of the concrete costs.

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 [25]. We refer to, *e.g.*, [23,24], 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 [23]. 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 $q^2 a$ ($= a$ 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 $\boldsymbol{Cl(\Delta)}$. 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 [25]. Although the strong root problem in $Cl(\Delta)$ is not as well studied, it is shown to be hard for generic group algorithms [31]. 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 2560-bit Δ offers approximately 128 bits of computational security.

The (resp. public-coin setup) position binding property of our first construction of SVC can be proven under either the (resp. public-coin setup) strong distinct-prime-product root assumption or the (resp. public-coin setup) adaptive root assumption. 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 private-coin setup setting, it is clear that the strong distinct-prime-product root assumption is implied by the standard strong root assumption. In the public-coin setup setting, it is conjectured [63,20] 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[7]. 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 with *public-coin setup* does not hold over $\mathbb{Z}_N^*$.

RSA-UFOs. The problem of constructing RSA-based accumulators without trapdoors was considered by Sander [58], 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 [58]. 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

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

RSA assumption in the presence of a random oracle. This result is implicitly shown by Sander [58] and a proof sketch is given in [47].

Acknoweldgements

We thank Yuval Ishai and Eran Tromer for insighful discussions and comments on an earlier draft of this work. Research supported in part by a gift from DoS Networks.

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 *2015 IEEE Symposium on Security and Privacy*, pages 271–286. IEEE Computer Society Press, May 2015.
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. Springer, Heidelberg, October 2003.
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. Springer, Heidelberg, August 2006.
8. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.
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 III*, volume 10212 of *LNCS*, pages 551–579. Springer, Heidelberg, April / May 2017.
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. Springer, Heidelberg, August 2013.
12. Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Heidelberg, May 2019.

13. 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. Springer, Heidelberg, October / November 2016.
14. 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. Springer, Heidelberg, August 2014.
15. 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.
16. Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In Shafi Goldwasser, editor, *ITCS 2012*, pages 326–349. ACM, January 2012.
17. 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. Springer, Heidelberg, March 2013.
18. Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching techniques for accumulators with applications to iops and stateless blockchains. Cryptology ePrint Archive, Report 2018/1188, 2018. https://eprint.iacr.org/2018/1188.
19. 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. Springer, Heidelberg, May 2004.
20. 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.
21. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357. Springer, Heidelberg, May 2016.
22. Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.
23. Johannes Buchmann and Safuat Hamdy. A survey on iq-cryptography. In *Tech. Report TI-4/01, Technische Universitäat Darmstadt, Fachbereich Informatik*, 2000.
24. 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.
25. Johannes Buchmann and Hugh C. Williams. A key-exchange system based on imaginary quadratic fields. *Journal of Cryptology*, 1(2):107–118, June 1988.
26. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.
27. Dario Catalano and Dario Fiore. Vector commitments and their applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 55–72. Springer, Heidelberg, February / March 2013.
28. 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. Springer, Heidelberg, April 2015.
29. Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno, and Samee Zahur. Geppetto: Versatile verifiable computation. In *2015 IEEE Symposium on Security and Privacy*, pages 253–270. IEEE Computer Society Press, May 2015.

30. Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 125–142. Springer, Heidelberg, December 2002.
31. 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. Springer, Heidelberg, April / May 2002.
32. 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. Springer, Heidelberg, December 2014.
33. 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.
34. Giovanni Di Crescenzo and Helger Lipmaa. Succinct np proofs from an extractability assumption. In *Conference on Computability in Europe*, pages 175–185. Springer, 2008.
35. 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. Springer, Heidelberg, August 1987.
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. Springer, Heidelberg, May 2013.
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. Springer, Heidelberg, December 2010.
40. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.
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. Springer, Heidelberg, December 2000.
42. 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.
43. 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. ACM Press, June 2007.
44. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Heidelberg, December 2010.
45. Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992.
46. Joe Kilian. Improved efficient arguments (preliminary version). In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 311–324. Springer, Heidelberg, August 1995.
47. Russell W.F. Lai and Giulio Malavolta. Subvector commitments with applications to succinct arguments. Cryptology ePrint Archive, Report 2018/705, 2018. https://eprint.iacr.org/2018/705.

48. 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. Schloss Dagstuhl, July 2016.
49. Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Output-compressing randomized encodings and applications. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 96–124. Springer, Heidelberg, January 2016.
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. Springer, Heidelberg, March 2012.
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. Springer, Heidelberg, June 2012.
52. Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 369–378. Springer, Heidelberg, August 1988.
53. Silvio Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, November 1994.
54. Silvio Micali, Michael O. Rabin, and Joe Kilian. Zero-knowledge sets. In *44th FOCS*, pages 80–91. IEEE Computer Society Press, October 2003.
55. Thilo Mie. Polylogarithmic two-round argument systems. *Journal of Mathematical Cryptology*, 2(4):343–363, 2008.
56. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992.
57. 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. ACM Press, June 2016.
58. Tomas Sander. Efficient accumulators without trapdoor extended abstracts. In Vijay Varadharajan and Yi Mu, editors, *ICICS 99*, volume 1726 of *LNCS*, pages 252–262. Springer, Heidelberg, November 1999.
59. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.
60. Victor Shoup. OAEP reconsidered. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 239–259. Springer, Heidelberg, August 2001.
61. Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 1–18. Springer, Heidelberg, March 2008.
62. Riad S Wahby, Ioanna Tzialla, Justin Thaler, and Michael Walfish. Doubly-efficient zksnarks without trusted setup.
63. Benjamin Wesolowski. Efficient verifiable delay functions. *IACR Cryptology ePrint Archive*, 2018:623, 2018.
64. 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. IEEE Computer Society Press, May 2017.