CRS-Updatable Asymmetric Quasi-Adaptive NIZK Arguments

Behzad Abdolmaleki[1] and Daniel Slamanig[2]

[1] Max Planck Institute for Security and Privacy, Bochum, Germany
behzad.abdolmaleki@mpi-sp.org
[2] AIT Austrian Institute of Technology, Vienna, Austria
daniel.slamanig@ait.ac.at

Abstract. A critical aspect for the practical use of non-interactive zero-knowledge (NIZK) arguments in the common reference string (CRS) model is the demand for a trusted setup, i.e., a trusted generation of the CRS. Recently, motivated by its increased use in real-world applications, there has been a growing interest in concepts that allow to reduce the trust in this setup. In particular one demands that the zero-knowledge and ideally also the soundness property hold even when the CRS generation is subverted. One important line of work in this direction is the so-called updatable CRS for NIZK by Groth *et al.* (CRYPTO'18). The basic idea is that everyone can update a CRS and there is a way to check the correctness of an update. This guarantees that if at least one operation (the generation or one update) have been performed honestly, the zero-knowledge and the soundness properties hold. Later, Lipmaa (SCN'20) adopted this notion of updatable CRS to quasi-adaptive NIZK (QA-NIZK) arguments.
In this work, we continue the study of CRS-updatable QA-NIZK and analyse the most efficient asymmetric QA-NIZKs by González *et al.* (ASIACRYPT'15) in a setting where the CRS is fully subverted and propose an updatable version of it. In contrast to the updatable QA-NIZK by Lipmaa (SCN'20) which represents a *symmetric* QA-NIZK and requires a new non-standard knowledge assumption for the subversion zero-knowledge property, our technique to construct updatable *asymmetric* QA-NIZK is under a well-known standard knowledge assumption, i.e., the Bilinear Diffie-Hellman Knowledge of Exponents assumption. Furthermore, we show the knowledge soundness of the (updatable) asymmetric QA-NIZKs, an open problem posed by Lipmaa, which makes them compatible with modular zk-SNARK frameworks such as LegoSNARK by Campanelli *et al.* (ACM CCS'19).

1 Introduction

Zero-knowledge proofs [24] are a fundamental concept which allows one party (the prover) by interacting with another party (the verifier) to convince the latter that a statement in any NP language is true without revealing any additional information (the zero-knowledge property). At the same time, the prover is not

able to make the verifier accept proofs about false statements (the soundness property). In many of its practical applications it is important to remove interaction in that the prover only needs to compute a single message (a proof), which can then be verified by everyone. These so called non-interactive zero-knowledge (NIZK) proofs, especially for algebraic languages in bilinear groups [26, 30, 31], play an important role in the design of cryptographic primitives and protocols. The non-interactivity, however, comes at a price and in particular (apart from NIZK secure in the random oracle model) demands a trusted setup that generates a so called common reference string (CRS). This CRS is an input to the prover and all potential verifiers. The critical issue is that if this setup is not performed honestly, i.e., the underlying trapdoor is known to some party, then all security is lost.

A long line of research has focused on obtaining very efficient NIZK proofs in this CRS model [23, 27, 28, 30–35, 39], covering efficient pairing-based zero-knowledge Succinct Non-interactive ARguments of Knowledge (zk-SNARKs) for any NP language and succinct Quasi-Adaptive Non-Interactive Zero-Knowledge arguments (QA-NIZKs) for restricted languages, i.e., membership in linear subspaces. QA-NIZKs are a relaxation of NIZK arguments, where the CRS is specialized to the linear space for which membership should be proven [7,8,25,32,33, 35, 37, 38]. This specialized part is called the language parameter. In this paper our focus will be on QA-NIZK arguments.

1.1 Motivation

For the practical application of NIZK primitives in general, a crucial question is how the CRS generation should be performed. While in theory it is simply assumed that some universally trusted party will perform the CRS generation, such a party is challenging to impossible to find in the real world. Consequently, this is typically a too strong assumption.

Now there are different approaches to reduce the required trust that needs to be put in the CRS generation. First, the CRS can be generated by a potentially huge set of parties via the use of secure multi-party computation (MPC), so called ceremonies, [1, 10, 11, 36]. And while this approach has seen use in the real world[3], such ceremonies are cumbersome and require significant effort even beyond the technical realisation. Despite the required efforts, however, it can give very strong guarantees, i.e., if at least as one party behaves honest then security is preserved. Second, to remove this additional effort, one can rely on so called subversion NIZKs [9], subversion zk-SNARKS [2, 20] and subversion QA-NIZKs [3,6]. In this subversion zero-knowledge model, one introduces a way to check the CRS and the prover does not require to trust the CRS, i.e., the zero-knowledge property (so-called subversion zero-knowledge) is still maintained even if the CRS generation is malicious. Unfortunately, the verifier is still required to trust the CRS generation and it is actually impossible to obtain subversion soundness when at the same time requiring zero-knowledge to hold [9]. Third, an interesting

[3] The "powers of tau" ceremony of Zcash: https://z.cash/technology/paramgen/.

middle ground is the recent technique of a so called *updatable CRS* introduced by Groth *et al.* [29], which is an increasingly popular model [5,12,14,18,22,40–43,46]. In this updatable CRS model, everyone can update a CRS along with providing update proofs such that the correctness of updates can be verified by everyone. This guarantees that zero-knowledge for the prover holds in the presence of an adversarial CRS generator. Moreover, the verifier can trust the CRS, i.e., soundness holds, as long as one operation, either the CRS generation itself or one of the updates of the CRS have been performed honestly. Thus, to be certain that soundness holds, a verifier could do a CRS update on its own and then send the updated CRS to the prover.

Initially, Groth *et al.* [29] defined CRS updates with a focus on zk-SNARKs, and then Lipmaa [40] proposed an updatable CRS version of the QA-NIZK construction of [3,35]. While Lipmaa considers so called symmetric QA-NIZK, i.e., where the language is defined in one of the source groups of a bilinear group, it is not known how this applies to asymmetric QA-NIZK [25], i.e., where the language is defined over both source groups (also called bilateral linear subspaces).[4] Asymmetric QA-NIZKs [17, 25, 45], however, are useful for many applications where commitments to the same value are available in both source groups of a bilinear group (e.g., proof aggregation, ring signatures, range proofs). As we will discuss soon, despite not being known how to construct it, having what we call an updatable asymmetric QA-NIZK does have interesting implications for concrete applications discussed below.

Applications. zk-SNARKs and QA-NIZKs are appealing as they are succinct, i.e., they allow proving circuits of arbitrary size and linear subspace languages respectively, with a compact proof. They are also concretely very efficient and in particular in bilinear groups we have constructions with proofs represented by three group elements for zk-SNARKs for arithmetic circuits [28], one group element for symmetric QA-NIZK for linear subspace languages [35], and two group elements for asymmetric QA-NIZK for bilateral linear subspace languages [25]. While (asymmetric) QA-NIZKs have many interesting applications (cf. [25, 32, 33]), our focus will be on their application in the modular design of zk-SNARKs and in particular on LegoSNARK [13].

LegoSNARK is a toolbox for commit-and-prove zk-SNARKs with the aim of constructing a *global* zk-SNARK for some computation C via the linking of *smaller* specialized zk-SNARKs for various subroutines that overall compose to C. The central idea is that by allowing each subroutine of C to be handled by a different proof system, one can select the one that maximizes a metric (e.g., efficiency) that is important for the concrete application. Now LegoSNARK uses succinct QA-NIZKs as efficient zk-SNARKs for linear subspace languages. Abdolmaleki and Slamanig [6] recently showed how one can construct a subversion zero-knowledge variant of symmetric [35] as well as asymmetric QA-NIZK [25] in

[4] To avoid confusion we intentionally do not call them QA-NIZK for *symmetric or asymmetric groups* as done in [25], as both types are instantiated in asymmetric, i.e., type-3, bilinear groups.

a setting where the CRS is subverted but the language parameters are generated honestly. As they mention, the honest language parameters do not represent a problem for practical applications, as they can typically be obtained in a transparent way without trust in their generation (e.g., by deriving them using a random oracle). Furthermore, they show how to integrate a knowledge-sound version of their subversion zero-knowledge symmetric QA-NIZK into LegoSNARK. This represents a step towards a subversion variant of the LegoSNARK toolbox and thus a way to use LegoSNARK with a reduced trust in the required setup.

As most of the recent zk-SNARK constructions focus on the updatable-CRS setting [5,12,14,18,22,41–43,46], it is desirable to enable composable zk-SNARK frameworks such as LegoSNARK also in the updatable CRS setting. If one thereby wants still to take advantage of using QA-NIZK as one of its building blocks, then updatable QA-NIZK are required. While, as mentioned above, there are numerous constructions of zk-SNARKs with an updatable CRS, to date there is only an updatable symmetric QA-NIZK by Lipmaa [40] available. To prove the zero-knowledge property, it requires a new and non-standard knowledge assumption (KW-KE). Adaptive soundness can be shown under a standard assumption, but achieving knowledge soundness, a property that would be required for composable zk-SNARKs, is left as an open problem in [40]. Lipmaa works in a model where the complete CRS including the language parameter (what he calls key) can be generated maliciously. Additionally proofs (what he calls arguments) under previous versions of the CRS can be updated to newer versions of the CRS. While latter extends potential applications, in the context of composable zk-SNARK frameworks, this feature is not required.

Now, apart from the missing knowledge-soundness property in [40], it could be tempting to think that two parallel symmetric QA-NIZK can be used to emulate what is provided by asymmetric QA-NIZK. However, the problem is that one would require an additional "linking proof" that would guarantee that both proofs use the same witness. And exactly this issue, which would increase the proof size and decrease efficiency, is what one can avoid when using asymmetric QA-NIZK in LegoSNARK, whenever the respective commitments are available in both source groups.

Consequently, when having updatable asymmetric QA-NIZKs, which avoid the aforementioned issue, this is another step towards an updatable variant of the LegoSNARK toolbox.

1.2 Our Results

We investigate the most efficient asymmetric QA-NIZK (denoted as Π'_{asy}) by González *et al.* (GHR) [25] in an updatable CRS setting. We show that for Π'_{asy} we can construct updatable asymmetric QA-NIZK arguments (which requires a witness samplable distribution [32]) by extending the CRS suitably and adding two new algorithms for updating the CRS and verify CRS updates. Compared to the recent updatable symmetric QA-NIZK in [40], we consider a variant where the CRS is subverted and can be updated, but the language parameter is chosen honestly. As already mentioned above that latter does not represent a problem

for practical applications and in particular composable zk-SNARK frameworks such as LegoSNARK [13].

In contrast to the updatable symmetric QA-NIZK in [40], which relies on a new non-standard knowledge assumption for their subversion zero-knowledge property (KW-KE), our construction of updatable QA-NIZK can be shown to have this property under the Bilinear Diffie-Hellman Knowledge of Exponents (BDH-KE) assumption [2, 4] and is asymmetric. Furthermore, under the discrete logarithm assumption in the Algebraic Group Model (AGM) due to Fuchsbauer *et al.* [21], we prove the knowledge soundness property of the proposed updatable asymmetric QA-NIZK. We also show that this also yields the knowledge soundness property of the original GHR asymmetric QA-NIZK.

Technical overview. In Section 3, we give constructions of succinct updatable *asymmetric* QA-NIZK arguments of membership in bilateral linear subspaces. Using implicit notation, we represent the elements of $\mathbb{G}_1$ (respectively of $\mathbb{G}_2$) as $[z]_1 \in \mathbb{G}_1$ (respectively, as $[z]_2 \in \mathbb{G}_2$). Given the language parameters $[\boldsymbol{M}]_1 \in \mathbb{G}_1^{n_1 \times m}$ and $[\boldsymbol{N}]_2 \in \mathbb{G}_2^{n_2 \times m}$, we consider QA-NIZK arguments of membership of the statements $([\boldsymbol{y}]_1, [\boldsymbol{x}]_2)$ in the language

$$\mathcal{L}_{[\boldsymbol{M}]_1,[\boldsymbol{N}]_2} = \left\{ ([\boldsymbol{y}]_1, [\boldsymbol{x}]_2) \in \mathbb{G}_1^{n_1} \times \mathbb{G}_2^{n_2} : \exists \boldsymbol{w} \in \mathbb{Z}_p^m \text{ s.t. } \boldsymbol{y} = \boldsymbol{M}\boldsymbol{w}, \boldsymbol{x} = \boldsymbol{N}\boldsymbol{w} \right\}.$$

As mentioned, to construct our updatable *asymmetric* QA-NIZK arguments we start from the *asymmetric* QA-NIZK by González *et al.* (GHR) [25] (cf. Fig. 1) and change GHR's QA-NIZK by adding extra elements to the CRS so that the CRS becomes publicly verifiable and trapdoor extractable. Importantly, our aim for the updatable asymmetric QA-NIZK, is to keep the prover and the verifier unchanged compared to GHR's QA-NIZK.

More precisely, the CRS of GHR's QA-NIZK contains $\mathsf{crs} = ([\boldsymbol{A}, \boldsymbol{C}_2, \boldsymbol{P}_2]_2, [\boldsymbol{A}, \boldsymbol{C}_1, \boldsymbol{P}_1]_1)$ where $[\boldsymbol{A}]_i \in \mathbb{G}_i^{k \times k}$, $\boldsymbol{C}_i \in \mathbb{Z}_p^{n_i \times k}$, and $\boldsymbol{P}_i \in \mathbb{Z}_p^{m \times k}$ for $i \in \{1, 2\}$ and integers n_i, m and k. The prover uses $[\boldsymbol{P}_2]_2$ and $[\boldsymbol{P}_1]_1$ to generate a proof and the verifier uses the rest of the CRS to verify the proof. We add two new elements $[\boldsymbol{C}_1]_2$ and $[\boldsymbol{C}_2]_1$ to the CRS of the GHR scheme to make the CRS publicly verifiable. The trapdoor extractability is guaranteed using the new elements $[\boldsymbol{C}_1]_2$ and $[\boldsymbol{C}_2]_1$ and under the Bilinear Diffie-Hellman Knowledge of Exponents assumption (the extracted trapdoor will be used to prove subversion zero-knowledge). To achieve the updatability property, we design two new algorithms Ucrs and Vcrs. The Ucrs algorithm takes the crs and updates it to a new $\mathsf{crs}_{\mathsf{up}}$ so that the update is publicly verifiable. More precisely, given the $\mathsf{crs}_{\mathsf{up}}$, the language parameters $[\boldsymbol{M}]_1$, and $[\boldsymbol{N}]_2$, the Vcrs algorithm checks the well-formedness of $\mathsf{crs}_{\mathsf{up}}$. The latter checking guarantees the existence of a trapdoor tc for the $\mathsf{crs}_{\mathsf{up}}$, which will be required to prove the zero-knowledge property (cf. Section 3.2).

This step is necessary and will be sufficient for subversion zero-knowledge (as the prover can check the well-formedness of the CRS) and updatable soundness (as the verifier can check and update the CRS) in the updatable setting. However, choosing which elements to add to the CRS is not straightfor-

ward since the QA-NIZK must remain secure even given this extended CRS as adding too much information into the CRS can easily break the security, i.e., zero-knowledge and/or soundness. For instance, one may achieve the aforementioned properties by adding $[\boldsymbol{P}_1]_2$ and $[\boldsymbol{P}_2]_1$ to the CRS of GHR's QA-NIZK. But adding such elements bring a fundamental issue that under the Bilinear Diffie-Hellman Knowledge of Exponents assumption, the simulator in the zero-knowledge proof can also extract the language parameters $\boldsymbol{M}$ and $\boldsymbol{N}$. Given statements $([\boldsymbol{y}]_1, [\boldsymbol{x}]_2)$, the simulator obtains more information of the witness $\boldsymbol{w}$ of the language $\mathcal{L}_{[\boldsymbol{M}]_1,[\boldsymbol{N}]_2}$, which would violate the zero-knowledge property.

2 Preliminaries

Let PPT denote probabilistic polynomial-time. Let $\lambda \in \mathbb{N}$ be the security parameter. By $x \leftarrow\!\!\$\ \mathcal{D}$ we denote that x is sampled according to distribution $\mathcal{D}$ or uniformly randomly if $\mathcal{D}$ is a set. We denote by $\mathsf{negl}(\lambda)$ an arbitrary negligible function. We write $a \approx_\lambda b$ if $|a-b| \leq \mathsf{negl}(\lambda)$. Algorithm $\mathsf{Pgen}(1^\lambda)$ returns $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$, where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are three additive cyclic groups of prime order p, and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate efficiently computable bilinear map (pairing). We use the implicit bracket notation of [19], that is, we write $[a]_\iota$ to denote ag_ι where g_ι is a fixed generator of $\mathbb{G}_\iota$. We denote $\hat{e}([a]_1, [b]_2)$ as $[a]_1[b]_2$. Thus, $[a]_1[b]_2 = [ab]_T$. By $y \leftarrow \mathcal{A}(x;\omega)$ we denote the fact that $\mathcal{A}$, given an input x and random coins ω, outputs y. Let $\mathsf{RND}(\mathcal{A})$ denote the random tape of $\mathcal{A}$, and let $\omega \leftarrow\!\!\$\ \mathsf{RND}(\mathcal{A})$ denote the random choice of the random coins ω from $\mathsf{RND}(\mathcal{A})$.

Computational Assumptions. We require the following assumptions.

Definition 1 (BDH-KE **Assumption [2, 4]).** *We say that* BDH-KE *holds relative to* K_0*, if for any PPT adversary* $\mathcal{A}$ *there exists a PPT extractor* $\mathsf{Ext}_{\mathcal{A}}^{\mathsf{BDH\text{-}KE}}$*, such that*

$$\Pr\left[\begin{array}{l}\mathsf{p} \leftarrow\!\!\$\ \mathsf{K}_0(1^\lambda); \omega_{\mathcal{A}} \leftarrow\!\!\$\ \mathsf{RND}(\mathcal{A}),\\ ([\alpha_1]_1, [\alpha_2]_2 || a) \leftarrow (\mathcal{A} || \mathsf{Ext}_{\mathcal{A}}^{\mathsf{BDH\text{-}KE}})(\mathsf{p}, \omega_{\mathcal{A}})\end{array} : [\alpha_1]_1[1]_2 = [1]_1[\alpha_2]_2 \wedge a \neq \alpha_1\right] \approx_\lambda 0\,.$$

Here $\mathsf{aux}_{\mathcal{R}}$ is the auxiliary information related to the relation generator of $\mathcal{R}$. Note that the BDH-KE assumption can be considered as a simple case of the PKE assumption of [16]. Also, BDH-KE can be seen as an asymmetric-pairing version of the original KoE assumption [15].

In the following definitions let $\mathcal{D}_k$ be a matrix distribution in $\mathbb{Z}_p^{(k+1)\times k}$.

Definition 2 ($\mathcal{D}_k$**-Matrix Diffie-Hellman (**$\mathcal{D}_k$-MDDH**) Assumption [44]).** *The* $\mathcal{D}_k$-MDDH *assumption for* $\iota \in \{1,2\}$ *holds relative to* K_0*, if for any PPT adversary* $\mathcal{A}$*,* $|\mathsf{Exp}_{\mathcal{A}}^{\mathsf{MDDH}}(\mathsf{p}) - 1/2| \approx_\lambda 0$*, where* $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{MDDH}}(\mathsf{p}) :=$

$$\Pr\left[\begin{array}{ll}\mathsf{p} \leftarrow\!\!\$\ \mathsf{K}_0(1^\lambda); \boldsymbol{A} \leftarrow\!\!\$\ \mathcal{D}_k; \boldsymbol{v} \leftarrow\!\!\$\ \mathbb{Z}_p^k; & \\ \boldsymbol{u} \leftarrow\!\!\$\ \mathbb{Z}_p^{k+1}; b \leftarrow\!\!\$\ \{0,1\}; & : b = b^*\\ b^* \leftarrow \mathcal{A}(\mathsf{p}, [\boldsymbol{A}]_\iota, [b \cdot \boldsymbol{A}\boldsymbol{v} + (1-b)\cdot \boldsymbol{u}]_\iota) & \end{array}\right].$$

Definition 3 ($\mathcal{D}_k$-SKerMDH **Assumption [25]**). *The* $\mathcal{D}_k$-SKerMDH *assumption holds relative to* K_0*, if for any PPT* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{l}\mathsf{p} \leftarrow \mathsf{K}_0(1^\lambda); \boldsymbol{A} \leftarrow\!\$\, \mathcal{D}_k; ([\boldsymbol{s}_1]_1, [\boldsymbol{s}_2]_2) \leftarrow \mathcal{A}(\mathsf{p}, [\boldsymbol{A}]_1, [\boldsymbol{A}]_2) : \\ \boldsymbol{s}_1 - \boldsymbol{s}_2 \neq \mathbf{0} \wedge \boldsymbol{A}^\top(\boldsymbol{s}_1 - \boldsymbol{s}_2) = \mathbf{0}_k\end{array}\right] \approx_\lambda 0\,.$$

Let $\mathcal{D}_{\ell k}$ be a probability distribution over matrices in $\mathbb{Z}_p^{\ell \times k}$, where $\ell > k$. Next, we define five commonly used distributions (see [19] for references), where $a, a_i, a_{ij} \leftarrow\!\$\, \mathbb{Z}_p^*$: $\mathcal{U}_k$ (uniform), $\mathcal{L}_k$ (linear), $\mathcal{IL}_k$ (incremental linear), $\mathcal{C}_k$ (cascade), $\mathcal{SC}_k$ (symmetric cascade):

$$\mathcal{U}_k\colon \boldsymbol{A} = \begin{pmatrix} a_{11} & \dots & a_{1k} \\ \dots & \dots & \dots \\ a_{k1} & \dots & a_{kk} \\ a_{k+1,1} & \dots & a_{k+1,k} \end{pmatrix}, \quad \mathcal{L}_k\colon \boldsymbol{A} = \begin{pmatrix} a_1 & 0 & \dots & 0 & 0 \\ 0 & a_2 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & a_k \\ 1 & 1 & \dots & 1 & 1 \end{pmatrix},$$

$$\mathcal{IL}_k\colon \boldsymbol{A} = \begin{pmatrix} a & 0 & \dots & 0 & 0 \\ 0 & a{+}1 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & a{+}k{-}1 \\ 1 & 1 & \dots & 1 & 1 \end{pmatrix}, \quad \mathcal{C}_k\colon \boldsymbol{A} = \begin{pmatrix} a_1 & 0 & \dots & 0 & 0 \\ 1 & a_2 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & a_k \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix},$$

$$\mathcal{SC}_k\colon \boldsymbol{A} = \begin{pmatrix} a & 0 & \dots & 0 & 0 \\ 1 & a & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & a \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix}.$$

Assume that $\mathcal{D}_{\ell k}$ outputs matrices $\boldsymbol{A}$ where the upper $k \times k$ submatrix $\bar{\boldsymbol{A}}$ is always invertible, i.e., $\mathcal{D}_{\ell k}$ is robust [32].

QA-NIZK Arguments. Let a language $\mathcal{L}_\varrho$ defined by a relation $\mathcal{R}_\varrho$ which is parametrized by some parameter ϱ, called the language parameter, chosen from a distribution $\mathcal{D}_\mathsf{p}$. We recall the definition of QA-NIZK arguments from Jutla and Roy [32]. A QA-NIZK argument provides a proof for membership of words x with according witnesses $\boldsymbol{w}$ in the language $\mathcal{L}_\varrho$. The distribution $\mathcal{D}_\mathsf{p}$ is witness samplable if there exist an efficient algorithm that samples $(\varrho, \mathtt{tc}_\varrho)$ so that the parameter ϱ is distributed according to $\mathcal{D}_\mathsf{p}$ and membership of the language parameter ϱ can be efficiently verified with $\mathtt{tc}_\varrho$. The CRS of a QA-NIZK depends on a language parameter ϱ and as mentioned in [32], it has to be chosen from a correct distribution $\mathcal{D}_\mathsf{p}$.

Let ϱ be sampled from a distribution $\mathcal{D}_\mathsf{p}$ over associated parameter language $\mathcal{L}_\mathsf{p}$. A QA-NIZK argument in the CRS model contains four PPT algorithms $\mathit{\Pi} = (\mathsf{Pgen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ for a set of witness-relations $\mathcal{R}_\mathsf{p} = \{\mathcal{R}_\varrho\}_{\varrho \in \mathrm{Supp}(\mathcal{D}_\mathsf{p})}$, if the following properties (i-iii) hold. We call the QA-NIZK knowledge sound if instead of (iii) the property (iv) holds. Here, Pgen is the parameter and the CRS generation algorithm, more precisely, Pgen consists of two algorithms K_0 (generates the parameter p) and K (generates the CRS), P is the prover, V is the verifier, and Sim is the simulator.

(i) **Completeness.** For any λ, and $(x, \boldsymbol{w}) \in \mathcal{R}_\varrho$,

$$\Pr\left[\begin{array}{l}\mathsf{p} \leftarrow \mathsf{K}_0(1^\lambda); \varrho \leftarrow\!\$\, \mathcal{D}_\mathsf{p}; (\mathsf{crs}, \mathtt{tc}) \leftarrow \mathsf{K}(\varrho); \pi \leftarrow \mathsf{P}(\varrho, \mathsf{crs}, x, \boldsymbol{w}) : \\ \mathsf{V}(\varrho, \mathsf{crs}, x, \pi) = 1\end{array}\right] = 1\,.$$

$\mathsf{K}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2)$

- $\boldsymbol{A} \leftarrow\!\$\ \hat{\mathcal{D}}_k$; $\boldsymbol{K}_1 \leftarrow\!\$\ \mathbb{Z}_p^{n_1 \times \hat{k}}$; $\boldsymbol{K}_2 \leftarrow\!\$\ \mathbb{Z}_p^{n_2 \times \hat{k}}$; $\boldsymbol{Z} \leftarrow\!\$\ \mathbb{Z}_p^{m \times \hat{k}}$; $\boldsymbol{C}_1 \leftarrow \boldsymbol{K}_1\boldsymbol{A} \in \mathbb{Z}_p^{n_1 \times k}$;
- $\boldsymbol{C}_2 \leftarrow \boldsymbol{K}_2\boldsymbol{A} \in \mathbb{Z}_p^{n_2 \times k}$; $[\boldsymbol{P}_1]_1 \leftarrow [\boldsymbol{M}]_1^\top \boldsymbol{K}_2 + [\boldsymbol{Z}]_1 \in \mathbb{Z}_p^{m \times \hat{k}}$;
- $[\boldsymbol{P}_2]_2 \leftarrow [\boldsymbol{N}]_2^\top \boldsymbol{K}_1 + [\boldsymbol{Z}]_2 \in \mathbb{Z}_p^{m \times \hat{k}}$;
- $\mathsf{tc} \leftarrow (\boldsymbol{K}_1, \boldsymbol{K}_2)$; $\mathsf{crs} \leftarrow ([\boldsymbol{A}, \boldsymbol{C}_2, \boldsymbol{P}_2]_2, [\boldsymbol{A}, \boldsymbol{C}_1, \boldsymbol{P}_1]_1)$;
- **return** $(\mathsf{tc}, \mathsf{crs})$;

$\mathsf{P}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2, \mathsf{crs}, [\boldsymbol{y}]_1, [\boldsymbol{x}]_2, \boldsymbol{w})$:

- $\boldsymbol{r} \leftarrow\!\$\ \mathbb{Z}_p^{\hat{k}}$;
- $[\boldsymbol{\pi}_1]_1 \leftarrow [\boldsymbol{P}_1]_1^\top \boldsymbol{w} + [\boldsymbol{r}]_1 \in \mathbb{G}_1^{\hat{k}}$;
- $[\boldsymbol{\pi}_2]_2 \leftarrow [\boldsymbol{P}_2]_2^\top \boldsymbol{w} + [\boldsymbol{r}]_2 \in \mathbb{G}_2^{\hat{k}}$;
- **return** $([\boldsymbol{\pi}_1]_1, [\boldsymbol{\pi}]_2)$;

$\mathsf{V}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2, \mathsf{crs}, [\boldsymbol{y}]_1, [\boldsymbol{x}]_2, [\boldsymbol{\pi}_1]_1, [\boldsymbol{\pi}_2]_2)$:

- **if** $[\boldsymbol{y}]_1^\top [\boldsymbol{C}_2]_2 - [\boldsymbol{\pi}_1]_1^\top [\boldsymbol{A}]_2 = [\boldsymbol{\pi}_2]_2^\top [\boldsymbol{A}]_1 - [\boldsymbol{x}]_2^\top [\boldsymbol{C}_1]_1$ **return** 1;
- **else** **return** 0;

$\mathsf{Sim}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2, \mathsf{crs}, \mathsf{tc}, [\boldsymbol{y}]_1)$:

- $\boldsymbol{r} \leftarrow\!\$\ \mathbb{Z}_p^{\hat{k}}$; - $[\boldsymbol{\pi}_1]_1 \leftarrow \boldsymbol{K}_2^\top [\boldsymbol{y}]_1 + [\boldsymbol{r}]_1 \in \mathbb{G}_1^{\hat{k}}$; - $[\boldsymbol{\pi}]_2 \leftarrow \boldsymbol{K}_1^\top [\boldsymbol{x}]_2 + [\boldsymbol{r}]_2 \in \mathbb{G}_1^{\hat{k}}$;
- **return** $([\boldsymbol{\pi}_1]_1, [\boldsymbol{\pi}]_2)$;

Fig. 1: Asymmetric QA-NIZK Π_{asy} ($\hat{\mathcal{D}}_k = \mathcal{D}_k$ and $\hat{k} = k+1$) and Π'_{asy} ($\hat{\mathcal{D}}_k = \bar{\mathcal{D}}_k$ and $\hat{k} = k$) from [25].

(ii) **Statistical Zero-Knowledge.** For any computationally unbounded adversary $\mathcal{A}$, $|\varepsilon_0^{zk} - \varepsilon_1^{zk}| \approx_\lambda 0$, where $\varepsilon_b^{zk} :=$

$$\Pr\left[\mathsf{p} \leftarrow \mathsf{K}_0(1^\lambda); \varrho \leftarrow\!\$\ \mathcal{D}_{\mathsf{p}}; (\mathsf{crs}, \mathsf{tc}) \leftarrow \mathsf{K}(\varrho); b \leftarrow\!\$\ \{0,1\} : \mathcal{A}^{\mathsf{O}_b(\cdot)}(\varrho, \mathsf{crs}) = 1\ \right].$$

The oracle $\mathsf{O}_0(x, \boldsymbol{w})$ returns $\perp$ (reject) if $(x, \boldsymbol{w}) \notin \mathcal{R}_\varrho$, and otherwise it returns $\mathsf{P}(\varrho, \mathsf{crs}, x, \boldsymbol{w})$. Similarly, $\mathsf{O}_1(x, \boldsymbol{w})$ returns $\perp$ (reject) if $(x, \boldsymbol{w}) \notin \mathcal{R}_\varrho$, and otherwise it returns $\mathsf{Sim}(\varrho, \mathsf{crs}, \mathsf{tc}, x)$.

(iii) **Adaptive Soundness.** For any PPT $\mathcal{A}$,

$$\Pr\left[\begin{array}{l}\mathsf{p} \leftarrow \mathsf{K}_0(1^\lambda); \varrho \leftarrow\!\$\ \mathcal{D}_{\mathsf{p}}; (\mathsf{crs}, \mathsf{tc}) \leftarrow \mathsf{K}(\varrho); (x, \pi) \leftarrow \mathcal{A}(\varrho, \mathsf{crs}) : \\ \mathsf{V}(\varrho, \mathsf{crs}, x, \pi) = 1 \wedge \neg(\exists \boldsymbol{w} : (x, \boldsymbol{w}) \in \mathcal{R}_\varrho)\end{array}\right] \approx_\lambda 0\ .$$

(vi) Adaptive Knowledge Soundness. For any PPT $\mathcal{A}$ there exists a non-uniform PPT extractor $\mathsf{Ext}_{\mathcal{A}}$,

$$\Pr\left[\begin{array}{l}\mathsf{p} \leftarrow \mathsf{K}_0(1^\lambda); \varrho \leftarrow\!\$\ \mathcal{D}_{\mathsf{p}}; (\mathsf{crs}, \mathsf{tc}) \leftarrow \mathsf{K}(\varrho); \omega_{\mathcal{A}} \leftarrow\!\$\ \mathsf{RND}(\mathcal{A}); \\ (x, \pi) \leftarrow \mathcal{A}(\omega_{\mathcal{A}}; \varrho, \mathsf{crs}); \boldsymbol{w} \leftarrow \mathsf{Ext}_{\mathcal{A}}(\omega_{\mathcal{A}}; \varrho, \mathsf{crs}) : (x, \boldsymbol{w}) \notin \mathcal{R}_\varrho \\ \wedge\, \mathsf{V}(\varrho, \mathsf{crs}, x, \pi) = 1\end{array}\right] \approx_\lambda 0\ .$$

Asymmetric QA-NIZK for Concatenation Languages. We recall the asymmetric QA-NIZK arguments of membership in bilateral linear subspaces of $\mathbb{G}_1^{n_1} \times$

$\mathbb{G}_2^{n_2}$ given by González *et al.* [25] for the language

$$\mathcal{L}_{[\boldsymbol{M}]_1,[\boldsymbol{N}]_2} = \left\{ ([\boldsymbol{y}]_1, [\boldsymbol{x}]_2) \in \mathbb{G}_1^{n_1} \times \mathbb{G}_2^{n_2} : \exists \boldsymbol{w} \in \mathbb{Z}_p^m \text{ s.t. } \boldsymbol{y} = \boldsymbol{M}\boldsymbol{w}, \boldsymbol{x} = \boldsymbol{N}\boldsymbol{w} \right\} .$$

This language is also known as the concatenation language, since one can define $\boldsymbol{R}$ as a concatenation of language parameters $[\boldsymbol{M}]_1$ and $[\boldsymbol{N}]_2$ so that $\boldsymbol{R} = \begin{pmatrix} [\boldsymbol{M}]_1 \\ [\boldsymbol{N}]_2 \end{pmatrix}$. In other words $([\boldsymbol{y}]_1, [\boldsymbol{x}]_2) \in \mathcal{L}_{[\boldsymbol{M}]_1,[\boldsymbol{N}]_2}$ iff $\begin{pmatrix} [\boldsymbol{y}]_1 \\ [\boldsymbol{x}]_2 \end{pmatrix}$ is in the span of $\boldsymbol{R}$. We recall the full construction of asymmetric QA-NIZK arguments in the CRS model in Fig. 1.

Notice that the QA-NIZK in Fig. 1 for $\mathcal{L}_{[\boldsymbol{M}]_1,[\boldsymbol{N}]_2}$ is a generalization of Π_{as} of [35] in two groups when we set $\hat{\mathcal{D}}_k = \mathcal{D}_k$ and $\hat{k} = k + 1$ (denoted as Π_{asy}). Also it is a generalization of Π'_{as} of [35] in two groups when we set $\hat{\mathcal{D}}_k = \bar{\mathcal{D}}_k$ and $\hat{k} = k$ (denoted as Π'_{asy}).

Theorem 1. *[Theorem 3 of [25]] If $\hat{\mathcal{D}}_k = \mathcal{D}_k$ and $\hat{k} = k+1$, the QA-NIZK proof system in Fig. 1 is perfect complete, computational adaptive soundness based on the $\mathcal{D}_k$-*SKerMDH *assumption, perfect zero-knowledge.*

Theorem 2. *[Theorem 4 of [25]] If $\hat{\mathcal{D}}_k = \bar{\mathcal{D}}_k$, $\hat{k} = k$ and $\mathcal{D}_{\mathsf{p}}$ is a witness samplable distribution, Fig. 1 describes a QA-NIZK proof system with perfect completeness, computational adaptive soundness based on the $\mathcal{D}_k$-*KerMDH *assumption, perfect zero-knowledge.*

3 Updatable Asymmetric QA-NIZK

In this section, we investigate asymmetric QA-NIZK arguments when the CRS can be maliciously generated and propose corresponding updatable asymmetric QA-NIZK arguments. Formally, we prove the following theorem:

Theorem 3. *Let $\Pi_{\mathsf{asy\text{-}up}}$ be an updatable asymmetric QA-NIZK argument for linear subspaces from Fig. 4. (i) $\Pi_{\mathsf{asy\text{-}up}}$ is* crs*-update correct,* crs*-update hiding, and complete, (ii) if the* BDH-KE *assumption hold, then $\Pi_{\mathsf{asy\text{-}up}}$ is statistically subversion zero-knowledge, and (iii) if the $\mathcal{D}_k$-*SKerMDH*, (for the case $\hat{\mathcal{D}}_k = \bar{\mathcal{D}}_k$, the distribution $\mathcal{D}_{\mathsf{p}}$ should be witness samplable) then $\Pi_{\mathsf{asy\text{-}up}}$ is computationally updatable sound.*

First, we discuss subversion security of QA-NIZKs in the updatable CRS setting, then propose an updatable version of the most efficient asymetric QA-NIZK construction Π'_{as} in [25] (cf. Fig. 1).

3.1 Security Definitions for Updatable QA-NIZK Arguments

As already mentioned, the notion of *updatability* to achieve subversion security for NIZKs in the CRS model with respect to zero-knowledge and soundness was introduced by Groth *et al.* in [29] with a focus on zk-SNARKs. Later, Lipma [40]

applied the underlying ideas to the setting of QA-NIZKs and in particular when both the language parameter ϱ and the CRS can be subverted. More precisely, Lipmaa obtains a version of the Kiltz-Wee QA-NIZK [35] (in the bare public-key (BPK) model) in the aforementioned setting under a new non-falsifiable $\mathsf{KW\text{-}KE}$ knowledge assumption. In this work, motivated by [6] and their application to composable zk-SNARK frameworks such as LegoSNARK [13], we investigate the security of QA-NIZKs in the CRS model when the CRS is subverted and can be updated but with *honestly chosen* ϱ[5]. Our security definition thus then enables us to construct an updatable asymmetric QA-NIZK that can be used to extend the LegoSANRK [13] with updatable CRS. More precisely such schemes can be used as the updatable zk-SNARKs for bilateral subspace languages as they provide better efficiency than general updatable zk-SNARKs for these types of languages.

Concretely, we define updatable QA-NIZKs security with some changes in the updatable CRS model. A tuple of PPT algorithms $\Pi = (\mathsf{Pgen}, \mathsf{Ucrs}, \mathsf{Vcrs}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ is an updatable QA-NIZK if properties (i-v) hold. We call an updatable QA-NIZK updatable knowledge sound if instead of (v) property (vi) holds. Here, $\mathsf{Ucrs}(\varrho, \mathsf{crs})$ is an algorithm to update the CRS that takes the language parameter ϱ and a CRS crs and outputs an updated CRS $\mathsf{crs}_{\mathsf{up}}$ and corresponding trapdoor $\mathtt{tc}_{\mathsf{up}}$. $\mathsf{Vcrs}(\varrho, \mathsf{crs}, \mathsf{crs}_{\mathsf{up}})$ is an algorithm to verify the correctness of a CRS update and takes an old crs to a new CRS $\mathsf{crs}_{\mathsf{up}}$ and checks the well-formedness of the updated CRS.

(i) CRS-update Correctness. For any λ,

$$\Pr\left[\begin{array}{l}\mathsf{p} \leftarrow \mathsf{K}_0(1^\lambda); \varrho \leftarrow\!\!\$\ \mathcal{D}_{\mathsf{p}}; (\mathsf{crs}, \mathtt{tc}) \leftarrow \mathsf{K}(\varrho); \\ (\mathsf{crs}_{\mathsf{up}}, \mathtt{tc}_{\mathsf{up}}) \leftarrow \mathsf{Ucrs}(\varrho, \mathsf{crs}) : \ \mathsf{Vcrs}(\varrho, \mathsf{crs}, \mathsf{crs}_{\mathsf{up}}) = 1\end{array}\right] = 1 \ .$$

(ii) CRS-update Hiding. For any λ,

$$\Pr\left[\begin{array}{l}\mathsf{p} \leftarrow \mathsf{K}_0(1^\lambda); \varrho \leftarrow\!\!\$\ \mathcal{D}_{\mathsf{p}}; (\mathsf{crs}, \mathtt{tc}) \leftarrow \mathsf{K}(\varrho); (\mathsf{crs}_{\mathsf{up}}, \mathtt{tc}_{\mathsf{up}}) \leftarrow \mathsf{Ucrs}(\varrho, \mathsf{crs}) \\ \mathsf{Vcrs}(\varrho, \mathsf{crs}, \mathsf{crs}_{\mathsf{up}}) = 1 : \mathsf{crs}_{\mathsf{up}} \approx_\lambda \mathsf{crs}\end{array}\right] = 1 \ .$$

Note that this property holds the initial crs is maliciously generated and an honest updater Ucrs updates it.

(iii) Completeness. For any λ, and $(x, \boldsymbol{w}) \in \mathcal{R}_\varrho$,

$$\Pr\left[\begin{array}{l}\mathsf{p} \leftarrow \mathsf{K}_0(1^\lambda); \varrho \leftarrow\!\!\$\ \mathcal{D}_{\mathsf{p}}; (\mathsf{crs}, \mathtt{tc}) \leftarrow \mathsf{K}(\varrho); (\mathsf{crs}_{\mathsf{up}}, \mathtt{tc}_{\mathsf{up}}) \leftarrow \mathsf{Ucrs}(\varrho, \mathsf{crs}); \\ \pi \leftarrow \mathsf{P}(\varrho, \mathsf{crs}_{\mathsf{up}}, x, \boldsymbol{w}) : \mathsf{Vcrs}(\varrho, \mathsf{crs}, \mathsf{crs}_{\mathsf{up}}) = 1 \ \wedge \ \mathsf{V}(\varrho, \mathsf{crs}_{\mathsf{up}}, x, \pi) = 1\end{array}\right] = 1 \ .$$

(iv) Statistical Subversion Zero-Knowledge. For any PPT subverter Z

[5] We recall that in such applications ϱ represents public keys of the commitment scheme and can typically derived in a way (e.g., via a random oracle) such that subversion is not possible.

there exists a PPT extractor $\mathsf{Ext}_{\mathsf{Z}}$, such that for any computationally unbounded adversary $\mathcal{A}$, $|\varepsilon_0^{zk} - \varepsilon_1^{zk}| \approx_\lambda 0$, where $\varepsilon_b^{zk} :=$

$$\Pr\left[\begin{array}{l}\mathsf{p} \leftarrow \mathsf{K}_0(1^\lambda); \varrho \leftarrow\!\!\$\ \mathcal{D}_{\mathsf{p}}; \omega_{\mathsf{Z}} \leftarrow\!\!\$\ \mathsf{RND}(\mathsf{Z}); (\mathsf{crs}, \mathsf{aux}_{\mathsf{Z}}) \leftarrow \mathsf{Z}(\varrho; \omega_{\mathsf{Z}}); \\ \mathsf{tc} \leftarrow \mathsf{Ext}_{\mathsf{Z}}(\varrho; \omega_{\mathsf{Z}}); b \leftarrow\!\!\$\ \{0,1\} : \mathsf{Vcrs}(\varrho, \mathsf{crs}) = 1 \wedge \mathcal{A}^{\mathsf{O}_b(\cdot,\cdot)}(\varrho, \mathsf{crs}, \mathsf{aux}_{\mathsf{Z}}) = 1\end{array}\right].$$

The oracle $\mathsf{O}_0(x, \boldsymbol{w})$ returns $\perp$ (reject) if $(x, \boldsymbol{w}) \notin \mathcal{R}_\varrho$, and otherwise it returns $\mathsf{P}(\varrho, \mathsf{crs}, x, \boldsymbol{w})$. Similarly, $\mathsf{O}_1(x, \boldsymbol{w})$ returns $\perp$ (reject) if $(x, \boldsymbol{w}) \notin \mathcal{R}_\varrho$, and otherwise it returns $\mathsf{Sim}(\varrho, \mathsf{crs}, \mathsf{tc}, x)$.

(v) Updatable Adaptive Soundness. For any PPT $\mathcal{A}$,

$$\Pr\left[\begin{array}{l}\mathsf{p} \leftarrow \mathsf{K}_0(1^\lambda); \varrho \leftarrow\!\!\$\ \mathcal{D}_{\mathsf{p}}; (\mathsf{crs}, \mathsf{tc}) \leftarrow \mathsf{K}(\varrho); (x, \pi, \mathsf{crs}') \leftarrow \mathcal{A}(\varrho, \mathsf{crs}) \\ : (x, \boldsymbol{w}) \notin \mathcal{R}_\varrho \wedge \mathsf{Vcrs}(\varrho, \mathsf{crs}, \mathsf{crs}') = 1 \wedge \mathsf{V}(\varrho, \mathsf{crs}', x, \pi) = 1\end{array}\right] \approx_\lambda 0 \ .$$

(vi) Updatable Adaptive Knowledge Soundness. For any PPT $\mathcal{A}$ there exists a non-uniform PPT extractor $\mathsf{Ext}_{\mathcal{A}}$,

$$\Pr\left[\begin{array}{l}\mathsf{p} \leftarrow \mathsf{K}_0(1^\lambda); \varrho \leftarrow\!\!\$\ \mathcal{D}_{\mathsf{p}}; (\mathsf{crs}, \mathsf{tc}) \leftarrow \mathsf{K}(\varrho); \omega_{\mathcal{A}} \leftarrow\!\!\$\ \mathsf{RND}(\mathcal{A}); \\ (x, \pi, \mathsf{crs}') \leftarrow \mathcal{A}(\omega_{\mathcal{A}}; \varrho, \mathsf{crs}); \boldsymbol{w} \leftarrow \mathsf{Ext}_{\mathcal{A}}(\omega_{\mathcal{A}}; \varrho, \mathsf{crs}, \mathsf{crs}') : (x, \boldsymbol{w}) \notin \mathcal{R}_\varrho \\ \wedge\ \mathsf{Vcrs}(\varrho, \mathsf{crs}, \mathsf{crs}') = 1 \wedge \mathsf{V}(\varrho, \mathsf{crs}', x, \pi) = 1\end{array}\right] \approx_\lambda 0 \ .$$

3.2 Construction of Updatable Asymmetric QA-NIZKs

In this section we describe our updatable QA-NIZK for bilateral subspace languages. We first recall some notation and the primitives used in the construction.

Ingredients and notation. Our updatable *asymmetric* QA-NIZK uses the following assumption and primitives:

- Asymmetric QA-NIZK in the CRS model, i.e., the asymmetric QA-NIZK Π'_{asy} of [25] (cf. Theorem 2).
- The knowledge assumption BDH-KE [2, 4]. (cf. Definition 1).
- The algorithm $\mathsf{MATV}([\boldsymbol{A}]_2)$ of [3] that checks if a matrix $[\boldsymbol{A}]_2$ from $\mathcal{D}_k \in \{\mathcal{L}_k, \mathcal{IL}_k, \mathcal{C}_k, \mathcal{SC}_k\}$ is efficiently verifiable (cf. Fig. 2).
- The algorithm $\mathsf{isinvertible}([\boldsymbol{A}]_2, [\boldsymbol{A}]_1)$ of [3] that checks the invertibility of a square matrix $\boldsymbol{A} \leftarrow\!\!\$\ \mathcal{D}_k = \mathcal{U}_k$ for $k \in \{1, 2\}$ given in both source groups (cf. Fig. 3).

Construction. We start with the asymmetric QA-NIZK argument Π'_{asy} from Fig. 1 and show how to obtain an updatable asymmetric QA-NIZK $\Pi'_{\mathsf{asy\text{-}up}}$. To this goal, similar as in previous work on updatable NIZK variants [29, 40], we design two new algorithms Ucrs and Vcrs. The Ucrs algorithm takes the original crs and updates it to a new $\mathsf{crs}_{\mathsf{up}}$ such that this update is publicly verifiable. Given

$\mathsf{MATV}([\boldsymbol{A}]_2)$ // $\mathcal{D}_k \in \{\mathcal{L}_k, \mathcal{IL}_k, \mathcal{C}_k, \mathcal{SC}_k\}$

check $[a_{11}]_2 \neq [0]_2 \wedge \ldots \wedge [a_{kk}]_2 \neq [0]_2$;
if $\mathcal{D}_k = \mathcal{L}_k$ **then** check $i \neq j \Rightarrow [a_{i,j}]_2 = [0]_2$;
elseif $\mathcal{D}_k = \mathcal{IL}_k$ **then** check $i \neq j \Rightarrow [a_{ij}]_2 = [0]_2$;
$\forall i, [a_{i,i}]_2 = [a_{1,1}]_2 + [i-1]_2$;
elseif $\mathcal{D}_k = \mathcal{C}_k$ **then** check $i \notin \{j, j+1\} \Rightarrow [a_{ij}]_2 = [0]_2$;
$\forall i, [a_{i+1,i}]_2 = [1]_2$;
elseif $\mathcal{D}_k = \mathcal{SC}_k$ **then** check $i \notin \{j, j+1\} \Rightarrow [a_{ij}]_2 = [0]_2$;
$\forall i \left([a_{i+1,i}]_2 = [1]_2 \wedge [a_{ii}]_2 = [a_{11}]_2\right)$; **fi**
return 1 if all checks pass and 0 otherwise;

Fig. 2: Auxiliary procedure MATV from [3] for $\mathcal{D}_k \in \{\mathcal{L}_k, \mathcal{IL}_k, \mathcal{C}_k, \mathcal{SC}_k\}$

.

$\mathsf{isinvertible}([\boldsymbol{A}]_2, [\boldsymbol{A}]_1)$ // $\mathcal{D}_k = \mathcal{U}_k$

if $k = 1$ **then** check $[a_{11}]_2 \neq [0]_2$
else check $[a_{11}, a_{12}]_1 \in \mathbb{G}_1^{1\times 2} \wedge [a_{11}]_1[1]_2 = [1]_1[a_{11}]_2 \wedge$
$[a_{12}]_1[1]_2 = [1]_1[a_{12}]_2 \wedge [a_{11}]_1[a_{22}]_2 - [a_{12}]_1[a_{21}]_2 \neq [0]_T$; **fi**

Fig. 3: Auxiliary procedure $\mathsf{isinvertible}$ for $\boldsymbol{A} \in \mathbb{Z}_p^{k\times k}$ and $k \in \{1, 2\}$.

the $\mathsf{crs}_{\mathsf{up}}$, the language parameters $[\boldsymbol{M}]_1$, and $[\boldsymbol{N}]_2$, the Vcrs algorithm checks the well-formedness of the $\mathsf{crs}_{\mathsf{up}}$. The latter checking guarantees the existence of a trapdoor $\mathtt{tc}$ for the $\mathsf{crs}_{\mathsf{up}}$, which will be required to prove the zero-knowledge property. Now, we take a closer look at the design of the update procedure.

Updating procedure. The updating phase is tricky as the updated elements need to be publicly verifiable via the Vcrs algorithm. Inspired by [40], we use both multiplicative and additive updating approaches. We let Ucrs adaptively update the element $\boldsymbol{P}_i$ for $i \in \{1, 2\}$, since due to the structure of crs of our updatable asymmetric QA-NIZK in Fig. 4, by using the crs, $\mathsf{crs}_{\mathsf{int}}$, and $\mathsf{crs}_{\mathsf{up}}$, the Vcrs algorithm can publicly verify them. But updating the element $\boldsymbol{A}$ is more tricky. In particular, if one updates it additively then in order to be able to verify the elements $\boldsymbol{C}_i$, which are needed to make trapdoor extraction possible, one would need to have $[\boldsymbol{K_i}]_i$ for $i \in \{1, 2\}$. More precisely, with additively updating $\boldsymbol{A}$, we would have $\boldsymbol{A}_{\mathsf{up}} = \boldsymbol{A} + \boldsymbol{A}_{\mathsf{int}}$ and the updating procedure of elements $\boldsymbol{C}_i$ is as follows:

$$\begin{aligned}[\boldsymbol{C}_{i,\mathsf{up}}]_i = [\boldsymbol{K}_{i,\mathsf{up}}\boldsymbol{A}_{\mathsf{up}}]_i &= [(\boldsymbol{K}_{\mathsf{int}} + \boldsymbol{K}_i)(\boldsymbol{A}_{\mathsf{int}} + \boldsymbol{A})]_i \\ &= [\boldsymbol{C}_{\mathsf{int}}]_i + [\boldsymbol{C}_i]_i + [\boldsymbol{K}_{\mathsf{int}}\boldsymbol{A}]_i + [\boldsymbol{K}_i\boldsymbol{A}_{\mathsf{int}}]_i,\end{aligned}$$

where for verifying $[\boldsymbol{K}_{\mathsf{int}}\boldsymbol{A}]_i$ and $[\boldsymbol{K}_i\boldsymbol{A}_{\mathsf{int}}]_i$ one needs to have $[\boldsymbol{K_i}]_i$. However, having these elements in the crs would leak information about the trapdoor. Thus, we need to update $\boldsymbol{A}$ multiplicatively as $\boldsymbol{A}_{\mathsf{up}} = \boldsymbol{A}\boldsymbol{A}_{\mathsf{int}}$.

Zero-knowledge property. In the zero-knowledge proof, we use the well-known BDH-KE knowledge assumption and show that if the possibly maliciously generated $\mathsf{crs}_{\mathsf{up}}$ passes the Vcrs algorithm, then under the knowledge assumptions there exists an extractor that extracts the trapdoor $\mathtt{tc}$ of $\mathsf{crs}_{\mathsf{up}}$. Using such a trapdoor $\mathtt{tc}$, the simulator can simulate proofs.

Soundness property. Since to achieve publicly verifiability of the $\mathsf{crs}_{\mathsf{up}}$, we add some new elements $[\boldsymbol{C}_2]_1$ and $[\boldsymbol{C}_1]_2$ in the CRS, we need to show that the soundness of the updatable asymmetric QA-NIZKs still holds. We prove the soundness under the standard SKerMDH assumption.

We depict the full construction of the updatable asymmetric QA-NIZK arguments in Fig. 4. Here, the elements with index int are intermediate elements generated by the algorithm Ucrs and can be viewed as update proofs, i.e., enabling to verify consistency of the old and updated crs. The elements with index up are the updated elements, i.e., the new crs. We note that our updatable asymmetric QA-NIZK in Fig. 4, the prover and the verifier are unchanged compared to GHR's QA-NIZK [25].

Remark 1. We note that, one can adapt the updatable asymmetric QA-NIZKs construction in Fig. 4 to other languages like as the *sum in subspace language* and obtain the updatable version of the *argument of sum in subspace* of [25].

3.3 Security Proof for Our Construction

In this section we prove Theorem 3.

Proof. The security properties (i-iii), crs-update correctness and completeness are straightforward from the construction. The crs-update hiding proof is similar [40] (see Appendix A for more details).

(iv: Subversion Zero-Knowledge:) For proving the zero-knowledge property, we need to construct a simulator that can construct proofs without knowing the witness but a trapdoor $\mathtt{tc}$. To this aim, in Lemma 1 (in the extraction phase), we show that from any adversary producing a valid crs from scratch it is possible to extract the trapdoors $(\boldsymbol{K}_1, \boldsymbol{K}_2)$. Then in the simulation phase, given the trapdoor $\mathtt{tc}$, we show how the zero-knowledge simulator can simulate proofs.

Extraction phase. Let the BDH-KE assumption hold. Let $\mathcal{A}$ be an adversary that computes crs so as to break the subversion zero-knowledge property of the updatable asymmetric QA-NIZK in Fig. 4. That is, $\mathcal{A}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2; \omega_{\mathcal{A}})$ outputs $(\mathsf{crs}, \mathsf{aux}_{\mathcal{A}})$. In Lemma 1, based on the BDH-KE assumption, we show how one can construct an extractor to extract the trapdoor $\mathtt{tc}$ of a possibly maliciously generated crs.

Lemma 1. *Let the* BDH-KE *assumption hold and let* $[\boldsymbol{M}]_1, [\boldsymbol{N}]_2 \leftarrow\!\!\$\ \mathcal{D}_{\mathsf{p}}$. *Then for any PPT adversary* $\mathcal{A}$ *there exists extractor* $\mathsf{Ext}_{\mathcal{A}}$ *such that the probability that* $\mathcal{A}$ *on input* $([\boldsymbol{M}]_1, [\boldsymbol{N}]_2)$ *and randomness* ω *outputs* crs *such that* $\mathsf{Vcrs}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2, \mathsf{crs}) = 1$ *and that* $\mathsf{Ext}_{\mathcal{A}}$ *on the same input, outputs* $\mathtt{tc} = (\boldsymbol{K}_1, \boldsymbol{K}_1)$, *is overwhelming.*

$\mathsf{K}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2)$

- $\boldsymbol{A} \leftarrow\!\!\$\ \hat{\mathcal{D}}_k$; $\boldsymbol{K}_1 \leftarrow\!\!\$\ \mathbb{Z}_p^{n_2 \times k}$; $\boldsymbol{K}_2 \leftarrow\!\!\$\ \mathbb{Z}_p^{n_1 \times k}$; $\boldsymbol{Z} \leftarrow\!\!\$\ \mathbb{Z}_p^{m \times k}$; $\boldsymbol{C}_1 \leftarrow \boldsymbol{K}_1\boldsymbol{A} \in \mathbb{Z}_p^{n_2 \times k}$;
- $\boldsymbol{C}_2 \leftarrow \boldsymbol{K}_2\boldsymbol{A} \in \mathbb{Z}_p^{n_1 \times k}$; $[\boldsymbol{P}_1]_1 \leftarrow [\boldsymbol{M}]_1^\top \boldsymbol{K}_2 + [\boldsymbol{Z}]_1 \in \mathbb{Z}_p^{m \times k}$;
- $[\boldsymbol{P}_2]_2 \leftarrow [\boldsymbol{N}]_2^\top \boldsymbol{K}_1 + [\boldsymbol{Z}]_2 \in \mathbb{Z}_p^{m \times k}$; $\mathsf{crs} \leftarrow ([\boldsymbol{A}, \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{P}_2]_2, [\boldsymbol{A}, \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{P}_1]_1)$;
- $\mathtt{tc} \leftarrow (\boldsymbol{K}_1, \boldsymbol{K}_2)$;
- **return** $(\mathtt{tc}, \mathsf{crs})$;

$\mathsf{Ucrs}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2, \mathsf{crs})$:

- $\boldsymbol{A}_{\mathsf{int}} \leftarrow\!\!\$\ \bar{\mathcal{D}}_k$; $\boldsymbol{K}_{1,\mathsf{int}} \leftarrow\!\!\$\ \mathbb{Z}_p^{n_2 \times k}$; $\boldsymbol{K}_{2,\mathsf{int}} \leftarrow\!\!\$\ \mathbb{Z}_p^{n_1 \times k}$; $\boldsymbol{Z}_{\mathsf{int}} \leftarrow\!\!\$\ \mathbb{Z}_p^{m \times k}$; $\boldsymbol{A}_{\mathsf{up}} \leftarrow \boldsymbol{A}\boldsymbol{A}_{\mathsf{int}}$;
- $\boldsymbol{C}_{1,\mathsf{int}} \leftarrow \boldsymbol{K}_{1,\mathsf{int}}\boldsymbol{A}_{\mathsf{up}} \in \mathbb{Z}_p^{n_1 \times k}$; $\boldsymbol{C}_{2,\mathsf{int}} \leftarrow \boldsymbol{K}_{2,\mathsf{int}}\boldsymbol{A}_{\mathsf{up}} \in \mathbb{Z}_p^{n_2 \times k}$;
- $[\boldsymbol{P}_{1,\mathsf{int}}]_1 \leftarrow [\boldsymbol{M}]_1^\top \boldsymbol{K}_{2,\mathsf{int}} + [\boldsymbol{Z}_{\mathsf{int}}]_1 \in \mathbb{Z}_p^{m \times k}$; $[\boldsymbol{P}_{2,\mathsf{int}}]_2 \leftarrow [\boldsymbol{N}]_2^\top \boldsymbol{K}_{1,\mathsf{int}} + [\boldsymbol{Z}_{\mathsf{int}}]_2 \in \mathbb{Z}_p^{m \times \hat{k}}$;
- $\boldsymbol{C}_{1,\mathsf{up}} \leftarrow \boldsymbol{C}_{1,\mathsf{int}} + \boldsymbol{C}_1\boldsymbol{A}_{\mathsf{int}}$; $\boldsymbol{C}_{2,\mathsf{up}} \leftarrow \boldsymbol{C}_{2,\mathsf{int}} + \boldsymbol{C}_2\boldsymbol{A}_{\mathsf{int}}$;
- $[\boldsymbol{P}_{1,\mathsf{up}}]_1 \leftarrow [\boldsymbol{P}_1]_1 + [\boldsymbol{P}_{1,\mathsf{int}}]_1$; $[\boldsymbol{P}_{2,\mathsf{up}}]_2 \leftarrow [\boldsymbol{P}_2]_2 + [\boldsymbol{P}_{2,\mathsf{int}}]_1$;
- $\mathsf{crs}_{\mathsf{int}} \leftarrow ([\boldsymbol{A}_{\mathsf{int}}, \boldsymbol{C}_{1,\mathsf{int}}, \boldsymbol{C}_{2,\mathsf{int}}.\boldsymbol{P}_{2,\mathsf{int}}]_2, [\boldsymbol{A}_{\mathsf{int}}, \boldsymbol{C}_{1,\mathsf{int}}, \boldsymbol{C}_{2,\mathsf{int}}, \boldsymbol{P}_{1,\mathsf{int}}]_1)$;
- **return** $\mathsf{crs}_{\mathsf{up}} \leftarrow ([\boldsymbol{A}_{\mathsf{up}}, \boldsymbol{C}_{1,\mathsf{up}}, \boldsymbol{C}_{2,\mathsf{up}}, \boldsymbol{P}_{2,\mathsf{up}}]_2, [\boldsymbol{A}_{\mathsf{up}}, \boldsymbol{C}_{1,\mathsf{up}}, \boldsymbol{C}_{2,\mathsf{up}}, \boldsymbol{P}_{1,\mathsf{up}}]_1, \mathsf{crs}_{\mathsf{int}})$;

$\mathsf{Vcrs}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2, \mathsf{crs}, \mathsf{crs}_{\mathsf{up}})$:

- **if**
  - for $i \in \{1, (1, \mathsf{int}), (1, \mathsf{up})\}$: : **if** $[\boldsymbol{C}_i]_1 \in \mathbb{G}_1^{n_1 \times k} \wedge [\boldsymbol{P}_i]_1 \in \mathbb{G}_1^{m \times k} \wedge [\boldsymbol{A}_i]_1 \in \mathbb{G}_1^{k \times k}$;
  - for $i \in \{2, (2, \mathsf{int}), (2, \mathsf{up})\}$: **if** $[\boldsymbol{C}_i]_2 \in \mathbb{G}_2^{n_2 \times k} \wedge [\boldsymbol{P}_i]_2 \in \mathbb{G}_2^{m \times k} \wedge [\boldsymbol{A}_i]_2 \in \mathbb{G}_2^{k \times k}$;
  - for $i \in \{1, 2, (1, \mathsf{int}), (1, \mathsf{up}), (2, \mathsf{int}), (2, \mathsf{up})\}$: **if** $[\boldsymbol{A}_i]_1[1]_2 = [1]_1[\boldsymbol{A}_i]_2; \wedge [\boldsymbol{C}_i]_1[1]_2 = [1]_1[\boldsymbol{C}_i]_2$;
  - $[\boldsymbol{A}_{\mathsf{up}}]_1[\boldsymbol{1}]_2 = [\boldsymbol{A}]_1[\boldsymbol{A}_{\mathsf{int}}]_2$
  - $[\boldsymbol{C}_{1,\mathsf{up}}]_1[\boldsymbol{1}]_2 = [\boldsymbol{C}_{1,\mathsf{int}}]_1[1]_2 + [\boldsymbol{C}_1]_1[\boldsymbol{A}_{\mathsf{int}}]_2 \wedge [\boldsymbol{C}_{2,\mathsf{up}}]_1[\boldsymbol{1}]_2 = [\boldsymbol{C}_{2,\mathsf{int}}]_1[1]_2 + [\boldsymbol{C}_2]_1[\boldsymbol{A}_{\mathsf{int}}]_2$
  - $[\boldsymbol{P}_{1,\mathsf{up}}]_1 = [\boldsymbol{P}_{1,\mathsf{int}}]_1 + [\boldsymbol{P}_1]_1 \wedge [\boldsymbol{P}_{2,\mathsf{up}}]_1 = [\boldsymbol{P}_{2,\mathsf{int}}]_1 + [\boldsymbol{P}_2]_1$;
  - $[\boldsymbol{P}_1]_1[\boldsymbol{A}]_2 - [\boldsymbol{A}]_1[\boldsymbol{P}_2]_2 = [\boldsymbol{M}]_1[\boldsymbol{C}_2]_2 - [\boldsymbol{N}]_2[\boldsymbol{C}_1]_1$;
  - $[\boldsymbol{P}_{1,\mathsf{int}}]_1[\boldsymbol{A}_{\mathsf{int}}]_2 - [\boldsymbol{A}_{\mathsf{int}}]_1[\boldsymbol{P}_{2,\mathsf{int}}]_2 = [\boldsymbol{M}]_1[\boldsymbol{C}_{2,\mathsf{int}}]_2 - [\boldsymbol{N}]_2[\boldsymbol{C}_{1,\mathsf{int}}]_1$;
  - $[\boldsymbol{P}_{1,\mathsf{up}}]_1[\boldsymbol{A}_{\mathsf{up}}]_2 - [\boldsymbol{A}_{\mathsf{up}}]_1[\boldsymbol{P}_{2,\mathsf{up}}]_2 = [\boldsymbol{M}]_1[\boldsymbol{C}_{2,\mathsf{up}}]_2 - [\boldsymbol{N}]_2[\boldsymbol{C}_{1,\mathsf{up}}]_1$;
  - for $i \in \mathsf{int}, \mathsf{up}$: **if** $\mathcal{D}_k$ is efficiently verifiable then $\mathsf{MATV}([\boldsymbol{A}_i]_2) = 1 \wedge \mathsf{MATV}([\boldsymbol{A}]_2) = 1$ else check $\mathsf{isinvertible}([\boldsymbol{A}_i]_2, [\boldsymbol{A}_i]_1) = 1$;
- **return** 1;
- **else return** 0;

$\mathsf{P}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2, \mathsf{crs}_{\mathsf{up}}, [\boldsymbol{y}]_1, [\boldsymbol{x}]_2, \boldsymbol{w})$:

- $\boldsymbol{r} \leftarrow\!\!\$\ \mathbb{Z}_p^k$; $[\boldsymbol{\pi}_1]_1 \leftarrow [\boldsymbol{P}_1]_1^\top \boldsymbol{w} + [\boldsymbol{r}]_1 \in \mathbb{G}_1^k \wedge [\boldsymbol{\pi}_2]_2 \leftarrow [\boldsymbol{P}_2]_2^\top \boldsymbol{w} + [\boldsymbol{r}]_2 \in \mathbb{G}_2^k$;
- **return** $([\boldsymbol{\pi}_1]_1, [\boldsymbol{\pi}]_2)$;

$\mathsf{V}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2, \mathsf{crs}_{\mathsf{up}}, [\boldsymbol{y}]_1, [\boldsymbol{x}]_2, [\boldsymbol{\pi}_1]_1, [\boldsymbol{\pi}_2]_2)$:

- **if** $[\boldsymbol{y}]_1^\top [\boldsymbol{C}_2]_2 - [\boldsymbol{\pi}_1]_1^\top [\boldsymbol{A}]_2 = [\boldsymbol{\pi}_2]_2^\top [\boldsymbol{A}]_1 - [\boldsymbol{x}]_2^\top [\boldsymbol{C}_1]_1$ **return** 1;
- **else return** 0;

$\mathsf{Sim}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2, \mathsf{crs}, \mathtt{tc}, [\boldsymbol{y}]_1)$:

- $\boldsymbol{r} \leftarrow\!\!\$\ \mathbb{Z}_p^k$; - $[\boldsymbol{\pi}_1]_1 \leftarrow \boldsymbol{K}_2^\top [\boldsymbol{y}]_1 + [\boldsymbol{r}]_1 \in \mathbb{G}_1^k$; - $[\boldsymbol{\pi}]_2 \leftarrow \boldsymbol{K}_1^\top [\boldsymbol{x}]_2 + [\boldsymbol{r}]_2 \in \mathbb{G}_1^k$;
- **return** $([\boldsymbol{\pi}_1]_1, [\boldsymbol{\pi}]_2)$;

Fig. 4: Updatable Asymmetric QA-NIZK $\Pi'_{\mathsf{asy\text{-}up}}$. Here k is an arbitrary value if $\mathcal{D}_k \in \{\mathcal{L}_k, \mathcal{IL}_k, \mathcal{C}_k, \mathcal{SC}_k\}$ and $k \in \{1, 2\}$ in $\mathcal{D}_k = \mathcal{U}_k$.

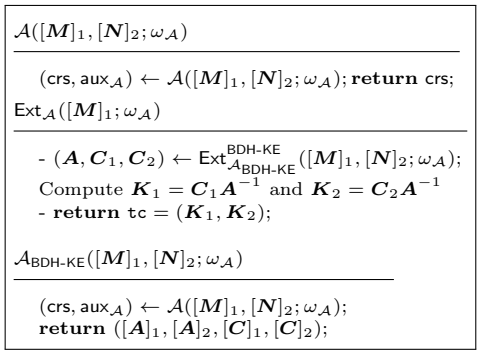

Fig. 5: The extractors and the constructed adversary $\mathcal{A}$ for Lemma 1.

Proof. Let adversary $\mathcal{A}$ output crs such that $\mathsf{Vcrs}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2, \mathsf{crs}) = 1$, which guarantees that elements from $\boldsymbol{P}_i$, $\boldsymbol{A}$ and $\boldsymbol{C}_i$ for $i \in \{1, 2\}$ are consistent and in particular that $[\boldsymbol{P}_1]_1[\boldsymbol{A}]_2 - [\boldsymbol{A}]_1[\boldsymbol{P}_2]_2 = [\boldsymbol{M}]_1[\boldsymbol{C}_2]_2 - [\boldsymbol{N}]_2[\boldsymbol{C}_1]_1$and $\boldsymbol{A}$ is invertible. Assume an internal subverter $\mathcal{A}_{\mathsf{BDH\text{-}KE}}$ against the $\mathsf{BDH\text{-}KE}$ assumption. We note that both the subverter and the adversary are in connection and separating them is just for readability of the proof. Let $\omega_{\mathcal{A}} = \omega_{\mathcal{A}_{\mathsf{BDH\text{-}KE}}}$. Let $\mathcal{A}_{\mathsf{BDH\text{-}KE}}$ run $\mathcal{A}$ and output $([\boldsymbol{A}]_1, [\boldsymbol{A}]_2, [\boldsymbol{C}_1, \boldsymbol{C}_2]_1, [\boldsymbol{C}_1, \boldsymbol{C}_2]_2)$. Then under the $\mathsf{BDH\text{-}KE}$ assumption, there exists an extractor $\mathsf{Ext}^{\mathsf{BDH\text{-}KE}}_{\mathcal{A}_{\mathsf{BDH\text{-}KE}}}$, such that if $\mathsf{Vcrs}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2, \mathsf{crs}) = 1$ then $\mathsf{Ext}^{\mathsf{BDH\text{-}KE}}_{\mathcal{A}_{\mathsf{BDH\text{-}KE}}}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2; \omega_{\mathcal{A}})$ outputs $(\boldsymbol{A}, \boldsymbol{C}_1, \boldsymbol{C}_2)$.

Let $\mathsf{Ext}_{\mathcal{A}}$ be an extractor that with input $([\boldsymbol{M}]_1, [\boldsymbol{N}]_2; \omega_{\mathcal{A}})$ and running $\mathsf{Ext}^{\mathsf{BDH\text{-}KE}}_{\mathcal{A}_{\mathsf{BDH\text{-}KE}}}$ as subroutine, extracts $\mathtt{tc} = (\boldsymbol{K}_1, \boldsymbol{K}_2)$. For the sake of simplicity, the full description of the algorithms is depicted in Fig. 5. More precisely, the extractor $\mathsf{Ext}_{\mathcal{A}}$ first runs $\mathsf{Ext}^{\mathsf{BDH\text{-}KE}}_{\mathcal{A}_{\mathsf{BDH\text{-}KE}}}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2; \omega_{\mathcal{A}})$ which outputs $(\boldsymbol{A}, \boldsymbol{C}_1, \boldsymbol{C}_2)$. Then, $\mathsf{Ext}_{\mathcal{A}}$ computes $(\boldsymbol{K}_1, \boldsymbol{K}_2)$. Indeed, by having $\boldsymbol{A}$, $\boldsymbol{C}_i$, and the fact that $\boldsymbol{A}$ is invertible, the extractor $\mathsf{Ext}_{\mathcal{A}}$ can compute $\boldsymbol{K}_i = \boldsymbol{C}_i\boldsymbol{A}^{-1}$ for $i \in \{1, 2\}$. □

Simulation phase. In the second step, given the trapdoor $\mathtt{tc}$, we show how a simulator Sim can simulate proofs. Fix concrete values of λ, $\mathsf{p} \in \mathrm{im}(\mathsf{Pgen}(1^\lambda))$, $([\boldsymbol{y}]_1, [\boldsymbol{x}]_2, \boldsymbol{w}) \in \mathcal{R}_{[\boldsymbol{M}]_1, [\boldsymbol{N}]_2}$, $\omega_{\mathcal{A}} \in \mathsf{RND}(\mathcal{A})$, and run $\mathsf{Ext}_{\mathcal{A}}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2; \omega_{\mathcal{A}})$ to obtain $(\boldsymbol{K}_1, \boldsymbol{K}_2)$. Thus, it suffices to show that if $\mathsf{Vcrs}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2, \mathsf{crs}) = 1$ and $([\boldsymbol{y}]_1, [\boldsymbol{x}]_2, \boldsymbol{w}) \in \mathcal{R}_{[\boldsymbol{M}]_1, [\boldsymbol{N}]_2}$ then

$$\begin{aligned}\mathsf{O}_0([\boldsymbol{y}]_1, [\boldsymbol{x}]_2, \boldsymbol{w}) &= \mathsf{P}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2, \mathsf{crs}, [\boldsymbol{y}]_1, [\boldsymbol{x}]_2, \boldsymbol{w}) \ , \\ \mathsf{O}_1([\boldsymbol{y}]_1, [\boldsymbol{x}]_2, \boldsymbol{w}) &= \mathsf{Sim}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2, \mathsf{crs}, [\boldsymbol{y}]_1, [\boldsymbol{x}]_2, \boldsymbol{K}_1, \boldsymbol{K}_2)\end{aligned}$$

have the same distribution. Since O_0 and O_1 have the same distribution, $\Pi'_{\mathsf{asy\text{-}up}}$ is zero-knowledge under the $\mathsf{BDH\text{-}KE}$ assumption.

(v: Updatable Adaptive Soundness:) The proof is similar to the adaptive soundness proof of Π'_{as} in [25] but with some modifications. Let $m' := n_1 + n_2$ and $\boldsymbol{W} := \left(\begin{smallmatrix}\boldsymbol{M}\\ \boldsymbol{N}\end{smallmatrix}\right)$. Let an adversary $\mathcal{B}$ against $\mathcal{D}_k$-SKerMDH assumption be given a challenge $([\boldsymbol{A}]_1, [\boldsymbol{A}]_2)$, $\boldsymbol{A} \leftarrow \mathcal{D}_k$.

$\mathcal{B}$ samples $([\boldsymbol{M}]_1, [\boldsymbol{N}]_2, \boldsymbol{M}, \boldsymbol{N}) \in \mathcal{R}_{\mathsf{p}}$ and computes $\boldsymbol{W}^{\perp} \in \mathbb{Z}_p^{m' \times (m'-r)}$, where $r = \mathsf{rank}(\boldsymbol{W})$, a basis of the kernel of $\boldsymbol{W}^{\top}$. By definition, $\boldsymbol{W}^{\top} = (\boldsymbol{M}^{\top} || \boldsymbol{N}^{\top})$ and $\boldsymbol{W}^{\top}\boldsymbol{W}^{\perp} = 0$ and thus we can write $\boldsymbol{W}^{\perp} = (\boldsymbol{W}_1, \boldsymbol{W}_2)$, for some matrices such that $\boldsymbol{M}^{\top}\boldsymbol{W}_1 = -\boldsymbol{N}^{\top}\boldsymbol{W}_2$.

Adversary $\mathcal{B}$ samples $\boldsymbol{R} \in \mathbb{Z}_p^{(m'-r-1)\times(k+1)}$ and for $i \in \{1,2\}$ defines,

$$[\boldsymbol{A}']_i \leftarrow \begin{pmatrix} [\boldsymbol{A}]_i \\ \boldsymbol{R} \cdot [\boldsymbol{A}]_i \end{pmatrix} \in \mathbb{Z}_p^{(k+m'-r)\times k}.$$

Then $\mathcal{B}$ samples $(\boldsymbol{K}'_1, \boldsymbol{K}'_2) \leftarrow \mathbb{Z}_p^{m' \times k}$. Let $\boldsymbol{A}_0$ be the first k rows of $\boldsymbol{A}'$ (or $\boldsymbol{A}$) and $\boldsymbol{A}'$ rows, and $\boldsymbol{T}_{\boldsymbol{A}'} = \boldsymbol{A}'_1\boldsymbol{A}_0^{-1}$. Then $\mathcal{B}$ implicitly sets $(\boldsymbol{K}_1, \boldsymbol{K}_2) := (\boldsymbol{K}'_1, \boldsymbol{K}'_2) + \boldsymbol{T}_{\boldsymbol{A}'}(\boldsymbol{W}_1 || \boldsymbol{W}_2)$, and for $i \in \{1,2\}$ and computes:

$$[\boldsymbol{C}_1]_i := \boldsymbol{K}_2[\boldsymbol{A}_0]_i = (\boldsymbol{K}'_2 + \boldsymbol{T}_{\boldsymbol{A}'}\boldsymbol{W}_2)[\boldsymbol{A}_0]_i = (\boldsymbol{K}'_2 || \boldsymbol{W}_2)[\boldsymbol{A}']_i,$$

and

$$[\boldsymbol{C}_2]_i := \boldsymbol{K}_1[\boldsymbol{A}_0]_i = (\boldsymbol{K}'_1 + \boldsymbol{T}_{\boldsymbol{A}'}\boldsymbol{W}_1)[\boldsymbol{A}_0]_i = (\boldsymbol{K}'_1 || \boldsymbol{W}_1)[\boldsymbol{A}']_i.$$

Adversary $\mathcal{B}$ also needs to compute $[\boldsymbol{M}]_1^{\top}\boldsymbol{K}_2 + [\boldsymbol{Z}]_1$ and $[\boldsymbol{N}]_2^{\top}\boldsymbol{K}_1 - [\boldsymbol{Z}]_2$. The adversary $\mathcal{B}$ does not know how to compute $\boldsymbol{N}^{\top}\boldsymbol{K}_1$ or $\boldsymbol{M}^{\top}\boldsymbol{K}_2$, but she can compute their sum in $\mathbb{Z}_p$ as:

$$\boldsymbol{N}^{\top}\boldsymbol{K}_1 + \boldsymbol{M}^{\top}\boldsymbol{K}_2 = \begin{pmatrix} \boldsymbol{M}^{\top} \\ \boldsymbol{N}^{\top} \end{pmatrix}(\boldsymbol{K}'_1, \boldsymbol{K}'_2) + \boldsymbol{T}_{\boldsymbol{A}'}(\boldsymbol{W}_1 || \boldsymbol{W}_2) = \boldsymbol{N}^{\top}\boldsymbol{K}'_1 + \boldsymbol{M}^{\top}\boldsymbol{K}'_2 := \boldsymbol{T},$$

due to the fact that $\boldsymbol{M}^{\top}\boldsymbol{W}_1 = -\boldsymbol{N}^{\top}\boldsymbol{W}_2$.

Thus, $\mathcal{B}$ picks $\boldsymbol{Z} \leftarrow\!\!\$\, \mathbb{Z}_p^{m \times k}$ and outputs $[\boldsymbol{P}]_2 := [\boldsymbol{T}]_2 - [\boldsymbol{Z}]_2$ and $[\boldsymbol{P}]_1 := [\boldsymbol{Z}]_1$. Now, when the adversary outputs a valid proof for some $([\boldsymbol{y}]_1, [\boldsymbol{x}]_2) \notin \mathcal{L}_{[\boldsymbol{M}]_1, [\boldsymbol{N}]_2}$, it holds that:

$$[\boldsymbol{y}]_1^{\top}[\boldsymbol{C}_2]_2 - [\boldsymbol{\pi}_1]_1^{\top}[\boldsymbol{A}_0]_2 = [\boldsymbol{\pi}_2]_2^{\top}[\boldsymbol{A}_0]_1 - [\boldsymbol{x}]_2^{\top}[\boldsymbol{C}_1]_1.$$

In which both the RHS and LHS of the last equation are:

$$\text{LHS} = [\boldsymbol{y}]_1^{\top}(\boldsymbol{K}'_2 || \boldsymbol{W}_1)[\boldsymbol{A}']_2 - ([\boldsymbol{\pi}_1]_1^{\top} || [\boldsymbol{0}_{1\times(m'-r)}]_1)^{\top}[\boldsymbol{A}']_2 = [\boldsymbol{s}_1^{\top}]_1[\boldsymbol{A}']_2,$$

$$\text{RHS} = ([\boldsymbol{\pi}_2]_2^{\top} || [\boldsymbol{0}_{1\times(m'-r)}]_2)[\boldsymbol{A}']_1 - [\boldsymbol{x}]_2^{\top}(\boldsymbol{K}'_1 || \boldsymbol{W}_2)[\boldsymbol{A}']_1 = [\boldsymbol{s}_2^{\top}]_2[\boldsymbol{A}']_1.$$

Here $[\boldsymbol{s}_1^{\top}]_1 := ([\boldsymbol{y}]_1^{\top}\boldsymbol{K}'_2 - [\boldsymbol{\pi}_1]_1^{\top} || [\boldsymbol{y}]_1^{\top}\boldsymbol{W}_1)$ and $[\boldsymbol{s}_2^{\top}]_2 := ([\boldsymbol{x}]_2^{\top}\boldsymbol{K}'_1 - [\boldsymbol{\pi}_2]_2^{\top} || - [\boldsymbol{x}]_2^{\top}\boldsymbol{W}_2)$.

This concludes that $(\boldsymbol{s}_1 - \boldsymbol{s}_2)$ is in the kernel space of $(\boldsymbol{A}')^{\top}$. In other words, we have that $(\boldsymbol{s}_1 - \boldsymbol{s}_2)^{\top}\boldsymbol{A}' = \boldsymbol{0}$, and by definition, $\boldsymbol{s}_1 - \boldsymbol{s}_2 = \boldsymbol{c}_1 + \boldsymbol{R}^{\top}\boldsymbol{c}_2$ and thus

$$(\boldsymbol{s}_1^{\top} - \boldsymbol{s}_2^{\top})\boldsymbol{A} = (\boldsymbol{c}_1^{\top} + \boldsymbol{c}_2^{\top}\boldsymbol{R})\boldsymbol{A} = \boldsymbol{c}^{\top}\boldsymbol{A}' = \boldsymbol{0}_{1\times k} \ .$$

Since $\boldsymbol{c} \neq \boldsymbol{0}$ and $\boldsymbol{R}$ leaks only through $\boldsymbol{A}'$ (in the definition of $\boldsymbol{C}_i$ for $i \in 1,2$) as $\boldsymbol{RA}$,

$$\Pr[\boldsymbol{c}_1 + \boldsymbol{R}^{\top}\boldsymbol{c}_2 = \boldsymbol{0} \mid \boldsymbol{RA}] \leq 1/p \ ,$$

where the probability is over $\boldsymbol{R}$. This solves the $\mathcal{D}_k$-SKerMDH. □

4 Knowledge Soundness of (Updatable) Asymmetric QA-NIZK Arguments

In the following we investigate a stronger soundness notion, i.e., the knowledge soundness, of (updatable) asymmetric QA-NIZK. We recall that for a proof system to be compatible with modular zk-SNARK frameworks such as LegoSNARK [13], it needs to provide knowledge soundness. Consequently, this guarantees that (updatable) asymmetric QA-NIZK can safely be used within such frameworks.

Similar to as it is done in [6,13], we analyse the knowledge soundness in the Algebraic Group Model (AGM) due to Fuchsbauer *et al.* [21]. In particular, we first directly prove the knowledge soundness property of the updatable asymmetric QA-NIZK in Fig. 4. Moreover, we show that this also yields the knowledge sound property of the original asymmetric QA-NIZK [25] in Fig. 1.

Theorem 4. *Let $\Pi'_{\mathsf{asy\text{-}up}}$ be an asymmetric QA-NIZK argument for linear subspaces from Fig. 1. Assume $\hat{\mathcal{D}}_k = \bar{\mathcal{D}}_k$ and the distribution $\mathcal{D}_{\mathsf{p}}$ is witness samplable matrix distribution. If the discrete logarithm assumption in asymmetric bilinear groups in the AGM holds, then the updatable asymmetric QA-NIZK $\Pi'_{\mathsf{asy\text{-}up}}$ in Fig. 4 is computationally updatable adaptive knowledge sound.*

Proof. We show the theorem under the discrete logarithm assumption in asymmetric bilinear groups in the AGM. Without loss of generality, we consider the updatable asymmetric QA-NIZK scheme $\Pi'_{\mathsf{asy\text{-}up}}$ for $\hat{\mathcal{D}}_k = \bar{\mathcal{D}}_k$, in the MDDH setting where $k = 1$.

Without loss of generality, we assume an algebraic adversary $\mathcal{A}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2, \mathsf{aux})$ against the updatable knowledge soundness of $\Pi'_{\mathsf{asy\text{-}up}}$, where aux is an associated auxiliary input, such that it first generates a $\mathsf{crs}_{\mathcal{A}}$ which verifies under the algorithm Vcrs. Then an honest updator $\mathsf{Ucrs}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2, \mathsf{crs}_{\mathcal{A}})$ outputs an updated $\mathsf{crs} = ([\boldsymbol{A}, \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{P}_2]_2, [\boldsymbol{A}, \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{P}_1]_1)$.

Let $[\boldsymbol{\zeta}]_1$ and $[\boldsymbol{\zeta}']_2$ be vectors that contain $\boldsymbol{M}$ (and the portion of aux that has elements from the group $\mathbb{G}_1$) and $\boldsymbol{N}$ (and the portion of aux that has elements from the group $\mathbb{G}_2$). We assume that $[\boldsymbol{\zeta}]_1$ and $[\boldsymbol{\zeta}']_2$ also contains the $\mathsf{crs}_{\mathcal{A}}$'s elements in $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Due to the CRS-update hiding property, the $\mathsf{crs}_{\mathcal{A}}$'s elements are indistinguishable from the crs generated by an honest Ucrs and so they will give no advantage to $\mathcal{A}$. Also assume that $[\boldsymbol{\zeta}]_1$ and $[\boldsymbol{\zeta}']_2$ include $[1]_1$ and $[1]_2$, respectively. $\mathcal{A}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2, \mathsf{crs}, \mathsf{aux})$ returns a tuple $([\boldsymbol{y}]_1, [\boldsymbol{x}]_2, [\pi_1]_1, [\pi_2]_2)$ along with coefficients that explain these elements as linear combinations of its input in the groups $\mathbb{G}_1$ and $\mathbb{G}_2$. Let these coefficients be:

$$
\begin{aligned}
[\boldsymbol{y}]_1 &= \boldsymbol{Y}_0[\boldsymbol{P}_1]_1 + \boldsymbol{Y}_1[\boldsymbol{\zeta}]_1 + \boldsymbol{Y}_2[\boldsymbol{A}]_1 + \boldsymbol{Y}_3[\boldsymbol{C}_1]_1 + \boldsymbol{Y}_4[\boldsymbol{C}_2]_1 \\
[\boldsymbol{\pi}]_1 &= \boldsymbol{Z}_0[\boldsymbol{P}_1]_1 + \boldsymbol{Z}_1[\boldsymbol{\zeta}]_1 + \boldsymbol{Z}_2[\boldsymbol{A}]_1 + \boldsymbol{Z}_3[\boldsymbol{C}_1]_1 + \boldsymbol{Z}_4[\boldsymbol{C}_2]_1 \\
[\boldsymbol{x}]_2 &= \boldsymbol{Y}'_0[\boldsymbol{P}_2]_2 + \boldsymbol{Y}'_1[\boldsymbol{\zeta}']_2 + \boldsymbol{Y}'_2[\boldsymbol{A}]_2 + \boldsymbol{Y}'_3[\boldsymbol{C}_1]_1 + \boldsymbol{Y}'_4[\boldsymbol{C}_2]_2 \\
[\boldsymbol{\pi}]_2 &= \boldsymbol{Z}'_0[\boldsymbol{P}_2]_2 + \boldsymbol{Z}'_1[\boldsymbol{\zeta}']_2 + \boldsymbol{Z}'_2[\boldsymbol{A}]_2 + \boldsymbol{Z}'_3[\boldsymbol{C}_1]_2 + \boldsymbol{Z}'_4[\boldsymbol{C}_2]_2
\end{aligned}
$$

Let the extractor $\mathsf{Ext}_{\mathcal{A}}([\boldsymbol{M}]_1, [\boldsymbol{N}]_2, \mathsf{crs}, \mathsf{aux})$ be the algorithm that runs $\mathcal{A}$ and returns $\boldsymbol{w} = \boldsymbol{Z}_0 = \boldsymbol{Z}'_0$. Then, we have to show that the probability that the output of $(\mathcal{A}, \mathsf{Ext}_{\mathcal{A}})$ satisfies verification while $\boldsymbol{y} \neq \boldsymbol{M}\boldsymbol{w}$ and $\boldsymbol{x} \neq \boldsymbol{N}\boldsymbol{w}$ are negligible. In other words, assume that the output of $\mathcal{A}$ is such that $[\boldsymbol{y}]_1 \neq [\boldsymbol{M}]_1\boldsymbol{Z}_0$, $[\boldsymbol{x}]_2 \neq [\boldsymbol{N}]_2\boldsymbol{Z}'_0$, and $[\boldsymbol{y}]_1^\top[\boldsymbol{K}_2 a]_2 - [\boldsymbol{\pi}_1]_1^\top[a]_2 = [\boldsymbol{\pi}_2]_2^\top[a]_1 - [\boldsymbol{x}]_2^\top[\boldsymbol{K}_1 a]_1$; If it happens with non-negligible probability, we can construct an algorithm $\mathcal{B}$ that on input $([\boldsymbol{K}_1, \boldsymbol{K}_2]_1, [\boldsymbol{K}_1, \boldsymbol{K}_2]_2)$ outputs nonzero elements $\boldsymbol{\alpha}, \boldsymbol{\alpha}' \in \mathbb{Z}_p^{\ell\times\ell}$, $\boldsymbol{\beta}, \boldsymbol{\beta}' \in \mathbb{Z}_p^{\ell}$, and $\gamma, \gamma' \in \mathbb{Z}_p$ such that

$$[\boldsymbol{K}_1^\top\boldsymbol{\alpha}\boldsymbol{K}_1 + \boldsymbol{K}_1^\top\boldsymbol{\beta} + \gamma]_1[1]_2 + [1]_1[\boldsymbol{K}_2^\top\boldsymbol{\alpha}'\boldsymbol{K}_2 + \boldsymbol{K}_2^\top\boldsymbol{\beta}' + \gamma']_2 = [0]_T.$$

Then we can construct an algorithm $\mathcal{C}$ against the discrete logarithm assumption in asymmetric bilinear groups such that given elements $([t, t']_1, [t, t']_2)$ it returns the exponent $t, t' \in \mathbb{Z}_p$. More precisely, the algorithm $\mathcal{B}([\boldsymbol{K}_1, \boldsymbol{K}_2]_1, [\boldsymbol{K_1}, \boldsymbol{K}_2]_2)$ proceeds as follows:

- Choose $([\boldsymbol{M}]_1, [\boldsymbol{N}]_2, \mathsf{aux})$ from $\mathcal{D}_{\mathsf{p}}$ along with corresponding elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ (i.e., vectors $\boldsymbol{\zeta}$, $\boldsymbol{\zeta}'$ of entries in $\mathbb{Z}_p$).
- Sample $a \leftarrow\!\$\ \mathbb{Z}_p$ and run $\mathcal{A}([\boldsymbol{\zeta}', \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{P}_1, a]_1, [\boldsymbol{\zeta}', a, \boldsymbol{P}_2, \boldsymbol{C}_1, \boldsymbol{C}_2]_2$. We note that $\mathcal{A}$'s input can be efficiently simulated.
- Once received the output of $\mathcal{A}$, it sets $\boldsymbol{\alpha} := \boldsymbol{Y}_0\boldsymbol{M}^\top$, $\boldsymbol{\beta} := \boldsymbol{Y}_1\boldsymbol{\zeta} + \boldsymbol{Y}_2 a + \boldsymbol{Y}_3\boldsymbol{C}_1 + \boldsymbol{Y}_4\boldsymbol{C}_2 - \boldsymbol{M}\boldsymbol{Z}_0$ and $\gamma := -\boldsymbol{Z}_1\boldsymbol{\zeta} - \boldsymbol{Z}_2 a - \boldsymbol{Z}_3\boldsymbol{C}_1 - \boldsymbol{Z}_4\boldsymbol{C}_2$.
- Also it sets $\boldsymbol{\alpha}' := \boldsymbol{Y}'_0\boldsymbol{N}^\top$, $\boldsymbol{\beta}' := \boldsymbol{Y}'_1\boldsymbol{\zeta}' + \boldsymbol{Y}'_2 a + \boldsymbol{Y}'_3\boldsymbol{C}_1 + \boldsymbol{Y}'_4\boldsymbol{C}_2 - \boldsymbol{N}\boldsymbol{Z}'_0$ and $\gamma' := -\boldsymbol{Z}'_1\boldsymbol{\zeta}' - \boldsymbol{Z}'_2 a - \boldsymbol{Z}'_3\boldsymbol{C}_1 - \boldsymbol{Z}'_4\boldsymbol{C}_2$.

Notice that,

$$\begin{aligned}
\boldsymbol{K}_1^\top\boldsymbol{\alpha}\boldsymbol{K}_1 + \boldsymbol{K}_1^\top\boldsymbol{\beta} + \gamma =& \boldsymbol{K}_1^\top\boldsymbol{Y}_0\boldsymbol{M}^\top\boldsymbol{K}_1 + \boldsymbol{K}_1^\top\boldsymbol{Y}_1\boldsymbol{\zeta} + \boldsymbol{K}_1^\top\boldsymbol{Y}_2 a + \boldsymbol{K}_1^\top\boldsymbol{Y}_3\boldsymbol{C}_1 + \\
& \boldsymbol{K}_1^\top\boldsymbol{Y}_4\boldsymbol{C}_2 - \boldsymbol{K}_1^\top\boldsymbol{M}\boldsymbol{Z}_0 - \boldsymbol{Z}_1\boldsymbol{\zeta} - \boldsymbol{Z}_2 a - \boldsymbol{Z}_3\boldsymbol{C}_2 - \boldsymbol{Z}_4\boldsymbol{C}_2 \\
=& \boldsymbol{K}_1^\top\boldsymbol{Y}_0\boldsymbol{M}^\top\boldsymbol{K}_1 + \boldsymbol{K}_1^\top\boldsymbol{Y}_1\boldsymbol{\zeta} + \boldsymbol{K}_1^\top\boldsymbol{Y}_2 a + \boldsymbol{K}_1^\top\boldsymbol{Y}_3\boldsymbol{C}_1 + \boldsymbol{K}_1^\top\boldsymbol{Y}_4\boldsymbol{C}_2 - \boldsymbol{\pi}_1 \\
=& \boldsymbol{K}_1^\top\boldsymbol{y} - \boldsymbol{\pi}_1,
\end{aligned}$$

and

$$\begin{aligned}
\boldsymbol{K}_2^\top\boldsymbol{\alpha}'\boldsymbol{K}_2 + \boldsymbol{K}_2^\top\boldsymbol{\beta}' + \gamma' =& \boldsymbol{K}_2^\top\boldsymbol{Y}'_0\boldsymbol{N}^\top\boldsymbol{K}_2 + \boldsymbol{K}_2^\top\boldsymbol{Y}'_1\boldsymbol{\zeta}' + \boldsymbol{K}_2^\top\boldsymbol{Y}'_2 a + \boldsymbol{K}_2^\top\boldsymbol{Y}'_3\boldsymbol{C}_1 + \\
& \boldsymbol{K}_2^\top\boldsymbol{Y}'_4\boldsymbol{C}_2 - \boldsymbol{K}_2^\top\boldsymbol{N}\boldsymbol{Z}'_0 - \boldsymbol{Z}'_1\boldsymbol{\zeta}' - \boldsymbol{Z}'_2 a - \boldsymbol{Z}'_3\boldsymbol{C}_2 - \boldsymbol{Z}'_4\boldsymbol{C}_2 \\
=& \boldsymbol{K}_2^\top\boldsymbol{Y}'_0\boldsymbol{N}^\top\boldsymbol{K}_2 + \boldsymbol{K}_2^\top\boldsymbol{Y}'_1\boldsymbol{\zeta}' + \boldsymbol{K}_2^\top\boldsymbol{Y}'_2 a + \boldsymbol{K}_2^\top\boldsymbol{Y}'_3\boldsymbol{C}_1 + \boldsymbol{K}_2^\top\boldsymbol{Y}'_4\boldsymbol{C}_2 - \boldsymbol{\pi}_2 \\
=& \boldsymbol{K}_2^\top\boldsymbol{x} - \boldsymbol{\pi}_2.
\end{aligned}$$

From the verification equation, we have

$$\begin{aligned}
&(\boldsymbol{K}_1^\top\boldsymbol{\alpha}\boldsymbol{K}_1 + \boldsymbol{K}_1^\top\boldsymbol{\beta} + \gamma) + (\boldsymbol{K}_2^\top\boldsymbol{\alpha}'\boldsymbol{K}_2 + \boldsymbol{K}_2^\top\boldsymbol{\beta}' + \gamma') \\
&= \boldsymbol{K}_1^\top\boldsymbol{y} - \boldsymbol{\pi}_1 + \boldsymbol{K}_2^\top\boldsymbol{x} - \boldsymbol{\pi}_2 = 0.
\end{aligned}$$

Note that, one among $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and γ ($\boldsymbol{\alpha}'$, $\boldsymbol{\beta}'$, and γ') must be nonzero. Indeed, if they are all zero then $\boldsymbol{Y}_1\boldsymbol{\zeta} + \boldsymbol{Y}_2 a + \boldsymbol{Y}_3\boldsymbol{C} - \boldsymbol{M}\boldsymbol{Z}_0 = 0$ ($\boldsymbol{Y}'_1\boldsymbol{\zeta}' + \boldsymbol{Y}'_2 a + \boldsymbol{Y}'_3\boldsymbol{C} - \boldsymbol{N}\boldsymbol{Z}'_0 = 0$), that is $\boldsymbol{y} = \boldsymbol{M}\boldsymbol{Z}_0$ ($\boldsymbol{x} = \boldsymbol{N}\boldsymbol{Z}'_0$), which contradicts our assumption on $\mathcal{A}$'s output.

Finally we show how the above problem can be reduced to discrete logarithm problem in asymmetric groups, i.e., the adversary $\mathcal{C}$ on input $([t, t']_1, [t, t']_2)$ returns t' and t'.

Indeed $\mathcal{C}$ samples $\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{r}', \boldsymbol{s}' \in \mathbb{Z}_p^\ell$ and implicitly sets $\boldsymbol{K}_1 = t\boldsymbol{r} + \boldsymbol{s}$ and $\boldsymbol{K}_2 = t\boldsymbol{r}' + \boldsymbol{s}'$. We see that $([\boldsymbol{K}_1, \boldsymbol{K}_2]_1, [\boldsymbol{K}_1, \boldsymbol{K}_2]_2)$ can be efficiently simulated with a distribution identical to the one expected by $\mathcal{B}$. Next, given a solution $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma, \boldsymbol{\alpha}', \boldsymbol{\beta}', \gamma')$ such that $\boldsymbol{K}_1^\top\boldsymbol{\alpha} + \boldsymbol{K}_1^\top\boldsymbol{\beta} + \gamma + \boldsymbol{K}_2^\top\boldsymbol{\alpha}' + \boldsymbol{K}_2^\top\boldsymbol{\beta}' + \gamma' = 0$, one can find $e_1, e_2, e_3 \in \mathbb{Z}_p$ and $e'_1, e'_2, e'_3 \in \mathbb{Z}_p$ such that:

$$
\begin{aligned}
0 =& (t\boldsymbol{r} + \boldsymbol{s})^\top\boldsymbol{\alpha}(t\boldsymbol{r} + \boldsymbol{s}) + (t\boldsymbol{r} + \boldsymbol{s})^\top\boldsymbol{\beta} + \gamma + (t\boldsymbol{r}' + \boldsymbol{s}')^\top\boldsymbol{\alpha}'(t'\boldsymbol{r}' + \boldsymbol{s}') + \\
& (t'\boldsymbol{r}' + \boldsymbol{s}')^\top\boldsymbol{\beta}' + \gamma' \\
=& t^2(\boldsymbol{r}^\top\boldsymbol{\alpha}\boldsymbol{r}) + t(\boldsymbol{r}^\top\boldsymbol{\alpha}\boldsymbol{s} + \boldsymbol{s}^\top\boldsymbol{\alpha}\boldsymbol{r} + \boldsymbol{r}^\top\boldsymbol{\beta}) + t'^2(\boldsymbol{r}'^\top\boldsymbol{\alpha}'\boldsymbol{r}') + t'(\boldsymbol{r}'^\top\boldsymbol{\alpha}'\boldsymbol{s}' + \\
& \boldsymbol{s}'^\top\boldsymbol{\alpha}'\boldsymbol{r}' + \boldsymbol{r}'^\top\boldsymbol{\beta}') + (\boldsymbol{s}^\top\boldsymbol{\alpha}\boldsymbol{s} + \boldsymbol{s}^\top\boldsymbol{\beta} + \gamma) + (\boldsymbol{s}'^\top\boldsymbol{\alpha}'\boldsymbol{s}' + \boldsymbol{s}'^\top\boldsymbol{\beta}' + \gamma') \\
=& e_1 t^2 + e_2 t + e_3 + e'_1 t'^2 + e'_2 t' + e'_3.
\end{aligned}
$$

In particular, with overwhelming probability (over the choice of $\boldsymbol{s}$ and $\boldsymbol{s}'$ that are information theoretically hidden from $\mathcal{B}$'s view) $e_3, e'_3 \neq 0$. From this solution, $\mathcal{C}$ can solve the system and extract t and t'. □

Theorem 5. *Let $\Pi'_{\mathsf{asy\text{-}up}}$ be an updatable knowledge sound asymmetric QA-NIZK argument for linear subspaces from Theorem 4. Then the asymmetric QA-NIZK Π'_{asy} [25] in Fig. 1 is computationally knowledge sound.*

Proof. We prove it by contradiction in a way that we assume that there is an adversary $\mathcal{A}_{\Pi'_{\mathsf{asy}}}$ that breaks the knowledge soundness of the asymmetric QA-NIZK Π'_{asy} . Then, one can build an adversary $\mathcal{B}_{\Pi'_{\mathsf{asy\text{-}up}}}$ against the updatable asymmetric QA-NIZK $\Pi'_{\mathsf{asy\text{-}up}}$ who runs $\mathcal{A}_{\Pi'_{\mathsf{asy}}}$ as a subroutine algorithm and breaks the updatable knowledge soundness $\Pi'_{\mathsf{asy\text{-}up}}$. More precisely, given $([\boldsymbol{M}]_1, [\boldsymbol{N}]_2, \mathsf{crs})$ where $\mathsf{crs} = ([\boldsymbol{A}, \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{P}_2]_2, [\boldsymbol{A}, \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{P}_1]_1)$, the adversary $\mathcal{B}_{\Pi'_{\mathsf{asy\text{-}up}}}$ sets $\mathsf{crs}_{\Pi'_{\mathsf{asy}}} := ([\boldsymbol{A}, \boldsymbol{C}_2, \boldsymbol{P}_2]_2, [\boldsymbol{A}, \boldsymbol{C}_1, \boldsymbol{P}_1]_1)$ and sends it to $\mathcal{A}_{\Pi'_{\mathsf{asy}}}$ and returns back a valid $\pi_{\Pi'_{\mathsf{asy}}}$ for $([\boldsymbol{y}]_1, [\boldsymbol{x}]_2) \notin \mathcal{L}_{[\boldsymbol{M}]_1, [\boldsymbol{N}]_2}$. This concludes the proof. □

5 Discussion and Future Work

In this paper we investigate QA-NIZKs in the full subversion setting via the updatable CRS model. In particular, we analyse the security of the most efficient asymmetric QA-NIZK Π'_{asy} by González et al. [25] for $k = 1, 2$ (when $\mathcal{D}_k = \mathcal{U}_k$) and for arbitrary k (when $\mathcal{D}_k \in \{\mathcal{L}_k, \mathcal{IL}_k, \mathcal{C}_k, \mathcal{SC}_k\}$), when the CRS is full subverted and propose an updatable version of these QA-NIZKs. Since in practice, due to increased efficiency, one is mostly interested in shorter proof size (smaller

k as proofs are of size $2k$) and thus even when relying on $\mathcal{D}_k = \mathcal{U}_k$ the focus on schemes for $k = 1, 2$ is most reasonable. Especially as for these values of k one obtains constructions from the most common standard assumptions. But from a theoretical point of view, it is interesting to construct an updatable version of asymmetric QA-NIZK (or even symmetric QA-NIZK) for arbitrary $k > 2$ even in the case of $\mathcal{D}_k = \mathcal{U}_k$. Here, the main obstacle is to design a general (efficient) version of the algorithm $\mathsf{isinvertible}(\cdot, \cdot)$ [3] for checking the invertibility of a matrix of group elements of size k (see Fig. 3).

We recall that our main motivation in this work was to fill the existing gap towards obtaining an updatable version of LegoSNARK [13]. An interesting question is to study how one can combine all the existing updatable CRS building blocks, i.e., updatable CRS SNARKs and QA-NIZKs as well as our construction, to construct an updatable LegoSNARK framework and investigate its efficiency.

Acknowledgements. This work was in part funded by the European Union's Horizon 2020 research and innovation programme under grant agreement n°871473 (KRAKEN) and n°890456 (SLOTMACHINE), and by the Austrian Science Fund (FWF) and netidee SCIENCE under grant agreement P31621-N38 (PROFET). This work has received funding by the German Federal Ministry of Education and Research BMBF (grant 16KISK038, project 6GEM).

References

1. Abdolmaleki, B., Baghery, K., Lipmaa, H., Siim, J., Zajac, M.: UC-secure CRS generation for SNARKs. In: Buchmann, J., Nitaj, A., eddine Rachidi, T. (eds.) AFRICACRYPT 19. LNCS, vol. 11627, pp. 99–117. Springer, Heidelberg (Jul 2019). https://doi.org/10.1007/978-3-030-23696-0_6
2. Abdolmaleki, B., Baghery, K., Lipmaa, H., Zajac, M.: A subversion-resistant SNARK. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part III. LNCS, vol. 10626, pp. 3–33. Springer, Heidelberg (Dec 2017). https://doi.org/10.1007/978-3-319-70700-6_1
3. Abdolmaleki, B., Lipmaa, H., Siim, J., Zajac, M.: On QA-NIZK in the BPK model. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part I. LNCS, vol. 12110, pp. 590–620. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45374-9_20
4. Abdolmaleki, B., Lipmaa, H., Siim, J., Zajac, M.: On subversion-resistant snarks. J. Cryptol. **34**(3), 17 (2021)
5. Abdolmaleki, B., Ramacher, S., Slamanig, D.: Lift-and-shift: Obtaining simulation extractable subversion and updatable SNARKs generically. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020. pp. 1987–2005. ACM Press (Nov 2020). https://doi.org/10.1145/3372297.3417228
6. Abdolmaleki, B., Slamanig, D.: Subversion-resistant quasi-adaptive NIZK and applications to modular zk-snarks. In: Conti, M., Stevens, M., Krenn, S. (eds.) Cryptology and Network Security - 20th International Conference, CANS 2021, Vienna, Austria, December 13-15, 2021, Proceedings. Lecture Notes in Computer Science, vol. 13099, pp. 492–512. Springer (2021)

7. Abe, M., Jutla, C.S., Ohkubo, M., Pan, J., Roy, A., Wang, Y.: Shorter QA-NIZK and SPS with tighter security. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 669–699. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-34618-8_23
8. Abe, M., Jutla, C.S., Ohkubo, M., Roy, A.: Improved (almost) tightly-secure simulation-sound QA-NIZK with applications. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part I. LNCS, vol. 11272, pp. 627–656. Springer, Heidelberg (Dec 2018). https://doi.org/10.1007/978-3-030-03326-2_21
9. Bellare, M., Fuchsbauer, G., Scafuro, A.: NIZKs with an untrusted CRS: Security in the face of parameter subversion. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 777–804. Springer, Heidelberg (Dec 2016). https://doi.org/10.1007/978-3-662-53890-6_26
10. Ben-Sasson, E., Chiesa, A., Green, M., Tromer, E., Virza, M.: Secure sampling of public parameters for succinct zero knowledge proofs. In: 2015 IEEE Symposium on Security and Privacy. pp. 287–304. IEEE (2015)
11. Bowe, S., Gabizon, A., Green, M.D.: A multi-party protocol for constructing the public parameters of the pinocchio zk-snark. Cryptology ePrint Archive, Report 2017/602 (2017), https://eprint.iacr.org/2017/602
12. Campanelli, M., Faonio, A., Fiore, D., Querol, A., Rodríguez, H.: Lunar: A toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part III. LNCS, vol. 13092, pp. 3–33. Springer, Heidelberg (Dec 2021). https://doi.org/10.1007/978-3-030-92078-4_1
13. Campanelli, M., Fiore, D., Querol, A.: LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 2075–2092. ACM Press (Nov 2019). https://doi.org/10.1145/3319535.3339820
14. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, N., Ward, N.P.: Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 738–768. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45721-1_26
15. Damgård, I.: Towards practical public key systems secure against chosen ciphertext attacks. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 445–456. Springer, Heidelberg (Aug 1992). https://doi.org/10.1007/3-540-46766-1_36
16. Danezis, G., Fournet, C., Groth, J., Kohlweiss, M.: Square span programs with applications to succinct NIZK arguments. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 532–550. Springer, Heidelberg (Dec 2014). https://doi.org/10.1007/978-3-662-45611-8_28
17. Daza, V., González, A., Pindado, Z., Ràfols, C., Silva, J.: Shorter quadratic QA-NIZK proofs. In: Lin, D., Sako, K. (eds.) PKC 2019, Part I. LNCS, vol. 11442, pp. 314–343. Springer, Heidelberg (Apr 2019). https://doi.org/10.1007/978-3-030-17253-4_11
18. Daza, V., Ràfols, C., Zacharakis, A.: Updateable inner product argument with logarithmic verifier and applications. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part I. LNCS, vol. 12110, pp. 527–557. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45374-9_18
19. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for Diffie-Hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 129–147. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40084-1_8

20. Fuchsbauer, G.: Subversion-zero-knowledge SNARKs. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part I. LNCS, vol. 10769, pp. 315–347. Springer, Heidelberg (Mar 2018). https://doi.org/10.1007/978-3-319-76578-5_11
21. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 33–62. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96881-0_2
22. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953 (2019), https://eprint.iacr.org/2019/953
23. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645. Springer, Heidelberg (May 2013). https://doi.org/10.1007/978-3-642-38348-9_37
24. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on computing **18**(1), 186–208 (1989)
25. González, A., Hevia, A., Ràfols, C.: QA-NIZK arguments in asymmetric groups: New tools and new constructions. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part I. LNCS, vol. 9452, pp. 605–629. Springer, Heidelberg (Nov / Dec 2015). https://doi.org/10.1007/978-3-662-48797-6_25
26. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (Dec 2006). https://doi.org/10.1007/11935230_29
27. Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340. Springer, Heidelberg (Dec 2010). https://doi.org/10.1007/978-3-642-17373-8_19
28. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 305–326. Springer, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_11
29. Groth, J., Kohlweiss, M., Maller, M., Meiklejohn, S., Miers, I.: Updatable and universal common reference strings with applications to zk-SNARKs. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 698–728. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96878-0_24
30. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (May / Jun 2006). https://doi.org/10.1007/11761679_21
31. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (Apr 2008). https://doi.org/10.1007/978-3-540-78967-3_24
32. Jutla, C.S., Roy, A.: Shorter quasi-adaptive NIZK proofs for linear subspaces. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 1–20. Springer, Heidelberg (Dec 2013). https://doi.org/10.1007/978-3-642-42033-7_1
33. Jutla, C.S., Roy, A.: Switching lemma for bilinear tests and constant-size NIZK proofs for linear subspaces. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 295–312. Springer, Heidelberg (Aug 2014). https://doi.org/10.1007/978-3-662-44381-1_17

34. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: 24th ACM STOC. pp. 723–732. ACM Press (May 1992). https://doi.org/10.1145/129712.129782
35. Kiltz, E., Wee, H.: Quasi-adaptive NIZK for linear subspaces revisited. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 101–128. Springer, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46803-6_4
36. Kohlweiss, M., Maller, M., Siim, J., Volkhov, M.: Snarky ceremonies. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part III. LNCS, vol. 13092, pp. 98–127. Springer, Heidelberg (Dec 2021). https://doi.org/10.1007/978-3-030-92078-4_4
37. Libert, B., Peters, T., Joye, M., Yung, M.: Non-malleability from malleability: Simulation-sound quasi-adaptive NIZK proofs and CCA2-secure encryption from homomorphic signatures. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 514–532. Springer, Heidelberg (May 2014). https://doi.org/10.1007/978-3-642-55220-5_29
38. Libert, B., Peters, T., Joye, M., Yung, M.: Compactly hiding linear spans - tightly secure constant-size simulation-sound QA-NIZK proofs and applications. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part I. LNCS, vol. 9452, pp. 681–707. Springer, Heidelberg (Nov / Dec 2015). https://doi.org/10.1007/978-3-662-48797-6_28
39. Lipmaa, H.: Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 169–189. Springer, Heidelberg (Mar 2012). https://doi.org/10.1007/978-3-642-28914-9_10
40. Lipmaa, H.: Key-and-argument-updatable QA-NIZKs. In: Galdi, C., Kolesnikov, V. (eds.) SCN 20. LNCS, vol. 12238, pp. 645–669. Springer, Heidelberg (Sep 2020). https://doi.org/10.1007/978-3-030-57990-6_32
41. Lipmaa, H.: A unified framework for non-universal snarks. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) Public-Key Cryptography - PKC 2022 - 25th IACR International Conference on Practice and Theory of Public-Key Cryptography, Virtual Event, March 8-11, 2022, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13177, pp. 553–583. Springer (2022)
42. Lipmaa, H., Siim, J., Zajac, M.: Counting vampires: From univariate sumcheck to updatable ZK-SNARK. Cryptology ePrint Archive, Report 2022/406 (2022), https://eprint.iacr.org/2022/406
43. Maller, M., Bowe, S., Kohlweiss, M., Meiklejohn, S.: Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 2111–2128. ACM Press (Nov 2019). https://doi.org/10.1145/3319535.3339817
44. Morillo, P., Ràfols, C., Villar, J.L.: The kernel matrix Diffie-Hellman assumption. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 729–758. Springer, Heidelberg (Dec 2016). https://doi.org/10.1007/978-3-662-53887-6_27
45. Ràfols, C., Silva, J.: QA-NIZK arguments of same opening for bilateral commitments. In: Nitaj, A., Youssef, A.M. (eds.) AFRICACRYPT 20. LNCS, vol. 12174, pp. 3–23. Springer, Heidelberg (Jul 2020). https://doi.org/10.1007/978-3-030-51938-4_1
46. Ràfols, C., Zapico, A.: An algebraic framework for universal and updatable SNARKs. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol.

12825, pp. 774–804. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84242-0_27

A CRS-update Hiding Proof

Lemma 2 ([40], Lemma 6.). *Assume that* $\boldsymbol{K}, \boldsymbol{K}_{\mathsf{int}} \in \mathcal{D}_{\boldsymbol{K}}$ *and* $\boldsymbol{A}, \boldsymbol{A}_{\mathsf{int}} \in \mathcal{D}_{\boldsymbol{A}}$ *, where* $\mathcal{D}_{\boldsymbol{K}}$ *and* $\mathcal{D}_{\boldsymbol{A}}$ *satisfy the following conditions for random variables* Y_1 *and* Y_2*: (i) if* $Y_1, Y_2 \leftarrow\!\!\$\ \mathcal{D}_{\boldsymbol{K}}$ *then* $Y_1 + Y_2 \in \mathcal{D}_{\boldsymbol{K}}$*, and (ii) if* $Y_1, Y_2 \leftarrow\!\!\$\ \mathcal{D}_{\boldsymbol{A}}$ *then* $Y_1 \cdot Y_2 \in \mathcal{D}_{\boldsymbol{A}}$*. Then,* $\Pi'_{\mathsf{asy\text{-}up}}$ *is key-update hiding.*

Proof. Since $\mathsf{Vcrs}(\mathsf{crs}, \mathsf{lpar}) = 1$, thus, crs is honestly created, $\boldsymbol{C} = \boldsymbol{K}\boldsymbol{A}$. So, $\boldsymbol{C}_{\mathsf{up}} = \boldsymbol{C}\boldsymbol{A}_{\mathsf{int}} + \boldsymbol{K}_{\mathsf{int}}\boldsymbol{A}\boldsymbol{A}_{\mathsf{int}} = (\boldsymbol{K} + \boldsymbol{K}_{\mathsf{int}})\boldsymbol{A}\boldsymbol{A}_{\mathsf{int}} = (\boldsymbol{K} + \boldsymbol{K}_{\mathsf{int}})\boldsymbol{A}_{\mathsf{up}} = \boldsymbol{K}_{\mathsf{up}}\boldsymbol{A}_{\mathsf{up}}$. Similarly holds for $\boldsymbol{P}$. Due to the assumption on $\mathcal{D}_{\boldsymbol{A}}$ and $\mathcal{D}_{\boldsymbol{K}}$, crs and $\mathsf{crs}_{\mathsf{up}}$ come from the same distribution.