Key-and-Argument-Updatable QA-NIZKs

Helger Lipmaa[0000−0001−8393−6821]
firstname.lastname@gmail.com

1 Simula UiB, Bergen, Norway
2 University of Tartu, Tartu, Estonia

Abstract. There are several new efficient approaches to decreasing trust in the CRS creators for NIZK proofs in the CRS model. Recently, Groth *et al.* (CRYPTO 2018) defined the notion of NIZK with updatable CRS (*updatable NIZK*) and described an updatable SNARK. We consider the same problem in the case of QA-NIZKs. We also define an important new property: we require that after updating the CRS, one should be able to update a previously generated argument to a new argument that is valid with the new CRS. We propose a general definitional framework for *key-and-argument-updatable QA-NIZKs.* After that, we describe a key-and-argument-updatable version of the most efficient known QA-NIZK for linear subspaces by Kiltz and Wee. Importantly, for obtaining soundness, it suffices to update a universal public key that just consists of a matrix drawn from a KerMDH-hard distribution and thus can be shared by *any pairing-based application that relies on the same hardness assumption.* After specializing the universal public key to the concrete language parameter, one can use the proposed key-and-argument updating algorithms to continue updating to strengthen the soundness guarantee.

Keywords: BPK model, CRS model, QA-NIZK, subversion security, updatable public key, updatable argument

1 Introduction

SNARKs. Zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs, [DL08,Gro10,Lip12,Lip13,GGPR13,PHGR13,Gro16,GM17]) have become widely researched and deployed, in particular because of their applicability in verifiable computation [PHGR13] and anonymous cryptocurrencies [DFKP13,BCG$^+$14]. However, due to a well-known impossibility result [GW11], the soundness of SNARKs can only be based on non-falsifiable assumptions [Nao03]. Moreover, a new security concern has arisen recently.

Most of the existing pairing-based zk-SNARKs are defined in the common reference string (CRS, [BFM88]) model assuming the existence of a trusted third party TTP that samples a CRS from the correct distribution and does not leak any trapdoors. The existence of such a TTP is often a too strong assumption. Recently, several efficient approaches have been proposed to decrease trust in the CRS creation, like the use of multi-party CRS generation [BCG$^+$15,BGG17,BGM17,ABL$^+$19] and the notion of subversion-resistant zero-knowledge (Sub-ZK) SNARKs [BFS16,ABLZ17,Fuc18]. A Sub-ZK SNARK guarantees to the prover P the zero-knowledge property even if the CRS was maliciously created, as long as P checks that a public algorithm $\mathsf{V}_{\mathsf{crs}}$ accepts the CRS. Existing Sub-ZK SNARKs use a non-falsifiable assumption to extract from a $\mathsf{V}_{\mathsf{crs}}$-accepted CRS its trapdoor td. Then, one simulates CRS by using td. Since one cannot at the same time achieve subversion-soundness and Sub-ZK [BFS16], Sub-ZK SNARKs only achieve the usual (knowledge-)soundness property.

Groth *et al.* [GKM$^+$18] defined CRS updating and showed how to implement it in the case of SNARKs. The main idea behind it is that given a CRS based on some trapdoor td, one can update the CRS to a new CRS crs' based on some trapdoor td'. Updating can be repeated many times, obtaining a sequence

$$\mathsf{crs}_0 \to \mathsf{crs}_1 \to \mathsf{crs}_2 \to \ldots \to \mathsf{crs}_n$$

of CRSs, updated by some parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$. The SNARK will be sound (and the CRS will be correctly distributed) if at least one of the CRS updaters was honest; this allows one to get soundness (if at least one updater was honest) and zero-knowledge (without any assumption on the updaters). At some moment, the prover will create an argument. The verifier only accepts when she trusts some updater *at the moment of argument creation.* Groth *et al.* [GKM$^+$18] constructed an updatable SNARK with a quadratically-long universal CRS (valid for all circuits of the given size) and linearly-long specialized CRS (constructed from the universal CRS when a circuit is fixed and actually used to create an argument). The subject of updatable SNARKs has become very popular, with many new schemes proposed within two years [MBKM19,GWC19,CHM$^+$20,DRZ20,ARS20].

As a drawback, since the argument itself is not updatable, the CRS updates after an argument has been created cannot be taken to account; in particular, it means that the verifier has to signal to the prover that she is ready to trust the argument created at some moment (since the CRS at that moment has been updated by a verifier-trusted party).

QA-NIZKs. Starting from the seminal work of Jutla and Roy [JR13], QA-NIZKs has been a (quite different) alternative research direction as compared to SNARKs with quite different applications and quite different underlying techniques. Intuitively, in QA-NIZKs, the prover and the verifier have access to an honestly generated language parameter lpar, and then prove a statement with respect to a lpar-dependent language $\mathcal{L}_{\mathsf{lpar}}$. Like SNARKs, QA-NIZKs offer succinct arguments and super-efficient verification. On the positive side, contemporary QA-NIZKs are based on very standard assumptions like MDDH [EHK$^+$13] (e.g., DDH) and KerMDH [MRV16] (e.g., CDH). On the negative side, efficient and succinct QA-NIZKs are known only for a much smaller class of languages like the language of linear subspaces [JR13,LPJY14,JR14,KW15,ABP15,LPJY15] and some related languages, including the language of quadratic relations [GHR15] and shuffles [GR16]. [DGP$^+$19] proposed a non-succinct QA-NIZK for SSP;

non-succinctness is expected due to known impossibility results [GW11]. QA-NIZKs have applications in the construction of efficient cryptographic primitives (like KDM-CCA2-secure encryption, IND-CCA2-secure IBE, and UC-secure commitments, [JR13], and linearly homomorphic structure-preserving signatures [KW15]) based on standard assumptions.

Abdolmaleki *et al.* [ALSZ20] recently studied subversion-resistant QA-NIZKs. They showed that Sub-ZK in the CRS model is equal to the known notion of no-auxiliary-string non-black-box zero knowledge [GO94] in the significantly weaker Bare Public Key (BPK, [CGGM00,MR01]) model. Like [ALSZ20], we will thus use the notions of "Sub-ZK" and "no-auxiliary-string non-black-box zero knowledge" interchangeably, but we will usually explicitly mention the trust model (CRS, BPK, plain). Due to known impossibility results [GO94], this provides a simple proof that one has to use no-auxiliary-string non-black-box NIZK to construct argument systems for non-trivial languages in the BPK model. Abdolmaleki *et al.* [ALSZ20] also proposed an efficient $\mathsf{V}_{\mathsf{crs}}$ algorithm for the most efficient known QA-NIZK Π'_{as} for linear subspaces by Kiltz and Wee [KW15] and proved that the resulting construction achieves Sub-ZK in the BPK model. In fact, they went one step further: they considered the case when the language parameter lpar itself is subverted and showed how to achieve soundness and Sub-ZK even in this case. More precisely, they defined separate Sub-ZK (*black-box* Sub-ZK, given an *honestly generated* lpar) and persistent Sub-ZK (*non-black-box* Sub-ZK, given a *subverted* lpar) properties, and showed that these two properties are in fact incomparable.[3]

The proof methods of [ALSZ20] are quite non-trivial. In particular, [ALSZ20] proved the Sub-ZK property under a new tautological $\mathsf{KW\text{-}KE}$ knowledge assumption, and then showed that $\mathsf{KW\text{-}KE}$ is secure in the subversion generic bilinear group model of [BFS16,ABLZ17] (named GBGM with hashing in [BFS16]). Especially the latter proof is quite complicated. Moreover, they proved so-called Sub-PAR soundness (soundness in the case lpar is subverted, but the CRS is untrusted) under natural but little-studied, non-falsifiable, interactive non-adaptive assumptions [Gjø06,Lip10].

As in the case of SNARKs, it is natural to ask if efficient QA-NIZKs like Π'_{as} can be updated. No published research has been done on this topic.

Our Contributions. We define updatable QA-NIZK by roughly following the definitional guidelines of [GKM+18] for updatable SNARKs. However, we make two significant changes to the model itself. The second change (the ability to update also the argument) is especially important, allowing for new applications. No succinct argument-updatable NIZKs, either SNARKs or QA-NIZKs, were known before. Crucially, for updating Π'_{as}, it is sufficient to update a single public key $\text{PK} = [\bar{\boldsymbol{A}}]_2$, where $[\boldsymbol{A}]_2$ is a KerMDH-hard matrix, and $[\bar{\boldsymbol{A}}]_2$ denotes

[3] To show incomparability, [ALSZ20] constructed a contrived persistent Sub-ZK argument where the simulator first uses a knowledge assumption on the language parameter to extract witness, and then uses this witness as input to the honest prover. Such an argument is obviously not black-box Sub-ZK.

its upmost square submatrix. This means that one can share the same updatable universal public key $\textsc{pk}$ between any applications where the security of one party relies on the (bare) public key, created by another party.

Firstly, since QA-NIZK security definitions differ from SNARKs (with lpar having an important and distinct role), we redefine updatable versions of completeness, soundness, and (persistent) zero-knowledge. We add to them the natural requirement of hiding (an updated key and a fresh key are indistinguishable). We will follow the framework of [ALSZ20] by relying on Sub-ZK QA-NIZK in the BPK model. According to [ALSZ20], the prover and the verifier of a QA-NIZK argument share a (possibly malformed) generated language parameter lpar together with a (possibly malformed) verifier's public key $\textsc{pk}$. We add the key-updating and update-verification algorithms with the corresponding security requirements: key-update completeness, key-update hiding, strong key-update hiding, key-update soundness, and key-update (persistent) Sub-ZK.

Secondly, and more importantly, we add to the QA-NIZK the ability to update the argument. That is, given a $\textsc{pk}$ and an argument π constructed while using $\textsc{pk}$, one can then update $\textsc{pk}$ (to a new key $\textsc{pk}'$) and π to a new valid argument π' (corresponding to $\textsc{pk}'$). There are two different ways to update the argument. First, the honest argument updater must know the witness (but no secret information about the key update). Second, the argument-update simulator must know some secret key-update trapdoor (but he does not have to know the witness). We require that these two different ways of updating are indistinguishable; thus, updating does not leak information about the witness.

Argument-updating has various non-obvious implications. The key-updater can, knowing the key-update trapdoor, simulate the argument-update; this means that we will not get soundness unless at least one of the argument-creators or argument-updaters does not collaborate with the corresponding key-creator or key-updater. (See Section 4.) One can obtain different trust models by handling the updating process differently. For example, the honest argument-updater can have additional anonymity since it is not revealed which of the argument-updaters knows the witness. On the other hand, if there exists at least one update such that the corresponding key-updater and argument-updater do not trust each other, we are guaranteed that one of the argument-updater actually "knows" the witness and thus the statement is true.

We will give rigorous security definitions for *key-and-argument-updatable* QA-NIZKs, requiring them to satisfy argument-update completeness, argument-update hiding, strong argument-update hiding, argument-update soundness, and argument-update (persistent) Sub-ZK. We use the terminology of convolution semigroups while arguing about the hiding properties; since this terminology is very natural, we argue that one should use it more widely in the context of updatable cryptographic protocols. We prove that argument-update soundness and argument-update (persistent) Sub-ZK follow from simpler security requirements and thus do not have to be proven anew in the case of each new QA-NIZK.

We implement the provided security definitions, by proposing an updatable version $\Pi_{\mathsf{bpk}}^{\mathsf{upd}}$ of the Kiltz-Wee QA-NIZK Π'_{as} for linear subspace [KW15]. Our construction uses crucially the fact that all operations (like the public key generation and proving) in Π'_{as} consist of only linear operations. Hence, the new update-related algorithms are relatively simple but still non-trivial. For example, we update some elements of the public key additively and some elements multiplicatively.[4] This is a major difference to (known to us) SNARKs, where one only does multiplicative updating. Interestingly, we update the argument π by adding to it an (honestly created or simulated) argument when using the appropriately defined"difference" $\widehat{\mathrm{PK}}$ of the new and the old public key as a one-time public key. This is why we can update arguments without the knowledge of either the witness or the trapdoor of either PK or PK′; instead, it suffices to use the trapdoor corresponding to the concrete update.

We prove that $\Pi^{\mathsf{upd}}_{\mathsf{bpk}}$ satisfies all defined security properties, and in particular, that (like the non-updatable Sub-ZK QA-NIZK of [ALSZ20]) it is Sub-PAR sound either under the KerMDH$^{\mathrm{dl}}$ assumption [MRV16,ALSZ20] or the SKerMDH$^{\mathrm{dl}}$ assumption [GHR15,ALSZ20] (depending on the values of the system parameters) and argument-update persistent Sub-ZK under the $\mathsf{KW\text{-}KE}$ assumption of [ALSZ20]. If the language parameter is trusted, then as in [ALSZ20], a falsifiable assumption (either KerMDH or SKerMDH) suffices for soundness. As in [ALSZ20], one can even obtain Sub-PAR knowledge-soundness.

The hiding properties rely on certain, well-defined, properties of the distributions of the secret key $\boldsymbol{K}$ and $\bar{\boldsymbol{A}}$: namely, these distributions are assumed to be (essentially) stable [Kle08]. We hope that this observation motivates study of other stable distributions for cryptographic purposes. In particular, stable distributions seem to be natural in the setting of various updatable primitives.

Updatable Universal Public Key. The goal of updatability [GKM+18] is to protect soundness in the case PK may be subverted since Sub-ZK can be obtained by running the public algorithm V_{pk} [ABLZ17]. In Π'_{as}, soundness is guaranteed by one of the elements of PK (namely, $[\bar{\boldsymbol{A}}]_2$, see Fig. 2) coming from a KerMDH-hard distribution, and another element $[\boldsymbol{C}]_2$ being correctly computed from $[\bar{\boldsymbol{A}}]_2$. Since the latter can be verified by the public-key verification algorithm, it suffices only to update $[\bar{\boldsymbol{A}}]_2$. Then, $[\bar{\boldsymbol{A}}]_2$ will be a "universal public key" [GKM+18] for all possible language parameters in all applications that rely on the concrete (i.e., using the same distribution) KerMDH *assumption.*

The possibility to rely just on $[\bar{\boldsymbol{A}}]_2$ is a major difference with known updatable SNARKs where the universal key is quite complex, and each universal key of length $\Theta(n)$ can only be used for circuits of size $\leq n$.

Importantly, one is not restricted to QA-NIZK: *any* application that relies on KerMDH and where it suffices to know $[\bar{\boldsymbol{A}}]_2$ (instead of $[\boldsymbol{A}]_2$) can use the same matrix $[\bar{\boldsymbol{A}}]_2$. A standard example is the 1-Lin [BBS04,EHK+13] distribution $\mathcal{L}_1 = \{\boldsymbol{A} = \left(\begin{smallmatrix} a \\ 1 \end{smallmatrix}\right) : a \leftarrow_{\$} \mathbb{Z}_p\}$. We emphasize that the possibility to rely just on

[4] Groth *et al.* [GKM+18] proved that in the case of multiplicative updating, each element of PK must be a monomial in secret trapdoors. Since we update various elements of PK either multiplicatively or additively, it is unclear whether this impossibility result holds. We leave this is an interesting open question.

$[\bar{\boldsymbol{A}}]_2$ is a major difference with updatable SNARKs [GKM$^+$18,MBKM19] where the universal key is quite complex and each universal key of length $\Omega(n)$ can only be used for circuits of size $\leq n$. See Section 7.

Some of the proofs are postponed to Appendix B.

2 Preliminaries

We denote the empty string by ϵ. Let PPT denote probabilistic polynomial-time and let $\lambda \in \mathbb{N}$ be the security parameter. All adversaries will be stateful. For an algorithm $\mathcal{A}$, let range$(\mathcal{A})$ be the range of $\mathcal{A}$, i.e., the set of valid outputs of $\mathcal{A}$, let $\mathsf{RND}_\lambda(\mathcal{A})$ denote the random tape of $\mathcal{A}$ (for fixed λ), and let $r \leftarrow_\$ \mathsf{RND}_\lambda(\mathcal{A})$ denote the uniformly random choice of the randomizer r from $\mathsf{RND}_\lambda(\mathcal{A})$. By $y \leftarrow \mathcal{A}(\mathsf{x}; r)$ we denote the fact that $\mathcal{A}$, given an input x and a randomizer r, outputs y. When we use this notation, then r represents the full random tape of $\mathcal{A}$. We denote by $\mathsf{negl}(\lambda)$ an arbitrary negligible function, and by $\mathsf{poly}(\lambda)$ an arbitrary polynomial function. We write $a \approx_\lambda b$ if $|a - b| = \mathsf{negl}(\lambda)$.

Probability theory. Let μ and ν be probability measures on $(\mathbb{Z}, 2^{\mathbb{Z}})$. The *convolution* [Kle08, Def. 14.46] $\mu * \nu$ is defined as the probability measure on $(\mathbb{Z}, 2^{\mathbb{Z}})$ such that $(\mu * \nu)(\{n\}) = \sum_{m=-\infty}^{\infty} \mu(\{m\})\nu(\{n-m\})$. We define the nth *convolution power* recursively by $\mu^{*1} = \mu$ and $\mu^{*(n+1)} = \mu^{*n} * \mu$. Let $I \subset [0, \infty)$ be a semigroup. A family $\nu = (\nu_t : t \in I)$ of probability distributions on $\mathbb{R}^d$ is called a *convolution semigroup* [Kle08, Def. 14.46] if $\nu_{s+t} = \nu_s * \nu_t$ holds for all $s, t \in I$. Let $X_1, X_2, \ldots$ be i.i.d random variables with distribution μ. The distribution μ is called *stable* [Kle08, Def. 16.20] with index $\alpha \in (0, 2]$ if $X_1 + \ldots + X_n = n^{1/\alpha} X_n$ for all $n \in \mathbb{N}$.

Bilinear pairings. A bilinear group generator $\mathsf{Pgen}(1^\lambda)$ returns $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ 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 pairing. We require the bilinear pairing to be Type-3 [GPS08], i.e., we assume that there is no efficient isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. We use the bracket notation of [EHK$^+$13], e.g., we write $[a]_\iota$ to denote $a g_\iota$ where $a \in \mathbb{Z}_p$ and g_ι is a fixed generator of $\mathbb{G}_\iota$. We denote $\hat{e}([a]_1, [b]_2)$ as $[a]_1 \cdot [b]_2$. We use the bracket notation freely together with matrix notation, e.g., $\boldsymbol{AB} = \boldsymbol{C}$ iff $[\boldsymbol{A}]_1 \cdot [\boldsymbol{B}]_2 = [\boldsymbol{C}]_T$.

Matrix Diffie-Hellman Assumptions. Kernel Matrix Diffie-Hellman Assumption (KerMDH) is a well-known assumption family formally introduced in [MRV16] and say, used in [KW15] to show the soundness of their QA-NIZK argument system for linear subspaces. The KerMDH assumption states that for a matrix $\boldsymbol{A}$ from some well-defined distribution, it is difficult to find a representation of a vector that belongs to the kernel of $\boldsymbol{A}^\top$ provided that the matrix is given in exponents only, i.e., as $[\boldsymbol{A}]_\iota$.

For fixed p, denote by $\mathcal{D}_{\ell k}$ a probability distribution over matrices in $\mathbb{Z}_p^{\ell \times k}$, where $\ell > k$. We assume that $\mathcal{D}_{\ell k}$ outputs matrices $\boldsymbol{A}$ where the upper $k \times k$ submatrix $\bar{\boldsymbol{A}}$ is always invertible. Let $\bar{\mathcal{D}}_{\ell k}$ that outputs $\bar{\boldsymbol{A}}$, where $\boldsymbol{A}$ is sampled from $\mathcal{D}_{\ell k}$. When $\ell = k + 1$, we denote $\mathcal{D}_k = \mathcal{D}_{\ell k}$. In Appendix A, we define five commonly used distributions. There, we also define assumptions, needed by the constructions in [KW15,ALSZ20], and thus, also by the current paper.

Bare Public Key (BPK) Model. In the BPK model [CGGM00,MR01], parties have access to a public file F, a polynomial-size collection of records $(id, \textsc{pk}_{id})$, where id is a string identifying a party (e.g., a verifier), and $\textsc{pk}_{id}$ is her (alleged) public key. In a typical zero-knowledge protocol in the BPK model, a key-owning party $\mathcal{P}_{id}$ works in two stages. In stage one (the *key-generation stage*), on input a security parameter 1^λ and randomizer r, $\mathcal{P}_{id}$ outputs a public key $\textsc{pk}_{id}$ and stores the corresponding secret key $\textsc{sk}_{id}$. We assume the *no-auxiliary-string BPK* model where from this it follows that $\mathcal{P}_{id}$ actually created $\textsc{pk}_{id}$. After that, F will include $(id, \textsc{pk}_{id})$. In stage two, each party has access to F, while $\mathcal{P}_{id}$ has possible access to $\textsc{sk}_{id}$ (however, the latter will be not required by us). It is commonly assumed that only the verifier of a NIZK argument system in the BPK model has a public key [MR01].

No-Auxiliary-String Non-Black-Box (Sub-ZK) QA-NIZK in the BPK Model. The original QA-NIZK security definitions, [JR13], were given in the CRS model. The following description of QA-NIZKs in the BPK model is taken from [ALSZ20], and we refer to [ALSZ20] for additional discussion. Since black-box [MR01,APV05] and even auxiliary-input non-black-box [GO94,Wee07] NIZK in the BPK model is impossible for non-trivial languages, we will give an explicit definition of no-auxiliary-string non-black-box NIZK. As explained in [ALSZ20], no-auxiliary-string non-black-box zero knowledge in the BPK model is the same as Sub-ZK [BFS16] in the CRS model.

As in [BFS16], we assume that the system parameters p are generated deterministically from λ; in particular, the choice of p could not be subverted. A QA-NIZK argument system enables one to prove membership in a language defined by a relation $\mathbf{R}_{\mathsf{lpar}} = \{(\mathsf{x}, \mathsf{w})\}$, which in turn is completely determined by a parameter lpar sampled (in the honest case) from a distribution $\mathcal{D}_{\mathsf{p}}$. We will assume implicitly that lpar contains p and thus not include p as an argument to algorithms that also input lpar; recall that we assumed that p cannot be subverted. A distribution $\mathcal{D}_{\mathsf{p}}$ on $\mathcal{L}_{\mathsf{lpar}}$ is *witness-sampleable* [JR13] if there exists a PPT algorithm $\mathcal{D}'_{\mathsf{p}}$ that samples $(\mathsf{lpar}, \mathsf{td}_{\mathsf{lpar}}) \in \mathbf{R}_{\mathsf{p}}$ such that lpar is distributed according to $\mathcal{D}_{\mathsf{p}}$.

The zero-knowledge simulator is usually required to be a single (non-black-box) PPT algorithm that works for the whole collection of relations $\mathbf{R}_{\mathsf{p}} = \{\mathbf{R}_{\mathsf{lpar}}\}_{\mathsf{lpar} \in \mathrm{im}(\mathcal{D}_{\mathsf{p}})}$; i.e., one requires *uniform simulation* (see [JR13] for a discussion). Following [ABLZ17], we accompany the universal simulator with an adversary-dependent extractor. We assume Sim also works in the case when one cannot efficiently establish whether $\mathsf{lpar} \in \mathrm{im}(\mathcal{D}_\mathsf{p})$. Sim is not allowed to create new lpar or PK but receive them as an input.

A tuple of PPT algorithms $\Pi = (\mathsf{Pgen}, \mathsf{K_{bpk}}, \mathsf{V_{par}}, \mathsf{V_{pk}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ is a *no-auxiliary-string non-black-box zero knowledge (Sub-ZK) QA-NIZK argument system* in the BPK model for a set of witness-relations $\mathbf{R}_\mathsf{p} = \{\mathbf{R}_\mathsf{lpar}\}_{\mathsf{lpar} \in \mathrm{Supp}(\mathcal{D}_\mathsf{p})}$, if the following Items i, ii, iv and v hold. Π is a *Sub-ZK QA-NIZK argument of knowledge*, if additionally Item iii holds. Here, Pgen is the parameter generation algorithm, $\mathsf{K_{bpk}}$ is the public key generation algorithm, $\mathsf{V_{par}}$ is the lpar-verification algorithm, $\mathsf{V_{pk}}$ is the public-key verification algorithm, P is the prover, V is the verifier, and Sim is the simulator. We abbreviate quasi-adaptive to QA.

(i) **Perfect Completeness:** for any λ, PPT $\mathcal{A}$, given $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, $\mathsf{lpar} \leftarrow_\$ \mathcal{D}_\mathsf{p}$, $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{K_{bpk}}(\mathsf{lpar})$, $(\mathsf{x}, \mathsf{w}) \leftarrow \mathcal{A}(\mathrm{PK})$, $\pi \leftarrow \mathsf{P}(\mathsf{lpar}, \mathrm{PK}, \mathsf{x}, \mathsf{w})$, it holds that $\mathsf{V_{par}}(\mathsf{lpar}) = 1$ and $\mathsf{V_{pk}}(\mathsf{lpar}, \mathrm{PK}) = 1$ and $((\mathsf{x}, \mathsf{w}) \not\in \mathbf{R}_\mathsf{lpar} \vee \mathsf{V}(\mathsf{lpar}, \mathrm{PK}, \mathsf{x}, \pi) = 1)$.

(ii) **Computational QA Sub-PAR Soundness:** $\forall$ PPT $\mathcal{A}$, given $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, $\mathsf{lpar} \leftarrow \mathcal{A}(\mathsf{p})$, $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{K_{bpk}}(\mathsf{lpar})$, and $(\mathsf{x}, \pi) \leftarrow \mathcal{A}(\mathrm{PK})$, the following holds with negligible probability: $\mathsf{V_{par}}(\mathsf{lpar}) = 1 \wedge \mathsf{V}(\mathsf{lpar}, \mathrm{PK}, \mathsf{x}, \pi) = 1 \wedge \neg(\exists \mathsf{w} : \mathbf{R}_\mathsf{lpar}(\mathsf{x}, \mathsf{w}) = 1))$.

(iii) **Computational QA Sub-PAR Knowledge-Soundness:** for any PPT $\mathcal{A}$, there exist a PPT extractor $\mathsf{Ext}_\mathcal{A}$, s.t. given $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, $r \leftarrow_\$ \mathsf{RND}_\lambda(\mathcal{A})$, $\mathsf{lpar} \leftarrow \mathcal{A}(\mathsf{p}; r)$, $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{K_{bpk}}(\mathsf{lpar})$, $(\mathsf{x}, \pi) \leftarrow \mathcal{A}(\mathrm{PK}; r)$, $\mathsf{w} \leftarrow \mathsf{Ext}_\mathcal{A}(\mathsf{p}, \mathrm{PK}; r)$, the following holds with a negligible probability: $\mathsf{V_{par}}(\mathsf{lpar}) = 1 \wedge \mathsf{V}(\mathsf{lpar}, \mathrm{PK}, \mathsf{x}, \pi) = 1 \wedge \mathbf{R}_\mathsf{lpar}(\mathsf{x}, \mathsf{w}) = 0$.

(iv) **Statistical Zero Knowledge:** for any unbounded $\mathcal{A}$, $|\varepsilon_0^{zk} - \varepsilon_1^{zk}| \approx_\lambda 0$, where ε_b^{zk} is the probability that given $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, $\mathsf{lpar} \leftarrow \mathcal{D}_\mathsf{p}$, $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{K_{bpk}}(\mathsf{lpar})$, it holds that $\mathcal{A}^{\mathsf{O}_b(\cdot,\cdot)}(\mathsf{lpar}, \mathrm{PK}) = 1$.
The oracle $\mathsf{O}_0(\mathsf{x}, \mathsf{w})$ returns $\perp$ (reject) if $(\mathsf{x}, \mathsf{w}) \not\in \mathbf{R}_\mathsf{lpar}$, and otherwise it returns $\mathsf{P}(\mathsf{lpar}, \mathrm{PK}, \mathsf{x}, \mathsf{w})$. Similarly, $\mathsf{O}_1(\mathsf{x}, \mathsf{w})$ returns $\perp$ (reject) if $(\mathsf{x}, \mathsf{w}) \not\in \mathbf{R}_\mathsf{lpar}$, and otherwise it returns $\mathsf{Sim}(\mathsf{lpar}, \mathrm{PK}, \mathrm{SK}, \mathsf{x})$.

(v) **Statistical Persistent Zero Knowledge:** for any PPT subverter Z, there exists a PPT extractor $\mathsf{Ext_Z}$, s.t. for any computationally unbounded adversary $\mathcal{A}$, $|\varepsilon_0^{zk} - \varepsilon_1^{zk}| \approx_\lambda 0$, where ε_b^{zk} is the probability that given $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, $r \leftarrow_\$ \mathsf{RND}_\lambda(\mathsf{Z})$, $(\mathsf{lpar}, \mathrm{PK}, \mathsf{aux}) \leftarrow \mathsf{Z}(\mathsf{p}; r)$, $\mathrm{SK} \leftarrow \mathsf{Ext_Z}(\mathsf{p}; r)$, the following holds with a negligible probability: $\mathsf{V_{par}}(\mathsf{lpar}) = 1 \wedge \mathsf{V_{pk}}(\mathsf{lpar}, \mathrm{PK}) = 1 \wedge \mathcal{A}^{\mathsf{O}_b(\cdot,\cdot)}(\mathsf{lpar}, \mathrm{PK}, \mathsf{aux}) = 1$.
The oracle $\mathsf{O}_0(\mathsf{x}, \mathsf{w})$ returns $\perp$ (reject) if $(\mathsf{x}, \mathsf{w}) \not\in \mathbf{R}_\mathsf{lpar}$, and otherwise it returns $\mathsf{P}(\mathsf{lpar}, \mathrm{PK}, \mathsf{x}, \mathsf{w})$. Similarly, $\mathsf{O}_1(\mathsf{x}, \mathsf{w})$ returns $\perp$ (reject) if $(\mathsf{x}, \mathsf{w}) \not\in \mathbf{R}_\mathsf{lpar}$, and otherwise it returns $\mathsf{Sim}(\mathsf{lpar}, \mathrm{PK}, \mathrm{SK}, \mathsf{x})$.

Π is *statistically no-auxiliary-string*[5] *non-black-box zero knowledge (Sub-ZK)* if it is both statistically zero-knowledge and statistically persistent zero-knowledge.

Kiltz-Wee QA-NIZK in the BPK model. Kiltz and Wee [KW15] described a very efficient QA-NIZK Π'_{as} for linear subspaces. Abdolmaleki *et al.* [ALSZ20]

[5] Auxiliary-string non-black-box ZK [GO94] means that definitions hold even if any $\mathsf{aux} \in \{0,1\}^{\mathsf{poly}(\lambda)}$ is given as an additional input to $\mathcal{A}$ and Z_{PK} (and $\mathsf{Ext_Z}$).

```
$\mathsf{Exp}^{\mathsf{zk}}_{\mathsf{Z},\mathcal{A}}(\mathsf{lpar})$
$r \leftarrow_{\$} \mathsf{RND}_\lambda(\mathsf{Z})$;
$(\textsc{pk}, \mathsf{aux}_\mathsf{Z}) \leftarrow \mathsf{Z}(\mathsf{lpar}; r)$;
$\textsc{sk} \leftarrow \mathsf{Ext}_\mathsf{Z}(\mathsf{lpar}; r)$;
$b \leftarrow_{\$} \{0,1\}$;
$b' \leftarrow \mathcal{A}^{\mathsf{O}_b(\cdot,\cdot)}(\mathsf{lpar}; \textsc{pk}, \mathsf{aux}_\mathsf{Z})$;
return $\mathsf{V}_{\mathsf{pk}}(\mathsf{lpar}; \textsc{pk}) = 1 \wedge b' = b$;

$\mathsf{O}_0(\mathsf{x},\mathsf{w})$:
if $(\mathsf{x},\mathsf{w}) \notin \mathbf{R}_{\mathsf{lpar}}$ then return $\bot$;
else return $\pi \leftarrow \mathsf{P}(\mathsf{lpar}, \textsc{pk}; \mathsf{x}, \mathsf{w})$; fi

$\mathsf{O}_1(\mathsf{x},\mathsf{w})$:
if $(\mathsf{x},\mathsf{w}) \notin \mathbf{R}_{\mathsf{lpar}}$ then return $\bot$;
else return $\pi_{\mathsf{Sim}} \leftarrow \mathsf{Sim}(\mathsf{lpar}, \textsc{pk}, \textsc{sk}; \mathsf{x})$; fi
```

Fig. 1. Experiment $\mathsf{Exp}^{\mathsf{zk}}_{\mathsf{Z},\mathcal{A}}(\mathsf{lpar})$

modified Π'_{as} and proved that the resulting QA-NIZK Π_{bpk} is Sub-ZK in the BPK model, assuming a novel KW-KE knowledge assumption. In addition, [ALSZ20] proved that the KW-KE assumption holds under a hash-knowledge assumption (HAK, [Lip19b]). The soundness of Π'_{as} holds in the BPK model under a suitable KerMDH assumption for any $k \geq 1$; one obtain optimal efficiency when $k = 1$.

The distribution $\mathcal{D}_k$ is *efficiently verifiable*, if there exists an algorithm $\mathsf{MATV}([\bar{\boldsymbol{A}}]_2)$ that outputs 1 if $\bar{\boldsymbol{A}}$ is invertible (recall that we assume that the matrix distribution is robust) and well-formed with respect to $\mathcal{D}_k$ and otherwise outputs 0. We refer to [ALSZ20] for the construction of MATV for common distributions. Fig. 2 describes the Sub-ZK QA-NIZK Π_{bpk} from [ALSZ20]. Here, as observed in [ALSZ20], the correctness of $[\boldsymbol{P}]_1$ is needed to guarantee zero knowledge and $[\bar{\boldsymbol{A}}, \boldsymbol{C}]_2$ is needed to guarantee soundness [ALSZ20]. Apart from restating Π'_{as} by using the terminology of the BPK model, Π_{bpk} differs from Π'_{as} only by having an additional entry $\textsc{pk}_{\mathsf{V_{pk}}}$ in the $\textsc{pk}$ and by including the $\mathsf{V_{pk}}$ algorithm. For the sake of completeness, we will next state the main security results of [ALSZ20]. The KW-KE assumption is defined in Appendix A.

Proposition 1 (Security of Π_{bpk}, [ALSZ20]). *Let Π_{bpk} be the QA-NIZK argument system for linear subspaces from Fig. 2. The following statements hold in the BPK model. Assume that $\mathcal{D}_\mathsf{p}$ is such that $\mathsf{V_{par}}$ is efficient. (i) Π_{bpk} is perfectly complete and perfectly zero-knowledge. (ii) If $(\mathcal{D}_\mathsf{p}, k, \mathcal{D}_k)$-$\mathsf{KW\text{-}KE}_{\mathbb{G}_1}$ holds relative to Pgen then Π_{bpk} is statistically persistent zero-knowledge. (iii) Assume $\mathcal{D}_k$ is efficiently verifiable (resp., $\mathcal{D}_k = \mathcal{U}_2$). If $\mathcal{D}_k$-KerMDH$^{\mathrm{dl}}$ (resp., $\mathcal{D}_k$-SKerMDH$^{\mathrm{dl}}$) holds relative to Pgen then Π_{bpk} is computationally quasi-adaptively Sub-PAR sound.*

3 Key-and-Argument-Updatable QA-NIZK: Definitions

Following [ALSZ20], we will consider QA-NIZK in the BPK model and thus with a public-key updating (and not CRS-updating like in [GKM$^+$18]) algorithm. Also, we allow updating of a previously created argument to one that corresponds to the new public key $\textsc{pk}$, obtaining what we will call a *key-and-argument-updatable* QA-NIZK. As in [GKM$^+$18], the updatable $\textsc{pk}$ and the corresponding secret key will be “shared” by more than one party. Thus, executing

```
$\mathsf{isinvertible}([\bar{\boldsymbol{A}}]_2, \mathrm{PK}_{\mathsf{Vpk}})$ // $\bar{\boldsymbol{A}} = (a_{ij})$
Check $\mathrm{PK}_{\mathsf{Vpk}} = [a^*_{11}, a^*_{12}]_1 \in \mathbb{G}_1^{1\times 2} \wedge [a^*_{11}]_1 \cdot [1]_2 = [1]_1 \cdot [a_{11}]_2 \wedge$
      $[a^*_{12}]_1 \cdot [1]_2 = [1]_1 \cdot [a_{12}]_2 \wedge [a^*_{11}]_1 \cdot [a_{22}]_2 - [a^*_{12}]_1 \cdot [a_{21}]_2 \neq [0]_T$;
```

```
$\mathsf{K_{bpk}}(\mathsf{lpar} := [\boldsymbol{M}]_1 \in \mathbb{G}_1^{n\times m})$: $\boldsymbol{A} \leftarrow_{\$} \mathcal{D}_k$; $\boldsymbol{K} \leftarrow_{\$} \mathbb{Z}_p^{n\times k}$; $[\boldsymbol{C}]_2 \leftarrow [\boldsymbol{K}\bar{\boldsymbol{A}}]_2 \in \mathbb{G}_2^{n\times k}$;
    $[\boldsymbol{P}]_1 \leftarrow [\boldsymbol{M}]_1^\top \boldsymbol{K} \in \mathbb{G}_1^{m\times k}$; if $\mathcal{D}_k$ is efficiently verifiable then $\mathrm{PK}_{\mathsf{Vpk}} \leftarrow \epsilon$;
    elseif $\mathcal{D}_k = \mathcal{U}_2$ then $\mathrm{PK}_{\mathsf{Vpk}} \leftarrow [a_{11}, a_{12}]_1$; fi ; $\mathrm{PK}_{\mathsf{snd}} \leftarrow [\bar{\boldsymbol{A}}, \boldsymbol{C}]_2$; $\mathrm{PK}_{\mathsf{zk}} \leftarrow [\boldsymbol{P}]_1$;
    $\mathrm{PK} \leftarrow (\mathrm{PK}_{\mathsf{snd}}, \mathrm{PK}_{\mathsf{zk}}, \mathrm{PK}_{\mathsf{Vpk}})$; $\mathrm{SK} \leftarrow \boldsymbol{K}$; return $(\mathrm{PK}, \mathrm{SK})$;
$\mathsf{P}([\boldsymbol{M}]_1, \mathbf{PK}, [\boldsymbol{y}]_1, \boldsymbol{w})$: return $[\boldsymbol{\pi}]_1 \leftarrow [\boldsymbol{P}]_1^\top \boldsymbol{w} \in \mathbb{G}_1^k$;
$\mathsf{Sim}([\boldsymbol{M}]_1, \mathbf{PK}, \mathbf{SK}, [\boldsymbol{y}]_1)$: // SK is extracted by using a knowledge assumption;
    return $[\boldsymbol{\pi}]_1 \leftarrow \boldsymbol{K}^\top [\boldsymbol{y}]_1 \in \mathbb{G}_1^k$;
$\mathsf{V}([\boldsymbol{M}]_1, \mathbf{PK}, [\boldsymbol{y}]_1, [\boldsymbol{\pi}]_1)$ : check that $[\boldsymbol{y}]_1^\top [\boldsymbol{C}]_2 = [\boldsymbol{\pi}]_1^\top [\bar{\boldsymbol{A}}]_2$; // $\in \mathbb{G}_T^{1\times k}$
$\mathsf{V_{pk}}([\boldsymbol{M}]_1, \mathbf{PK})$: Return 1 only if the following checks all succeed:
        $\mathrm{PK} = (\mathrm{PK}_{\mathsf{snd}}, \mathrm{PK}_{\mathsf{zk}}, \mathrm{PK}_{\mathsf{Vpk}}) \wedge \mathrm{PK}_{\mathsf{snd}} = [\bar{\boldsymbol{A}}, \boldsymbol{C}]_2 \wedge \mathrm{PK}_{\mathsf{zk}} = [\boldsymbol{P}]_1$;
        $[\boldsymbol{P}]_1 \in \mathbb{G}_1^{m\times k} \wedge [\bar{\boldsymbol{A}}]_2 \in \mathbb{G}_2^{k\times k} \wedge [\boldsymbol{C}]_2 \in \mathbb{G}_2^{n\times k}$;
    (*) $[\boldsymbol{M}]_1^\top [\boldsymbol{C}]_2 = [\boldsymbol{P}]_1 [\bar{\boldsymbol{A}}]_2$;
        if $\mathcal{D}_k$ is efficiently verifiable then $\mathsf{MATV}([\bar{\boldsymbol{A}}]_2)$;
        else check $\mathsf{isinvertible}([\bar{\boldsymbol{A}}]_2, \mathrm{PK}_{\mathsf{Vpk}})$; fi
```

Fig. 2. Sub-ZK QA-NIZK Π_{bpk} for $[\boldsymbol{y}]_1 = [\boldsymbol{M}]_1 \boldsymbol{w}$ in the BPK model, where either (1) $\mathcal{D}_k$ is efficiently verifiable or (2) $\mathcal{D}_k = \mathcal{U}_2$.

multiple updates of PK by independent parties means that the updated version of PK is not "created" solely by a single verifier. To achieve soundness, it suffices that V (or an entity trusted by her) was one of the parties involved in the creation or updating of PK. It even suffices if, up to the currently last available updated argument, at least one key-updater does not collaborate with the corresponding proof-updater.

This moves us out from designated-verifier arguments, typical for the BPK model, to (somewhat-)transferable arguments. The CRS model corresponds to the case where PK belongs to a universally trusted third party (TTP); updating the public key of the TTP by another party decreases trust requirements in the TTP. E.g., the PK can originally belong to the TTP, and then updated by two interested verifiers.

New Algorithms. An (argument-)updatable Sub-ZK QA-NIZK has the following additional PPT algorithms on top of $(\mathsf{Pgen}, \mathsf{K_{bpk}}, \mathsf{V_{pk}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$:

$\mathsf{K_{upd}}(\mathsf{lpar}, \mathbf{PK})$**:** a randomized *key updater* algorithm that, given an old PK, generates a new updated public key PK', and returns $(\mathrm{PK}', \widehat{\mathrm{SK}})$ where $\widehat{\mathrm{SK}}$ is a trapdoor corresponding to the PK-update.

$\mathsf{V_{Kupd}}(\mathsf{lpar}, \mathbf{PK}, \mathbf{PK}')$**:** a deterministic *key-update verifier* algorithm that, given PK and PK', verifies that PK' is a correct update of PK.

$\mathsf{P_{upd}}(\mathsf{lpar}, \mathbf{PK}, \mathbf{PK}'; \mathsf{x}, \mathsf{w}, \pi)$**:** a possibly randomized *argument-updater* algorithm that, given $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}_{\mathsf{lpar}}$, an argument π (made by using the old public key PK), and the updated public key PK', outputs an argument π' that corresponds to the updated public key PK'. $\mathsf{P}_{\mathsf{upd}}$ must be executable without the knowledge of either SK, SK' (the secret key corresponding to PK'), or any trapdoor $\widehat{\mathrm{SK}}$ about the update. Hence, $\mathsf{P}_{\mathsf{upd}}$ can be used to update either a prover-generated or a simulated argument, but only honestly, i.e., when the prover knows the witness.

$\mathsf{Sim}_{\mathsf{upd}}(\mathsf{lpar}, \widehat{\mathbf{SK}}; \mathsf{x}, \pi)$**:** a (randomized) *argument-update simulator* algorithm that, given an argument π (made with an old public key PK) and a PK-update trapdoor $\widehat{\mathrm{SK}}$ (corresponding to the update from π to π'), outputs an argument π' with an updated public key PK'. $\mathsf{Sim}_{\mathsf{upd}}$ is executed without the knowledge of either w, SK, or SK' (the secret keys corresponding to PK and PK', respectively). Thus, $\mathsf{Sim}_{\mathsf{upd}}$ can be used to update either a prover-generated or a simulated argument, but only when knowing the trapdoor $\widehat{\mathrm{SK}}$ of the key-update. $\mathsf{Sim}_{\mathsf{upd}}$ can have more inputs (like PK); in our constructions, we do not need them.

$\mathsf{V}_{\mathsf{Pupd}}(\mathsf{lpar}, \mathbf{PK}, \mathbf{PK}'; \mathsf{x}, \pi, \pi')$**:** a deterministic *argument-update verifier* algorithm that verifies that π' is a correct (updated either by $\mathsf{P}_{\mathsf{upd}}$ or $\mathsf{Sim}_{\mathsf{upd}}$ on correct inputs) update of π when PK was updated to PK'.

We require that there exists an efficient algorithm Comb that, on input $(\mathsf{lpar}; \mathrm{SK}, \widehat{\mathrm{SK}})$ (where SK is the secret key corresponding to PK and $\widehat{\mathrm{SK}}$ is the trapdoor of the update $\mathrm{PK} \Rightarrow \mathrm{PK}'$), returns SK' (the secret key corresponding to PK').

New Security Requirements. We introduce several new security requirements that accompany the new algorithms. They include requirements that guarantee that the standard definitions of completeness, (computational) soundness, and (statistical) zero-knowledge also hold after the key or the key-and-argument updates. We complete them with the various hiding requirements that guarantee that an updated key (and argument) are indistinguishable from the freshly generated key (and argument), assuming that either the pre-update key (and argument) were honestly created or the update was honest. While hiding is a natural security objective by itself, we will see that it allows us to get general reductions between soundness (and key/argument-update soundness) and Sub-ZK (and key/argument-update Sub-ZK).

We will consider two versions of argument-update soundness. Argument-update soundness (I) holds when PK was created honestly, but the updater is malicious, and argument-update soundness (II) holds when PK was created maliciously, but the updater is honest. Argument-update soundness (I) (resp., (II)) is defined in the case when PK and π were created honestly (resp., updated honestly) since it is impossible to get both subversion-soundness (which corresponds to the case both the PK creator and the updater are malicious) and zero-knowledge, [BFS16]. We want to guarantee soundness even in the case when only one of the key and argument updaters was honest, but various verification algorithms accept the updates of other updaters.

In the case of key-update Sub-ZK, we are interested in the case when only the public key has been updated, but the update could have been done maliciously. Since the argument is not updated, we require that an argument and simulated argument, given with the updated key (where both the old key and the key-update verify), are indistinguishable.

Similarly, in the case of argument-update Sub-ZK, the key update does not depend on the witness and may be done maliciously (possibly not by P). However, the argument is updated by P who uses the witness but does not have to know the key-update trapdoor $\widehat{\mathrm{SK}}$. This motivates the use of $\mathsf{K}_{\mathsf{upd}}$ and $\mathsf{Sim}_{\mathsf{upd}}$ in the update process in $\mathsf{Exp}^{\mathsf{au-pzk}}_{\mathsf{Z},\mathcal{A}}(\mathsf{lpar})$. Thus, key-update Sub-ZK and argument-update Sub-ZK are different notions and have to be handled separately.

We give all security notions for a single update; this results in simple reductions between these notions, and simple security proofs of the QA-NIZK. All notions can be composed over several updates, and the composed properties can then be proved by using standard hybrid arguments. We will omit further discussion, see [GKM+18] for more information. We divide the considerable number of definitions into completeness, hiding, soundness, and zero-knowledge sections.

Completeness.

Key-update completeness: $\forall\lambda$, $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, $\forall\mathsf{lpar} \in \mathcal{D}_{\mathsf{p}}$, $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{K}_{\mathsf{bpk}}(\mathsf{lpar})$, $(\mathrm{PK}', \widehat{\mathrm{SK}}) \leftarrow \mathsf{K}_{\mathsf{upd}}(\mathsf{lpar}, \mathrm{PK})$: $\mathsf{V}_{\mathsf{Kupd}}(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}') = 1$. Moreover, if $\mathsf{V}_{\mathsf{Kupd}}(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}') = 1$ then $\mathsf{V}_{\mathsf{pk}}(\mathsf{lpar}; \mathrm{PK}) = 1$ iff $\mathsf{V}_{\mathsf{pk}}(\mathsf{lpar}; \mathrm{PK}') = 1$.

Argument-update completeness: $\forall\lambda$, $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, $\forall\mathsf{lpar} \in \mathcal{D}_{\mathsf{p}}$, $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{K}_{\mathsf{bpk}}(\mathsf{lpar})$, $\forall(\mathsf{x}, \mathsf{w}) \in \mathbf{R}_{\mathsf{lpar}}$, $\pi \leftarrow \mathsf{P}(\mathsf{lpar}, \mathrm{PK}; \mathsf{x}, \mathsf{w})$, $(\mathrm{PK}', \widehat{\mathrm{SK}}) \leftarrow \mathsf{K}_{\mathsf{upd}}(\mathsf{lpar}, \mathrm{PK})$, $\pi' \leftarrow \mathsf{P}_{\mathsf{upd}}(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}'; \mathsf{x}, \mathsf{w}, \pi)$: $\mathsf{V}_{\mathsf{Pupd}}(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}'; \mathsf{x}, \pi, \pi') = 1$. Moreover, if $\mathsf{V}_{\mathsf{Kupd}}(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}') = 1$ and $\mathsf{V}_{\mathsf{Pupd}}(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}'; \mathsf{x}, \pi, \pi') = 1$ then $\mathsf{V}(\mathsf{lpar}, \mathrm{PK}; \mathsf{x}, \pi) = 1$ iff $\mathsf{V}(\mathsf{lpar}, \mathrm{PK}'; \mathsf{x}, \pi') = 1$.

Simulator-update completeness: $\forall\lambda$, $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, $\forall\mathsf{lpar} \in \mathcal{D}_{\mathsf{p}}$, $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{K}_{\mathsf{bpk}}(\mathsf{lpar})$, $\forall(\mathsf{x}, \mathsf{w}) \in \mathbf{R}_{\mathsf{lpar}}$, $\pi_{\mathsf{Sim}} \leftarrow \mathsf{Sim}(\mathsf{lpar}, \mathrm{PK}, \mathrm{SK}; \mathsf{x})$, $(\mathrm{PK}', \widehat{\mathrm{SK}}) \leftarrow \mathsf{K}_{\mathsf{upd}}(\mathsf{lpar}, \mathrm{PK})$, $\pi' \leftarrow \mathsf{Sim}_{\mathsf{upd}}(\mathsf{lpar}, \widehat{\mathrm{SK}}; \mathsf{x}, \pi)$: $\mathsf{V}_{\mathsf{Pupd}}(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}'; \mathsf{x}, \pi, \pi') = 1$. Moreover, if $\mathsf{V}_{\mathsf{Kupd}}(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}') = 1$ and $\mathsf{V}_{\mathsf{Pupd}}(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}'; \mathsf{x}, \pi, \pi') = 1$ then $\mathsf{V}(\mathsf{lpar}, \mathrm{PK}; \mathsf{x}, \pi) = 1$ iff $\mathsf{V}(\mathsf{lpar}, \mathrm{PK}'; \mathsf{x}, \pi') = 1$.

Hiding.

Key-update hiding: $\forall\lambda$, $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, $\forall\mathsf{lpar} \in \mathcal{D}_{\mathsf{p}}$: if $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{K}_{\mathsf{bpk}}(\mathsf{lpar})$ and $(\mathrm{PK}', \widehat{\mathrm{SK}}) \leftarrow \mathsf{K}_{\mathsf{upd}}(\mathsf{lpar}, \mathrm{PK})$, then $\mathrm{PK}' \approx_\lambda \mathsf{K}_{\mathsf{bpk}}(\mathsf{lpar})$.

Strong key-update hiding: $\forall\lambda$, $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, $\forall\mathsf{lpar} \in \mathcal{D}_{\mathsf{p}}$, $\forall(\mathrm{PK}, \mathrm{PK}')$: $\mathrm{PK}' \approx_\lambda \mathsf{K}_{\mathsf{bpk}}(\mathsf{lpar})$ holds if either

1. the old public key was honestly generated and the key-update verifies: $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{K}_{\mathsf{bpk}}(\mathsf{lpar})$ and $\mathsf{V}_{\mathsf{Kupd}}(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}') = 1$, or
2. the old public key verifies and the key-update was honest: $\mathsf{V}_{\mathsf{pk}}(\mathsf{lpar}, \mathrm{PK}) = 1$ and $(\mathrm{PK}', \widehat{\mathrm{SK}}) \leftarrow \mathsf{K}_{\mathsf{upd}}(\mathsf{lpar}, \mathrm{PK})$.

Argument-update hiding: $\forall\lambda$, $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, $\forall\mathsf{lpar} \in \mathcal{D}_{\mathsf{p}}$, $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{K}_{\mathsf{bpk}}$ (lpar), $(\mathrm{PK}', \widehat{\mathrm{SK}}) \leftarrow \mathsf{K}_{\mathsf{upd}}(\mathsf{lpar}, \mathrm{PK})$, $\pi \leftarrow \mathsf{P}(\mathsf{lpar}, \mathrm{PK}; \mathsf{x}, \mathsf{w})$, $\pi_{\mathsf{Sim}} \leftarrow \mathsf{Sim}(\mathsf{lpar},$

$\mathrm{PK}, \mathrm{SK}; \mathsf{x})$, $\pi' \leftarrow \mathsf{P}_{\mathsf{upd}}(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}'; \mathsf{x}, \mathsf{w}, \pi)$, $\pi'_{\mathsf{Sim}} \leftarrow \mathsf{Sim}_{\mathsf{upd}}(\mathsf{lpar}, \widehat{\mathrm{SK}}; \mathsf{x}, \pi)$: $\pi' \approx_\lambda \mathsf{P}(\mathsf{lpar}, \mathrm{PK}'; \mathsf{x}, \mathsf{w})$ and $\pi'_{\mathsf{Sim}} \approx_\lambda \mathsf{Sim}(\mathsf{lpar}, \mathrm{PK}', \mathrm{SK}'; \mathsf{x})$.

Strong argument-update hiding: $\forall \lambda$, $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, $\forall \mathsf{lpar} \in \mathcal{D}_{\mathsf{p}}$, $\forall (\mathsf{x}, \mathsf{w}) \in \mathbf{R}_{\mathsf{lpar}}$, $\forall (\mathrm{PK}, \mathrm{PK}'; \mathsf{x}, \pi, \pi')$: $\mathrm{PK}' \approx_\lambda \mathsf{K}_{\mathsf{bpk}}(\mathsf{lpar})$, $\pi' \approx_\lambda \mathsf{P}(\mathsf{lpar}, \mathrm{PK}'; \mathsf{x}, \mathsf{w})$, and $\pi'_{\mathsf{Sim}} \approx_\lambda \mathsf{Sim}(\mathsf{lpar}, \mathrm{PK}', \mathrm{SK}'; \mathsf{x})$ hold if either

(i) the old public key and argument were honestly generated and the updates verify: $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{K}_{\mathsf{bpk}}(\mathsf{lpar})$, $\pi \leftarrow \mathsf{P}(\mathsf{lpar}, \mathrm{PK}; \mathsf{x}, \mathsf{w})$, $\pi_{\mathsf{Sim}} \leftarrow \mathsf{Sim}(\mathsf{lpar}, \mathrm{PK}, \mathrm{SK}; \mathsf{x})$, $\mathsf{V}_{\mathsf{Kupd}}(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}') = 1$, $\mathsf{V}_{\mathsf{Pupd}}(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}'; \mathsf{x}, \pi, \pi') = 1$, and $\mathsf{V}_{\mathsf{Pupd}}(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}'; \mathsf{x}, \pi_{\mathsf{Sim}}, \pi'_{\mathsf{Sim}}) = 1$, or

(ii) the old public key and argument verify and the updates were honestly generated: $\mathsf{V}_{\mathsf{pk}}(\mathsf{lpar}, \mathrm{PK}) = 1$, $\mathsf{V}(\mathsf{lpar}, \mathrm{PK}; \mathsf{x}, \pi) = 1$, $\mathsf{V}(\mathsf{lpar}, \mathrm{PK}; \mathsf{x}, \pi_{\mathsf{Sim}}) = 1$, $(\mathrm{PK}', \widehat{\mathrm{SK}}) \leftarrow \mathsf{K}_{\mathsf{upd}}(\mathsf{lpar}, \mathrm{PK})$, $\pi' \leftarrow \mathsf{P}_{\mathsf{upd}}(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}'; \mathsf{x}, \mathsf{w}, \pi)$, and $\pi'_{\mathsf{Sim}} \leftarrow \mathsf{Sim}_{\mathsf{upd}}(\mathsf{lpar}, \widehat{\mathrm{SK}}; \mathsf{x}, \pi_{\mathsf{Sim}})$.

Soundness. Here, we abbreviate quasi-adaptive to QA.

(Computational QA) Sub-PAR key-update soundness (I): for any PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{sndku1}}_{\mathcal{A},\Pi}(\lambda) \approx_\lambda 0$, where $\mathsf{Adv}^{\mathrm{sndku1}}_{\mathcal{A},\Pi}(\lambda)$ is the probability that given $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, $\mathsf{lpar} \leftarrow_\$ \mathcal{A}(\mathsf{p}); (\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{K}_{\mathsf{bpk}}(\mathsf{lpar})$, $(\mathrm{PK}', \mathsf{x}, \pi') \leftarrow \mathcal{A}(\mathrm{PK})$, the following holds: $\mathsf{V}_{\mathsf{par}}(\mathsf{lpar}) = 1 \wedge \mathsf{V}_{\mathsf{Kupd}}(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}') = 1 \wedge \mathsf{V}(\mathsf{lpar}, \mathrm{PK}'; \mathsf{x}, \pi') = 1 \wedge \neg(\exists \mathsf{w} : \mathbf{R}_{\mathsf{lpar}}(\mathsf{x}, \mathsf{w}) = 1)$.

(Computational QA) Sub-PAR key-update soundness (II): for any PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{sndku2}}_{\mathcal{A},\Pi}(\lambda) \approx_\lambda 0$, where $\mathsf{Adv}^{\mathrm{sndku2}}_{\mathcal{A},\Pi}(\lambda)$ is the probability that given $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, $(\mathsf{lpar}, \mathrm{PK}) \leftarrow_\$ \mathcal{A}(\mathsf{p})$, $(\mathrm{PK}', \widehat{\mathrm{SK}}) \leftarrow \mathsf{K}_{\mathsf{upd}}(\mathsf{lpar}, \mathrm{PK})$, $\pi' \leftarrow \mathcal{A}(\mathrm{PK}')$, the following holds: $\mathsf{V}_{\mathsf{par}}(\mathsf{lpar}) = 1 \wedge \mathsf{V}_{\mathsf{pk}}(\mathsf{lpar}, \mathrm{PK}) = 1 \wedge \mathsf{V}(\mathsf{lpar}, \mathrm{PK}'; \mathsf{x}, \pi') = 1 \wedge \neg(\exists \mathsf{w} : \mathbf{R}_{\mathsf{lpar}}(\mathsf{x}, \mathsf{w}) = 1)$.

(Computational QA) Sub-PAR key-update soundness: iff both Sub-PAR key-update soundness (I) and (II) hold.

(Computational QA) argument-update soundness (I): for any PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{sndpu1}}_{\mathcal{A},\Pi}(\lambda) \approx_\lambda 0$, where $\mathsf{Adv}^{\mathrm{sndpu1}}_{\mathcal{A},\Pi}(\lambda)$ is the probability that given $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \mathsf{lpar} \leftarrow_\$ \mathcal{A}(\mathsf{p}); (\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{K}_{\mathsf{bpk}}(\mathsf{lpar})$, $(\mathrm{PK}', \mathsf{x}, \pi, \pi') \leftarrow \mathcal{A}(\mathrm{PK})$, the following holds: $\mathsf{V}_{\mathsf{par}}(\mathsf{lpar}) = 1 \wedge \mathsf{V}_{\mathsf{Kupd}}(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}') = 1 \wedge \mathsf{V}_{\mathsf{Pupd}}(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}'; \mathsf{x}, \pi, \pi') = 1 \wedge \mathsf{V}(\mathsf{lpar}, \mathrm{PK}; \mathsf{x}, \pi) = 1 \wedge \neg(\exists \mathsf{w} : \mathbf{R}_{\mathsf{lpar}}(\mathsf{x}, \mathsf{w}) = 1)$.

(Computational QA) Sub-PAR argument-update soundness (II): for any PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{sndpu2}}_{\mathcal{A},\Pi}(\lambda) \approx_\lambda 0$, where $\mathsf{Adv}^{\mathrm{sndpu2}}_{\mathcal{A},\Pi}(\lambda)$ is the probability that given $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, $(\mathsf{lpar}, \mathrm{PK}, \pi) \leftarrow_\$ \mathcal{A}(\mathsf{p})$, $(\mathrm{PK}', \widehat{\mathrm{SK}}) \leftarrow \mathsf{K}_{\mathsf{upd}}(\mathsf{lpar}, \mathrm{PK})$, $\pi' \leftarrow \mathsf{P}_{\mathsf{upd}}(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}'; \mathsf{x}, \mathsf{w}, \pi)$, the following holds: $\mathsf{V}_{\mathsf{par}}(\mathsf{lpar}) = 1 \wedge \mathsf{V}_{\mathsf{pk}}(\mathsf{lpar}, \mathrm{PK}) = 1 \wedge \mathsf{V}(\mathsf{lpar}, \mathrm{PK}; \mathsf{x}, \pi) = 1 \wedge \neg(\exists \mathsf{w} : \mathbf{R}_{\mathsf{lpar}}(\mathsf{x}, \mathsf{w}))$.

(Computational QA) Sub-PAR argument-update soundness: iff both Sub-PAR argument-update soundness (I) and Sub-PAR argument-update soundness (II) hold.

Zero-Knowledge. Here, all experiments are described in Fig. 3.

```
$\mathsf{Exp}^{\mathsf{ku-zk}}_{\mathcal{A}}(\mathsf{lpar})$
$\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \mathsf{lpar} \leftarrow \mathcal{D}_{\mathsf{p}};$
$(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{K}_{\mathsf{bpk}}(\mathsf{lpar});$
$(\mathrm{PK}', \widehat{\mathrm{SK}}) \leftarrow \mathsf{K}_{\mathsf{upd}}(\mathsf{lpar}, \mathrm{PK});$
$\mathrm{SK}' \leftarrow \mathsf{Comb}(\mathsf{lpar}; \mathrm{SK}, \widehat{\mathrm{SK}});$
$b \leftarrow_\$ \{0, 1\};$
$b' \leftarrow \mathcal{A}^{\mathsf{O}^k_b(\cdot,\cdot)}(\mathsf{lpar}; \mathrm{PK}, \mathrm{PK}');$
**return** $\mathsf{V}_{\mathsf{Kupd}}(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}') = 1 \wedge$
    $b' = b;$

$\mathsf{O}^k_0(\mathsf{x}, \mathsf{w})$:
**if** $(\mathsf{x}, \mathsf{w}) \not\in \mathbf{R}_{\mathsf{lpar}}$ **then return** $\bot$;
**else** $\pi' \leftarrow \mathsf{P}(\mathsf{lpar}, \mathrm{PK}'; \mathsf{x}, \mathsf{w});$
    **return** $\pi'$; **fi**

$\mathsf{O}^a_0(\mathsf{x}, \mathsf{w})$:
**if** $(\mathsf{x}, \mathsf{w}) \not\in \mathbf{R}_{\mathsf{lpar}}$ **then return** $\bot$;
**else** $\pi \leftarrow \mathsf{P}(\mathsf{lpar}, \mathrm{PK}; \mathsf{x}, \mathsf{w});$
    $\pi' \leftarrow \mathsf{P}_{\mathsf{upd}}(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}'; \mathsf{x}, \mathsf{w}, \pi);$
    **return** $(\pi, \pi')$; **fi**
```

```
$\mathsf{Exp}^{\mathsf{ku-pzk}}_{\mathsf{Z},\mathcal{A}}(\mathsf{lpar})$
$\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); r \leftarrow_\$ \mathsf{RND}_\lambda(\mathsf{Z});$
$(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}', \mathsf{aux}_{\mathsf{Z}}) \leftarrow \mathsf{Z}(\mathsf{p}; r);$
$(\mathrm{SK}, \widehat{\mathrm{SK}}) \leftarrow \mathsf{Ext}_{\mathsf{Z}}(\mathsf{p}; r);$
$\mathrm{SK}' \leftarrow \mathsf{Comb}(\mathsf{lpar}; \mathrm{SK}, \widehat{\mathrm{SK}});$
$b \leftarrow_\$ \{0, 1\};$
$b' \leftarrow \mathcal{A}^{\mathsf{O}^k_b(\cdot,\cdot)}(\mathsf{lpar}; \mathrm{PK}, \mathrm{PK}', \mathsf{aux}_{\mathsf{Z}});$
**return** $\mathsf{V}_{\mathsf{par}}(\mathsf{lpar}) = 1 \wedge \mathsf{V}_{\mathsf{pk}}(\mathsf{lpar}; \mathrm{PK}) = 1 \wedge$
    $\mathsf{V}_{\mathsf{Kupd}}(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}') = 1 \wedge b' = b;$

$\mathsf{O}^k_1(\mathsf{x}, \mathsf{w})$:
**if** $(\mathsf{x}, \mathsf{w}) \not\in \mathbf{R}_{\mathsf{lpar}}$ **then return** $\bot$;
**else** $\pi'_{\mathsf{Sim}} \leftarrow \mathsf{Sim}(\mathsf{lpar}, \mathrm{PK}', \mathrm{SK}'; \mathsf{x});$
    **return** $\pi'_{\mathsf{Sim}}$; **fi**

$\mathsf{O}^a_1(\mathsf{x}, \mathsf{w})$:
**if** $(\mathsf{x}, \mathsf{w}) \not\in \mathbf{R}_{\mathsf{lpar}}$ **then return** $\bot$;
**else** $\pi_{\mathsf{Sim}} \leftarrow \mathsf{Sim}(\mathsf{lpar}, \mathrm{PK}, \mathrm{SK}; \mathsf{x});$
    $\pi'_{\mathsf{Sim}} \leftarrow \mathsf{Sim}_{\mathsf{upd}}(\mathsf{lpar}, \widehat{\mathrm{SK}}; \mathsf{x}, \pi_{\mathsf{Sim}});$
    **return** $(\pi_{\mathsf{Sim}}, \pi'_{\mathsf{Sim}})$; **fi**
```

Fig. 3. Zero-knowledge experiments. Experiments $\mathsf{Exp}^{\mathsf{ku-zk}}_{\mathsf{Z},\mathcal{A}}(\mathsf{lpar})$ and $\mathsf{Exp}^{\mathsf{ku-pzk}}_{\mathsf{Z},\mathcal{A}}(\mathsf{lpar})$ are described first. Experiments $\mathsf{Exp}^{\mathsf{au-zk}}_{\mathsf{Z},\mathcal{A}}(\mathsf{lpar})$ and $\mathsf{Exp}^{\mathsf{au-pzk}}_{\mathsf{Z},\mathcal{A}}(\mathsf{lpar})$ are like $\mathsf{Exp}^{\mathsf{ku-zk}}_{\mathsf{Z},\mathcal{A}}(\mathsf{lpar})$ and $\mathsf{Exp}^{\mathsf{ku-pzk}}_{\mathsf{Z},\mathcal{A}}(\mathsf{lpar})$, except the adversary has access to oracle O^a_b instead of O^k_b.

(Statistical) key-update ZK: for any computationally unbounded $\mathcal{A}$, $|\mathsf{Exp}^{\mathsf{ku-zk}}_{\mathsf{Z},\mathcal{A}}(\mathsf{lpar}) - 1/2| \approx_\lambda 0$.

(Statistical) argument-update ZK: for any computationally unbounded $\mathcal{A}$, $|\mathsf{Exp}^{\mathsf{au-zk}}_{\mathsf{Z},\mathcal{A}}(\mathsf{lpar}) - 1/2| \approx_\lambda 0$.

(Statistical) key-update persistent Sub-ZK: for any PPT subverter Z there exists a PPT $\mathsf{Ext}_{\mathsf{Z}}$, such that for any computationally unbounded $\mathcal{A}$, $|\mathsf{Exp}^{\mathsf{ku-pzk}}_{\mathsf{Z},\mathcal{A}}(\mathsf{lpar}) - 1/2| \approx_\lambda 0$.

(Statistical) argument-update persistent Sub-ZK: for any PPT subverter Z there exists a PPT $\mathsf{Ext}_{\mathsf{Z}}$, such that for any computationally unbounded $\mathcal{A}$, $|\mathsf{Exp}^{\mathsf{au-pzk}}_{\mathsf{Z},\mathcal{A}}(\mathsf{lpar}) - 1/2| \approx_\lambda 0$.

Argument-updatable variants of ZK are stronger than key-updatable variants. Our constructions satisfy the stronger definitions, but for the sake of completeness, it is interesting to consider also the weaker notions.

We will now show that the key-update soundness, argument-update soundness, key-update Sub-ZK, and argument-update Sub-ZK properties follow from simpler security requirements. Hence, in the case of a concrete updatable QA-NIZK, it will suffice to prove computational soundness, Sub-ZK, (key-update and argument-update) completeness and strong (key-update and argument-update) hiding. Dependency between security properties is summarized in Table 1.

Table 1. Relations between security requirements due to Lemmas 1 to 4

Requirement	Follows from
Sub-PAR key-update soundness	key-update complete, Sub-PAR sound, strong key-update hiding
Sub-PAR argument-update soundness	argument-update complete, Sub-PAR sound, strong key-update hiding, strong argument-update hiding
(Persistent) key-update Sub-ZK	key-update complete, (persistent) Sub-ZK
(Persistent) argument-update Sub-ZK	key-update complete, (persistent) Sub-ZK

Lemma 1. *Assume Π is a Sub-PAR sound and strongly key-update hiding non-interactive argument system. (i) Π is Sub-PAR key-update sound (I). (ii) If Π is additionally key-update complete, then Π is Sub-PAR key-update sound (II).*

Proof. **(i)** By strong key-update hiding, $\textsc{pk}'$ comes from the correct distribution. Thus, by Sub-PAR soundness, it is computationally hard to come up with an acceptable argument π' unless x belongs to the language.

(ii) From key-update completeness it follows that in the definition of Sub-PAR key-update soundness (II), we can replace the condition $\mathsf{V}_{\mathsf{pk}}(\mathsf{lpar}, \textsc{pk}) = 1$ with the condition $\mathsf{V}_{\mathsf{pk}}(\mathsf{lpar}, \textsc{pk}') = 1$. Because of the strong key-update hiding, $\textsc{pk}'$ is indistinguishable from an honestly generated $\textsc{pk}'$. From Sub-PAR soundness, we now obtain Sub-PAR key-update soundness (II). □

Lemma 2. *Assume Π is a Sub-PAR sound non-interactive argument system. (i) Π is Sub-PAR argument-update sound (I). (ii) If Π is also argument-update complete, strongly key-update hiding, and strongly argument-update hiding, then Π is Sub-PAR argument-update sound (II).*

Proof. **(i)** Let $\mathcal{A}$ be an adversary against Sub-PAR argument-update soundness (I). We construct the following adversary $\mathcal{B}$ against Sub-PAR soundness. If $\mathcal{A}$ returns lpar, $\mathcal{B}$ returns the same lpar. After the generation of $\textsc{pk}$, $\mathcal{B}(\textsc{pk})$ obtains $(\textsc{pk}'; \mathsf{x}, \pi, \pi') \leftarrow \mathcal{A}(\textsc{pk})$ and returns (x, π). Clearly, if $\mathcal{A}$ is successful then V accepts π with honestly chosen $\textsc{pk}$ but $\mathsf{x} \notin \mathcal{L}_{\mathsf{lpar}}$. Thus, $\mathcal{B}$ is successful.

(ii) From argument-update completeness, it follows that in the definition of Sub-PAR argument-update soundness (II), we can replace the condition $\mathsf{V}(\mathsf{lpar}, \textsc{pk}; \mathsf{x}, \pi) = 1$ with the condition $\mathsf{V}(\mathsf{lpar}, \textsc{pk}'; \mathsf{x}, \pi') = 1$. Because of the strong key-update hiding, $\textsc{pk}'$ is indistinguishable from an honestly generated $\textsc{pk}'$. Because of the strong argument-update hiding, π' is indistinguishable from an honestly generated argument given $\textsc{pk}'$. From Sub-PAR soundness, we get Sub-PAR argument-update soundness. □

Lemma 3. *Assume Π is a key-update complete non-interactive argument system. (i) if Π is zero-knowledge then Π is key-update zero-knowledge. (ii) if Π is persistent Sub-ZK then Π is persistent key-update Sub-ZK.*

Proof. **(i)** Consider a creation of $(\textsc{pk}, \pi)$ (that returns $\textsc{sk}$) followed by a update of $(\textsc{pk}, \pi)$ to $(\textsc{pk}', \pi')$ (that returns $\widehat{\textsc{sk}}$). Due to key-update completeness, $\mathsf{V_{pk}}(\mathsf{lpar}; \textsc{pk}') = 1$. Then, by the zero-knowledge property, for $\textsc{sk}' \leftarrow \mathsf{Comb}(\mathsf{lpar}; \textsc{sk}, \widehat{\textsc{sk}})$, $\mathsf{Sim}(\mathsf{lpar}, \textsc{pk}', \textsc{sk}'; \mathsf{x}) \approx_\lambda \mathsf{P}(\mathsf{lpar}, \textsc{pk}'; \mathsf{x}, \mathsf{w})$.

(ii) Consider a possibly malicious creation of $(\textsc{pk}, \pi)$ followed by a possibly malicious update of $(\textsc{pk}, \pi)$ to $(\textsc{pk}', \pi')$, such that all verifications accept. Due to key-update completeness, we have $\mathsf{V_{pk}}(\mathsf{lpar}; \textsc{pk}') = 1$. Then, by the Sub-ZK property, there exist an extractor $\mathsf{Ext_Z}$ that extracts $(\textsc{sk}, \widehat{\textsc{sk}})$, such that for $\textsc{sk}' \leftarrow \mathsf{Comb}(\mathsf{lpar}; \textsc{sk}, \widehat{\textsc{sk}})$, $\mathsf{Sim}(\mathsf{lpar}, \textsc{pk}', \textsc{sk}'; \mathsf{x}) \approx_\lambda \mathsf{P}(\mathsf{lpar}, \textsc{pk}'; \mathsf{x}, \mathsf{w})$. □

Lemma 4. *Assume that $\widehat{\textsc{sk}}$ is efficiently computable from $\textsc{sk}$ and $\textsc{sk}'$. Assume Π is a key-update complete and simulator-update complete non-interactive argument system. (i) If Π is zero-knowledge then Π is persistent argument-update zero-knowledge. (ii) If Π is persistent Sub-ZK then Π is persistent argument-update Sub-ZK.*

Proof (Sketch.). **(i)** Consider an honest creation of $(\textsc{pk}, \pi)$ (that also returns $\textsc{sk}$) followed by an honest update of $(\textsc{pk}, \pi)$ to $(\textsc{pk}', \pi')$ (that also returns $\widehat{\textsc{sk}}$). By zero-knowledge, $\pi \leftarrow \mathsf{P}(\mathsf{lpar}, \textsc{pk}; \mathsf{x}, \mathsf{w})$ and $\pi_{\mathsf{Sim}} \leftarrow \mathsf{Sim}(\mathsf{lpar}, \textsc{pk}, \textsc{sk}; \mathsf{x})$ are indistinguishable. By the key-update completeness, $\mathsf{V_{pk}}(\mathsf{lpar}; \textsc{pk}') = 1$ and thus, by zero-knowledge, for $\textsc{sk}' \leftarrow \mathsf{Comb}(\textsc{sk}, \widehat{\textsc{sk}})$, π' and $\pi''_{\mathsf{Sim}} \leftarrow \mathsf{Sim}(\mathsf{lpar}, \textsc{pk}', \textsc{sk}'; \mathsf{x})$ are indistinguishable. By the simulator-update completeness, $\pi''_{\mathsf{Sim}} \approx_\lambda \pi'_{\mathsf{Sim}}$, where $\pi'_{\mathsf{Sim}} \leftarrow \mathsf{Sim_{upd}}(\mathsf{lpar}, \widehat{\textsc{sk}}; \mathsf{x}, \pi_{\mathsf{Sim}})$. Thus, the joint distributions (π, π') and $(\pi_{\mathsf{Sim}}, \pi'_{\mathsf{Sim}})$ are indistinguishable.

(ii) Consider a possibly malicious creation of $(\textsc{pk}, \pi)$ followed by a possibly malicious update of $(\textsc{pk}, \pi)$ to $(\textsc{pk}', \pi')$, such that all verifications accept. By persistent Sub-ZK, there exists an extractor $\mathsf{Ext}_{\mathsf{Z}_1}$ that extracts $\textsc{sk}$, such that $\pi \leftarrow \mathsf{P}(\mathsf{lpar}, \textsc{pk}; \mathsf{x}, \mathsf{w})$ and $\pi_{\mathsf{Sim}} \leftarrow \mathsf{Sim}(\mathsf{lpar}, \textsc{pk}, \textsc{sk}; \mathsf{x})$ are indistinguishable. By the key-update completeness, $\mathsf{V_{pk}}(\mathsf{lpar}; \textsc{pk}') = 1$ and thus, by persistent Sub-ZK, there exists an extractor $\mathsf{Ext}_{\mathsf{Z}_2}$ that extracts $\textsc{sk}'$ such that π' and $\pi''_{\mathsf{Sim}} \leftarrow \mathsf{Sim}(\mathsf{lpar}, \textsc{pk}', \textsc{sk}'; \mathsf{x})$ are indistinguishable. By the simulator-update completeness, $\pi''_{\mathsf{Sim}} \approx_\lambda \pi'_{\mathsf{Sim}}$, where $\pi'_{\mathsf{Sim}} \leftarrow \mathsf{Sim_{upd}}(\mathsf{lpar}, \widehat{\textsc{sk}}; \mathsf{x}, \pi_{\mathsf{Sim}})$. Thus, the joint distributions (π, π') and $(\pi_{\mathsf{Sim}}, \pi'_{\mathsf{Sim}})$ are indistinguishable. □

Handling Multiple Updates. All security notions given above are for a single update, but they can be generalized for many updates, by using standard hybrid arguments. We will omit further details and point to [GKM$^+$18] for more discussion.

4 Updatable Kiltz-Wee QA-NIZK

Since the public key of Π'_{as} consists of (bracketed) matrices, one may hope to construct a quite simple updating process where all $\textsc{pk}$ elements are updated additively. In such a case, an updater would create a "difference" public key $\widehat{\textsc{pk}}$ (by choosing the trapdoor privately) and update $\textsc{pk}$ by adding $\widehat{\textsc{pk}}$ componentwise to it, $\textsc{pk}' \leftarrow \textsc{pk} + \widehat{\textsc{pk}}$. However, this simple idea does not work since in the

```
$\mathsf{K_{bpk}}$, $\mathsf{P}$, $\mathsf{Sim}$, $\mathsf{V}$, $\mathsf{V_{pk}}$: exactly as in Fig. 2.
----------------------------------------------------------------
$\mathsf{K_{upd}}([\boldsymbol{M}]_1, \textbf{PK})$: // Updates $\mathrm{PK} = ([\boldsymbol{P}]_1, [\bar{\boldsymbol{A}}, \boldsymbol{C}]_2)$ to $\mathrm{PK}' = ([\boldsymbol{P}']_1, [\bar{\boldsymbol{A}}', \boldsymbol{C}']_2)$
    $\widehat{\boldsymbol{A}} \leftarrow_{\$} \mathcal{D}_k$; $[\bar{\boldsymbol{A}}']_2 \leftarrow [\bar{\boldsymbol{A}}]_2 \widehat{\boldsymbol{A}}$; $\widehat{\boldsymbol{K}} \leftarrow_{\$} \mathbb{Z}_p^{n \times k}$;
    $[\widehat{\boldsymbol{C}}]_2 \leftarrow \widehat{\boldsymbol{K}} [\bar{\boldsymbol{A}}']_2$; $[\boldsymbol{C}']_2 \leftarrow ([\boldsymbol{C}]_2 \widehat{\boldsymbol{A}} + [\widehat{\boldsymbol{C}}]_2)/\beta$;
    $[\widehat{\boldsymbol{P}}]_1 \leftarrow [\boldsymbol{M}]_1^\top \widehat{\boldsymbol{K}}$; $[\boldsymbol{P}']_1 \leftarrow ([\boldsymbol{P}]_1 + [\widehat{\boldsymbol{P}}]_1)/\beta$; // Implicitly, $\boldsymbol{K}' = (\boldsymbol{K} + \widehat{\boldsymbol{K}})/\beta$
    $\mathrm{PK}_{\mathsf{upd}} \leftarrow ([\widehat{\boldsymbol{A}}]_1, [\widehat{\boldsymbol{A}}, \widehat{\boldsymbol{C}}]_2)$; $\mathrm{PK}' \leftarrow ([\boldsymbol{P}']_1, [\bar{\boldsymbol{A}}', \boldsymbol{C}']_2, \mathrm{PK}_{\mathsf{upd}})$; $\widehat{\mathrm{SK}} \leftarrow \widehat{\boldsymbol{K}}$;
    return $(\mathrm{PK}', \widehat{\mathrm{SK}})$;
----------------------------------------------------------------
$\mathsf{V_{Kupd}}([\boldsymbol{M}]_1, \textbf{PK}, \textbf{PK}')$: $[\widehat{\boldsymbol{P}}]_1 \leftarrow [\beta \boldsymbol{P}' - \boldsymbol{P}]_1$; $\widehat{\mathrm{PK}} \leftarrow ([\widehat{\boldsymbol{P}}]_1, [\bar{\boldsymbol{A}}', \widehat{\boldsymbol{C}}]_2)$;
    if $\mathsf{isinvertible}([\bar{\boldsymbol{A}}]_1, \mathrm{PK}_{\mathsf{V_{pk}}}) \wedge \mathsf{isinvertible}([\widehat{\boldsymbol{A}}]_1, \mathrm{PK}_{\mathsf{V_{pk}}}) \wedge [\bar{\boldsymbol{A}}']_1 \cdot [1]_2 = [\bar{\boldsymbol{A}}]_1 \cdot [\widehat{\boldsymbol{A}}]_2$
    $\wedge\ [\widehat{\boldsymbol{A}}]_1 \cdot [1]_2 = [1]_1 \cdot [\widehat{\boldsymbol{A}}]_2\ \wedge$
      $[1]_1 \cdot [\boldsymbol{C}']_2 = ([\boldsymbol{C}]_2 \cdot [\widehat{\boldsymbol{A}}]_1 + [1]_1 \cdot [\widehat{\boldsymbol{C}}]_2)/\beta \wedge \mathsf{V_{pk}}([\boldsymbol{M}]_1, \widehat{\mathrm{PK}}) = 1$
    then return 1 else return 0 fi
----------------------------------------------------------------
$\mathsf{P_{upd}}([\boldsymbol{M}]_1, \textbf{PK}; [\boldsymbol{y}]_1, [\mathbf{w}]_1, [\boldsymbol{\pi}]_1)$: $[\widehat{\boldsymbol{\pi}}]_1 \leftarrow [\widehat{\boldsymbol{P}}]_1^\top \mathbf{w}$; return $[\boldsymbol{\pi}']_1 \leftarrow ([\boldsymbol{\pi}]_1 + [\widehat{\boldsymbol{\pi}}]_1)/\beta$;
----------------------------------------------------------------
$\mathsf{Sim_{upd}}([\boldsymbol{M}]_1, \widehat{\textbf{SK}}; [\boldsymbol{y}]_1, [\boldsymbol{\pi}]_1)$: $[\widehat{\boldsymbol{\pi}}_{\mathsf{Sim}}]_1 \leftarrow \widehat{\boldsymbol{K}}^\top [\boldsymbol{y}]_1$; return $[\boldsymbol{\pi}']_1 \leftarrow ([\boldsymbol{\pi}]_1 +$
    $[\widehat{\boldsymbol{\pi}}_{\mathsf{Sim}}]_1)/\beta$;
----------------------------------------------------------------
$\mathsf{V_{Pupd}}([\boldsymbol{M}]_1, \textbf{PK}, \textbf{PK}'; [\boldsymbol{y}]_1, [\boldsymbol{\pi}]_1, [\boldsymbol{\pi}']_1)$:
    $[\widehat{\boldsymbol{P}}]_1 \leftarrow [\beta \boldsymbol{P}' - \boldsymbol{P}]_1$; $\widehat{\mathrm{PK}} \leftarrow ([\widehat{\boldsymbol{P}}]_1, [\bar{\boldsymbol{A}}', \widehat{\boldsymbol{C}}]_2)$; $[\widehat{\boldsymbol{\pi}}]_1 \leftarrow [\beta \boldsymbol{\pi}' - \boldsymbol{\pi}]_1$;
    if $\mathsf{V}([\boldsymbol{M}]_1, \widehat{\mathrm{PK}}; [\boldsymbol{y}]_1; [\widehat{\boldsymbol{\pi}}]_1) = 1$ then return 1 else return 0 fi
```

Fig. 4. Variant $\Pi_{\mathsf{bpk}}^{\mathsf{upd}}$ of Kiltz-Wee QA-NIZK for $[\boldsymbol{y}]_1 = [\boldsymbol{M}]_1 \mathbf{w}$ in the Sub-ZK model. Here, $k \in \{1, 2\}$, and $\alpha, \beta \geq 1$.

case of additive updating (see Fig. 4 for notation), when we define $\bar{\boldsymbol{A}}' \leftarrow \bar{\boldsymbol{A}} + \widehat{\boldsymbol{A}}$, we need to compute

$$[\boldsymbol{C}']_2 = [\boldsymbol{K}' \bar{\boldsymbol{A}}']_2 = [(\boldsymbol{K} + \widehat{\boldsymbol{K}})(\bar{\boldsymbol{A}} + \widehat{\boldsymbol{A}})]_2 = [\boldsymbol{C}]_2 + [\boldsymbol{K}]_2 \widehat{\boldsymbol{A}} + \widehat{\boldsymbol{K}}([\bar{\boldsymbol{A}}]_2 + [\widehat{\boldsymbol{A}}]_2) \ .$$

An arbitrary party cannot compute the last formula since $[\boldsymbol{K}]_2$ is not public. To overcome this issue, we have chosen to update the square matrix $\bar{\boldsymbol{A}}$ multiplicatively, that is, $\bar{\boldsymbol{A}}' \leftarrow \bar{\boldsymbol{A}} \widehat{\boldsymbol{A}}$. On the other hand, we cannot update $[\boldsymbol{K}]_2$ multiplicatively since $[\boldsymbol{K}]_2$ is (usually) not a square matrix. Thus, we use different updating strategies for different elements of PK, updating some of them additively, and some of them multiplicatively. This differs significantly from the known updating procedures for SNARKs like [GKM$^+$18] (and all subsequent works that we are aware of), where all PK elements are updated multiplicatively. Finally, to allow for a larger variety of distributions $\mathcal{D}_{\boldsymbol{K}}$ of $\boldsymbol{K}$, we introduce a scaling factor β. That is, we update $\boldsymbol{K}$ to $\boldsymbol{K}' = (\boldsymbol{K} + \widehat{\boldsymbol{K}})/\beta$. (For example, with $\beta = 2$, strong key-update and argument-updating hold even when $\mathcal{D}_K = \mathcal{L}_k$.) We recommend to usually take $\beta = 2$, but other choices of β may be appropriate. We leave it as an interesting open question to similarly generalize the updating of $\bar{\boldsymbol{A}}$. We depict an updatable version of Π'_{as} in Fig. 4.

Note that (i) $\mathsf{P_{upd}}$ updates a QA-NIZK argument $[\boldsymbol{\pi}]_1$ by adding to it a honest argument $[\widehat{\boldsymbol{\pi}}]_1$ under the "difference" public key $\widehat{\mathrm{PK}}$, given the witness $\mathbf{w}$.

Thus, $\mathsf{P}_{\mathsf{upd}}$ can be run by a party who knows $\mathbf{w}$. (ii) $\mathsf{Sim}_{\mathsf{upd}}$ updates the existing QA-NIZK argument $[\boldsymbol{\pi}]_1$ by adding to it a simulation $[\widehat{\boldsymbol{\pi}}_{\mathsf{Sim}}]_1$ of the argument given $\widehat{\mathrm{SK}} = \widehat{\boldsymbol{K}}$ (that is known to the key-updater) as the trapdoor. Thus, $\mathsf{Sim}_{\mathsf{upd}}$ can be run by the key-updater. Thus, to be sure that at least one update was made by a party who knows the witness, one should make sure that at least one key-updater will not collude with the argument-updater of the same round.

5 Security of $\Pi_{\mathsf{bpk}}^{\mathsf{upd}}$

Lemma 5. *$\Pi_{\mathsf{bpk}}^{\mathsf{upd}}$ is (i) key-update complete, (ii) argument-update complete, and (iii) simulator-update complete.*

Proof. **(i: Key-update completeness)** We need to show that for $(\mathrm{PK}', \widehat{\mathrm{SK}}) \leftarrow \mathsf{K}_{\mathsf{upd}}(\mathsf{lpar}, \mathrm{PK})$, $\mathsf{V}_{\mathsf{Kupd}}(\mathsf{lpar}, \mathrm{PK}, \mathrm{PK}') = 1$.

Really, PK' is defined by $\bar{\boldsymbol{A}}' = \bar{\boldsymbol{A}}\widehat{\boldsymbol{A}}$, $\boldsymbol{C}' = (\boldsymbol{C}\widehat{\boldsymbol{A}} + \widehat{\boldsymbol{K}}\bar{\boldsymbol{A}}')/\beta$, and $\boldsymbol{P}' = (\boldsymbol{P} + \boldsymbol{M}^\top\widehat{\boldsymbol{K}})/\beta$. Thus, the first two verification equations in $\mathsf{V}_{\mathsf{Kupd}}$ hold by the definition of $\bar{\boldsymbol{A}}$ and $\widehat{\boldsymbol{A}}$ (they are invertible), and the next three ones hold trivially. Let $\widetilde{\mathrm{PK}} \leftarrow ([\boldsymbol{P}]_1, [\bar{\boldsymbol{A}}, \boldsymbol{C}]_2)$ and $\widetilde{\mathrm{PK}}' \leftarrow ([\boldsymbol{P}']_1, [\bar{\boldsymbol{A}}', \boldsymbol{C}']_2)$. We get $\mathsf{V}_{\mathsf{pk}}([\boldsymbol{M}]_1, \widetilde{\mathrm{PK}}') = 1$ from

$$\begin{aligned}\boldsymbol{M}^\top\boldsymbol{C}' - \boldsymbol{P}'\bar{\boldsymbol{A}}' =&\boldsymbol{M}^\top(\boldsymbol{C}\widehat{\boldsymbol{A}} + \widehat{\boldsymbol{K}}\bar{\boldsymbol{A}}')/\beta - (\boldsymbol{P} + \boldsymbol{M}^\top\widehat{\boldsymbol{K}})/\beta \cdot \bar{\boldsymbol{A}}\widehat{\boldsymbol{A}}\\ =&\left((\boldsymbol{M}^\top\boldsymbol{C} - \boldsymbol{P}\bar{\boldsymbol{A}})\widehat{\boldsymbol{A}} + \boldsymbol{M}^\top\widehat{\boldsymbol{K}}\left(\bar{\boldsymbol{A}}' - \bar{\boldsymbol{A}}\widehat{\boldsymbol{A}}\right)\right)/\beta = \boldsymbol{0}\end{aligned}$$

since $\mathsf{V}_{\mathsf{pk}}([\boldsymbol{M}]_1, \widetilde{\mathrm{PK}}) = 1$ (and thus $\boldsymbol{M}^\top\boldsymbol{C} = \boldsymbol{P}\bar{\boldsymbol{A}}$) and $\bar{\boldsymbol{A}}' = \bar{\boldsymbol{A}}\widehat{\boldsymbol{A}}$.

On the other hand, if $\mathsf{V}_{\mathsf{Kupd}}([\boldsymbol{M}]_1; \mathrm{PK}, \mathrm{PK}') = 1$ then

$$\begin{aligned}\boldsymbol{0} = \boldsymbol{M}^\top\widehat{\boldsymbol{C}} - \widehat{\boldsymbol{P}}\bar{\boldsymbol{A}}' =&\boldsymbol{M}^\top(\beta\boldsymbol{C}' - \boldsymbol{C}\widehat{\boldsymbol{A}}) - (\beta\boldsymbol{P}' - \boldsymbol{P})\bar{\boldsymbol{A}}'\\ =&\beta(\boldsymbol{M}^\top\boldsymbol{C}' - \boldsymbol{P}'\bar{\boldsymbol{A}}') - (\boldsymbol{M}^\top\boldsymbol{C} - \boldsymbol{P}\bar{\boldsymbol{A}})\widehat{\boldsymbol{A}}\end{aligned}$$

and thus, since $\widehat{\boldsymbol{A}}$ is invertible, $\mathsf{V}_{\mathsf{pk}}([\boldsymbol{M}]_1; \mathrm{PK}') = 1$ iff $\mathsf{V}_{\mathsf{pk}}([\boldsymbol{M}]_1; \mathrm{PK}) = 1$.

(ii: Argument-update completeness) Clearly, $\boldsymbol{y}^\top\widehat{\boldsymbol{C}} - \widehat{\boldsymbol{\pi}}^\top\bar{\boldsymbol{A}}' = \boldsymbol{y}^\top\widehat{\boldsymbol{K}}\bar{\boldsymbol{A}}' - (\widehat{\boldsymbol{P}}^\top\mathbf{w})^\top\bar{\boldsymbol{A}}' = (\mathbf{w}^\top\boldsymbol{M}^\top\widehat{\boldsymbol{K}} - \mathbf{w}^\top\boldsymbol{M}^\top\widehat{\boldsymbol{K}})\bar{\boldsymbol{A}}' = \boldsymbol{0}$ and thus $\mathsf{V}([\boldsymbol{M}]_1, \widehat{\mathrm{PK}}; [\boldsymbol{y}]_1; [\widehat{\boldsymbol{\pi}}]_1) = 1$. On the other hand, $\boldsymbol{y}^\top\widehat{\boldsymbol{C}} - \widehat{\boldsymbol{\pi}}^\top\bar{\boldsymbol{A}}' = \boldsymbol{y}^\top(\beta\boldsymbol{C}' - \boldsymbol{C}\widehat{\boldsymbol{A}}) - (\beta\boldsymbol{\pi}' - \boldsymbol{\pi})^\top\bar{\boldsymbol{A}}\widehat{\boldsymbol{A}} = \beta\left(\boldsymbol{y}^\top\boldsymbol{C}' - \boldsymbol{\pi}'^\top\bar{\boldsymbol{A}}'\right) - \left(\boldsymbol{y}^\top\boldsymbol{C} - \boldsymbol{\pi}^\top\bar{\boldsymbol{A}}\right)\widehat{\boldsymbol{A}}$ and thus if $\mathsf{V}_{\mathsf{Pupd}}$ accepts then, since $\widehat{\boldsymbol{A}}$ is invertible, $\mathsf{V}([\boldsymbol{M}]_1, \mathrm{PK}'; [\boldsymbol{y}]_1, [\boldsymbol{\pi}']_1) = 1$ iff $\mathsf{V}([\boldsymbol{M}]_1, \mathrm{PK}; [\boldsymbol{y}]_1, [\boldsymbol{\pi}]_1) = 1$.

(iii: Simulator-update completeness) Clearly, $\boldsymbol{y}^\top\widehat{\boldsymbol{C}} - \widehat{\boldsymbol{\pi}}^\top\bar{\boldsymbol{A}}' = (\boldsymbol{y}^\top\widehat{\boldsymbol{K}} - (\widehat{\boldsymbol{K}}^\top\boldsymbol{y})^\top) = \boldsymbol{0}$ and thus $\mathsf{V}([\boldsymbol{M}]_1, \widehat{\mathrm{PK}}; [\boldsymbol{y}]_1; [\widehat{\boldsymbol{\pi}}]_1) = 1$. The proof that if $\mathsf{V}_{\mathsf{Pupd}}$ accepts then $\mathsf{V}([\boldsymbol{M}]_1, \mathrm{PK}'; [\boldsymbol{y}]_1, [\boldsymbol{\pi}']_1) = 1$ iff $\mathsf{V}([\boldsymbol{M}]_1, \mathrm{PK}; [\boldsymbol{y}]_1, [\boldsymbol{\pi}]_1) = 1$ is the same as in the case (ii). □

Lemma 6 (Key-update hiding and argument-update hiding). *Assume that $\boldsymbol{K}, \widehat{\boldsymbol{K}} \sim \mathcal{D}_{\boldsymbol{K}}$ and $\bar{\boldsymbol{A}}, \widehat{\boldsymbol{A}} \sim \mathcal{D}_{\bar{\boldsymbol{A}}}$, where $\mathcal{D}_{\boldsymbol{K}}$ and $\mathcal{D}_{\bar{\boldsymbol{A}}}$ satisfy the following conditions: for i.i.d random variables X_1 and X_2,*

– *if $X_i \sim \mathcal{D}_{\boldsymbol{K}}$ for both i then $X_1 + X_2 \sim \beta\mathcal{D}_{\boldsymbol{K}}$. (Thus, $\mathcal{D}_{\boldsymbol{K}}^{*2} = \beta\mathcal{D}_{\boldsymbol{K}}$, where $\mathcal{D}^{*s}$ is the sth convolution power of $\mathcal{D}$. That is, $\mathcal{D}_{\boldsymbol{K}}$ is a stable distribution with index $1/\log_2\beta$.)*

– *if $X_i \sim \mathcal{D}_{\bar{\boldsymbol{A}}}$ for both i then $X_1 \cdot X_2 \sim \mathcal{D}_{\bar{\boldsymbol{A}}}$.*

Then, $\Pi_{\mathsf{bpk}}^{\mathsf{upd}}$ is (i) key-update hiding and (ii) (assuming perfect simulation) argument-update hiding.

Proof. (i) Since PK is honestly created, $\boldsymbol{C} = \boldsymbol{K}\bar{\boldsymbol{A}}$ and thus $\boldsymbol{C}' = (\boldsymbol{C}\widehat{\boldsymbol{A}} + \widehat{\boldsymbol{K}}\bar{\boldsymbol{A}}')/\beta = (\boldsymbol{K}\bar{\boldsymbol{A}}\widehat{\boldsymbol{A}} + \widehat{\boldsymbol{K}}\bar{\boldsymbol{A}}')/\beta = (\boldsymbol{K} + \widehat{\boldsymbol{K}})/\beta \cdot \bar{\boldsymbol{A}}' = \boldsymbol{K}'\bar{\boldsymbol{A}}'$. Similarly, $\boldsymbol{P} = \boldsymbol{M}^\top \boldsymbol{K}$ and $\boldsymbol{P}' = (\boldsymbol{P} + \boldsymbol{M}^\top \widehat{\boldsymbol{K}})/\beta = \boldsymbol{M}^\top(\boldsymbol{K} + \widehat{\boldsymbol{K}})/\beta = \boldsymbol{M}^\top \boldsymbol{K}'$. Due to the assumption on $\mathcal{D}_{\bar{\boldsymbol{A}}}$ and $\mathcal{D}_{\boldsymbol{K}}$, PK and PK$'$ come from the same distribution.

(ii) We already know PK and PK$'$ come from the same distribution. Assume that $[\boldsymbol{y}]_1 = [\boldsymbol{M}]_1\mathsf{w}$. Due to the perfect simulation, $\boldsymbol{\pi} = \mathsf{Sim}([\boldsymbol{M}]_1, \text{PK}; [\boldsymbol{y}]_1, [\boldsymbol{\pi}]_1) = \boldsymbol{K}^\top \boldsymbol{y}$. Thus, $\boldsymbol{\pi}' = (\boldsymbol{K}^\top \boldsymbol{y} + \widehat{\boldsymbol{K}}^\top \boldsymbol{y})/\beta = \boldsymbol{K}'^\top \boldsymbol{y} = \mathsf{Sim}([\boldsymbol{M}]_1, \text{PK}'; [\boldsymbol{y}]_1, [\boldsymbol{\pi}']_1) = \mathsf{P}([\boldsymbol{M}]_1, \text{PK}', [\boldsymbol{y}]_1, \mathsf{w})$. □

Remark 1. In Lemma 6, we need a convolution semigroup consisting of a single element. It is possible to generalize to the case of a general convolution semigroup (i.e., allowing $\widehat{\boldsymbol{K}}$ to come from a different distribution than K).

Theorem 1 (Strong key-update hiding and strong argument-update hiding). *Assume that $\boldsymbol{K}, \widehat{\boldsymbol{K}} \sim \mathcal{D}_{\boldsymbol{K}}$ and $\bar{\boldsymbol{A}}, \widehat{\boldsymbol{A}} \sim \mathcal{D}_{\bar{\boldsymbol{A}}}$, where $\mathcal{D}_{\boldsymbol{K}}$ and $\mathcal{D}_{\bar{\boldsymbol{A}}}$ satisfy the following conditions: for i.i.d random variables X_1 and X_2,*

– *if $X_i \sim \mathcal{D}_{\boldsymbol{K}}$ for at least one i then $X_1 + X_2 \sim \beta\mathcal{D}_{\boldsymbol{K}}$. (Thus, the convolution of $\mathcal{D}_{\boldsymbol{K}}$ with any other distribution — over the support of $\mathcal{D}_{\boldsymbol{K}}$ — is $\beta\mathcal{D}_{\boldsymbol{K}}$, or $\mathcal{D}_{\boldsymbol{K}}$ is belongs to a generalized ideal of a convolution semigroup.)*

– *if $X_i \sim \mathcal{D}_{\bar{\boldsymbol{A}}}$ for at least one i then $X_1 \cdot X_2 \sim \mathcal{D}_{\bar{\boldsymbol{A}}}$. (Thus, the log of $\mathcal{D}_{\bar{\boldsymbol{A}}}$ belongs to an ideal of a convolution semigroup.)*

Then, $\Pi_{\mathsf{bpk}}^{\mathsf{upd}}$ is (i) strong key-update hiding and (ii) (assuming perfect simulation) strong argument-update hiding.

Proof. We will prove (i) and (ii) together in two different cases: (1) when PK (and the argument) was honestly created, and (2) when PK$'$ (and the argument) was honestly updated.

(1: PK / π were honestly created and the updates verify) Since PK is honestly created, $\boldsymbol{C} = \boldsymbol{K}\bar{\boldsymbol{A}}$ and $\boldsymbol{P} = \boldsymbol{M}^\top \boldsymbol{K}$. Since $\mathsf{V}_{\mathsf{Kupd}}$ accepts, we have $\bar{\boldsymbol{A}}' = \bar{\boldsymbol{A}}\widehat{\boldsymbol{A}}$ (thus, by the assumption on $\mathcal{D}_{\bar{\boldsymbol{A}}}$, $\bar{\boldsymbol{A}}'$ comes from the correct distribution), $\boldsymbol{C}' = (\boldsymbol{C}\widehat{\boldsymbol{A}} + \widehat{\boldsymbol{C}})/\beta$, and $\boldsymbol{M}^\top \widehat{\boldsymbol{C}} = \widehat{\boldsymbol{P}}\bar{\boldsymbol{A}}'$ where $\widehat{\boldsymbol{P}} = \beta\boldsymbol{P}' - \boldsymbol{P}$. Thus, $\boldsymbol{C}' = (\boldsymbol{C}\widehat{\boldsymbol{A}} + \widehat{\boldsymbol{C}})/\beta = (\boldsymbol{K}\bar{\boldsymbol{A}}\widehat{\boldsymbol{A}} + \widehat{\boldsymbol{C}})/\beta = (\boldsymbol{K}\bar{\boldsymbol{A}}' + \widehat{\boldsymbol{C}})/\beta$. Define implicitly $\boldsymbol{K}' := \boldsymbol{C}'\bar{\boldsymbol{A}}'^{-1} = (\boldsymbol{K}\bar{\boldsymbol{A}}' + \widehat{\boldsymbol{C}})/\beta \cdot \bar{\boldsymbol{A}}'^{-1} = (\boldsymbol{K} + \widehat{\boldsymbol{C}}\bar{\boldsymbol{A}}'^{-1})/\beta$ (note that $\bar{\boldsymbol{A}}'$ is invertible). Thus, obviously, $\boldsymbol{C}' = \boldsymbol{K}'\bar{\boldsymbol{A}}'$. On the other hand, $\boldsymbol{M}^\top \boldsymbol{K}' = \boldsymbol{M}^\top(\boldsymbol{K} + \widehat{\boldsymbol{C}}\bar{\boldsymbol{A}}'^{-1})/\beta = (\boldsymbol{P} + \boldsymbol{M}^\top \widehat{\boldsymbol{C}}\bar{\boldsymbol{A}}'^{-1})/\beta = (\boldsymbol{P} + \widehat{\boldsymbol{P}}\bar{\boldsymbol{A}}'\bar{\boldsymbol{A}}'^{-1})/\beta = (\boldsymbol{P} + \widehat{\boldsymbol{P}})/\beta = \boldsymbol{P}'$ and thus $\boldsymbol{P}' = \boldsymbol{M}^\top \boldsymbol{K}'$. To show that PK and PK$'$ come from the same distribution, we now only need to show that $\boldsymbol{K}' = (\boldsymbol{K} + \widehat{\boldsymbol{C}}\bar{\boldsymbol{A}}'^{-1})/\beta$ comes from the same distribution as $\boldsymbol{K}$. This holds assuming that $\mathcal{D}_{\boldsymbol{K}}$ is a generalized ideal of a convolution semigroup.

Consider the argument $[\boldsymbol{\pi'}]_1$. We have, in addition to equations above, that

- since $[\boldsymbol{\pi}]_1$ is honestly created: $\boldsymbol{\pi} = \boldsymbol{P}^\top \mathbf{w}$.
- since $[\boldsymbol{\pi}']_1$ verifies: $\boldsymbol{y}^\top \widehat{\boldsymbol{C}} = \widehat{\boldsymbol{\pi}} \bar{\boldsymbol{A}}$.

Due to completeness, $\boldsymbol{y}^\top \boldsymbol{C} = \boldsymbol{\pi}^\top \bar{\boldsymbol{A}}$. Thus,

$$\begin{aligned}\beta(\boldsymbol{y}^\top \boldsymbol{C}' - \boldsymbol{\pi}' \bar{\boldsymbol{A}}') =& \boldsymbol{y}^\top (\boldsymbol{C}\widehat{\boldsymbol{A}} + \widehat{\boldsymbol{C}}) - (\boldsymbol{\pi} + \widehat{\boldsymbol{\pi}})\bar{\boldsymbol{A}}' \\ =& \underbrace{(\boldsymbol{y}^\top \boldsymbol{C} - \boldsymbol{\pi}\bar{\boldsymbol{A}})}_{=\mathbf{0}} \widehat{\boldsymbol{A}} + \underbrace{(\boldsymbol{y}^\top \widehat{\boldsymbol{C}} - \widehat{\boldsymbol{\pi}}\bar{\boldsymbol{A}}')}_{=\mathbf{0}}\end{aligned}$$

and thus $\boldsymbol{C}'$ verifies. But then clearly, $\boldsymbol{\pi}' = \boldsymbol{y}^\top \boldsymbol{C}' \bar{\boldsymbol{A}}'^{-1}$ is the unique correct argument for $\boldsymbol{y} \in \mathrm{ColSpace}(\boldsymbol{M})$ when using the public key PK$'$.

(2: PK verifies and the update was honestly done) We have the following equations:

- since PK verifies: $\boldsymbol{M}^\top \boldsymbol{C} = \boldsymbol{P} \bar{\boldsymbol{A}}$,
- since PK$'$ was honestly updated: for correctly distributed $\widehat{\boldsymbol{A}}$ and $\widehat{\boldsymbol{K}}$, $\bar{\boldsymbol{A}}' = \bar{\boldsymbol{A}}\widehat{\boldsymbol{A}}$, $\widehat{\boldsymbol{C}} = \widehat{\boldsymbol{K}}\bar{\boldsymbol{A}}'$, $\boldsymbol{C}' = (\boldsymbol{C}\widehat{\boldsymbol{A}} + \widehat{\boldsymbol{C}})/\beta$, $\widehat{\boldsymbol{P}} = \boldsymbol{M}^\top \widehat{\boldsymbol{K}}$, $\boldsymbol{P}' = (\boldsymbol{P} + \widehat{\boldsymbol{P}})/\beta$.

Due to the assumption on $\mathcal{D}_{\bar{\boldsymbol{A}}}$, $\bar{\boldsymbol{A}}'$ comes from the correct distribution. Define implicitly $\boldsymbol{P} := \boldsymbol{M}^\top \boldsymbol{C} \bar{\boldsymbol{A}}^{-1}$ (note that $\bar{\boldsymbol{A}}$ is invertible) and $\boldsymbol{K} := \boldsymbol{C}\bar{\boldsymbol{A}}^{-1}$. Then, $\boldsymbol{K}' = (\boldsymbol{K} + \widehat{\boldsymbol{K}})/\beta = (\boldsymbol{K} + \widehat{\boldsymbol{C}}\bar{\boldsymbol{A}}'^{-1})/\beta = (\boldsymbol{C}\widehat{\boldsymbol{A}} + \widehat{\boldsymbol{C}})/\beta \cdot \bar{\boldsymbol{A}}'^{-1} = \boldsymbol{C}'\bar{\boldsymbol{A}}'^{-1}$. Next, $\boldsymbol{K}' = (\boldsymbol{K} + \widehat{\boldsymbol{K}})/\beta$ has the same distribution as $\widehat{\boldsymbol{K}}$ by the assumption on $\mathcal{D}_{\boldsymbol{K}}$. Because both $\boldsymbol{K}'$ and $\bar{\boldsymbol{A}}'$ have correct distributions, also $\boldsymbol{C}' = \boldsymbol{K}'\bar{\boldsymbol{A}}'$ has correct distribution.

Next, obviously $\boldsymbol{P}' = \boldsymbol{M}^\top (\boldsymbol{C}\bar{\boldsymbol{A}}^{-1} + \widehat{\boldsymbol{K}})/\beta = \boldsymbol{M}^\top (\boldsymbol{K} + \beta\boldsymbol{K}' - \boldsymbol{K})/\beta = \boldsymbol{M}^\top \boldsymbol{K}'$ has the correct distribution. This proves strong key-updating.

In the case of strong argument-updating, additionally, the following holds:

- the original argument verifies: $\boldsymbol{y}^\top \boldsymbol{C} = \boldsymbol{\pi}^\top \bar{\boldsymbol{A}}$,
- $[\boldsymbol{\pi}]_1$ was updated honestly: $\boldsymbol{\pi}' = (\boldsymbol{\pi} + \widehat{\boldsymbol{K}}^\top \boldsymbol{y})/\beta$ for honestly distributed $\widehat{\boldsymbol{K}} \sim \mathcal{D}_{\boldsymbol{K}}$.

From this, we get that $\boldsymbol{\pi} = (\boldsymbol{y}^\top \boldsymbol{C}\bar{\boldsymbol{A}}^{-1})^\top = (\boldsymbol{y}^\top \boldsymbol{K})^\top = \boldsymbol{K}^\top \boldsymbol{y}$. Thus, $\boldsymbol{\pi}' = (\boldsymbol{\pi} + \widehat{\boldsymbol{K}}^\top \boldsymbol{y})/\beta = (\boldsymbol{K}^\top \boldsymbol{y} + \widehat{\boldsymbol{K}}^\top \boldsymbol{y})/\beta = (\boldsymbol{K} + \widehat{\boldsymbol{K}})^\top/\beta \cdot \boldsymbol{y} = \boldsymbol{K}'^\top \boldsymbol{y}$ which is equal to the simulated argument of $\boldsymbol{y} = \boldsymbol{M}\mathbf{w}$ (when using the public key PK$'$) and by perfect simulation, thus also to the real argument (when using PK$'$). □

Example 1 (Of the required distributions). $\mathcal{D}_{\boldsymbol{K}}$ is the uniform distribution over $k \times k$ matrices over $\mathbb{Z}_p$ and $\mathcal{D}_{\bar{\boldsymbol{A}}}$ is the uniform distribution over $k \times k$ invertible matrices over $\mathbb{Z}_p$ where as the sanity check, one checks that both matrices $[\bar{\boldsymbol{A}}]_1$ and $[\widehat{\boldsymbol{A}}]_1$ are invertible.[6] As mentioned before, in the actual instantiation of Π'_{as} (at least in the most efficient case $k = 1$) both $\mathcal{D}_{\boldsymbol{K}}$ and $\mathcal{D}_{\bar{\boldsymbol{A}}}$ are equal to the uniform distribution over $\mathbb{Z}_p$ and thus satisfy the required properties. □

Theorem 2. *Let $\mathcal{D}_k$ be efficiently verifiable (resp., $\mathcal{D}_k = \mathcal{U}_2$). If the $\mathcal{D}_k$-*KerMDH$^{\mathrm{dl}}$ *(resp., $\mathcal{D}_k$-*SKerMDH$^{\mathrm{dl}}$*) assumption holds relative to* Pgen *then* $\Pi^{\mathsf{upd}}_{\mathsf{bpk}}$

[6] Intuitively, we require the family ν of probability distributions to be an ideal of a convolution semigroup: $\nu_s * \mu = \nu_t$, for some t, for any element μ of the semigroup.

is (i) computationally quasi-adaptively Sub-PAR argument-update sound (I), and (ii) assuming that the preconditions of Theorem 1 are fulfilled, also computationally quasi-adaptively Sub-PAR argument-update sound (II) in the BPK model.

Proof. **(i: Sub-PAR argument-update soundness (I))** follows from Proposition 1 (Π_{bpk} is computationally quasi-adaptively Sub-PAR sound under the KerMDH$^{\mathrm{dl}}$ / SKerMDH$^{\mathrm{dl}}$ assumption) and Lemma 2 (any Sub-PAR sound argument system is also Sub-PAR argument-update sound (I)).

(ii: Sub-PAR argument-update soundness (II)) follows from Proposition 1 (Π_{bpk} is computationally quasi-adaptively Sub-PAR sound under the KerMDH$^{\mathrm{dl}}$ / SKerMDH$^{\mathrm{dl}}$ assumption), Lemma 2 (any Sub-PAR sound, argument-update complete, strongly-key-update hiding, and strongly argument-update hiding argument system is also Sub-PAR argument-update sound (II)), Lemma 5 ($\Pi_{\mathsf{bpk}}^{\mathsf{upd}}$ is argument-update complete), and Theorem 1 ($\Pi_{\mathsf{bpk}}^{\mathsf{upd}}$ is strongly key-update hiding and strongly argument-update hiding). □

We emphasize that Sub-PAR argument-update soundness (II) follows only when the update has been done by a honest prover (who knows the witness and does not know the key-update secret key $\widehat{\mathrm{SK}}$).

Interestingly, next, we rely on $\mathsf{KW\text{-}KE}$ that is a tautological knowledge assumption for Π_{bpk}, but not for $\Pi_{\mathsf{bpk}}^{\mathsf{upd}}$. This gives more credence to $\mathsf{KW\text{-}KE}$ as an assumption that is of independent interest.

Lemma 7. *Let $\Pi_{\mathsf{bpk}}^{\mathsf{upd}}$ be the updatable QA-NIZK argument system for linear subspaces from Fig. 4. Assume that $\mathcal{D}_{\mathsf{p}}$ is such that $\mathsf{V}_{\mathsf{par}}$ is efficient. (i) Π_{bpk} is key-update statistical zero-knowledge in the BPK model. (ii) If the $(\mathcal{D}_{\mathsf{p}}, k, \mathcal{D}_k)$-$\mathsf{KW\text{-}KE}_{\mathbb{G}_1}$ assumption holds relative to Pgen then Π_{bpk} is key-update statistical persistent Sub-ZK in the BPK model.*

Lemma 8. *Let $\Pi_{\mathsf{bpk}}^{\mathsf{upd}}$ be the updatable QA-NIZK argument system for linear subspaces from Fig. 4. Assume that $\mathcal{D}_{\mathsf{p}}$ is such that $\mathsf{V}_{\mathsf{par}}$ is efficient. (i) Π_{bpk} is argument-update statistical zero-knowledge in the BPK model. (ii) If the $(\mathcal{D}_{\mathsf{p}}, k, \mathcal{D}_k)$-$\mathsf{KW\text{-}KE}_{\mathbb{G}_1}$ assumption holds relative to Pgen then Π_{bpk} is argument-update statistical persistent Sub-ZK in the BPK model.*

6 Discussion

Why updating $\boldsymbol{M}$ might be difficult. In certain applications, one might also be interested in updating $\boldsymbol{M}$ (e.g., if $[\boldsymbol{M}]_1$ is a public key of some trusted third party). Assume $[\boldsymbol{M}]_1$ is updated to $[\boldsymbol{M}']_1 = [\boldsymbol{M}]_1 + [\widehat{\boldsymbol{M}}]_1$, then $[\beta \boldsymbol{P}']_1 - [\boldsymbol{P}]_1 = [\beta \boldsymbol{M}'^\top \boldsymbol{K}']_1 - [\boldsymbol{M}^\top \boldsymbol{K}_{i-1}]_1 = [\beta \boldsymbol{M}'^\top (\boldsymbol{K}_{i-1} + \widehat{\boldsymbol{K}})/\beta]_1 - [\boldsymbol{M}^\top \boldsymbol{K}_{i-1}]_1 = [\widehat{\boldsymbol{M}}^\top \boldsymbol{K}_{i-1}]_1 + [\boldsymbol{M}'^\top]_1 \widehat{\boldsymbol{K}}$ that can be computed assuming the updater knows either (1) both $\widehat{\boldsymbol{M}}$ and $[\boldsymbol{K}_{i-1}]_1$ or (2) $\boldsymbol{K}_{i-1}$, or if all previous parties help him to compute $[\widehat{\boldsymbol{M}}^\top \boldsymbol{K}_{i-1}]_1$. Since neither possibility seems realistic (note that even $[\boldsymbol{K}_{i-1}]_1$ is not public), one cannot probably update the language parameter.

7 Updatable Universal Public Key

Consider updatability in a more generic setting of pairing-based protocols, where the language parameter may not exist at all. The goal of updatability is to protect soundness in the case PK may be subverted, since Sub-ZK can be obtained by running the public algorithm V_{pk} [ABLZ17]. In Π'_{as}, soundness is guaranteed by one of the elements of PK (namely, $[\bar{\boldsymbol{A}}]_2$, see Fig. 2) coming from a KerMDH-hard distribution[7], and another element $[\boldsymbol{C}]_2$ being correctly computed from $[\bar{\boldsymbol{A}}]_2$. Since the latter can be verified by V_{pk}, to obtain soundness it suffices to update $[\bar{\boldsymbol{A}}]_2$. (The procedure of this is the same as creating $[\bar{\boldsymbol{A}}']_2$ by $\mathsf{K}_{\mathsf{upd}}$ in Fig. 4, and verifying this update consists of the first four verification equations in $\mathsf{V}_{\mathsf{Kupd}}$.) Then, $[\bar{\boldsymbol{A}}]_2$ will be a "universal public key" [GKM+18] for all possible language parameters in all applications that trust the concrete KerMDH *assumption.*

Importantly, one is here not restricted to Π'_{as} or even QA-NIZK: *any* application that relies on KerMDH-hardness of $\mathcal{D}_{\bar{\boldsymbol{A}}}$, where it suffices to know $[\bar{\boldsymbol{A}}]_2$ (instead of $[\boldsymbol{A}]_2$), and where $\mathcal{D}_{\bar{\boldsymbol{A}}}$ satisfies the conditions of Theorem 1, can use the same matrix $[\bar{\boldsymbol{A}}]_2$. A standard example is the 1-Lin [BBS04,EHK+13] distribution $\mathcal{L}_1 = \{\boldsymbol{A} = \left(\begin{smallmatrix} a \\ 1 \end{smallmatrix}\right) : a \leftarrow_{\$} \mathbb{Z}_p\}$. After potentially many updates of $[\bar{\boldsymbol{A}}]_2$, one can create the whole public key PK corresponding to a concrete language parameter. Thereafter, one can continue updating $[\bar{\boldsymbol{A}}]_2$ together with all known public keys and arguments that use (the same version of) $[\bar{\boldsymbol{A}}]_2$ for soundness. Such one-phase updating can be formalized like in [GKM+18], adding to it QA-NIZK and argument-update specifics, and is out of the scope of the current paper.

We emphasize that the possibility to rely just on $[\bar{\boldsymbol{A}}]_2$ is a major difference with updatable SNARKs [GKM+18] where the universal key is quite complex and each universal key of length $\Theta(n)$ can only be used for circuits of size $\leq n$.

History Further Work. The first version of this paper was written a few days after [GKM+18] was posted on eprint; and then eprinted as [Lip19a]. The current version mainly differs by taking in the account the newer version of [ALSZ20]. We leave the definition and study of updatable Sub-PAR *knowledge-sound* QA-NIZKs as an open question. We also leave the study of other applications that can make use of the described universal public key updating method to the further work. We conjecture that many other such applications will be found shortly.

Acknowledgment. We would like to thank Dario Fiore and Markulf Kohlweiss for useful comments. The authors were partially supported by the Estonian Research Council grant (PRG49).

[7] Technically, $\boldsymbol{A}$ comes from a KerMDH-hard distribution, not $\bar{\boldsymbol{A}}$. However, in the case of some distributions (like DLIN-related distributions), $\boldsymbol{A}$ has an extra constant column compared to $\bar{\boldsymbol{A}}$. The knowledge of $[\boldsymbol{A}]_2$ and $[\bar{\boldsymbol{A}}]_2$ is equivalent in such a case.

References

ABL$^+$19. Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, Janno Siim, and Michal Zajac. UC-secure CRS generation for SNARKs. In Johannes Buchmann, Abderrahmane Nitaj, and Tajje eddine Rachidi, editors, *AFRICACRYPT 19*, volume 11627 of *LNCS*, pages 99–117. Springer, Heidelberg, July 2019. `doi:10.1007/978-3-030-23696-0_6`.

ABLZ17. Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, and Michal Zajac. A subversion-resistant SNARK. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 3–33. Springer, Heidelberg, December 2017. `doi:10.1007/978-3-319-70700-6_1`.

ABP15. Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. Disjunctions for hash proof systems: New constructions and applications. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 69–100. Springer, Heidelberg, April 2015. `doi:10.1007/978-3-662-46803-6_3`.

ALSZ20. Behzad Abdolmaleki, Helger Lipmaa, Janno Siim, and Michal Zajac. On QA-NIZK in the BPK model. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 590–620. Springer, Heidelberg, May 2020. `doi:10.1007/978-3-030-45374-9_20`.

APV05. Joël Alwen, Giuseppe Persiano, and Ivan Visconti. Impossibility and feasibility results for zero knowledge with public keys. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 135–151. Springer, Heidelberg, August 2005. `doi:10.1007/11535218_9`.

ARS20. Behzad Abdolmaleki, Sebastian Ramacher, and Daniel Slamanig. Lift-and-Shift: Obtaining Simulation Extractable Subversion and Updatable SNARKs Generically. Technical Report 2020/062, IACR, January 21, 2020. URL: `https://eprint.iacr.org/2020/062`.

BBS04. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, August 2004. `doi:10.1007/978-3-540-28628-8_3`.

BCG$^+$14. Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014. `doi:10.1109/SP.2014.36`.

BCG$^+$15. Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. Secure sampling of public parameters for succinct zero knowledge proofs. In *2015 IEEE Symposium on Security and Privacy*, pages 287–304. IEEE Computer Society Press, May 2015. `doi:10.1109/SP.2015.25`.

BFM88. Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988. `doi:10.1145/62212.62222`.

BFS16. Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume

10032 of *LNCS*, pages 777–804. Springer, Heidelberg, December 2016. doi:10.1007/978-3-662-53890-6_26.

BGG17. Sean Bowe, Ariel Gabizon, and Matthew D. Green. A multi-party protocol for constructing the public parameters of the pinocchio zk-SNARK. Cryptology ePrint Archive, Report 2017/602, 2017. http://eprint.iacr.org/2017/602.

BGM17. Sean Bowe, Ariel Gabizon, and Ian Miers. Scalable multi-party computation for zk-SNARK parameters in the random beacon model. Cryptology ePrint Archive, Report 2017/1050, 2017. http://eprint.iacr.org/2017/1050.

CGGM00. Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *32nd ACM STOC*, pages 235–244. ACM Press, May 2000. doi:10.1145/335305.335334.

CHM$^+$20. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45721-1_26.

DFKP13. George Danezis, Cédric Fournet, Markulf Kohlweiss, and Bryan Parno. Pinocchio coin: building zerocoin from a succinct pairing-based proof system. pages 27–30, Berlin, Germany, November 4, 2013. ACM.

DGP$^+$19. Vanesa Daza, Alonso González, Zaira Pindado, Carla Ràfols, and Javier Silva. Shorter quadratic QA-NIZK proofs. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part I*, volume 11442 of *LNCS*, pages 314–343. Springer, Heidelberg, April 2019. doi:10.1007/978-3-030-17253-4_11.

DL08. Giovanni Di Crescenzo and Helger Lipmaa. Succinct NP Proofs from an Extractability Assumption. In Arnold Beckmann, Costas Dimitracopoulos, and Benedikt Löwe, editors, *Computability in Europe, CIE 2008*, volume 5028 of *LNCS*, pages 175–185, Athens, Greece, June 15–20, 2008. Springer, Heidelberg.

DRZ20. Vanesa Daza, Carla Ràfols, and Alexandros Zacharakis. Updateable inner product argument with logarithmic verifier and applications. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 527–557. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45374-9_18.

EHK$^+$13. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, August 2013. doi:10.1007/978-3-642-40084-1_8.

Fuc18. Georg Fuchsbauer. Subversion-zero-knowledge SNARKs. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 315–347. Springer, Heidelberg, March 2018. doi:10.1007/978-3-319-76578-5_11.

GGPR13. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013. doi:10.1007/978-3-642-38348-9_37.

GHR15. Alonso González, Alejandro Hevia, and Carla Ràfols. QA-NIZK arguments in asymmetric groups: New tools and new constructions. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 605–629. Springer, Heidelberg, November / December 2015. doi:10.1007/978-3-662-48797-6_25.

Gjø06. Kristian Gjøsteen. A new security proof for damgård's ElGamal. In David Pointcheval, editor, *CT-RSA 2006*, volume 3860 of *LNCS*, pages 150–158. Springer, Heidelberg, February 2006. doi:10.1007/11605805_10.

GKM$^+$18. Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Heidelberg, August 2018. doi:10.1007/978-3-319-96878-0_24.

GM17. Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 581–612. Springer, Heidelberg, August 2017. doi:10.1007/978-3-319-63715-0_20.

GO94. Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, December 1994. doi:10.1007/BF00195207.

GPS08. Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for Cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

GR16. Alonso González and Carla Ràfols. New techniques for non-interactive shuffle and range arguments. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *ACNS 16*, volume 9696 of *LNCS*, pages 427–444. Springer, Heidelberg, June 2016. doi:10.1007/978-3-319-39555-5_23.

Gro10. Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, December 2010. doi:10.1007/978-3-642-17373-8_19.

Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016. doi:10.1007/978-3-662-49896-5_11.

GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011. doi:10.1145/1993636.1993651.

GWC19. Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. https://eprint.iacr.org/2019/953.

JR13. Charanjit S. Jutla and Arnab Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2013. doi:10.1007/978-3-642-42033-7_1.

JR14. Charanjit S. Jutla and Arnab Roy. Switching lemma for bilinear tests and constant-size NIZK proofs for linear subspaces. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of

LNCS, pages 295–312. Springer, Heidelberg, August 2014. doi:10.1007/978-3-662-44381-1_17.

Kle08. Achim Klenke. *Probability Theory: A Comprehensive Course*. Universitext. Springer, 1 edition, 2008.

KW15. Eike Kiltz and Hoeteck Wee. Quasi-adaptive NIZK for linear subspaces revisited. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 101–128. Springer, Heidelberg, April 2015. doi:10.1007/978-3-662-46803-6_4.

Lip10. Helger Lipmaa. On the CCA1-Security of Elgamal and Damgård's Elgamal. In Xuejia Lai, Moti Yung, and Dongdai Lin, editors, *Inscrypt 2010*, volume 6584 of *LNCS*, pages 18–35, Shanghai, China, October 20–23, 2010. Springer, Heidelberg. doi:https://doi.org/10.1007/978-3-642-21518-6_2.

Lip12. Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Heidelberg, March 2012. doi:10.1007/978-3-642-28914-9_10.

Lip13. Helger Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 41–60. Springer, Heidelberg, December 2013. doi:10.1007/978-3-642-42033-7_3.

Lip19a. Helger Lipmaa. Key-and-Argument-Updatable QA-NIZKs. Technical Report 2019/333, IACR, March 27, 2019. https://eprint.iacr.org/2019/333.

Lip19b. Helger Lipmaa. Simulation-Extractable ZK-SNARKs Revisited. Technical Report 2019/612, IACR, May 31, 2019. https://eprint.iacr.org/2019/612, updated on 8 Feb 2020.

LPJY14. Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. Non-malleability from malleability: Simulation-sound quasi-adaptive NIZK proofs and CCA2-secure encryption from homomorphic signatures. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 514–532. Springer, Heidelberg, May 2014. doi:10.1007/978-3-642-55220-5_29.

LPJY15. Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. Compactly hiding linear spans - tightly secure constant-size simulation-sound QA-NIZK proofs and applications. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 681–707. Springer, Heidelberg, November / December 2015. doi:10.1007/978-3-662-48797-6_28.

MBKM19. Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019. doi:10.1145/3319535.3339817.

MR01. Silvio Micali and Leonid Reyzin. Soundness in the public-key model. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 542–565. Springer, Heidelberg, August 2001. doi:10.1007/3-540-44647-8_32.

MRV16. Paz Morillo, Carla Ràfols, and Jorge Luis Villar. The kernel matrix Diffie-Hellman assumption. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 729–758.

Springer, Heidelberg, December 2016. doi:10.1007/978-3-662-53887-6_27.

Nao03. Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 96–109. Springer, Heidelberg, August 2003. doi:10.1007/978-3-540-45146-4_6.

PHGR13. Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013. doi:10.1109/SP.2013.47.

Wee07. Hoeteck Wee. Lower bounds for non-interactive zero-knowledge. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 103–117. Springer, Heidelberg, February 2007. doi:10.1007/978-3-540-70936-7_6.

A Assumptions

Next, we define five commonly used distributions (see [EHK$^+$13] 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):

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

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

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

$\mathcal{D}_{\ell k}$-KerMDH$_{\mathbb{G}_\iota}$ [MRV16] holds relative to Pgen, if for all PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{kermdh}}_{\mathcal{A},\mathcal{D}_{\ell k},\iota,\mathsf{Pgen}}(\lambda) \approx_\lambda 0$, where $\mathsf{Adv}^{\mathrm{kermdh}}_{\mathcal{A},\mathcal{D}_{\ell k},\iota,\mathsf{Pgen}}(\lambda)$ is the probability that given $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, $\boldsymbol{A} \leftarrow_\$ \mathcal{D}_{\ell k}$, $[\boldsymbol{c}]_{3-\iota} \leftarrow \mathcal{A}(\mathsf{p}, [\boldsymbol{A}]_\iota)$, the following holds: $\boldsymbol{A}^\top \boldsymbol{c} = \boldsymbol{0}_k \wedge \boldsymbol{c} \neq \boldsymbol{0}_\ell$.

$\mathcal{D}_{\ell k}$-SKerMDH [GHR15] holds relative to Pgen, if for all PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{skermdh}}_{\mathcal{A},\mathcal{D}_{\ell k},\mathsf{Pgen}}(\lambda) \approx_\lambda 0$, where $\mathsf{Adv}^{\mathrm{skermdh}}_{\mathcal{A},\mathcal{D}_{\ell k},\mathsf{Pgen}}(\lambda)$ is the probability that given $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, $\boldsymbol{A} \leftarrow_\$ \mathcal{D}_{\ell k}$, $([\boldsymbol{c}_1]_1, [\boldsymbol{c}_2]_2) \leftarrow \mathcal{A}(\mathsf{p}, [\boldsymbol{A}]_1, [\boldsymbol{A}]_2)$, the following holds: $\boldsymbol{A}^\top (\boldsymbol{c}_1 - \boldsymbol{c}_2) = \boldsymbol{0}_k \wedge \boldsymbol{c}_1 - \boldsymbol{c}_2 \neq \boldsymbol{0}_\ell$.

According to Lem. 1 of [GHR15], if $\mathcal{D}_{\ell k}$-KerMDH holds in generic symmetric bilinear groups then $\mathcal{D}_{\ell k}$-SKerMDH holds in generic asymmetric bilinear groups. KerMDH holds also for Type-I pairings, where $\mathbb{G}_1 = \mathbb{G}_2$. However, in such case we need $k \geq 2$, which affects efficiency of the arguments relying on it.

The $\mathcal{D}_{\ell k}$-KerMDH$^{\mathrm{dl}}_{\mathbb{G}_1}$ *assumption* [ALSZ20] holds relative to Pgen, if for any PPT $\mathcal{A}$, given $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$; $st \leftarrow \mathcal{A}^{\mathrm{dl}(\cdot)}(\mathsf{p})$; $\boldsymbol{A} \leftarrow_\$ \mathcal{D}_{\ell k}$; $[\boldsymbol{c}]_2 \leftarrow \mathcal{A}(\mathsf{p}, st, [\boldsymbol{A}]_1)$, the following holds with a negligible probability: $\boldsymbol{A}^\top \boldsymbol{c} = \boldsymbol{0}_k \wedge \boldsymbol{c} \neq \boldsymbol{0}_\ell$.

The $\mathcal{D}_{\ell k}$-SKerMDH$^{\mathrm{dl}}$ *assumption* [ALSZ20] holds relative to Pgen, if for any PPT $\mathcal{A}$, given $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, $st \leftarrow \mathcal{A}^{\mathrm{dl}(\cdot)}(\mathsf{p})$, $\boldsymbol{A} \leftarrow_\$ \mathcal{D}_{\ell k}$, $([\boldsymbol{c}_1]_1, [\boldsymbol{c}_2]_2) \leftarrow$

$\mathcal{A}(\mathsf{p}, st, [\boldsymbol{A}]_1, [\boldsymbol{A}]_2)$, the following holds with a negligible probability: $\boldsymbol{A}^\top(\boldsymbol{c}_1 - \boldsymbol{c}_2) = \boldsymbol{0}_k \wedge \boldsymbol{c}_1 - \boldsymbol{c}_2 \neq \boldsymbol{0}_\ell$.

The following assumption is tautological to the Kiltz-Wee QA-NIZK Π'_{as}. Here, $\mathrm{PK}_{\mathsf{V}_{\mathsf{pk}}}$ denotes the part of PK added to PK only to make V_{pk} efficient. As explained in [ALSZ20], $\mathrm{PK}_{\mathsf{V}_{\mathsf{pk}}} = \epsilon$ if $k = 1$; in general, $\mathrm{PK}_{\mathsf{V}_{\mathsf{pk}}}$ contains information needed to efficiently check that $\bar{\boldsymbol{A}}$ is invertible..

Fix $k \geq 1$, $n > m \geq 1$, and a distribution $\mathcal{D}_k$. Let V_{pk} be as in Fig. 2. Then $(\mathcal{D}_{\mathsf{p}}, k, \mathcal{D}_k)$-$\mathsf{KW}$-$\mathsf{KE}_{\mathbb{G}_1}$ holds relative to Pgen if for any PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathsf{Ext}_{\mathcal{A}}$, s.t. given $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, $r \leftarrow_{\$} \mathsf{RND}_\lambda(\mathcal{A})$, $(\mathsf{lpar} := [\boldsymbol{M}]_1, \mathrm{PK}) \leftarrow \mathcal{A}(\mathsf{p}; r)$, $\boldsymbol{K} \leftarrow \mathsf{Ext}_{\mathcal{A}}(\mathsf{p}; r)$, the following holds with a negligible probability: $\mathrm{PK} = ([\bar{\boldsymbol{A}}, \boldsymbol{C}]_2, [\boldsymbol{P}]_1, \mathrm{PK}_{\mathsf{V}_{\mathsf{pk}}}) \wedge \mathsf{V}_{\mathsf{par}}([\boldsymbol{M}]_1) = 1 \wedge \mathsf{V}_{\mathsf{pk}}([\boldsymbol{M}]_1, \mathrm{PK}) = 1 \wedge \boldsymbol{P} \neq \boldsymbol{M}^\top \boldsymbol{K}$.

Assume that either $\mathcal{D}_k$ is efficiently verifiable or $\mathcal{D}_k = \mathcal{U}_2$. Assume $k/p \approx_\lambda 0$. Then $(\mathcal{D}_{\mathsf{p}}, k, \mathcal{D}_k)$-$\mathsf{KW}$-$\mathsf{KE}_{\mathbb{G}_1}$ holds under the $\mathcal{D}_k$-HAK assumption from [Lip19b], [ALSZ20].

B Remaining Proofs

B.1 Proof of Lemma 7

Proof (Of Lemma 7). The proof follows from Lemma 5 ($\Pi^{\mathsf{upd}}_{\mathsf{bpk}}$ is key-update complete), Proposition 1 (Π_{bpk} is zero-knowledge / persistent Sub-ZK), and Lemma 3 (key-update zero-knowledge / persistent Sub-ZK follows from key-update completeness and zero-knowledge / persistent Sub-ZK). □

B.2 Proof of Lemma 8

Proof (Of Lemma 8). By Lemma 4, argument-update zero-knowledge / persistent Sub-ZK follows from key-update completeness, simulator-update completeness (both proven in Lemma 5) and zero-knowledge / persistent Sub-ZK (proven in Proposition 1). The result now follows from the precise security statement in Proposition 1. □