Towards Instantiating the Algebraic Group Model

Julia Kastner[1] and Jiaxin Pan*[2]

[1] Karlsruhe Institute of Technology, Karlsruhe, Germany
julia.kastner@kit.edu
[2] Norwegian University of Science and Technology, Trondheim, Norway
jiaxin.pan@ntnu.no

Abstract. The Generic Group Model (GGM) is one of the most important tools for analyzing the hardness of a cryptographic problem. Although a proof in the GGM provides a certain degree of confidence in the problem's hardness, it is a rather strong and limited model, since it does not allow an algorithm to exploit any property of the group structure. To bridge the gap between the GGM and the Standard Model, Fuchsbauer, Kiltz, and Loss proposed a model, called the Algebraic Group Model (AGM, CRYPTO 2018). In the AGM, an adversary can take advantage of the group structure, but it needs to provide a representation of its group element outputs, which seems weaker than the GGM but stronger than the Standard Model. Due to this additional information we learn about the adversary, the AGM allows us to derive simple but meaningful security proofs.

In this paper, we take the first step to bridge the gap between the AGM and the Standard Model. We instantiate the AGM under Standard Assumptions. More precisely, we construct two algebraic groups under the Knowledge of Exponent Assumption (KEA). In addition to the KEA, our first construction requires symmetric pairings, and our second construction needs an additively homomorphic Non-Interactive Zero-Knowledge (NIZK) argument system, which can be implemented by a standard variant of Diffie-Hellman Assumption in the asymmetric pairing setting. Furthermore, we show that both of our constructions provide cryptographic hardness which can be used to construct secure cryptosystems. We note that the KEA provably holds in the GGM. Our results show that, instead of instantiating the seemingly complex AGM directly, one can try to instantiate the GKEA under falsifiable assumptions in the Standard Model. Thus, our results can serve as a stepping stone towards instantiating the AGM under falsifiable assumptions.

Keywords. Public-key cryptography, algebraic group model, generic group model, knowledge of exponent assumption.

* Work was conducted while he was employed by KIT, Germany under the DFG grant HO 4534/4-1.

1 Introduction

1.1 Motivation

Modern public-key cryptographic schemes are usually proposed with security proofs. A security proof is commonly a reduction relating the security of a scheme to the hardness of the underlying cryptographic problem (for instance, the discrete logarithm (DLOG) problem). Naturally, the harder the underlying problem is, the stronger the security guarantee that a scheme can provide.

From generic groups to algebraic groups. Since 1994, several works have been analysing the hardness of cryptographic problems over cyclic groups in the generic group model (GGM) [37,16,43,36,35] and some works have used the GGM to directly argue the security of cryptographic schemes [19,41,5,7]. In the GGM, an algorithm can only perform abstract group operations and check the equality of two group elements. On the one hand, due to the fact that the algorithm's operations are generic, algorithms in the GGM can be carried on to any cyclic group, and it allows us to show information-theoretical lower bounds of a generic algorithm's running time. On the other hand, a generic algorithm cannot exploit any special properties of the encoding of a group element, and thus a security argument in the generic group model can only provide weaker guarantees than one in the Standard Model. Therefore, a security proof in the GGM only shows that there is no "trivial" attack on the proven scheme or problem.

In order to overcome this limitation, the Algebraic Group Model (AGM) which models all adversaries as *algebraic* was proposed at CRYPTO 2018 [25]. Informally, an algorithm $\mathcal{A}_{\mathsf{alg}}$ is algebraic if, whenever $\mathcal{A}_{\mathsf{alg}}$ outputs a group element, one can efficiently extract a representation of the output element w.r.t $\mathcal{A}_{\mathsf{alg}}$'s input group elements. To be a bit more formal, let $\mathbb{H}$ be a cyclic group with prime order q. Whenever an algebraic algorithm $\mathcal{A}_{\mathsf{alg}}$ outputs an element $[y]_{\mathbb{H}} \in \mathbb{H}$[3], there is an efficient extractor $\mathcal{E}$ that computes a *representation* $\mathbf{z} := (z_1, ..., z_n) \in \mathbb{Z}_q^n$ of $[y]_{\mathbb{H}}$ such that $[y]_{\mathbb{H}} = \prod_{i=1}^{n} [z_i x_i]_{\mathbb{H}}$, where $[\mathbf{x}]_{\mathbb{H}} := ([x_1]_{\mathbb{H}}, \ldots, [x_n]_{\mathbb{H}}) \in \mathbb{H}^n$ is the list of all group elements which were given to $\mathcal{A}_{\mathsf{alg}}$ during its run so far.

Applications of the AGM. Algebraic algorithms were firstly considered by Boneh and Venkatesan [18] to analyse the hardness of the RSA problem based on the factoring problem. After that, algebraic algorithms have been widely used to prove impossibility results on cryptographic schemes (for instance, [38,6,3]). Interestingly, all the early stage research only considers reductions to be algebraic, in order to disprove the existence of an algebraic security reduction between two primitives with certain good quality. For instance, it has been used to show that the security of Schnorr's signature [40] cannot be proven based on the DLOG assumption without random oracles via algebraic reductions [38], and that the non-tight security reduction [39] of Schnorr's signature based on DLOG is an optimal algebraic reduction in terms of security loss [42]. In particular, algebraic

[3] In this paper, we follow the implicit notion [23] of a group element. We write $[y]_{\mathbb{H}}$ as an element from a group $\mathbb{H}$ instead of h^y, where h is the generator of $\mathbb{H}$ and $y \in \mathbb{Z}_q$.

algorithms are important tools (or even the only tool) to show the lower bounds of structure-preserving cryptographic primitives [6,3,2].

Different to the previous works, Fuchsbauer, Kiltz, and Loss [25] not only define algebraic algorithms formally via the AGM, but also consider algebraic algorithms in the role of adversaries in a security game and prove positive results on cryptographic assumptions and schemes. Security implications in the AGM are similar to the standard model, namely, they are done by constructing reductions, but the reductions have the representation of a group element from the adversary as additional helper information. Due to the additional information available about the adversary, the reduction becomes simpler. In particular, some of the security implications in [25] even bypass certain known lower bounds on the reduction quality.

We highlight their security reductions for two important schemes: the BLS signature [17] and the most efficient zero-knowledge SNARK scheme so far by Groth [27]. Interestingly, their tight reduction on the BLS signature based on the DLOG assumption has bypassed its security loss lower bound [21,31] and it also has the practical impact that one can implement the BLS signature with shorter keys an an algebraic group. Moreover, the previous proof on the Groth SNARK is only in the GGM. A security proof in the AGM can offer stronger security guarantee, since an adversary is allowed exploit special properties of the group structure.

The problem. Behind all these interesting results stands the fundamental question whether we can trust the AGM or, more precisely, whether there is an algebraic group in the Standard Model.

Intuitively, it seems reasonable to assume that group-specific algorithms and reductions are algebraic, since most of them are indeed algebraic (cf. reductions in [42,12,1]). However, there is still no formal proof on the existence of an algebraic group. Furthermore, both the impossibility results (for instance, [6,42]) and the positive results in [25] require an explicit extraction of the representation. If one wants to carry the proven results in the AGM over to the Standard Model, a promising way is to simulate the AGM with assumptions in the Standard Model. Thus, the question whether there exists a group where algorithms are algebraic becomes very interesting.

1.2 Our contribution

We carry out the *first* formal treatment on the feasibility of the AGM. More precisely, we construct groups that are algebraic under the (Generalized) Knowledge of Exponent Assumption ((G)KEA) [22,10,44]. We propose two different constructions. They are both based on the GKEA. Our two constructions require additional primitives to check group membership. Our first construction uses pairings of Type 1 or 2[4] for membership verification, and our second construction

[4] Type 2 pairing $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ is an asymmetric pairing, namely, $\mathbb{G} \neq \hat{\mathbb{G}}$, where there exists an efficient homomorphism from $\mathbb{G}$ to $\hat{\mathbb{G}}$.

uses an additively homomorphic Non-Interactive Zero-Knowledge (NIZK) Argument system for subspace membership. An example for such a proof system is the Kiltz-Wee scheme [33] which requires a Type 3 pairing where the Kernel Matrix Diffie-Hellman assumption holds in the second source group. We note however that it is possible to use any additively homomorphic NIZK Argument system of subspace membership for our construction, even one without a pairing, if such a scheme exists. Our results offer a foundation for the aforementioned positive and negative results on cryptographic schemes. In particular, in both algebraic groups constructed by us, all the previous security results can be transformed to the Standard Model by assuming the GKEA. We note that the GKEA holds provably in the GGM. Our results are just to bring the AGM closer to the Standard Model and essentially we show that, instead of instantiating the seemingly complex AGM directly, one can try to instantiate the GKEA under falsifiable assumptions in the Standard Model.

We provide a technical overview of our construction. By definition an algebraic group (denoted by $\mathbb{H}$) needs to offer two properties:

- Extractability: there is an efficient extraction algorithm that extracts the representation of an $\mathbb{H}$ element output by $\mathcal{A}_{\mathsf{alg}}$ w.r.t its input group elements from $\mathbb{H}$.
- Cryptographic hardness: since the algebraic group is also used to show security implications of schemes, there should be some cryptographic hard problem over the algebraic group.

Task 1: Achieving extractability. Intuitively, it seems hard to compute any representation over $\mathbb{Z}_q$ from only group elements, since the DLOG problem is hard. However, if the given group elements satisfy certain relations, the KEA (cf. [22] and KEA1 in [10]) can provide an alternative way to extract a representation. More precisely, let $\mathbb{G}$ be a group with prime order q, where the KEA holds and $[1]_{\mathbb{G}} := g$ is a random generator of $\mathbb{G}$.

Definition 1 (The KEA, informal). *For an efficient adversary $\mathcal{A}$ that, given $(q, [1]_{\mathbb{G}}, [a]_{\mathbb{G}})$ and a random tape $r_{\mathcal{A}}$, outputs $([b]_{\mathbb{G}}, [ab]_{\mathbb{G}})$, there exists an efficient extractor $\mathcal{E}_{\mathsf{kea}}$ that, given the same inputs as $\mathcal{A}$ and its output elements $([b]_{\mathbb{G}}, [ab]_{\mathbb{G}})$, returns $b \in \mathbb{Z}_q$.*

Essentially, the KEA states that the only way to compute $([b]_{\mathbb{G}}, [ab]_{\mathbb{G}})$ efficiently, given $(q, [1]_{\mathbb{G}}, [a]_{\mathbb{G}})$, is to "know" b over $\mathbb{Z}_q$.

Starting with $\mathbb{G}$ where the KEA holds, we have our first candidate of an algebraic group $\mathbb{H}$. Choosing $a \xleftarrow{\$} \mathbb{Z}_q$, we define $\mathbb{H} \subsetneq \mathbb{G} \times \mathbb{G}$ and an element $[x]_{\mathbb{H}}$ $(x \in \mathbb{Z}_q)$ has the form

$$[x]_{\mathbb{H}} := ([x]_{\mathbb{G}}, [ax]_{\mathbb{G}}). \tag{1}$$

It seems that $\mathbb{H}$ almost achieves extractability with the KEA extractor. For simplicity, we consider a very simple adversary $\mathcal{A}$, which only takes the group generator of $\mathbb{H}$, $[1]_{\mathbb{H}}$, and outputs a vector $([y_1]_{\mathbb{G}}, [y_2]_{\mathbb{G}})$. In order to make $\mathbb{H}$ extractable, we need to solve the following two problems: (i) We need to ensure the output vector $([y_1]_{\mathbb{G}}, [y_2]_{\mathbb{G}})$ is in $\mathbb{H}$, namely, the vector satisfies Equation (1). That is mainly because the KEA extractor can only compute the representation

y_1 w.r.t $[1]_{\mathbb{H}}$ if $y_2 = ay_1$. Moreover, checking whether an element belongs to a group is usually implicitly used in a scheme (for instance, in the BLS signature). This group membership check should be publicly available. (ii) The extraction should work in the general case, where $\mathcal{A}$ gets more than only a group generator. For instance, $\mathcal{A}$ can receive a vector of group elements.

Solution to Problem (i): Pairing-based validation. In order to verify group membership, we use a base group $\mathbb{G}$ for which a pairing $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ exists. We consider Type 1 ($\mathbb{G} = \hat{\mathbb{G}}$) and 2 ($\mathbb{G} \neq \hat{\mathbb{G}}$, but it is efficient to compute $[a]_{\hat{\mathbb{G}}}$ from $[a]_{\mathbb{G}}$) pairing groups, respectively. In both cases, we can efficiently check the group membership of a pair $([x]_{\mathbb{G}}, [y]_{\mathbb{G}})$ by checking the pairing equation $e([x]_{\mathbb{G}}, [a]_{\hat{\mathbb{G}}}) = e([y]_{\mathbb{G}}, [1]_{\hat{\mathbb{G}}})$. We note that some decisional assumptions, such as the Decisional Diffie-Hellman assumption, are easy in $\mathbb{G}$ when $\mathbb{G}$ is a base group of a Type 1 or 2 pairing. Therefore we cannot hope for the same decisional assumption in $\mathbb{H}$ which is constructed from $\mathbb{G}$, since by the definition of our construction the problem challenge in $\mathbb{H}$ can be easily mapped back to $\mathbb{G}$. Thus, we need to consider a more general construction.

More General Solution to Problem (i): NIZK. We observe that Equation (1) defines a linear subspace in $\mathbb{G} \times \mathbb{G}$ generated by $[\mathbf{M}]_{\mathbb{G}} := \left[\begin{pmatrix} 1 \\ a \end{pmatrix}\right]_{\mathbb{G}}$. To achieve public membership verification, we can attach a NIZK argument in the encoding of an $\mathbb{H}$ element (defined by Equation (1)) and the NIZK argument proves that the vector $([y_1]_{\mathbb{G}}, [y_2]_{\mathbb{G}})^\top$ is in the span of $[\mathbf{M}]_{\mathbb{G}}$. Our second construction of an algebraic group $\mathbb{H} := \mathsf{Span}([\mathbf{M}]_{\mathbb{G}}) \times \Pi$ and an element $[x]_{\mathbb{H}}$ ($x \in \mathbb{Z}_q$) has the form

$$[x]_{\mathbb{H}} := (([x]_{\mathbb{G}}, [ax]_{\mathbb{G}})^\top, \pi), \tag{2}$$

where $\mathsf{Span}([\mathbf{M}]_{\mathbb{G}}) := \{[\mathbf{M}w]_{\mathbb{G}} | w \in \mathbb{Z}_q\}$, π is a NIZK argument for language $\mathcal{L}_{[\mathbf{M}]_{\mathbb{G}}} := \left\{[\mathbf{y}]_{\mathbb{G}} \in \mathbb{G}^2 \middle| \exists w \in \mathbb{Z}_q \text{ such that } [\mathbf{y}]_{\mathbb{G}} = [\mathbf{M}w]_{\mathbb{G}}\right\}$ and Π is the set of the π values from the NIZK system.

We reconsider the previous simple adversary $\mathcal{A}$, which outputs $[y]_{\mathbb{H}} := (([y_1]_{\mathbb{G}}, [y_2]_{\mathbb{G}})^\top, \pi')$, given $[1]_{\mathbb{H}} := (([1]_{\mathbb{G}}, [a]_{\mathbb{G}})^\top, \pi)$. The soundness of the NIZK system guarantees that the output vector $([y_1]_{\mathbb{G}}, [y_2]_{\mathbb{G}})^\top \in \mathsf{Span}([\mathbf{M}]_{\mathbb{G}})$ (i.e. $y_2 = ay_1$), and thus one can use the KEA extractor to compute the representation y_1 of $[y]_{\mathbb{H}}$ w.r.t. $\mathcal{A}$'s input $[1]_{\mathbb{H}}$.

This NIZK system can be implemented with the Groth-Sahai system [28]. In this paper, we use a more efficient and simpler Quasi-Adaptive NIZK (QANIZK) for linear subspaces from Kiltz and Wee [33], since the basis $[\mathbf{M}]_{\mathbb{G}}$ can be generated in the setup phase. The notion of QANIZK was first proposed by Jutla and Roy [30]. The Kiltz-Wee scheme requires an asymmetric Type 3 pairing, $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$, and its soundness is based on the Kernel Matrix Diffie-Hellman (KerMDH) assumption in $\hat{\mathbb{G}}$. Thus, our construction also needs these two requirements.

Additionally, one should be able to add two group elements in $\mathbb{H}$, and, therefore, our construction needs an additively homomorphic property of the QANIZK system: Given two valid proofs π_1 and π_2 for two vector $[\mathbf{v}_1]_{\mathbb{G}}, [\mathbf{v}_2]_{\mathbb{G}} \in \mathsf{Span}([\mathbf{M}]_{\mathbb{G}})$,

respectively, one can publicly compute the sum of π_1 and π_2; moreover, the sum of π_1 and π_2 should be a valid proof for the sum of $[\mathbf{v}_1]_{\mathbb{G}}$ and $[\mathbf{v}_2]_{\mathbb{G}}$. We observe that the Kiltz-Wee scheme has this additively homomorphic property.

We emphasize that our second construction is not necessarily limited to pairings, although we only have an instantiation based on pairings. If we have an additively homomorphic QANIZK system without pairings, then we can instantiate the AGM without pairings via our second construction. We leave constructing an additively homomorphic QANIZK system without pairings as an interesting open problem.

Solution to Problem (ii): GKEA. We require the generalized KEA (GKEA) [44] to compute the representation in the general case.

Definition 2 (The GKEA, informal). *For an efficient adversary $\mathcal{A}$, given $(q, [1]_{\mathbb{G}}, [a]_{\mathbb{G}}, [x_1]_{\mathbb{G}}, [ax_1]_{\mathbb{G}}, \ldots, [x_n]_{\mathbb{G}}, [ax_n]_{\mathbb{G}})$ (for some $x_1, ..., x_n \in \mathbb{Z}_q$ and $a \xleftarrow{\boxtimes} \mathbb{Z}_q$) and a random tape $r_{\mathcal{A}}$, outputs $([b]_{\mathbb{G}}, [ab]_{\mathbb{G}})$, there is an efficient extractor $\mathcal{E}_{\mathsf{gkea}}$ given the same inputs as $\mathcal{A}$ returns a vector $(z_0, ..., z_n)^\top =: \mathbf{z} \in \mathbb{Z}_q^{n+1}$ such that $b = z_0 + \sum_{i=1}^{n} x_i z_i$.*

Let $\mathcal{A}$ be a general adversary which gets $q, [1]_{\mathbb{H}}, [x_1]_{\mathbb{H}}, ..., [x_n]_{\mathbb{H}}$ as input and outputs an element in $\mathbb{H}$, $[y]_{\mathbb{H}} := (([y_1]_{\mathbb{G}}, [y_2]_{\mathbb{G}})^\top, \pi_y)$, where $[x_i]_{\mathbb{H}} := (([x_{i,1}]_{\mathbb{G}}, [x_{i,2}]_{\mathbb{G}})^\top, \pi_i)$. Similar to the simple adversary, by the correctness of the group membership validation algorithm, the input and output elements of the general adversary have the right format, namely, $x_{i,2} = ax_{i,1}$ and $y_2 = ay_1$. Hence, by running the extractor of the GKEA, we can obtain the representation of $[y]_{\mathbb{H}}$ w.r.t. $([1]_{\mathbb{H}}, [x_1]_{\mathbb{H}}, ..., [x_n]_{\mathbb{H}})$.

If an algorithm $\mathcal{A}$ outputs more than one (say, ℓ-many) elements in $\mathbb{H}$, then we run the above extraction ℓ times to compute the representations of all these elements.

Task 2: Achieving cryptographic hardness. We provide hard cryptographic problems in $\mathbb{H}$, since security implications in the AGM rely on assumptions. We are interested in the DLOG and variants of Diffie-Hellman assumptions, which are used to prove security of schemes such as the BLS and CL [20] signatures and Groth's SNARK. Instead of going through these assumptions one by one, we use the notion of *security games* where the challenger provides a set of oracle procedures to the adversary, including an initialisation oracle that starts the game, and a finalisation oracle that the adversary calls with his final output.

We prove that if the game is hard to win in the base group $\mathbb{G}$ then the same game is also hard to win in the resulting algebraic group $\mathbb{H}$ (denoted by $\mathcal{P}_{\mathbb{G}} \Rightarrow \mathcal{P}_{\mathbb{H}}$). We use the DLOG problem to sketch our strategy here and a formal treatment is provided in Sections 3.1 and 4.1.

Let $\mathcal{A}$ be an adversary that breaks the DLOG problem in $\mathbb{H}$. We get the DLOG challenge $([1]_{\mathbb{G}}, [x]_{\mathbb{G}})$ in the base group $\mathbb{G}$ from the initialisation procedure and we make use of $\mathcal{A}$ to compute $x \in \mathbb{Z}_q$. We choose a random $a \xleftarrow{\boxtimes} \mathbb{Z}_q$ and, in case of construction with membership verification through QANIZK, we also generate the CRS of the QANIZK and keep its trapdoor. Now we need to provide $\mathcal{A}$ with $([1]_{\mathbb{H}}, [x]_{\mathbb{H}})$. It is easy to generate $[1]_{\mathbb{H}}$. For $[x]_{\mathbb{H}}$, we can

compute $[ax]_{\mathbb{G}}$ by multiplying $[x]_{\mathbb{G}}$ with an explicit $a \in \mathbb{Z}_q$. However, since $[x]_{\mathbb{G}}$ is from the DLOG challenger in $\mathbb{G}$, we do not have the witness $x \in \mathbb{Z}_q$ to show $([x]_{\mathbb{G}}, [ax]_{\mathbb{G}})^{\top} \in \mathsf{Span}([\mathbf{M}]_{\mathbb{G}})$. Instead, we use the zero-knowledge simulator to simulate the proof for the above statement.

To capture all our ideas formally, we use the notion of *group schemes* [9,8] to define our algebraic group $\mathbb{H}$.

1.3 More related work and open problems

Our representation extraction is heavily based on the KEA. We suppose that it is very unlikely that the KEA does not hold, due to an analysis of the KEA over bilinear groups in the GGM [4]. Moreover, it is worth mentioning that the (generalized) KEA is widely used in the literature. It has been used to bypass some impossibility results in the (non-interactive) zero-knowledge protocols [10,4], as a means to construct extractable collision resistant hash functions and SNARK [13], and to prove the the security of the well-known key exchange protocol HMQV (cf. Theorem 18 in the full version of [34]).

As a further generalisation of the KEA3, the GKEA and AI-KEA were introduced by [44] to use for identification protocols. We note here that the GKEA – the assumption that it is possible to extract exponents even when the adversary is given multiple KEA-tuples – implies the AI-KEA – the assumption that it is possible to extract exponents from a set of multiple turing machines that communicate with each other, sending and receiving KEA-tuples.

The KEA is a *non-black-box* assumption, since its extractor needs to know an adversary's random tape. A non-black-box assumption seems crucial for instantiating the AGM. Imagine the BLS signature: there is a tight reduction for it in the AGM, but it is provably impossible in the Standard Model. If we instantiate the AGM without any non-black-box access to the adversary, then we will contradict the proven impossibility results on the BLS signature [21,31]. We leave as an open problem to instantiate the AGM with other non-black-box assumptions in such a way that we can have more confidence in the AGM.

2 Preliminaries

2.1 Notations

Let A be a set. $a \xleftarrow{\boxtimes} A$ denotes picking a from A according to the uniform distribution. Let $\mathcal{A}$ be an probabilistic polynomial time (PPT) algorithm. $a \xleftarrow{\boxtimes} \mathcal{A}(b)$ denotes the random variable which is defined as the output of $\mathcal{A}$ on input b. If we want to denote the randomness $r_{\mathcal{A}}$ of $\mathcal{A}$ explicitly, we write $\mathcal{A}(b; r_{\mathcal{A}})$. We denote our security parameter as λ.

2.2 Cryptographic assumptions

Pairing groups and matrix Diffie-Hellman. We consider finite cyclic groups with prime order in this paper. We say that a group is *cyclic* if it is generated by a single element, the *generator* g.

We also use pairing groups for which we recall the following definition from [32]: Let GGen be a probabilistic polynomial time algorithm that on input 1^λ returns $\mathcal{PG} = \left(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, g_1, g_2, e\right)$ where $\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T$ are groups of order q for a λ-bit prime q and g_1, g_2 are generators of $\mathbb{G}$ and $\hat{\mathbb{G}}$, respectively. $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ is an efficiently computable non-degenerate bilinear map where $g_T = e(g_1, g_2)$ is a generator in $\mathbb{G}_T$. We consider three different types of pairings as in [26]:

- Type 1: $\mathbb{G} = \hat{\mathbb{G}}$,
- Type 2: $\mathbb{G} \neq \hat{\mathbb{G}}$ but there is an efficiently computable homomorphism[5] $\psi : \mathbb{G} \to \hat{\mathbb{G}}$, and
- Type 3: $\mathbb{G} \neq \hat{\mathbb{G}}$ and there is no efficiently computable homomorphism between $\mathbb{G}$ and $\hat{\mathbb{G}}$.

We follow [23] to use the implicit notation for a group element. For a cyclic group $\mathbb{G}$ with prime order q and generator g we write $[a]_\mathbb{G}$ for elements g^a with $a \in \mathbb{Z}_q$. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ with entries a_{ij} ($1 \leq i \leq n$ and $1 \leq j \leq m$), we write $[\mathbf{A}]_\mathbb{G}$ for the matrix in $\mathbb{G}^{n \times m}$ that has the entries $[a_{ij}]_\mathbb{G}$. For $[\mathbf{X}]_\mathbb{G} \in \mathbb{G}^{a \times b}$ and $[\mathbf{Y}]_{\hat{\mathbb{G}}} \in \hat{\mathbb{G}}^{b \times c}$, we write $[\mathbf{X}]_\mathbb{G} \circ [\mathbf{Y}]_{\hat{\mathbb{G}}}$ for $e([\mathbf{X}]_\mathbb{G}, [\mathbf{Y}]_{\hat{\mathbb{G}}}) := [\mathbf{XY}]_{\mathbb{G}_T}$.

Definition 3 (The Kernel Matrix Diffie-Hellman assumption). *Let $\mathcal{D}_k$ be a matrix distribution and $s \in \{1, 2\}$. We say that the $\mathcal{D}_k$ Kernel Matrix Diffie-Hellman ($\mathcal{D}_k$-KerMDH) Assumption holds relative to Gen in group $\mathbb{G}_s$ if for all PPT-adversaries $\mathcal{A}$*

$$\mathrm{Adv}^{\mathcal{D}_k\text{-}\mathsf{kmdh}}_{\mathbb{G}_s,\mathcal{A}}(\lambda) := \Pr\left[\begin{bmatrix}\mathbf{c}^\top\mathbf{A} = \mathbf{0}\\ \wedge \\ \mathbf{c} \neq \mathbf{0}\end{bmatrix} \middle| [\mathbf{c}]_{\mathbb{G}_{3-s}} \xleftarrow{\$} \mathcal{A}(\mathsf{par}, [\mathbf{A}]_{\mathbb{G}_s})\right]$$

is negligible, where the probability is taken over $\mathsf{par} \xleftarrow{\$} \mathsf{GGen}(1^\lambda), \mathbf{A} \xleftarrow{\$} \mathcal{D}_k$.

We recall the Knowledge of Exponent assumption that was introduced in [22] and has since been used in several papers [29,10,34].

Definition 4 (The Knowledge of Exponent Assumption, KEA). *Let $\mathbb{G}$ be a cyclic group of prime order q. The Knowledge of exponent assumption states that for any adversary $\mathcal{A}$ that on input $[1]_\mathbb{G}, [a]_\mathbb{G}$ outputs a tuple $[b]_\mathbb{G}, [c]_\mathbb{G}$ there exists a PPT-extractor $\mathcal{E}_\mathcal{A}$ that takes inputs of $\mathcal{A}$ and any random coins $r_\mathcal{A}$ of $\mathcal{A}$ and outputs $b' \in \mathbb{Z}_q$ such that*

$$\mathrm{Adv}^{\mathsf{kea}}_{\mathbb{G},\mathcal{A},\mathcal{E}_\mathcal{A}}(\lambda) := \Pr\left[\begin{bmatrix}[b']_\mathbb{G} \neq [b]_\mathbb{G}\\ \wedge \\ [ab]_\mathbb{G} = [c]_\mathbb{G}\end{bmatrix} \middle| \begin{matrix}([b]_\mathbb{G}, [c]_\mathbb{G}) \xleftarrow{\$} \mathcal{A}(\mathsf{par}, [1]_\mathbb{G}, [a]_\mathbb{G}; r_\mathcal{A})\\ b' \xleftarrow{\$} \mathcal{E}_\mathcal{A}(\mathsf{par}, [1]_\mathbb{G}, [a]_\mathbb{G}, r_\mathcal{A})\end{matrix}\right]$$

is negligible.

We recall the Generalized Knowledge of Exponent assumption from [44]. It is also known as the n-KEA [13]. We add that similar to the KEA, the extractor is given the random coins of the adversary.

[5] To simplify the presentation, we define the computable homomorphism mapping from $\mathbb{G}$ to $\hat{\mathbb{G}}$ instead of from $\hat{\mathbb{G}}$ to $\mathbb{G}$. It is only a syntactic difference.

Definition 5 (The Generalized Knowledge of Exponent Assumption, GKEA). *Let $\mathbb{G}$ be a cyclic group of prime order q. The Knowledge of exponent assumption states that for any adversary $\mathcal{A}$ that on input $X := ([x_0 := 1]_{\mathbb{G}}, [a]_{\mathbb{G}}, [x_1]_{\mathbb{G}}, [ax_1]_{\mathbb{G}}, \ldots, [x_n]_{\mathbb{G}}, [ax_n]_{\mathbb{G}})$ outputs a tuple $[b]_{\mathbb{G}}, [c]_{\mathbb{G}}$, there exists an extractor $\mathcal{E}_{\mathcal{A}}$ that takes inputs of $\mathcal{A}$ and any random coins of $\mathcal{A}$ and outputs $\mathbf{z} := (z_1, ..., z_n)^{\top} \in \mathbb{Z}_q^n$ such that*

$$\mathrm{Adv}^{\mathsf{gkea}}_{\mathbb{G},\mathcal{A},\mathcal{E}_{\mathcal{A}}}(\lambda) := \Pr\left[\begin{array}{c} \prod_{i=0}^{n} [x_i \cdot z_i]_{\mathbb{G}} \neq [b]_{\mathbb{G}} \\ \wedge \\ [ab]_{\mathbb{G}} = [c]_{\mathbb{G}} \end{array} \middle| \begin{array}{c} ([b]_{\mathbb{G}}, [c]_{\mathbb{G}}) \xleftarrow{\$} \mathcal{A}(\mathsf{par}, X; r_{\mathcal{A}}) \\ \mathbf{z} \xleftarrow{\$} \mathcal{E}_{\mathcal{A}}(\mathsf{par}, X, r_{\mathcal{A}}) \end{array} \right]$$

is negligible.

We present our security games and cryptographic assumptions with the code-based game-playing framework similar to [11,15].

Definition 6 (Security Game). *A security game* Game *consists of an initialisation procedure* Init*, a finalise procedure* Final*, and possibly some additional procedures* $\mathrm{P}_1, \ldots, \mathrm{P}_n$*. All procedures are described in pseudo-codes. We say that an adversary $\mathcal{A}$ is playing a game* Game *if he first calls* Init*, obtaining its output. It may then make oracle queries to procedures* $\mathrm{P}_1, \ldots, \mathrm{P}_n$ *until it makes its final call to* Final*.*

We say that the adversary $\mathcal{A}$ has won Game *if* Final *outputs* 1 *on the input received from the adversary. The adversary has lost if* Final *outputs* 0*. We will denote an adversary $\mathcal{A}$ playing a game* Game $= (\textsc{Init}, \mathrm{P}_1, \ldots, \mathrm{P}_n, \textsc{Final})$ *by* $\mathcal{A}^{\mathrm{P}_1,\ldots,\mathrm{P}_n}$*.*

We call a game non-interactive if there are no procedures besides Init *and* Final*.*

Similar to the notion of non-interactive problems in groups from [24], we distinguish between group elements in the inputs and outputs of procedures, and other inputs or outputs that are not group elements. More precisely, every input or output X to a procedure P consists of the following parts:

- group elements $x_1, \ldots, x_u$
- and a bit string x'.

We note that this distinction is especially relevant in the AGM, as we want to exclude pathological cases where an adversary receives some encoding of a group element in the bit string part x' that would not be counted towards his input elements. It is obvious that this might cause problems when trying to extract, as the adversary and the extractor might not know a representation of this encoded group element.

Furthermore, in the AGM, any vector $\mathbf{v}$ of group elements defines an extractable OWF (EOWF) if the DLOG is hard (simply define $f_{\mathbf{v}} \colon \mathbb{Z}_q^n \to \mathbb{G}; \mathbf{z} \mapsto \mathbf{v}^{\top} \cdot \mathbf{z}$. In [14] it was shown that EOWFs with auxiliary input are impossible assuming indistinguishability obfuscation. They conjecture, however, that if the distribution from which the auxiliary input is drawn is *benign* (i.e. it is unlikely that the auxiliary input contains an obfuscated circuit) EOWFs may still exist.

We therefore require that all bitstrings x' contained in oracle responses are drawn from benign distributions.

We say that an adversary *solves* a problem $\mathcal{P}$ if he wins the corresponding security game. We say that a problem is hard if the probability that a PPT adversary solves the problem is negligible in the security parameter λ, more formally if

$$\Pr\left[\textsc{Final}(S) = 1 \,\middle|\, \begin{array}{c} C \xleftarrow{\$} \textsc{Init} \\ S \xleftarrow{\$} \mathcal{A}^{\mathrm{P}_1,\ldots,\mathrm{P}_n}(C, 1^\lambda) \end{array}\right]$$

is negligible.

As an example, we describe the Discrete Logarithm Problem as a game with Init and Final in Figure 1.

$\textsc{Init}()$:	$\textsc{Final}(y)$:
$x \xleftarrow{\$} \mathbb{Z}_q$	Return $y = x$
Return $([1]_{\mathbb{G}}, [x]_{\mathbb{G}})$	

Fig. 1. The Discrete Logarithm Game $\textsc{Game}_{\textsc{dlog}}$ with respect to a cyclic group $\mathbb{G}$

2.3 The Algebraic Group Model

A cryptographic group allows us to perform the group operation (usually denoted as multiplication), exponentiation, inversion, validity check, and equality check on group elements. We abstract all these properties as a group scheme [9,8] defined as follows.

Definition 7 (Group Scheme). *A group scheme* $\Gamma := (\mathsf{GGen}, \mathsf{Multi}, \mathsf{Exp}, \mathsf{Inv}, \mathsf{V}, \mathsf{Eq})$ *has the following 6 algorithms:*

- *The probabilistic setup algorithm* $\mathsf{GGen}(1^\lambda)$ *outputs a public group description* gp *of a group* $\mathbb{H}$*. Particularly,* gp *contains a generator* $h = [1]_{\mathbb{H}}$ *and the group order* q*. For simplicity, we assume that* gp *is implicitly given to all the following algorithms.*
- *The deterministic group operation algorithm* $\mathsf{Multi}([x_1]_{\mathbb{H}}, [x_2]_{\mathbb{H}})$ *outputs the multiplication of* $[x_1]_{\mathbb{H}}$ *and* $[x_2]_{\mathbb{H}}$ *for* $[x_1]_{\mathbb{H}}, [x_2]_{\mathbb{H}} \in \mathbb{H}$ *and we denote the result by* $[x_1]_{\mathbb{H}} \cdot [x_2]_{\mathbb{H}} = [x_1 + x_2]_{\mathbb{H}}$*.*
- *The deterministic exponentiation algorithm* $\mathsf{Exp}([x]_{\mathbb{H}}, y)$ *for* $x, y \in \mathbb{Z}_p$ *outputs* $[xy]_{\mathbb{H}}$*.*
- *The deterministic inverse algorithm* $\mathsf{Inv}([x]_{\mathbb{H}})$ *outputs the inversion of* $[x]_{\mathbb{H}}$*, denoted by* $[-x]_{\mathbb{H}}$*, for* $[x]_{\mathbb{H}} \in \mathbb{H}$*.*
- *The deterministic validity check algorithm* $\mathsf{V}([x]_{\mathbb{H}})$ *outputs* 1 *if* $[x]_{\mathbb{H}}$ *is a valid element from* $\mathbb{H}$ *and* 0 *otherwise.*
- *The deterministic equality check algorithm* $\mathsf{Eq}([x]_{\mathbb{H}}, [y]_{\mathbb{H}})$ *outputs* 1 *if* $[x]_{\mathbb{H}}$ *and* $[y]_{\mathbb{H}}$ *are the same group element.*

To save space we sometimes write $[x]_{\mathbb{H}} \cdot [y]_{\mathbb{H}}$ or $[x+y]_{\mathbb{H}}$ instead of $\mathsf{Multi}([x]_{\mathbb{H}}, [y]_{\mathbb{H}})$, $[x]_{\mathbb{H}} = [y]_{\mathbb{H}}$ of $\mathsf{Eq}([x]_{\mathbb{H}}, [y]_{\mathbb{H}}) = 1$, and $[x \cdot y]_{\mathbb{H}}$ of $\mathsf{Exp}([x]_{\mathbb{H}}, y)$.

Definition 8 (Correctness of a Group Scheme). *Let* $\Gamma := (\mathsf{GGen}, \mathsf{Multi}, \mathsf{Exp}, \mathsf{Inv}, \mathsf{V}, \mathsf{Eq})$ *be a group scheme. We say that a group scheme is* correct *if for all* $\mathsf{gp} \in \mathsf{GGen}(1^\lambda)$ *(which define the group* $\mathbb{H}$*) the following holds:*

- *The group is closed, namely, for all valid group elements* $x, y \in \mathbb{H}$ *(i.e.* $\mathsf{V}(x) = \mathsf{V}(y) = 1$*),* $\mathsf{V}(\mathsf{Multi}(x, y)) = 1$,
- *There exists a neutral element, namely, there exists an element* $a \in \mathbb{H}$ *such that for all group elements* x $\mathsf{Eq}(\mathsf{Multi}(x, a), x) = 1$,
- *Each group element has an inverse, namely, for all valid group encodings* x, $\mathsf{V}(\mathsf{Inv}(x)) = 1$ *and* $\mathsf{Eq}(\mathsf{Multi}(x, \mathsf{Inv}(x)), a) = 1$ *where* a *is an encoding of the neutral element, and*
- *The group operation is associative, i.e. for group elements* x, y, z, $\mathsf{Eq}($ $\mathsf{Multi}(\mathsf{Multi}(x, y), z), \mathsf{Multi}(x, \mathsf{Multi}(y, z)))$

Remark 1. In this work, we only consider abelian groups, i.e. groups where $\mathsf{Eq}(\mathsf{Multi}(x, y), \mathsf{Multi}(y, x)) = 1$.

In the following, we define the algebraic group model [25]. The algebraic group model is a computational model where all adversaries are modeled as algebraic algorithms. We recall the definition of an algebraic algorithm, which capture the intuition that the only way for an algebraic algorithm to output a new group element is to perform group multiplications from known group elements. Furthermore, we say a group is algebraic if all PPT adversaries operate on its elements in an algebraic way.

Definition 9 (Algebraic Group). *Let* $\mathbb{H}$ *be a cyclic group of prime order* q *defined by a group scheme* $\Gamma := (\mathsf{GGen}, \mathsf{Multi}, \mathsf{Exp}, \mathsf{Inv}, \mathsf{V}, \mathsf{Eq})$*. We say that a PPT algorithm* $\mathcal{A}$ *is* algebraic *if there exists an efficient extractor* $\mathcal{E}$ *that, given the inputs* $([x_1]_{\mathbb{H}}, \dots [x_n]_{\mathbb{H}}) \in \mathbb{H}^n$ *of* $\mathcal{A}$*, which are generated according to some distribution defined by the security game, and* $\mathcal{A}$*'s random tape, outputs a representation* $\mathbf{z} := (z_1, \dots, z_n)^\top \in \mathbb{Z}_q^n$ *for every group element* $[y]_{\mathbb{H}} \in \mathbb{H}$ *in the output of* $\mathcal{A}$ *such that*

$$\mathrm{Adv}^{\mathsf{alg}}_{\mathbb{H},\mathcal{A}}(\lambda) := \Pr\left[\prod_{i=1}^{n} [x_i \cdot z_i]_{\mathbb{H}} \neq [y]_{\mathbb{H}} \;\middle|\; \begin{matrix} [y]_{\mathbb{H}} \xleftarrow{\$} \mathcal{A}([x_1]_{\mathbb{H}}, \dots, [x_n]_{\mathbb{H}}; r_{\mathcal{A}}) \\ \mathbf{z} \xleftarrow{\$} \mathcal{E}_{\mathcal{A}}([x_1]_{\mathbb{H}}, \dots, [x_n]_{\mathbb{H}}, [y]_{\mathbb{H}}, r_{\mathcal{A}}) \end{matrix}\right]$$

is negligible.

We call the group $\mathbb{H}$ algebraic *if every algorithm that operates on its elements is algebraic.*

We have a syntactic difference here, namely, we define an additional algorithm to extract a representation and it is similar to [38], while [25] required the adversary outputs a representation. Moreover, we require the outputted representation to be valid with overwhelming probability, which is (slightly) weaker than [25].

2.4 Quasi-Adaptive Non-interactive Zero-Knowledge Arguments

In this paper, we use a quasi-adaptive non-interactive zero-knowledge argument system (QANIZK) for linear subspaces, and we recall the useful definitions from [30,33] as follows.

In the following, we consider public parameters par generated by $\mathsf{Gen}_{\mathsf{par}}$. Since we use a QANIZK argument system for a linear subspace only, we define it specifically for linear subspaces. Let $\mathcal{D}_{\mathsf{par}}$ be a probability distribution over a set of matrices $\mathcal{M} = \{\mathbf{M} \in \mathbb{Z}_q^{n \times m}\}$ for some integers $n > m$ where the associated language to a matrix is $\mathcal{L}_{[\mathbf{M}]_\mathbb{G}} = \{[\mathbf{y}]_\mathbb{G} \in \mathbb{G}^n \mid \exists \mathbf{x} \in \mathbb{Z}_m : [\mathbf{M}\mathbf{x}]_\mathbb{G} = [\mathbf{y}]_\mathbb{G}\}$.

Definition 10 (Quasi-Adaptive Non-interactive Zero-Knowledge Arguments). *A* Quasi-Adaptive Non-Interactive Zero-Knowledge Argument *(QANIZK) Π for a language distribution $\mathcal{D}_{\mathsf{par}}$ consists of five PPT-Algorithms $\Phi = (\mathsf{Gen}_{\mathsf{par}}, \mathsf{Gen}_{\mathsf{crs}}, \mathsf{Prove}, \mathsf{Sim}, \mathsf{Ver})$*

- *The probabilistic key generation algorithm $\mathsf{Gen}_{\mathsf{par}}(1^\lambda)$ returns the public parameters par, which implicitly defines the proof space Π.*
- *The probabilistic algorithm $\mathsf{Gen}_{\mathsf{crs}}(\mathsf{par}, [\mathbf{M}]_\mathbb{G})$ returns a common reference string crs and a trapdoor td.*
- *The probabilistic proving algorithm $\mathsf{Prove}(\mathsf{crs}, \mathbf{x}, [\mathbf{y}]_\mathbb{G})$ returns a proof π.*
- *The deterministic verification algorithm $\mathsf{Ver}(\mathsf{crs}, [\mathbf{y}]_\mathbb{G}, \pi)$ returns 1 or 0 where 1 means that π is a valid proof of $[\mathbf{y}]_\mathbb{G} \in \mathcal{L}_{[\mathbf{M}]_\mathbb{G}}$.*
- *The probabilistic simulation algorithm $\mathsf{Sim}(\mathsf{crs}, \mathsf{td}, [\mathbf{y}]_\mathbb{G})$ returns a proof π for y (note that $[\mathbf{y}]_\mathbb{G}$ is not necessarily in $\mathcal{L}_{[\mathbf{M}]_\mathbb{G}}$).*

We require that the algorithms satisfy the following properties:

Perfect Completeness. *For all λ, all $\mathsf{par} \xleftarrow{\boxdot} \mathsf{Gen}_{\mathsf{par}}(1^\lambda)$, all $\mathbf{M} \xleftarrow{\boxdot} \mathcal{D}_{\mathsf{par}}$, all $(\mathsf{crs}, \mathsf{td}) \xleftarrow{\boxdot} \mathsf{Gen}_{\mathsf{crs}}(\mathsf{par}, [\mathbf{M}]_\mathbb{G})$ all $([\mathbf{y}]_\mathbb{G}, \mathbf{x})$ with $[\mathbf{M}\mathbf{x}]_\mathbb{G} = [\mathbf{y}]_\mathbb{G}$ and all $\pi \xleftarrow{\boxdot} \mathsf{Prove}(\mathsf{crs}, [\mathbf{y}]_\mathbb{G}, \mathbf{x})$, we have $\mathsf{Ver}(\mathsf{crs}, [\mathbf{y}]_\mathbb{G}, \pi) = 1$.*

Perfect Zero-Knowledge. *For all λ, all par output by $\mathsf{Gen}_{\mathsf{par}}(1^\lambda)$, all $[\mathbf{M}]_\mathbb{G} \xleftarrow{\boxdot} \mathcal{D}_{\mathsf{par}}$, all $(\mathsf{crs}, \mathsf{td}) \xleftarrow{\boxdot} \mathsf{Gen}_{\mathsf{crs}}(\mathsf{par}, [\mathbf{M}]_\mathbb{G})$, and all $([\mathbf{y}]_\mathbb{G}, \mathbf{x})$ with $[\mathbf{M}\mathbf{x}]_\mathbb{G} = [\mathbf{y}]_\mathbb{G}$ the distributions*

$$\mathsf{Prove}(\mathsf{crs}, [\mathbf{y}]_\mathbb{G}, \mathbf{x}) \text{ and } \mathsf{Sim}(\mathsf{crs}, \mathsf{td}, [\mathbf{y}]_\mathbb{G})$$

are identical (where the coin tosses are taken over Prove, Sim).

Adaptive soundess. *We define the adaptive soundness game $\text{Game}_{\mathsf{as}}^{\Phi}$ for argument system Φ in Figure 2. We say Φ is adaptively sound if for all PPT*

$\underline{\text{Init}():}$	$\underline{\text{Final}([\mathbf{y}^*]_\mathbb{G}, \pi^*):}$
$\mathsf{par} \xleftarrow{\boxdot} \mathsf{Gen}_{\mathsf{par}}; \mathbf{M} \xleftarrow{\boxdot} \mathcal{D}_{\mathsf{par}}$	Return $(\mathbf{M}^\perp [\mathbf{y}^*]_\mathbb{G} = [\mathbf{0}]_\mathbb{G} \wedge \mathsf{Ver}(\mathsf{crs}, [\mathbf{y}^*]_\mathbb{G}, \pi^*))$
$(\mathsf{crs}, \mathsf{td}) \xleftarrow{\boxdot} \mathsf{Gen}_{\mathsf{crs}}(\mathsf{par}, [\mathbf{M}]_\mathbb{G})$	
Return $(\mathsf{par}, \mathsf{crs}, [\mathbf{M}]_\mathbb{G})$	

Fig. 2. Game $\text{Game}_{\mathsf{as}}^{\Phi}$ for the adaptive soundness of Φ. Given $\mathbf{M} \in \mathbb{Z}_q^{n \times m}$ it is efficient to compute a non-zero kernel matrix $\mathbf{M}^\perp \in \mathbb{Z}_q^{(n-m) \times n}$ such that $\mathbf{M}^\perp \cdot \mathbf{M} = \mathbf{0}$

adversaries $\mathcal{A}$

$$\mathrm{Adv}^{\mathsf{as}}_{\Phi,\mathcal{A}}(\lambda) := \Pr\left[\textsc{Final}([\mathbf{y}^*]_{\mathbb{G}}, \pi^*) = 1 \;\middle|\; \begin{array}{c} C := (\mathsf{par}, \mathsf{crs}, [\mathbf{M}]_{\mathbb{G}}) \xleftarrow{\boxdot} \textsc{Init}() \\ ([\mathbf{y}^*]_{\mathbb{G}}, \pi^*) \xleftarrow{\boxdot} \mathcal{A}(C, 1^\lambda) \end{array}\right]$$

is negligible.

We will later use the QANIZK system from [33]. We recall it in Figure 3. Its adaptive soundness is based on the KerMDH (Definition 3) for the matrix distribution $\mathcal{D}_k$.

$\underline{\mathsf{Gen}_{\mathsf{crs}}(\mathsf{par}, [\mathbf{M}]_{\mathbb{G}} \in \mathbb{G}^{n\times m})}$:	$\underline{\mathsf{Prove}(\mathsf{crs}, [\mathbf{y}]_{\mathbb{G}}, \mathbf{x})}$: $\quad /\!/\, \mathbf{y} := \mathbf{M}\mathbf{x}$	$\underline{\mathsf{Sim}(\mathsf{crs}, \mathsf{td}, [\mathbf{y}]_{\mathbb{G}})}$:
$\mathbf{A} \xleftarrow{\boxdot} \mathcal{D}_k$	$\pi := \left[\mathbf{x}^\top \mathbf{P}\right]_{\mathbb{G}}$	$\pi := \left[\mathbf{y}^\top \mathbf{K}\right]_{\mathbb{G}}$
$\mathbf{K} \xleftarrow{\boxdot} \mathbb{Z}_q^{n\times(k+1)}$	Return $\pi \in \mathbb{G}^{1\times(k+1)}$	Return π
$\mathbf{C} := \mathbf{K}\mathbf{A} \in \mathbb{Z}_q^{n\times k}$		
$\mathbf{P} := \mathbf{M}^\top \mathbf{K}$	$\underline{\mathsf{Ver}(\mathsf{crs}, [\mathbf{y}]_{\mathbb{G}}, \pi)}$:	
$\mathsf{td} := \mathbf{K}$	Return $(\pi \circ [\mathbf{A}]_{\hat{\mathbb{G}}} = \left[\mathbf{y}^\top\right]_{\mathbb{G}} \circ [\mathbf{C}]_{\hat{\mathbb{G}}})$	
$\mathsf{crs} := ([\mathbf{P}]_{\mathbb{G}}, [\mathbf{C}]_{\hat{\mathbb{G}}}, [\mathbf{A}]_{\hat{\mathbb{G}}})$		
Return $(\mathsf{crs}, \mathsf{td})$		

Fig. 3. A QANIZK scheme from [33]

We recall the theorem from [33] for the soundness of the system shown in Figure 3.

Theorem 1 (Adaptive soundness, Theorem 1 in [33]). *Protocol Π_{as} from Figure 3 is a Quasi-Adaptive Non-Interactive Zero Knowledge Argument. Furthermore, under the $\mathcal{D}_k$-KerMDH Assumption in $\hat{\mathbb{G}}$, it has adaptive soundness.*

This argument system is additively homomorphic. More precisely, given two valid proofs π_1 and π_2 for two vectors $[\mathbf{y}_1]_{\mathbb{G}}$ and $[\mathbf{y}_2]_{\mathbb{G}}$, respectively, and a integer $x \in \mathbb{Z}_q$, one can efficiently compute a valid proof for $[\mathbf{y}_1 + \mathbf{y}_2]_{\mathbb{G}}$ and also a valid proof for $[\mathbf{y}_1 \cdot x]_{\mathbb{G}}$.

Formally, this property is captured by the algorithms PAdd and PMult defined in Figure 4. The correctness of PAdd and PMult is defined as: For all λ, all $\mathsf{par} \xleftarrow{\boxdot} \mathsf{Gen}_{\mathsf{par}}(1^\lambda)$, all $[\mathbf{M}]_{\mathbb{G}} \xleftarrow{\boxdot} \mathcal{D}_{\mathsf{par}}$, all $\mathsf{crs} \xleftarrow{\boxdot} \mathsf{Gen}_{\mathsf{crs}}(\mathsf{par}, [\mathbf{M}]_{\mathbb{G}})$, all $([\mathbf{y}_1]_{\mathbb{G}}, \mathbf{x}_1)$ with $[\mathbf{M}\mathbf{x}_1]_{\mathbb{G}} = [\mathbf{y}_1]_{\mathbb{G}}$, all $([\mathbf{y}_2]_{\mathbb{G}}, \mathbf{x}_2)$ with $[\mathbf{M}\mathbf{x}_2]_{\mathbb{G}} = [\mathbf{y}_2]_{\mathbb{G}}$, all $\pi_1 \xleftarrow{\boxdot} \mathsf{Prove}(\mathsf{crs}, [\mathbf{y_1}]_{\mathbb{G}}, \mathbf{x}_1)$, all $\pi_2 \xleftarrow{\boxdot} \mathsf{Prove}(\mathsf{crs}, [\mathbf{y_2}]_{\mathbb{G}}, \mathbf{x}_2)$, $\pi_{\mathsf{add}} := \mathsf{PAdd}(\pi_1, \pi_2)$ and $\pi_{\mathsf{mult}} := \mathsf{PMult}(x, \pi_1)$ and all $x \in \mathbb{Z}_q$, we have

$$\mathsf{Ver}(\mathsf{crs}, [\mathbf{y}_1 + \mathbf{y}_2]_{\mathbb{G}}, \pi_{\mathsf{add}}) = 1 \text{ and } \mathsf{Ver}(\mathsf{crs}, [\mathbf{y}_1 \cdot x]_{\mathbb{G}}, \pi_{\mathsf{mult}}) = 1.$$

3 Construction with Type 1 or 2 Pairings

We construct an algebraic group with the Generalized Knowledge of Exponent Assumption (GKEA) and pairings. Our pairing $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ can be implemented

$\mathsf{PAdd}(\pi_1, \pi_2)$	$\mathsf{PMult}(\pi_1, x \in \mathbb{Z}_q)$
Parse $\pi_1 := [\mathbf{s}]_{\mathbb{G}}$	Parse $\pi_1 := [\mathbf{s}]_{\mathbb{G}}$
Parse $\pi_2 := [\mathbf{t}]_{\mathbb{G}}$	Return $\pi_r := [\mathbf{s} \cdot x]_{\mathbb{G}}$
Return $\pi_r := [\mathbf{s} + \mathbf{t}]_{\mathbb{G}}$	

Fig. 4. Algorithms PAdd and PMult for Φ from Figure 3

with a (symmetric) Type 1 pairing (where $\mathbb{G} = \hat{\mathbb{G}}$) or an (asymmetric) Type 2 pairing (where $\mathbb{G} \neq \hat{\mathbb{G}}$ and there is an efficiently computable homomorphism $\psi : \mathbb{G} \to \hat{\mathbb{G}}$).

The constructed group $\mathbb{H} \subsetneq \mathbb{G} \times \hat{\mathbb{G}}$ and an element of $\mathbb{H}$ has the form

$$([x]_{\mathbb{G}}, [a \cdot x]_{\mathbb{G}}),$$

where a is a random element in $\mathbb{Z}_q$. The group parameter gp of $\mathbb{H}$ contains the pairing group and a generator of $\mathbb{H}$, $[1]_{\mathbb{H}} := ([1]_{\mathbb{G}}, [a]_{\mathbb{G}})$. To verify if $([x_1]_{\mathbb{G}}, [x_2]_{\mathbb{G}}) \in \mathbb{H}$, we check if $[x_1]_{\mathbb{G}} \circ [a]_{\hat{\mathbb{G}}} = [x_2]_{\mathbb{G}} \circ [1]_{\hat{\mathbb{G}}}$ and $[a]_{\hat{\mathbb{G}}}$ is easy to have in Type 1 or 2 pairings: For Type 1 $\mathbb{G} = \hat{\mathbb{G}}$ and thus $[a]_{\hat{\mathbb{G}}} = [a]_{\mathbb{G}}$, and for Type 2 $[a]_{\hat{\mathbb{G}}} = \psi([a]_{\mathbb{G}})$. Our algebraic group is defined by the group scheme in Figure 5.

$\mathsf{GGen}(1^\lambda)$:	$\mathsf{Multi}(X, Y)$:
$a \xleftarrow{\$} \mathbb{Z}_q$	If $\mathsf{V}(X) = 0$ or $\mathsf{V}(Y) = 0$ then
$\mathcal{PG} \xleftarrow{\$} \mathsf{GGen}$	return $\perp$
$[1]_{\mathbb{H}} = ([1]_{\mathbb{G}}, [a]_{\mathbb{G}})$	Parse $X =: ([x_1]_{\mathbb{G}}, [x_2]_{\mathbb{G}})$
$\mathsf{gp} := (\mathcal{PG}, q, [1]_{\mathbb{H}})$	Parse $Y =: ([y_1]_{\mathbb{G}}, [y_1]_{\mathbb{G}})$
	Return $([x_1 + y_1]_{\mathbb{G}}, [x_2 + y_2]_{\mathbb{G}})$
$\mathsf{Eq}(X, Y)$:	
If $\mathsf{V}(X) = 0$ or $\mathsf{V}(Y) = 0$ then	$\mathsf{Exp}(X, y \in \mathbb{Z}_q)$:
return $\perp$	If $\mathsf{V}(X) = 0$ then return $\perp$
Parse $X =: ([x_1]_{\mathbb{G}}, [x_2]_{\mathbb{G}})$	Parse $X =: ([x_1]_{\mathbb{G}}, [x_2]_{\mathbb{G}})$
Parse $Y =: ([y_1]_{\mathbb{G}}, [y_2]_{\mathbb{G}})$	Return $([x_1 \cdot y]_{\mathbb{G}}, [x_2 \cdot y]_{\mathbb{G}})$
If $([x_1]_{\mathbb{G}}, [x_2]_{\mathbb{G}}) = ([y_1]_{\mathbb{G}}, [y_2]_{\mathbb{G}})$ then	
return 1	$\mathsf{V}(X)$:
Else return 0	Parse $X =: ([x_1]_{\mathbb{G}}, [x_2]_{\mathbb{G}})$
	Return $[x_1]_{\mathbb{G}} \circ \psi([a]_{\mathbb{G}}) = [x_2]_{\mathbb{G}} \circ [1]_{\hat{\mathbb{G}}}$
$\mathsf{Inv}(X)$:	
If $\mathsf{V}(X) = 0$ then return $\perp$	
Return $\mathsf{Exp}(X, q - 1)$	

Fig. 5. Construction of an algebraic group with a pairing based verification. $\psi : \mathbb{G} \to \hat{\mathbb{G}}$ is either the identity function (for Type 1 pairings) or an efficiently computable homomorphism (for Type 2 pairings).

Correctness of the Group Scheme. The group generated by $[1]_{\mathbb{H}}$ is a subgroup of $\mathbb{G} \times \mathbb{G}$ and has order q. The group operation Multi is associative and

commutative due to the associativity and commutativity of the group operation in the base group $\mathbb{G}$. The group is closed w.r.t. Multi because each valid element lies in the span of $\begin{pmatrix}1\\a\end{pmatrix}$. This is a one-dimensional subspace of $\mathbb{G}\times\mathbb{G}$ and therefore contains q elements. The neutral element is $([0]_{\mathbb{G}}, [0\cdot a]_{\mathbb{G}}) = ([0]_{\mathbb{G}}, [0]_{\mathbb{G}})$. The the exponentiation algorithm is correct due to the correctness of the corresponding algorithm in $\mathbb{G}$, and the inversion algorithm is correct due to the correctness of Exp

The group membership is verified through pairings: $\mathsf{V}([x_1]_{\mathbb{G}}, [x_2]_{\mathbb{G}})$ outputs 1 if and only if $[x_1]_{\mathbb{G}} \circ [a]_{\hat{\mathbb{G}}} = [x_2]_{\mathbb{G}} \circ [1]_{\hat{\mathbb{G}}} \Leftrightarrow x_2 = x_1 \cdot a$.

Algebraicity of the Group. The algebraicity of the group scheme in Figure 5 is based on the extraction through the GKEA. If an adversary outputs valid group elements, the GKEA-extractor can be used to extract a representation vector $\mathbf{z}$. We state this in the following theorem.

Theorem 2. *Under the GKEA in $\mathbb{G}$ and the existence of a Type 1 or 2 pairing $e : \mathbb{G}\times\hat{\mathbb{G}} \to \mathbb{G}_T$, the group $\mathbb{H}$ from Figure 5 is algebraic in the sense of Definition 9.*

Proof. Let $\mathcal{A}([x_1]_{\mathbb{H}}, \ldots, [x_n]_{\mathbb{H}}; r_{\mathcal{A}})$ be any algorithm that takes a list of elements $([x_1]_{\mathbb{H}}, \ldots, [x_n]_{\mathbb{H}})$ from $\mathbb{H}^n$ $(n \geq 1)$ and output an element $[y]_{\mathbb{H}} \in \mathbb{H}$. We show that $\mathcal{A}$ is algebraic so that $\mathbb{H}$ is an algebraic group. To show it, we construct an extractor $\mathcal{E}_{\mathbb{H}}$ that takes group elements in $\mathcal{A}$'s input and output and $\mathcal{A}$'s random tape as inputs and uses the GKEA extractor $\mathcal{E}_{\mathrm{GKEA}}$ as a subroutine. The construction of $\mathcal{E}_{\mathbb{H}}$ is in Figure 6. For simplicity, we ignore non-$\mathbb{H}$ elements in the input and output of $\mathcal{A}$ and, for an algorithm $\mathcal{A}'$ that outputs more than one $\mathbb{H}$ elements, we run $\mathcal{E}_{\mathbb{H}}$ defined in Figure 6 multiple time to show that $\mathcal{A}'$ is algebraic.

$\mathcal{E}_{\mathbb{H}}([x_1]_{\mathbb{H}}, \ldots, [x_n]_{\mathbb{H}}, Y, r_{\mathcal{A}})$:
If $\mathsf{V}(Y) = 0$ Return $\bot$
Parse $[x_i]_{\mathbb{H}} =: ([s_i]_{\mathbb{G}}, [t_i]_{\mathbb{G}})$
Parse $Y =: ([y_1]_{\mathbb{G}}, [y_2]_{\mathbb{G}})$
$\mathbf{z} \xleftarrow{\boxtimes} \mathcal{E}_{\mathrm{GKEA}}(([s_i]_{\mathbb{G}}, [t_i]_{\mathbb{G}})_i, ([y_1]_{\mathbb{G}}, [y_2]_{\mathbb{G}}), r_{\mathcal{A}})$
Return $\mathbf{z}$

Fig. 6. Extractor for the group $\mathbb{H}$ from Figure 5

Here our group $\mathbb{H}$ is defined without knowing $a \in \mathbb{Z}_q$ but using $([1]_{\mathbb{G}}, [a]_{\mathbb{G}})$ from the GKEA. With the generator $[1]_{\mathbb{H}} := ([1]_{\mathbb{G}}, [a]_{\mathbb{G}}) \in \mathbb{H}$ and a Type 1 or 2 pairing, one can perform any group operation and verification publicly. It is easy to see that if $\mathcal{A}$ outputs an element in $\mathbb{H}$ then $\mathcal{E}_{\mathbb{H}}$ outputs a correct representation with the same probability as $\mathcal{E}_{\mathrm{GKEA}}$ returns a correct vector. Thus, $\mathrm{Adv}^{\mathsf{alg}}_{\mathbb{H},\mathcal{A}}(\lambda) = \mathrm{Adv}^{\mathsf{gkea}}_{\mathbb{G},\mathcal{A},\mathcal{E}_{\mathcal{A}}}(\lambda)$. We note that this is a non-black-box extractor, because it needs the random tape of $\mathcal{A}$ as an input. □

3.1 Cryptographic Hardness in $\mathbb{H}$

We show that any computational problem that is hard in $\mathbb{G}$ is also hard in $\mathbb{H}$. We note that problems that are rendered easy due to the pairing are already easy in $\mathbb{G}$, because an adversary in $\mathbb{G}$ can also use the pairing to solve the problem in $\mathbb{G}$.

We state this in the following theorem:

Theorem 3 ($\mathcal{P}_{\mathbb{G}} \Rightarrow \mathcal{P}_{\mathbb{H}}$). *Let $\mathcal{P}$ be a computational problem defined by a game* $\textsc{Game} := (\textsc{Init}, \mathrm{P}_1, \ldots, \mathrm{P}_n, \textsc{Final})$. *We denote $\mathcal{P}$ in group $\mathbb{X} \in \{\mathbb{G}, \mathbb{H}\}$ by $\mathcal{P}_{\mathbb{X}}$ and the corresponding security is defined by* $\textsc{Game}_{\mathbb{X}} := (\textsc{Init}_{\mathbb{X}}, \mathrm{P}_{\mathbb{X},1}, \ldots, \mathrm{P}_{\mathbb{X},n}, \textsc{Final}_{\mathbb{X}})$.

If $\mathcal{P}_{\mathbb{G}}$ is hard then the problem $\mathcal{P}_{\mathbb{H}}$ is hard as well. More precisely, if there is an adversary $\mathcal{A}$ that solves $\mathcal{P}_{\mathbb{H}}$, then there is a reduction $\mathcal{R}$ that solves $\mathcal{P}_{\mathbb{G}}$ with success probability

$$\mathrm{Adv}^{\mathcal{P}}_{\mathbb{G},\mathcal{R}}(\lambda) = \mathrm{Adv}^{\mathcal{P}}_{\mathbb{H},\mathcal{A}}(\lambda).$$

Proof. Let $\mathcal{A}$ be an adversary against the hardness of $\mathcal{P}_{\mathbb{H}}$. We prove the theorem by constructing a reduction that calls $\mathcal{A}$ and tries to win $\textsc{Game}_{\mathbb{G}}$ in $\mathbb{G}$ by providing $\mathcal{A}$ oracle access to the corresponding procedures in group $\mathbb{H}$. Note that the constructed algebraic group $\mathbb{H}$ is defined in Figure 5. The construction of our reduction $\mathcal{R}$ is shown in Figure 7.

$\underline{\textsc{Init}_{\mathbb{H}}}$:
$C = (\mathcal{PG}, X_{\mathbb{G}} =$
$([x_1]_{\mathbb{G}}, \ldots, [x_n]_{\mathbb{G}}, x')) \xleftarrow{\boxdot} \textsc{Init}_{\mathbb{G}}$
$a \xleftarrow{\boxdot} \mathbb{Z}_q$
$[1]_{\mathbb{H}} = ([1]_{\mathbb{G}}, [a]_{\mathbb{G}})$
$v = [a]_{\hat{\mathbb{G}}}$
$\mathsf{gp} := (\mathcal{PG}, q, [1]_{\mathbb{H}}, v)$
For $i = 1$ to n:
  $[x_i]_{\mathbb{H}} = ([x_i]_{\mathbb{G}}, [a \cdot x_i]_{\mathbb{G}})$
Return $C_{\mathbb{H}} = (\mathcal{PG}, X_{\mathbb{H}} =$
$([x_1]_{\mathbb{H}}, \ldots, [x_n]_{\mathbb{H}}, x'))$

$\underline{\textsc{Final}_{\mathbb{H}}(S)}$:
Parse $S =: (s_1, \ldots, s_m, s')$
For $i = 1$ to n:
  If $\mathsf{V}(s_i) = 0$
    Abort
  Parse $s_i =: (s_{i,1}, s_{i,2})$
$S_{\mathbb{G}} := (s_{1,1}, \ldots, s_{m,1}, s')$
Return $\textsc{Final}_{\mathbb{G}}(S_{\mathbb{G}})$

$\underline{\mathrm{P}_{\mathbb{H},j}(x_1, \ldots x_u, x')}$: // $1 \le j \le n$
For $i = 1$ to u
  If $\mathsf{V}(x_i) = 0$
    Abort
  Parse $x_i = (x_{i,1}, x_{i,2})$
$([y_1]_{\mathbb{G}}, \ldots, [y_v]_{\mathbb{G}}, y') \xleftarrow{\boxdot} \mathrm{P}_{\mathbb{G},j}(x_{1,1}, \ldots x_{n,1}, x')$
For $i = 1$ to v:
  $[y_i]_{\mathbb{H}} := ([y_i]_{\mathbb{G}}, [y_i \cdot a]_{\mathbb{G}})$
Return $Y_{\mathbb{H}} = ([y_1]_{\mathbb{H}}, \ldots, [y_v]_{\mathbb{H}}, y')$

Fig. 7. Reduction $\mathcal{R}$ against $\mathcal{P}$ in $\mathbb{H}$

In the $\textsc{Init}$ procedure for the problem in $\mathbb{H}$, the reduction $\mathcal{R}$ sets up the group by choosing an exponent a and setting the other group parameters accordingly.

It calls the INIT procedure from its own security game in $\mathbb{G}$ and transforms the received outputs into outputs for the game in $\mathbb{H}$. Specifically, it replaces any group element with a group element from the constructed $\mathbb{H}$ group, and replaces the public group parameters $\mathcal{PG}$ with the public parameters gp of $\mathbb{H}$. For any procedure $\mathrm{P}_{\mathbb{G}}$ that $\mathcal{R}$ is allowed to use by $\textsc{Game}_{\mathbb{G}}$, $\mathcal{R}$ provides an equivalent procedure $\mathrm{P}_{\mathbb{H}}$ in $\mathbb{H}$. It does so by "forwarding" the requests and responses. In order to forward the request, it first checks the validity of the supposed group elements contained in the request, and then removes the second part in order to obtain a representation as group elements in $\mathbb{G}$. For forwarding the response, it adds a second part $[a \cdot y]_{\mathbb{G}}$ to every group element $[y]_{\mathbb{G}}$ contained in the response, thus providing a representation in the group $\mathbb{H}$. For the FINAL procedure, it forwards the final output of $\mathcal{A}$ in a similar way. Thus, the adversary only wins when the reduction wins and vice versa. □

We note that in order to construct the problem instance, the reduction $\mathcal{R}$ needs to have the exponent a as an element of $\mathbb{Z}_q$. It is therefore necessary for the reduction to generate its own instance of the group scheme instead of re-using an instance where it does not know the corresponding exponent a. The adversary used as a subroutine needs to be an adversary who works on the group scheme in general, not just certain instances of the scheme.

We further note that there might be problems that are easy in $\mathbb{G}$ but hard in $\mathbb{H}$, because adversaries in $\mathbb{G}$ are not required to be algebraic.

4 Construction with QANIZK

We present another construction of a group which is algebraic with respect to Definition 9. Similar to the construction in Section 3, extractability is based on the GKEA, but instead of using the pairing directly for group membership verification, we use additively homomorphic QANIZK arguments of linear subspace membership.

We construct a group $\mathbb{H}$ with elements of the form

$$([x]_{\mathbb{G}}, [ax]_{\mathbb{G}}, \pi)$$

where $[x]_{\mathbb{G}}$ and $[ax]_{\mathbb{G}}$ are group elements of $\mathbb{G}$, and π is a QANIZK argument proving that $([x]_{\mathbb{G}}, [ax]_{\mathbb{G}})^{\top} \in \mathsf{Span}\begin{pmatrix}[1]_{\mathbb{G}}\\ [a]_{\mathbb{G}}\end{pmatrix}$, where a is chosen uniformly at random from $\mathbb{Z}_q$. This homomorphic argument of subspace membership can be instantiated with the QANIZK system in Figure 3. We define our construction in terms of a group scheme.

Let $\mathbb{G}$ be a cyclic group of prime oder q and $\Phi = (\mathsf{Gen}_{\mathsf{par}}, \mathsf{Gen}_{\mathsf{crs}}, \mathsf{Prove}, \mathsf{Sim}, \mathsf{Ver})$ be a QANIZK system. We define our construction of an algebraic group $\mathbb{H}$ via the following group scheme $\Gamma_{\mathsf{alg}} := (\mathsf{GGen}, \mathsf{Multi}, \mathsf{Exp}, \mathsf{Inv}, \mathsf{V}, \mathsf{Eq})$.

CORRECTNESS OF THE GROUP SCHEME. We argue that the scheme described in Figure 8 describes a group. The Multi algorithm describes a group operation that is associative (due to the associativity of the group operation in $\mathbb{G}$) and

$\underline{\mathsf{GGen}(1^\lambda)}$:
$a \xleftarrow{\$} \mathbb{Z}_q, \mathbf{M} := \begin{pmatrix} 1 \\ a \end{pmatrix}$
$\mathsf{par}_{\mathsf{qa}} \xleftarrow{\$} \mathsf{Gen}_{\mathsf{par}}(1^\lambda)$
$\mathsf{crs} \xleftarrow{\$} \mathsf{Gen}_{\mathsf{crs}}(\mathsf{par}_{\mathsf{qa}}, [\mathbf{M}]_{\mathbb{G}})$
$\pi_1 \xleftarrow{\$} \mathsf{Prove}(\mathsf{crs}, [\mathbf{M}]_{\mathbb{G}}, 1)$
$[1]_{\mathbb{H}} := ([1]_{\mathbb{G}}, [a]_{\mathbb{G}}, \pi_1)$
$\mathsf{gp} := (\mathbb{G}, q, \mathsf{par}_{\mathsf{qa}}, \mathsf{crs}, [1]_{\mathbb{H}})$

$\underline{\mathsf{Eq}(X, Y)}$:
If $\mathsf{V}(X) = 0$ or $\mathsf{V}(Y) = 0$ then
  return $\bot$
Parse $X =: ([x_1]_{\mathbb{G}}, [x_2]_{\mathbb{G}}, \pi_x)$
Parse $Y =: ([y_1]_{\mathbb{G}}, [y_2]_{\mathbb{G}}, \pi_y)$
If $([x_1]_{\mathbb{G}}, [x_2]_{\mathbb{G}}) = ([y_1]_{\mathbb{G}}, [y_2]_{\mathbb{G}})$ then
  return 1
Else return 0

$\underline{\mathsf{Inv}(X)}$:
If $\mathsf{V}(X) = 0$ then return $\bot$
Return $\mathsf{Exp}(X, q-1)$

$\underline{\mathsf{Multi}(X, Y)}$:
If $\mathsf{V}(X) = 0$ or $\mathsf{V}(Y) = 0$ then
  return $\bot$
Parse $X =: ([x_1]_{\mathbb{G}}, [x_2]_{\mathbb{G}}, \pi_1)$
Parse $Y =: ([y_1]_{\mathbb{G}}, [y_1]_{\mathbb{G}}, \pi_2)$
$\pi_3 := \mathsf{PAdd}(\pi_1, \pi_2)$
Return $([x_1 + y_1]_{\mathbb{G}}, [x_2 + y_2]_{\mathbb{G}}, \pi_3)$

$\underline{\mathsf{Exp}(X, y \in \mathbb{Z}_q)}$:
If $\mathsf{V}(X) = 0$ then return $\bot$
Parse $X =: ([x_1]_{\mathbb{G}}, [x_2]_{\mathbb{G}}, \pi)$
$\pi' := \mathsf{PMult}(\pi, y)$
Return $([x_1 \cdot y]_{\mathbb{G}}, [x_2 \cdot y]_{\mathbb{G}}, \pi')$

$\underline{\mathsf{V}(X)}$:
Parse $X =: ([x_1]_{\mathbb{G}}, [x_2]_{\mathbb{G}}, \pi)$
Return $\mathsf{Ver}(\mathsf{crs}, ([x_1]_{\mathbb{G}}, [x_2]_{\mathbb{G}}), \pi)$

Fig. 8. Group scheme for $\mathbb{H}$

commutative (due to commutativity of the group operation in $\mathbb{G}$), as well as associativity and commutativity of PAdd. It is closed and has order q because only valid group elements can be added, and there are only q valid group elements. Furthermore the inversion and exponentiation algorithms are correct due to the correctness of the corresponding algorithms in the group scheme of $\mathbb{G}$ as well as the correctness of the PMult algorithm.

Algebraicity of the group $\mathbb{H}$. We show that any adversary in the group $\mathbb{H}$ is algebraic. First, we state this as a theorem:

Theorem 4. *The group $\mathbb{H}$ as constructed above is an algebraic group in the sense of Definition 9 under the GKEA in $\mathbb{G}$ and the adaptive soundness of the QANIZK scheme.*

Proof. An extractor for a representation of the group elements output by the adversary can be seen in Figure 9. The extractor $\mathcal{E}_{\mathrm{GKEA}}$ of the GKEA is used as a subroutine to extract a representation from the elements.

There are two scenarios that lead to the extractor $\mathcal{E}$ of the AGM not being able to output a representation. Either the elements do not have the correct form, i.e. there is an element $([s]_{\mathbb{G}}, [t]_{\mathbb{G}}, \pi)$ where $t \neq a \cdot s$, or we are in the unlikely case that the GKEA-extractor fails even though the elements have the correct form.

$\underline{\mathcal{E}([x_1]_{\mathbb{H}}, \ldots [x_n]_{\mathbb{H}}, ([y_1]_{\mathbb{G}}, [y_2]_{\mathbb{G}}, \pi); r_{\mathcal{A}})}$
If $(\neg \mathsf{V}(([y_1]_{\mathbb{G}}, [y_2]_{\mathbb{G}}, \pi)))$ return $\bot$
Parse $[x_i]_{\mathbb{H}} =: ([s_i]_{\mathbb{G}}, [t_i]_{\mathbb{G}}, \pi_i)$
Parse $\left[\mathbf{y}^\top\right]_{\mathbb{G}} := ([y_1]_{\mathbb{G}}, [y_2]_{\mathbb{G}})$
$\mathbf{z} \xleftarrow{\boxdot} \mathcal{E}_{\text{GKEA}}(([s_i]_{\mathbb{G}}, [t_i]_{\mathbb{G}})_i, \left[\mathbf{y}^\top\right]_{\mathbb{G}}, r_{\mathcal{A}})$
Return $\mathbf{z}$

$\underline{\mathcal{R}([\mathbf{M}]_{\mathbb{G}}, \mathsf{crs})}$
// generate challenge elements:
Define par as $([\mathbf{M}]_{\mathbb{G}}, \mathsf{crs})$
$\left(([x_i]_{\mathbb{H}})_i, C'\right) \xleftarrow{\boxdot} \textsc{Init}_{\mathcal{A}}(\mathsf{par})$
$r_{\mathcal{A}} \xleftarrow{\boxdot} \{0,1\}^{p(\lambda)}$
For $([y_1]_{\mathbb{G}}, [y_2]_{\mathbb{G}}, \pi^*) \xleftarrow{\boxdot} \mathcal{A}(([x_i]_{\mathbb{H}})_i; r_{\mathcal{A}})$
do
  If $(\mathsf{Ver}(\mathsf{crs}, ([y_1]_{\mathbb{G}}, [y_2]_{\mathbb{G}}), \pi^*)$ then
    $\mathbf{z} \xleftarrow{\boxdot} \mathcal{E}(([x_i]_{\mathbb{H}})_i, ([y_1]_{\mathbb{G}}, [y_2]_{\mathbb{G}}, \pi^*), r_{\mathcal{A}})$
    If $(\prod_{i=1}^n [s_i \cdot z_i]_{\mathbb{G}} \neq [y_1]_{\mathbb{G}} \vee$
$\prod_{i=1}^n [t_i \cdot z_i]_{\mathbb{G}} \neq [y_2]_{\mathbb{G}}$) then
      $\textsc{Final}_{\mathsf{as}}([y_1]_{\mathbb{G}}, [y_2]_{\mathbb{G}}, \pi^*)$
end for
$y_1, y_2 \xleftarrow{\boxdot} \mathbb{Z}_q$, $\pi^* \xleftarrow{\boxdot} \Pi$
$\textsc{Final}_{\mathsf{as}}([y_1]_{\mathbb{G}}, [y_2]_{\mathbb{G}}, \pi^*)$

Fig. 9. Extractor for our algebraic group as well as a reduction to the adaptive soundness of Π_{as} from Figure 3.

In case that the elements do not have the correct form, but the proof part π is valid anyway, this output element breaks the soundness of the QANIZK scheme.

Figure 9 shows a reduction that uses an adversary for which the extractor is unable to extract a representation to attack the soundness of the QANIZK scheme.

The reduction generates the input elements for $\mathcal{A}$ through the $\textsc{Init}_{\mathcal{A}}$ procedure of the game $\textsc{Game}_{\mathcal{A}}$ that $\mathcal{A}$ is designed to play. We also note that in order to be used for this game, the adversary $\mathcal{A}$ does not need to win the game $\textsc{Game}_{\mathcal{A}}$, it merely needs to be an adversary for which the extractor fails, i.e. it either outputs nothing or an incorrect representation. If the extracted representation $\mathbf{z}$ only matches one of the elements $[y_1]_{\mathbb{G}}, [y_2]_{\mathbb{G}}$ but not the other, the reduction $\mathcal{R}$ knows that the adversary must have broken the soundness of the QANIZK scheme. However, in the cases where $\mathbf{z}$ does not match either element of $\mathbb{G}$ output by the adversary, or the extractor is unable to output any representation at all, the reduction may not be able to decide whether the adversary has broken the soundness or the extractor failed due to the negligible failure probability of the GKEA-extractor. In fact, if the DDH problem is hard in $\mathbb{G}$, it is difficult for the reduction to decide why the extraction failed in these cases. Therefore, the reduction attempts to use the output in any case where the extraction is not a completely correct representation of the output of $\mathcal{A}$. If $\mathcal{A}$'s output is correctly extractable or the proof is incorrect, the reduction chooses its solution at random instead.

We obtain the following probability that the reduction wins its game $\textsc{Game}_{\mathsf{as}}$ against the soundness of the QANIZK scheme.

$$\text{Adv}^{\mathsf{as}}_{\Phi,\mathcal{R}}(\lambda) = \text{Adv}^{\mathsf{alg}}_{\mathbb{H},\mathcal{A}}(\lambda) - \text{Adv}^{\mathsf{gkea}}_{\mathbb{G},\mathcal{A},\mathcal{E}_{\mathcal{A}}}(\lambda),$$

where $\mathrm{Adv}^{\mathsf{alg}}_{\mathbb{H},\mathcal{A}}(\lambda)$ is the advantage of $\mathcal{A}$ in the algebraic game, i.e. the probability that $\mathcal{A}$ outputs a group element for which $\mathcal{E}_{\mathsf{GKEA}}$ can not extract a representation. □

Remark 2. We note that the QANIZK argument system by Kiltz and Wee requires an asymmetric Type 3 pairing and the KerMDH assumption. Beyond that, any additively homomorphic QANIZK argument system for linear subspace can be used to instantiate this construction. It is possible to instantiate algebraic groups without pairings if the underlying additively homomorphic QANIZK system does not require pairings.

4.1 Cryptographic Hardness in $\mathbb{H}$

We show that all computational hardness assumptions that hold in $\mathbb{G}$ also hold in $\mathbb{H}$. This means that our construction can be used to turn any cyclic prime-order group into an algebraic group while preserving computational hardness assumptions.

Theorem 5 ($\mathcal{P}_{\mathbb{G}} \Rightarrow \mathcal{P}_{\mathbb{H}}$)**.** *Let $\mathcal{P}$ be a computational problem in group $\mathbb{G}$ (denoted by $\mathcal{P}_{\mathbb{G}}$) and $\mathcal{P}_{\mathbb{H}}$ be the same problem but in group $\mathbb{H}$. Then, if the argument system Φ is perfectly zero-knowledge and the problem $\mathcal{P}$ is hard in $\mathbb{G}$ then the problem $\mathcal{P}$ is hard in $\mathbb{H}$.*

More precisely, if there is an adversary $\mathcal{A}$ that solves $\mathcal{P}$ in $\mathbb{H}$, then there is a reduction $\mathcal{R}$ that solves $\mathcal{P}$ in $\mathbb{G}$ with success probability

$$\mathrm{Adv}^{\mathcal{P}}_{\mathbb{G},\mathcal{R}}(\lambda) = \mathrm{Adv}^{\mathcal{P}}_{\mathbb{H},\mathcal{A}}(\lambda)$$

Proof. This proof works the same way as the proof for Theorem 3 with the only difference being the way the group elements are "forwarded". Group element "forwarding" involves checking the attached NIZK arguments (in the case of forwarding from $\mathcal{A}$ to the procedures in $\mathbb{G}$), and simulating arguments to attach (in the case of forwarding responses from procedures to $\mathcal{A}$). The procedure implementations are shown in Figure 10. We note that since the elements constructed by $\mathcal{R}$ are in the span of $\mathbf{M}$, the distribution of the simulated arguments is the same as the distribution of arguments generated with Prove would be. □

5 Conclusion

In this paper, we propose the first formal treatment on instantiating the Algebraic Group Model (AGM). More precisely, we have two constructions of algebraic groups from different primitives. Both constructions require the Generalized Knowledge of Exponent Assumption (GKEA) to achieve the algebraic property. Additionally, they require either pairings or additively homomorphic Non-Interactive Zero-Knowledge (NIZK) Argument systems. The additively homomorphic NIZK can be implemented by a variant of Diffie-Hellman assumption in pairing groups. We leave it as an interesting open problem to construct such a

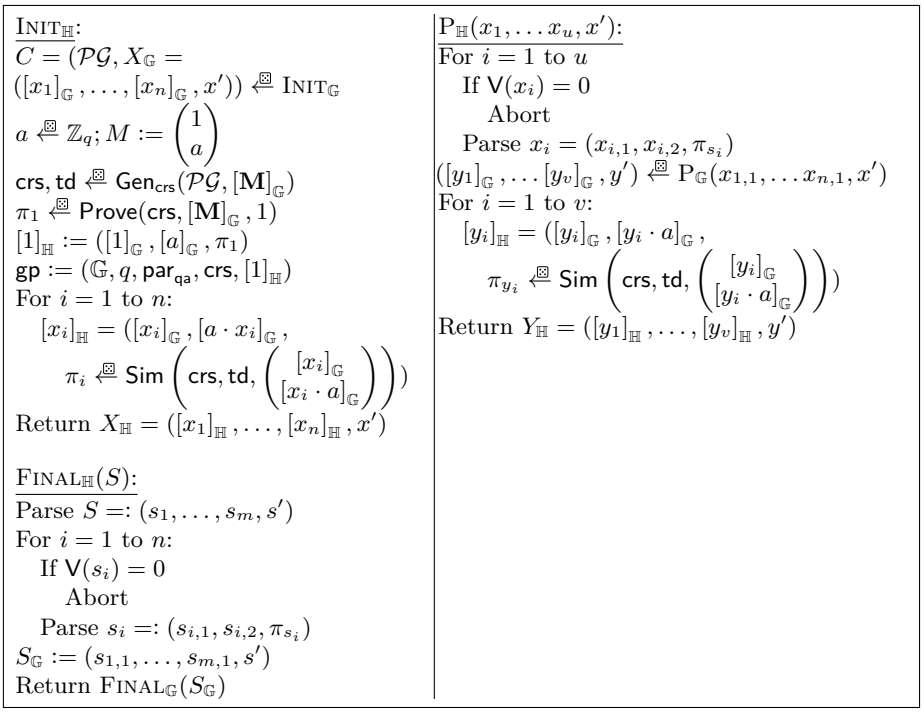

Fig. 10. The oracles provided by the reduction $\mathcal{R}$ in $\mathbb{H}$

NIZK argument system without pairings. By our construction, that will give us an algebraic group without using pairings. In the end, our constructions show that all the results in the AGM also hold in the Standard Model by assuming the KEA and pairings.

We propose instantiating the AGM with other (weaker) assumptions and primitives as the main future direction.

Acknowledgments. We thank one of the Eurocrypt 2019 reviewers for pointing us to the construction with Type 1 pairings, and one of the Asiacrypt 2019 reviewers for the remark on extractable one-way functions and indistinguishability obfuscations.

References

1. Abdalla, M., Benhamouda, F., MacKenzie, P.: Security of the J-PAKE password-authenticated key exchange protocol. In: 2015 IEEE Symposium on Security and Privacy. pp. 571–587. IEEE Computer Society Press, San Jose, CA, USA (May 17–21, 2015)
2. Abe, M., Ambrona, M., Ohkubo, M., Tibouchi, M.: Lower bounds on structure-preserving signatures for bilateral messages. In: Catalano, D., De Prisco, R. (eds.)

SCN 18: 11th International Conference on Security in Communication Networks. Lecture Notes in Computer Science, vol. 11035, pp. 3–22. Springer, Heidelberg, Germany, Amalfi, Italy (Sep 5–7, 2018)
3. Abe, M., Camenisch, J., Dowsley, R., Dubovitskaya, M.: On the impossibility of structure-preserving deterministic primitives. In: Lindell, Y. (ed.) TCC 2014: 11th Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 8349, pp. 713–738. Springer, Heidelberg, Germany, San Diego, CA, USA (Feb 24–26, 2014)
4. Abe, M., Fehr, S.: Perfect NIZK with adaptive soundness. In: Vadhan, S.P. (ed.) TCC 2007: 4th Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 4392, pp. 118–136. Springer, Heidelberg, Germany, Amsterdam, The Netherlands (Feb 21–24, 2007)
5. Abe, M., Groth, J., Haralambiev, K., Ohkubo, M.: Optimal structure-preserving signatures in asymmetric bilinear groups. In: Rogaway, P. (ed.) Advances in Cryptology – CRYPTO 2011. Lecture Notes in Computer Science, vol. 6841, pp. 649–666. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2011)
6. Abe, M., Groth, J., Ohkubo, M.: Separating short structure-preserving signatures from non-interactive assumptions. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology – ASIACRYPT 2011. Lecture Notes in Computer Science, vol. 7073, pp. 628–646. Springer, Heidelberg, Germany, Seoul, South Korea (Dec 4–8, 2011)
7. Abe, M., Kohlweiss, M., Ohkubo, M., Tibouchi, M.: Fully structure-preserving signatures and shrinking commitments. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015, Part II. Lecture Notes in Computer Science, vol. 9057, pp. 35–65. Springer, Heidelberg, Germany, Sofia, Bulgaria (Apr 26–30, 2015)
8. Agrikola, T., Hofheinz, D.: Interactively secure groups from obfuscation. In: Abdalla, M., Dahab, R. (eds.) PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part II. Lecture Notes in Computer Science, vol. 10770, pp. 341–370. Springer, Heidelberg, Germany, Rio de Janeiro, Brazil (Mar 25–29, 2018)
9. Albrecht, M.R., Farshim, P., Hofheinz, D., Larraia, E., Paterson, K.G.: Multilinear maps from obfuscation. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A: 13th Theory of Cryptography Conference, Part I. Lecture Notes in Computer Science, vol. 9562, pp. 446–473. Springer, Heidelberg, Germany, Tel Aviv, Israel (Jan 10–13, 2016)
10. Bellare, M., Palacio, A.: The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In: Franklin, M. (ed.) Advances in Cryptology – CRYPTO 2004. Lecture Notes in Computer Science, vol. 3152, pp. 273–289. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 2004)
11. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) Advances in Cryptology – EUROCRYPT 2006. Lecture Notes in Computer Science, vol. 4004, pp. 409–426. Springer, Heidelberg, Germany, St. Petersburg, Russia (May 28 – Jun 1, 2006)
12. Bernhard, D., Fischlin, M., Warinschi, B.: On the hardness of proving CCA-security of signed ElGamal. In: Cheng, C.M., Chung, K.M., Persiano, G., Yang, B.Y. (eds.) PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part I. Lecture Notes in Computer Science, vol. 9614, pp. 47–69. Springer, Heidelberg, Germany, Taipei, Taiwan (Mar 6–9, 2016)
13. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In:

Goldwasser, S. (ed.) ITCS 2012: 3rd Innovations in Theoretical Computer Science. pp. 326–349. Association for Computing Machinery, Cambridge, MA, USA (Jan 8–10, 2012)
14. Bitansky, N., Canetti, R., Paneth, O., Rosen, A.: On the existence of extractable one-way functions. In: Shmoys, D.B. (ed.) 46th Annual ACM Symposium on Theory of Computing. pp. 505–514. ACM Press, New York, NY, USA (May 31 – Jun 3, 2014)
15. Blazy, O., Kiltz, E., Pan, J.: (Hierarchical) identity-based encryption from affine message authentication. In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology – CRYPTO 2014, Part I. Lecture Notes in Computer Science, vol. 8616, pp. 408–425. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2014)
16. Boneh, D., Lipton, R.J.: Algorithms for black-box fields and their application to cryptography (extended abstract). In: Koblitz, N. (ed.) Advances in Cryptology – CRYPTO'96. Lecture Notes in Computer Science, vol. 1109, pp. 283–297. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 1996)
17. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: Boyd, C. (ed.) Advances in Cryptology – ASIACRYPT 2001. Lecture Notes in Computer Science, vol. 2248, pp. 514–532. Springer, Heidelberg, Germany, Gold Coast, Australia (Dec 9–13, 2001)
18. Boneh, D., Venkatesan, R.: Breaking RSA may not be equivalent to factoring. In: Nyberg, K. (ed.) Advances in Cryptology – EUROCRYPT'98. Lecture Notes in Computer Science, vol. 1403, pp. 59–71. Springer, Heidelberg, Germany, Espoo, Finland (May 31 – Jun 4, 1998)
19. Brown, D.R.L.: Generic groups, collision resistance, and ecdsa. Designs, Codes and Cryptography 35(1), 119–152 (Apr 2005), https://doi.org/10.1007/s10623-003-6154-z
20. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) Advances in Cryptology – CRYPTO 2004. Lecture Notes in Computer Science, vol. 3152, pp. 56–72. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 2004)
21. Coron, J.S.: Optimal security proofs for PSS and other signature schemes. In: Knudsen, L.R. (ed.) Advances in Cryptology – EUROCRYPT 2002. Lecture Notes in Computer Science, vol. 2332, pp. 272–287. Springer, Heidelberg, Germany, Amsterdam, The Netherlands (Apr 28 – May 2, 2002)
22. Damgård, I.: Towards practical public key systems secure against chosen ciphertext attacks. In: Feigenbaum, J. (ed.) Advances in Cryptology – CRYPTO'91. Lecture Notes in Computer Science, vol. 576, pp. 445–456. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 11–15, 1992)
23. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for Diffie-Hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013, Part II. Lecture Notes in Computer Science, vol. 8043, pp. 129–147. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2013)
24. Fleischhacker, N., Jager, T., Schröder, D.: On tight security proofs for Schnorr signatures. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology – ASIACRYPT 2014, Part I. Lecture Notes in Computer Science, vol. 8873, pp. 512–531. Springer, Heidelberg, Germany, Kaoshiung, Taiwan, R.O.C. (Dec 7–11, 2014)
25. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018, Part II. Lecture Notes in Computer Science, vol. 10992, pp. 33–62. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2018)

26. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics 156(16), 3113 – 3121 (2008)
27. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.S. (eds.) Advances in Cryptology – EUROCRYPT 2016, Part II. Lecture Notes in Computer Science, vol. 9666, pp. 305–326. Springer, Heidelberg, Germany, Vienna, Austria (May 8–12, 2016)
28. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) Advances in Cryptology – EUROCRYPT 2008. Lecture Notes in Computer Science, vol. 4965, pp. 415–432. Springer, Heidelberg, Germany, Istanbul, Turkey (Apr 13–17, 2008)
29. Hada, S., Tanaka, T.: On the existence of 3-round zero-knowledge protocols. In: Krawczyk, H. (ed.) Advances in Cryptology – CRYPTO'98. Lecture Notes in Computer Science, vol. 1462, pp. 408–423. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 23–27, 1998)
30. Jutla, C.S., Roy, A.: Shorter quasi-adaptive NIZK proofs for linear subspaces. In: Sako, K., Sarkar, P. (eds.) Advances in Cryptology – ASIACRYPT 2013, Part I. Lecture Notes in Computer Science, vol. 8269, pp. 1–20. Springer, Heidelberg, Germany, Bengalore, India (Dec 1–5, 2013)
31. Kakvi, S.A., Kiltz, E.: Optimal security proofs for full domain hash, revisited. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012. Lecture Notes in Computer Science, vol. 7237, pp. 537–553. Springer, Heidelberg, Germany, Cambridge, UK (Apr 15–19, 2012)
32. Kiltz, E., Pan, J., Wee, H.: Structure-preserving signatures from standard assumptions, revisited. In: Gennaro, R., Robshaw, M.J.B. (eds.) Advances in Cryptology – CRYPTO 2015, Part II. Lecture Notes in Computer Science, vol. 9216, pp. 275–295. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2015)
33. Kiltz, E., Wee, H.: Quasi-adaptive NIZK for linear subspaces revisited. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015, Part II. Lecture Notes in Computer Science, vol. 9057, pp. 101–128. Springer, Heidelberg, Germany, Sofia, Bulgaria (Apr 26–30, 2015)
34. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) Advances in Cryptology – CRYPTO 2005. Lecture Notes in Computer Science, vol. 3621, pp. 546–566. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2005)
35. Maurer, U.M.: Abstract models of computation in cryptography (invited paper). In: Smart, N.P. (ed.) 10th IMA International Conference on Cryptography and Coding. Lecture Notes in Computer Science, vol. 3796, pp. 1–12. Springer, Heidelberg, Germany, Cirencester, UK (Dec 19–21, 2005)
36. Maurer, U.M., Wolf, S.: Lower bounds on generic algorithms in groups. In: Nyberg, K. (ed.) Advances in Cryptology – EUROCRYPT'98. Lecture Notes in Computer Science, vol. 1403, pp. 72–84. Springer, Heidelberg, Germany, Espoo, Finland (May 31 – Jun 4, 1998)
37. Nechaev, V.I.: Complexity of a determinate algorithm for the discrete logarithm. Mathematical Notes 55(2), 165–172 (Feb 1994), https://doi.org/10.1007/BF02113297
38. Paillier, P., Vergnaud, D.: Discrete-log-based signatures may not be equivalent to discrete log. In: Roy, B.K. (ed.) Advances in Cryptology – ASIACRYPT 2005. Lecture Notes in Computer Science, vol. 3788, pp. 1–20. Springer, Heidelberg, Germany, Chennai, India (Dec 4–8, 2005)
39. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. Journal of Cryptology 13(3), 361–396 (Jun 2000)

40. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of Cryptology 4(3), 161–174 (Jan 1991)
41. Schnorr, C.P.: Security of blind discrete log signatures against interactive attacks. In: Qing, S., Okamoto, T., Zhou, J. (eds.) ICICS 01: 3rd International Conference on Information and Communication Security. Lecture Notes in Computer Science, vol. 2229, pp. 1–12. Springer, Heidelberg, Germany, Xian, China (Nov 13–16, 2001)
42. Seurin, Y.: On the exact security of Schnorr-type signatures in the random oracle model. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012. Lecture Notes in Computer Science, vol. 7237, pp. 554–571. Springer, Heidelberg, Germany, Cambridge, UK (Apr 15–19, 2012)
43. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) Advances in Cryptology – EUROCRYPT'97. Lecture Notes in Computer Science, vol. 1233, pp. 256–266. Springer, Heidelberg, Germany, Konstanz, Germany (May 11–15, 1997)
44. Wu, J., Stinson, D.: An efficient identification protocol and the knowledge-of-exponent assumption. Cryptology ePrint Archive, Report 2007/479 (2007), `http://eprint.iacr.org/2007/479`