On Instantiating the Algebraic Group Model from Falsifiable Assumptions

Thomas Agrikola[1](✉), Dennis Hofheinz[2](✉), and Julia Kastner[2]

[1] Karlsruhe Institute of Technology, Karlsruhe, Germany
thomas.agrikola@kit.edu
[2] ETH Zürich, Zürich, Switzerland
{hofheinz,julia.kastner}@inf.ethz.ch

Abstract. We provide a standard-model implementation (of a relaxation) of the algebraic group model (AGM, [Fuchsbauer, Kiltz, Loss, CRYPTO 2018]). Specifically, we show that every algorithm that uses our group is algebraic, and hence "must know" a representation of its output group elements in terms of its input group elements. Here, "must know" means that a suitable extractor can extract such a representation efficiently. We stress that our implementation relies only on falsifiable assumptions in the standard model, and in particular does not use any knowledge assumptions.

As a consequence, our group allows to transport a number of results obtained in the AGM into the standard model, under falsifiable assumptions. For instance, we show that in our group, several Diffie-Hellman-like assumptions (including computational Diffie-Hellman) are equivalent to the discrete logarithm assumption. Furthermore, we show that our group allows to prove the Schnorr signature scheme tightly secure in the random oracle model.

Our construction relies on indistinguishability obfuscation, and hence should not be considered as a practical group itself. However, our results show that the AGM is a realistic computational model (since it can be instantiated in the standard model), and that results obtained in the AGM are also possible with standard-model groups.

Keywords: Indistinguishability obfuscation · Algebraic group model · Schnorr signatures

1 Introduction

The generic group model. In order to analyze the plausibility and relative strength of computational assumptions in cyclic groups, Shoup [38] and Maurer [31] have

Work done while all authors were at Karlsruhe Institute of Technology.
T. Agrikola and J. Kastner—Supported by ERC Project PREP-CRYPTO 724307.
D. Hofheinz—Supported by ERC Project PREP-CRYPTO 724307, and by DFG project GZ HO 4534/4-2.

A. Canteaut and Y. Ishai (Eds.): EUROCRYPT 2020, LNCS 12106, pp. 96–126, 2020.
https://doi.org/10.1007/978-3-030-45724-2_4

proposed the *generic group model* (GGM). In the GGM, any adversary can only interact with the modeled group through an oracle. In particular, all computations in that group must be explicitly expressed in terms of the group operation. To prevent an adversary from locally performing computations, that adversary gets to see only truly random strings (in [38]) or independent handles (in [31]) as representations of group elements.[1]

The discrete logarithm and even many Diffie-Hellman-style problems are hard generically (i.e., when restricting group operations in the above way) [32,38]. Hence, the only way to break such a generically hard assumption in a concrete group is to use the underlying group representation in a nontrivial way. In that sense, the GGM can be very useful as a sanity check for the validity of a given assumption, or even the security of a given cryptographic scheme. However, generic groups cannot be implemented: there exist cryptographic schemes that are secure in the GGM, but insecure when instantiated with *any* concrete group [15].

The algebraic group model. The algebraic group model (AGM, [21]) is a relaxation of the GGM that tries to avoid impossibilities as in [15] while preserving the GGM's usefulness. Specifically, the AGM only considers *algebraic* (rather than generic) adversaries. An algebraic adversary $\mathcal{A}$ can make arbitrary use of the representation of group elements, but must supply an explicit decomposition for any of its output group elements in terms of input group elements. In other words, $\mathcal{A}$ must also output an explanation of how any group element in its output was computed from its input using the group operation.

Now [21] show that many GGM proofs only use this type of algebraicity of an adversary, and carry over to the AGM. At the same time, GGM impossibilities like [15] do not apply to the AGM, since algebraic adversaries are able to work with the actual group (and not only with random or abstract representations of group elements).

The AGM and knowledge assumptions. The AGM is closely related to the notions of knowledge assumptions and extractability. To illustrate, assume that for any (possibly non-algebraic) adversary $\mathcal{A}$, we can find an extractor $\mathcal{E}$ that manages to extract from $\mathcal{A}$ a decomposition of $\mathcal{A}$'s output in terms of $\mathcal{A}$'s input. Then, composing $\mathcal{E}$ and $\mathcal{A}$ yields an algebraic adversary $\mathcal{A}_{\mathsf{alg}}$. In this situation, we can then say that without loss of generality, any adversary can be assumed to be algebraic.[2] Conversely, any algebraic adversary by definition yields the results of such an extraction in its output.

This observation also provides a blueprint to *instantiating* the AGM: simply prove that any adversary $\mathcal{A}$ can be replaced by an algebraic adversary $\mathcal{A}_{\mathsf{alg}}$, possibly using an extraction process as above. If this extraction requires $\mathcal{A}$'s code

[1] Other black-box abstractions of groups with similar ramifications exist [6,34].

[2] This observation about algebraic adversaries has already been made in [9,35]. Also, similar but more specific knowledge assumptions have been used to prove concrete cryptographic constructions secure, e.g., [4,14,16,25].

and randomness but no other trapdoor, we obtain an AGM instantiation based on a knowledge assumption such as the knowledge of exponent assumption [14]. Indeed, this was recently done by [30] under a very strong generalized version of the knowledge of exponent assumption. Unfortunately, such knowledge assumptions are not falsifiable in the sense of Naor [33]. It is thus not entirely clear how to assess the plausibility of such a universal and strong knowledge assumption. Naturally, the question arises whether an AGM implementation inherently requires such strong and non-falsifiable assumptions. Or, more generally:

Can we achieve knowledge-type properties
from falsifiable assumptions?

Note that in the AGM, the discrete logarithm assumption implies the existence of extractable one-way functions (EOWFs) with unbounded auxiliary input. The existence of such EOWFs, however, conflicts with the existence of indistinguishability obfuscation, [5]. Due to this barrier, we can only hope for an instantiation of some suitably relaxed variant of the AGM from falsifiable assumptions.

Our strategy: private extraction. There is also another way to instantiate the AGM: show that it is possible to extract a decomposition of $\mathcal{A}$'s outputs *from these outputs* and a suitable (secret) extraction trapdoor. In other words, our idea is to avoid non-falsifiable knowledge assumptions by assuming that extraction requires a special trapdoor that can be generated alongside the public parameters of the group. This entails a number of technical difficulties (see below), but allows us to rely entirely on falsifiable assumptions.

Specifically, our main result is an *algebraic wrapper* that transforms a given cyclic group into a new one which allows for an extraction of representations. More specifically, an element of the new group carries an encrypted representation of this group element relative to a fixed basis (i.e., set of group elements). Upon group operations, this representation is updated, and a special trapdoor (generated alongside the public parameters) allows to extract it.

Our results. Our strategy allows us to retrieve several AGM results (from [21,22]) in the standard model, in the sense that the group can be concretely implemented from falsifiable assumptions.[3] In particular, we show that in our group,

– the discrete logarithm assumption, the computational Diffie-Hellman assumption, the square Diffie-Hellman assumption, and the linear-combination Diffie-Hellman assumption (see [21]) are all equivalent,

[3] Note that by "standard model", we mean that the group itself is formulated without idealizations and can be concretely implemented. While our construction itself does not rely on the ROM, we still can transfer some ROM proofs in the AGM to ROM proofs using our concrete group instantiation. We stress that a standard model instantiation of the (full-fledged) AGM from very strong *non-falsifiable* assumptions is already known due to [30].

– the security of the Schnorr signature scheme [37] can be *tightly* reduced to the discrete logarithm assumption escaping impossibility results due to [19].[4]

While, on a technical level, the AGM proofs from [21,22] need to be adapted, the general AGM proof strategies (that rely on extraction) can be replicated.

Limitations. We note that not all known AGM proofs can be transported to the standard model. For instance, [21] also prove the Boneh-Lynn-Shacham [7] signature scheme tightly secure in the AGM. Their reduction relies on the fact that the view of a signature forger is statistically independent of how simulated signatures are prepared by the reduction. However, with our algebraic wrapper, group elements (and thus BLS signatures) always carry an encrypted representation of how they were generated. In this case, our private extraction strategy also reveals additional (statistical, computationally hidden) information to an adversary. This additional information is problematic in the AGM-based BLS proof of [21]. We believe it is an interesting open problem to obtain a tight security proof for the BLS scheme with our group.[5]

Furthermore, as we will detail below, the amount of information we can extract from a group element is limited by the size of that group element. In particular, in settings in which no a-priori bound on the size of a desired algebraic representation is known, our techniques do not apply. This can be problematic, e.g., for constructions that depend on q-type assumptions.

Our assumptions. We stress that our algebraic wrapper relies on a strong (but falsifiable) computational assumption: the existence of subexponentially strong indistinguishability obfuscation (subexp-iO).[6] Additionally, we assume a re-randomizable encryption scheme. Together with subexp-iO, this implies a number of other strong primitives that we use: a variant of probabilistic iO (see [11]), fully homomorphic encryption (see [11]), and dual-mode non-interactive zero-knowledge (see [27]).

Interpretation. Due to their inefficiency, we view algebraic wrappers not as a tool to obtain practical cryptographic primitives. Rather, we believe that algebraic wrappers show that the AGM is a useful and realistic abstraction and not merely an idealized model which heuristically captures known adversaries: we show that AGM proofs *can* be replicated in the standard model, and even without resorting to knowledge assumptions.

[4] Tight security reductions provide a tight relation between the security of cryptographic schemes and the hardness of computational problems. Apart from their theoretical importance, tight reductions are also beneficial for practice, since they allow smaller keylength recommendations.

[5] We note that impossibility results for tight reductions of schemes like BLS (e.g., [12]) do not apply in our case, as the representation of our group elements is not unique.

[6] We note that iO and knowledge assumptions contradict each other [5]. However, we stress that the notion of *private* extractability we obtain does *not* contradict iO.

On implementing idealized models. Replacing idealized (heuristic) models with concrete standard-model implementations is a widely studied intriguing problem. A well-known example for this is the line of work on programmable hash functions. A programmable hash function due to [26] is a cryptographic primitive which can be used to replace random oracles in several cryptographic schemes. Following their introduction, a line of work [20,28,29] leveraged multi-linear maps or indistinguishability obfuscation to transport proofs from the random oracle model to the standard model. Our results can be interpreted as following this endeavor by leveraging indistinguishability obfuscation to replace the AGM with a standard model implementation (from falsifiable assumptions). From this angle, our algebraic wrapper relates to the AGM as programmable hash functions relate to the ROM.

1.1 Technical Overview

Algebraic wrappers. In the following, we speak of *group schemes* ([3], also called encoding schemes in [23]) as a generalization of groups with potentially non-unique encodings of group elements. This implies that a dedicated algorithm is required to determine if two given group elements are equal.[7] Our algebraic wrapping process takes a group $\mathbb{G}$ (which we call "base group") as input, and outputs a new group scheme $\mathbb{H}$ which allows for an efficient extraction process. Concretely, every $\mathbb{H}$-element $\widehat{h}$ can be viewed as a $\mathbb{G}$-element $h \in \mathbb{G}$, plus auxiliary information aux.

Intuitively, aux carries (encrypted) information that allows to express h as a linear combination of fixed base elements $b_1, \ldots, b_n \in \mathbb{G}$. The corresponding decryption key (generated alongside the group parameters) allows to extract this information, and essentially yields the information any algebraic adversary (in the sense of the AGM) would have to provide for any output group element. However, we are facing a number of technical problems:

(a) The group operation algorithm should update aux (in the sense that the linear combinations encrypted in the input elements should be added).
(b) Validity of aux should be ensured (so that no adversary can produce an $\mathbb{H}$-element from which no valid linear combination can be extracted from aux).
(c) It should be possible to switch the basis elements $b_1, \ldots, b_n$ to an application-dependent basis. (For instance, to prove a signature scheme like Schnorr's [37] secure, one would desire to set the basis vectors to elements from an externally given computational challenge.)
(d) To preserve tightness of reductions from the AGM (which is necessary in some of our applications), it should be possible to re-randomize group element encodings statistically.

[7] That is, formally, the group is defined as the quotient set of all well-formed bitstrings modulo the equivalence relation induced by the equality test.

Our solution largely follows the group scheme from [3]. In particular, (a) will be solved by encrypting the coefficients $z_1, \ldots, z_n$ with $h = \sum_i b_i^{z_i}$ using a homomorphic encryption scheme in aux. Hence, such coefficient vectors can be added homomorphically during the group operation. For (b), we will add a suitable non-interactive zero-knowledge proof of consistency in aux.[8] For (c), we adapt a "switching" lemma from [3]. In [3], that lemma allows to switch between two different representations of the same group element, but under a fixed basis. In our case, we show that similar techniques allow to also switch the group elements that form this basis. This switching property already implies a notion of computational re-randomizability. Finally, for (d), we introduce a re-randomization lemma using techniques from (c) in conjunction with a novel notion for probabilistic iO.

At this point, one main conceptual difference to the line of work [1,3,17] is that the basis elements $b_1, \ldots, b_n$ appear as part of the functionality of the new group scheme $\mathbb{H}$, not only in a proof. In particular, our construction must be able to deal with arbitrary b_i that are not necessarily randomly chosen. This issue is dealt with by additional linear randomization of the base group elements.

Another main conceptual difference to [1,3,17] is the notion of statistical re-randomizability of group elements. The group schemes from [1,3,17] do not satisfy this property. This will be resolved by developing a stronger notion of *statistically correct* probabilistic iO which may be of independent interest.

We note, however, that our techniques are inherently limited in the following sense: our extraction can only extract as much information as contained in (the auxiliary information of) group elements. Technically speaking, we cannot treat settings in which the size of the basis $b_1, \ldots, b_n$ is not known in advance (e.g., in case of constructions based on q-type assumptions).

Applications. The applications we consider have already been considered for the AGM in [21,22]. Hence, in this description, we focus on the technical differences that our extraction approach entails for these proofs.

First, recall that in the AGM by [21], an adversary outputs an algebraic representation of each output group element to the basis of its input group elements. Therefore, this basis depends also on the respective security game. On the other hand, in security proofs with our algebraic wrapper, a reduction needs to select such a basis in advance. The appropriate selection of such a basis is one of the main challenges when transferring proofs from the AGM to our setting. Namely, even though the basis as well as the representation of each group element is hidden, the choice of representations will still be information-theoretically known to the adversary. Therefore, security games that are identically distributed in the AGM might only be computationally indistinguishable in the wrapper, depending on the choice of a basis.

When transferring proofs from the AGM to our new group scheme, we thus use a technique we call *symmetrization* to extend the basis in such a way that

[8] Note that this approach is related to [8] in the sense that we restrict the homomorphic operations an adversary can perform on encodings by requiring a consistency proof.

security games are identically distributed in the relevant situations. In a nutshell, symmetrization achieves a uniform way to express challenge elements across most games of a security proof, and yields statistical security guarantees.

Another challenge is the implementation of tight security reductions in the wrapper. In some security reductions, the basis of the group and the algebraic representations of oracle responses need to be switched in order to be able to extract a useful algebraic representation. However, as we only achieve computationally indistinguishable group element representations, switching the representations of q oracle responses would lead to a q-fold computational loss, compromising the tightness of the reduction.

We show that it is possible to circumvent this loss by constructing oracle responses via the group operation from so-called *origin elements*, reducing the number of elements whose representation gets switched to a constant. In a nutshell, we derive many coordinated oracle answers from just few group elements (the "origin elements"), such that switching these origin elements affects (and changes) all oracle answers.

1.2 Related Work

This work builds upon the line of work [1,3,17] who build group schemes from iO. [3] lays the conceptual foundations for the construction of group schemes with non-unique encodings from iO and uses this framework to equip groups with multilinear maps. [17] extends this approach by allowing partial evaluations of the multilinear map yielding a graded encoding scheme. In contrast to [1,3,17] does not extend the functionality of an underlying group, but builds a group scheme with reduced functionality (group elements lack a unique representation). The resulting group scheme allows to mimic commonly used proof techniques from the generic group model. This is demonstrated by proving the validity of an adaptive variant of the Uber assumption family [10] in the constructed group scheme. Our results can hence be viewed as an extension of [1].

[30] make a first step towards instantiating the AGM. The authors identify an equivalence between the AGM and a very strong generalized version of the knowledge of exponent assumption [14], thus giving rise to the first instantiation of the AGM.

Roadmap

In Sect. 2, we recall some preliminaries and develop the mentioned variant of probabilistic iO. In Sect. 3, we present our notion of algebraic wrappers and give an iO-based instantiation. Section 4 contains results transported from the AGM to our wrapper setting, along with a description of how AGM proof techniques can be adapted. In the full version of this paper [2], we provide (besides further standard definitions and more motivation) an analysis of the Schnorr-signed ElGamal encryption scheme with our algebraic wrapper.

2 Preliminaries

Notation

Throughout this paper λ denotes the security parameter. For a natural number $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, \ldots, n\}$. A function $\mathrm{negl}\colon \mathbb{N} \to \mathbb{R}$ is negligible in λ if for every constant $c \in \mathbb{N}$, there exists a bound $n_c \in \mathbb{R}$, such that for all $n \geq n_c$, $|\mathrm{negl}(n)| \leq n^{-c}$. Given a finite set S, the notation $x \leftarrow S$ means a uniformly random assignment of an element of S to the variable x. Given an algorithm A, the notation $y \leftarrow A(x)$ means evaluation of A on input of x with fresh random coins and assignment to the variable y. The notation $\mathcal{A}^{\mathcal{O}}$ indicates that the algorithm $\mathcal{A}$ is given oracle access to $\mathcal{O}$. Given a random variable B, $\mathsf{supp}(B)$ denotes the support of B.

Let $\mathbb{G}$ be a finite cyclic group with generator g and order p. For $x \in \mathbb{Z}_p$, the notation $[x]_{\mathbb{G}}$ denotes the group element g^x. Note that using this notation does not imply knowledge of x. Let $\mathbb{K}$ be a field and V be a vector space over $\mathbb{K}$ of finite dimension n. For $i \in [n]$, $\mathbf{e_i}$ denotes the vector which carries 1 in its i-th entry and 0 in all other entries.

In game based proofs, out_i denotes the output of game G_i.

2.1 Subset Membership Problem

Let $\mathcal{L} = (\mathcal{L}_\lambda)_{\lambda \in \mathbb{N}}$ be a family of families of languages $L \subseteq X_\lambda$ in a universe $X_\lambda = X$. Further, let R be an efficiently computable witness relation, such that $x \in L$ if and only if there exists a witness $w \in \{0,1\}^{\mathsf{poly}(|x|)}$ with $R(x,w) = 1$ (for a fixed polynomial poly). We assume that we are able to efficiently and uniformly sample elements from L together with a corresponding witness, and that we are able to efficiently and uniformly sample elements from $X \setminus L$.

Definition 1 (Subset membership problem, [13]). *A subset membership problem $L \subseteq X$ is hard, if for any PPT adversary $\mathcal{A}$, the advantage*

$$\mathrm{Adv}^{\mathtt{smp}}_{L,\mathcal{A}}(\lambda) := \Pr[x \leftarrow L\colon \mathcal{A}(1^\lambda, x) = 1] - \Pr[x \leftarrow X \setminus L\colon \mathcal{A}(1^\lambda, x) = 1]$$

is negligible in λ.

We additionally require that for every L and every $x \in L$, there exists exactly one witness $r \in \{0,1\}^*$ with $R(x,w) = 1$. Note that given a cyclic group $\mathbb{G}$ of prime order p in which DDH is assumed to hold, the Diffie-Hellman language $L_{[(1,x)]_{\mathbb{G}}} := \{[(y, xy)]_{\mathbb{G}} \mid y \in \mathbb{Z}_p\}$ (for randomly chosen generators $[1]_{\mathbb{G}}, [x]_{\mathbb{G}}$) satisfies this definition. Another instantiation of Definition 1 is the language containing all commitments to a fixed value using a perfectly binding commitment scheme with unique opening.

2.2 Dual-mode NIWI

A dual-mode NIWI proof system is a variant of NIWI proofs [18] offering two computationally indistinguishable modes to setup the common reference string (CRS). A binding mode CRS provides perfect soundness guarantees whereas a hiding mode CRS provides perfect witness indistinguishability guarantees.

Definition 2 (Dual-mode NIWI proof system (syntax), [3,24]**).** *A* dual mode non-interactive witness-indistinguishable (NIWI) proof system *for a relation $\mathcal{R}$ is a tuple of PPT algorithms $\Pi = (\mathsf{Setup}, \mathsf{HSetup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Ext})$.*

$\mathsf{Setup}(1^\lambda)$**.** *On input of* 1^λ*,* Setup *outputs a* perfectly binding *common reference string* crs *and a corresponding extraction trapdoor* td_{ext}.

$\mathsf{HSetup}(1^\lambda)$**.** *On input of* 1^λ*,* HSetup *outputs a* perfectly hiding *common reference string* crs.

$\mathsf{Prove}(\mathsf{crs}, x, w)$**.** *On input of the CRS* crs*, a statement x and a corresponding witness w,* Prove *produces a proof π.*

$\mathsf{Verify}(\mathsf{crs}, x, \pi)$**.** *On of the CRS* crs*, a statement x and a proof π,* Verify *outputs* 1 *if the proof is valid and* 0 *otherwise.*

$\mathsf{Ext}(td_{\mathsf{ext}}, x, \pi)$**.** *On input the extraction trapdoor* td_{ext}*, a statement x and a proof π,* Ext *outputs a witness w.*

We require Π to satisfy the CRS indistinguishability, perfect completeness, perfect soundness, perfect extractability and perfect witness-indistinguishability.

For a more detailed definition, we refer the reader to the full version [2]. There are several instantiations of dual-mode NIWI proof systems satisfying the above definition (or statistical variants), [24,27,36].

2.3 Probabilistic Indistinguishability Obfuscation

Let $\mathcal{C} = (\mathcal{C}_\lambda)_{\lambda \in \mathbb{N}}$ be a family of sets $\mathcal{C}_\lambda$ of probabilistic circuits. A *circuit sampler* for $\mathcal{C}$ is defined as a family of (efficiently samplable) distributions $S = (S_\lambda)_{\lambda \in \mathbb{N}}$, where S_λ is a distribution over triplets (C_0, C_1, z) with $C_0, C_1 \in \mathcal{C}_\lambda$ such that C_0 and C_1 take inputs of the same length and $z \in \{0,1\}^{poly(\lambda)}$.

Definition 3 (X-ind sampler, [11]**).** *Let $X(\lambda)$ be a function upper bounded by 2^λ. The class $\mathcal{S}^{X\text{-ind}}$ of X-*ind samplers *for a circuit family $\mathcal{C}$ contains all circuit samplers $S = (S_\lambda)_{\lambda \in \mathbb{N}}$ for $\mathcal{C}$ such that for all $\lambda \in \mathbb{N}$, there exists a set $\mathcal{X}_\lambda \subseteq \{0,1\}^*$ with $|\mathcal{X}| \leq X(\lambda)$, such that*

X-differing inputs. *With overwhelming probability over the choice of $(C_0, C_1, z) \leftarrow S_\lambda$, for every $x \notin \mathcal{X}_\lambda$, for all $r \in \{0,1\}^{m(\lambda)}$, $C_0(x; r) = C_1(x; r)$.*

X-indistinguishability. *For all (non-uniform) adversaries $\mathcal{A}$, the advantage*

$$X(\lambda) \cdot \left(\Pr[\mathrm{Exp}^{\texttt{sel-ind}}_{S,\mathcal{A}}(\lambda) = 1] - \frac{1}{2}\right)$$

is negligible, where $\mathrm{Exp}_{\mathcal{S},\mathcal{A}}^{\texttt{sel-ind}}(\lambda)$ *requires* $\mathcal{A}$ *to statically choose an input, samples circuits* C_0, C_1 *(and auxiliary information* z*) afterwards, evaluates the circuit* C_b *(for randomly chosen* b*) on the adversarially chosen input (let the output be* y*) and outputs* 1 *if* $\mathcal{A}$ *on input of* (C_0, C_1, z, y) *guesses* b *correctly.*

Definition 4 (Probabilistic indistinguishability obfuscation for a class of samplers $\mathcal{S}$ (syntax), [11]). *A* probabilistic indistinguishability obfuscator *(pIO) for a class of samplers* $\mathcal{S}$ *is a uniform PPT algorithm* piO*, such that correctness and security with respect to* $\mathcal{S}$ *hold.*

For a more detailed definition, we refer the reader to the full version [2].

[11] present the to date only known construction of pIO for X-ind samplers over the family of all polynomial sized probabilistic circuits.

2.4 Re-randomizable and Fully Homomorphic Encryption

We define an IND-CPA secure PKE scheme as a tuple of PPT algorithms $\mathsf{PKE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ in the usual sense. Furthermore, without loss of generality, we assume that sk is the random tape used for key generation. Therefore, making the random tape of KGen explicit, we write $(pk, sk) = \mathsf{KGen}(1^\lambda; sk)$.

A re-randomizable PKE scheme additionally provides an algorithm Rerand which re-randomizes a given ciphertext perfectly.

Finally, a fully homomorphic PKE scheme additionally provides an algorithm Eval which given the public key pk, an circuit C (expecting a inputs from the message space) and a ciphertexts $C_1, \ldots, C_a$, produces a ciphertext encrypting $C(\mathsf{Dec}(sk, C_1), \ldots, \mathsf{Dec}(sk, C_a))$.

Due to [11], probabilistic indistinguishability obfuscation in conjunction with (slightly super-polynomially secure) perfectly correct and perfectly re-randomizable public-key encryption yields a perfectly correct and perfectly re-randomizable fully homomorphic encryption scheme.

We refer the reader to the full version [2] for more detailed definitions.

2.5 Statistically Correct Input Expanding pIO

Looking ahead, instead of computationally correct pIO, we require a notion of statistically correct pIO, i.e. statistical closeness between evaluations of the original (probabilistic) circuit and the obfuscated (deterministic) circuit. Clearly, in general, this is impossible since the obfuscated circuit is deterministic and hence has no source of entropy other than its input. However, as long as a portion of the circuit's input is guaranteed to be outside the view of the adversary (and has sufficiently high min-entropy), the output of the obfuscated circuit and the actual probabilistic circuit can be statistically close. Therefore, we compile probabilistic circuits such that they receive an auxiliary input aux but simply ignore this input in their computation. Even though the obfuscated circuit is deterministic, the auxiliary input can be used as a source of actual entropy.

First try. We recall that the pIO construction from [11] obfuscates a probabilistic circuit C by using IO to obfuscate the deterministic circuit $\overline{C}(x) := C(x; F_K(x))$. A natural idea to achieve statistical correctness is to modify this construction such that the auxiliary input aux is directly XORed on the random tape which is derived using F, i.e. to obfuscate the circuit $\overline{C}(x, aux; F_K(x) \oplus aux)$. For uniform auxiliary input aux, statistical correctness follows immediately. However, security breaks down. Consider two circuits C_1 and C_2 such that C_1 outputs the first bit on its random tape and C_2 outputs the second bit on its random tape. Since C_1 and C_2 produce identical output distributions, it is desirable that a probabilistic indistinguishability obfuscator conceals which of the two circuits was obfuscated. However, this construction admits a successful attack. An adversary can evaluate the obfuscated circuit Λ on inputs (x, aux) and $(x, aux \oplus 1)$. If both evaluations yield identical outputs, C_2 was obfuscated, otherwise C_1 was obfuscated.

Using an extracting PRF. Our construction of statistically correct pIO applies an extracting puncturable PRF on the entire input (including the auxiliary input) of the circuit to derive the random tape for the probabilistic circuit. An extracting PRF guarantees that PRF outputs are uniformly distributed (even given the PRF key) as long as the input has high min-entropy. This is achieved using a universal hash function and the leftover hash lemma. For more details, we refer the reader to the full version [2].

Let $\{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of sets $\mathcal{C}_\lambda$ of probabilistic circuits of polynomial size $p(\lambda)$ expecting inputs from $\{0,1\}^{n'(\lambda)}$ and randomness from $\{0,1\}^{r(\lambda)}$. Let $\mathcal{E}_\ell$ denote a compiler which on input of a probabilistic circuit $C \in \mathcal{C}_\lambda$ appends $\ell(\lambda)$ input gates (without any additional edges) to the original circuit. The expanded circuit $\widehat{C}$ is of size $p'(\lambda) = p(\lambda) + \ell(\lambda)$, expects inputs from $\{0,1\}^{n'(\lambda)+\ell(\lambda)}$ and randomness from $\{0,1\}^{r(\lambda)}$. We refer to these additional input bits as auxiliary input $aux \in \{0,1\}^{\ell(\lambda)}$.

Our input expanding pIO scheme satisfies similar correctness and security properties as defined in [11] but additionally guarantees statistical correctness.

Definition 5 (ℓ-expanding pIO for the class of samplers $\mathcal{S}$). *An ℓ-expanding* probabilistic indistinguishability obfuscator *for the class of samplers $\mathcal{S}$ over $\mathcal{C} = (\mathcal{C}_\lambda)_{\lambda \in \mathbb{N}}$ is a uniform PPT algorithm $\mathsf{piO}^\star_\ell$, satisfying the following properties.*

Input expanding correctness. *For all PPT adversaries $\mathcal{A}$, all circuits $C \in \mathcal{C}$,*

$$\left| \Pr[\mathcal{A}^{\mathcal{O}_C(\cdot,\cdot)}(1^\lambda, C) = 1] - \Pr[\Lambda \leftarrow \mathsf{piO}^\star_\ell(1^{p(\lambda)}, C) \colon \mathcal{A}^{\mathcal{O}_\Lambda(\cdot,\cdot)}(1^\lambda, C) = 1] \right|$$

is negligible, where the oracles must not be called twice on the same input (x, aux).

```
$\underline{\mathcal{O}_C(x, aux)}$
$r \leftarrow \{0,1\}^m$
return $C(x; r)$

$\underline{\mathcal{O}_\Lambda(x, aux)}$
return $\Lambda(x, aux)$
```

Security with respect to $\mathcal{S}$. *For all circuit samplers $S \in \mathcal{S}$, for all PPT adversaries $\mathcal{A}$, the advantage*

$$\begin{aligned}&\mathrm{Adv}^{\texttt{pio-ind}(\star)}_{\mathsf{piO}^\star_\ell,S,\mathcal{A}}(\lambda) := \\ &\Big| \Pr\Big[(C_0,C_1,z) \leftarrow S(1^\lambda)\colon \mathcal{A}(1^\lambda, C_0, C_1, z, \mathsf{piO}^\star_\ell(1^{p(\lambda)}, C_0)) = 1\Big] \\ &- \Pr\Big[(C_0,C_1,z) \leftarrow S(1^\lambda)\colon \mathcal{A}(1^\lambda, C_0, C_1, z, \mathsf{piO}^\star_\ell(1^{p(\lambda)}, C_1)) = 1\Big] \Big|\end{aligned}$$

is negligible in λ.

Support respecting. *For all circuits $C \in \mathcal{C}_\lambda$, all inputs $x \in \{0,1\}^{n'(\lambda)}$, all $aux \in \{0,1\}^{\ell(\lambda)}$, all $\Lambda \in \mathsf{supp}(\mathsf{piO}^\star_\ell(1^{p(\lambda)}, C))$, $\Lambda(x, aux) \in \mathsf{supp}(C(x))$.*

Statistical correctness with error $2^{-e(\lambda)}$. *For all $C \in \mathcal{C}_\lambda$ and all joint distributions (X_1, X_2) over $\{0,1\}^{n'(\lambda)} \times \{0,1\}^{\ell(\lambda)}$ with average min-entropy $\ell(\lambda) \geq \widetilde{\mathrm{H}}_\infty(X_2 \mid X_1) > m(\lambda) + 2e(\lambda) + 2$, the statistical distance between*

$$\Big\{\Lambda \leftarrow \mathsf{piO}^\star_\ell(1^{p(\lambda)}, C)\colon (\Lambda, \Lambda(X_1, X_2))\Big\}$$
$$\text{and } \Big\{\Lambda \leftarrow \mathsf{piO}^\star_\ell(1^{p(\lambda)}, C)\colon (\Lambda, C(X_1; U_{m(\lambda)}))\Big\}$$

is at most $2^{-e(\lambda)}$.

We note that setting $\ell := 0$ recovers the original definition of pIO for X-ind samplers due to [11]. Looking ahead, our application does not require input expanding correctness.

Let S be a circuit sampler and let $\widehat{S}$ denote the circuit sampler which calls S and outputs ℓ-expanded circuits. Unfortunately, if S is an X-ind sampler does not imply that $\widehat{S}$ also satisfies the requirements to be an X-ind sampler. On a high level this is because $\widehat{X}(\lambda) := X(\lambda) \cdot 2^{\ell(\lambda)}$ is necessary for $\widehat{S}$ to satisfy the X-differing inputs property. Then, however, X-indistinguishability of S does not suffice to prove $\widehat{X}$-indistinguishability of $\widehat{S}$. Thus, we introduce the notion of ℓ-expanding X-ind samplers.

Definition 6 (ℓ-expanding X-ind sampler). *Let S be a circuit sampler. With $\widehat{S}$ we denote the circuit sampler which on input of $1^{p(\lambda)+\ell(\lambda)}$ samples $(C_0, C_1, z) \leftarrow S(1^{p(\lambda)})$ and outputs the circuits $\widehat{C}_0 := \mathcal{E}_\ell(C_0), \widehat{C}_1 := \mathcal{E}_\ell(C_1)$ and auxiliary information $\widehat{z} := (C_0, C_1, z)$. The class $\mathcal{S}_\ell^{X\text{-}(\star)\text{-ind}}$ of* ℓ-expanding X-ind samplers *for a circuit family $\mathcal{C}$ contains all circuit samplers $S = (S_\lambda)_{\lambda\in\mathbb{N}}$ for $\mathcal{C}$ such that the circuit sampler $\widehat{S}$ is an X-ind sampler according to Definition 3, i.e. $\widehat{S} \in \mathcal{S}^{X\text{-ind}}$.*

On a high level, we instantiate the construction of pIO for X-ind samplers due to [11] with a suitably extracting puncturable pseudorandom function (pPRF). By suitably extracting we mean that the PRF output is guaranteed to be statistically close to uniform randomness as long as the average min-entropy of the input of the PRF is sufficiently high. Such a pPRF can be constructed by composing a pPRF with a universal hash function.

Theorem 1. *Let e be an efficiently computable function. Let F be a sub-exponentially secure special extracting PRF family with distinguishing advantage $2^{-\lambda^{\epsilon}}$ (for some constant ϵ) and error $2^{-e(\lambda)}$ mapping $n(\lambda) = n'(\lambda) + \ell(\lambda)$ bits to $m(\lambda)$ bits which is extracting if the input average min-entropy is greater than $m(\lambda) + 2e(\lambda) + 2$. Then, there exists a statistically correct input expanding pIO $\mathsf{piO}_{\ell}^{\star}$ for the class of samplers $\mathcal{S}_{\ell}^{X\text{-}(\star)\text{-ind}}$.*

For additional explanations and a formal proof, we refer the reader to the full version [2].

3 How to Simulate Extraction – Algebraic Wrappers

In order to instantiate the AGM, we need to first find a way to conceptualize what it means to be a group in a cryptographic sense. This is captured by the notion of a *group scheme* or *encoding scheme*, [23]. In a nutshell, a group scheme provides an interface of algorithms abstracting the handling of a cryptographic group. As we want to prove hardness of certain problems based on hardness assumptions in an already existing base group, we incorporate this existing group into our group scheme.

More specifically, we introduce the concept of an *algebraic wrapper*, i.e. a group scheme that allows to extract a representation which – similar to the AGM – can be used in a security reduction. A similar approach has already been taken by [30]. [30] define their group scheme as a linear subspace of $\mathbb{G} \times \mathbb{G}$ for an existing group $\mathbb{G}$ in such a way that the Generalized Knowledge of Exponent Assumption (GKEA) can be used to extract a representation (membership can for instance be tested via a symmetric pairing). Hence, that group scheme can also be viewed as an extension, or a *wrapper*, for the underlying base group. However, [30] relies on GKEA in the base group which more or less directly yields an equivalence between algebraic groups and GKEA. The existence of algebraic groups, however, implies the existence of extractable one-way functions with unbounded auxiliary input (since the AGM allows an additional unstructured input from $\{0,1\}^*$) which in turn conflicts with the existence of indistinguishability obfuscation, [5]. Due to this contradiction and the difficulty to assess the plausibility of knowledge-type assumptions, we strive for a weaker model which can purely be based on falsifiable assumptions.

Extraction trapdoors. In [30], extraction is possible as long as the code and the randomness which where used to produce a group element are known. Since we strive to avoid knowledge-type assumptions, we need to find a different mechanism of what enables extraction. We observe that in order to reproduce proof strategies from the algebraic group model, extraction is only necessary during security reductions. Since the reduction to some assumption in the base group is in control of the group parameters of the wrapper, the reduction may use corresponding trapdoor information which we define to enable extraction. We call this notion *private extractability.*

3.1 Group Schemes

A group scheme or encoding scheme [23] abstracts the properties of mathematical groups used in cryptography. Group schemes have recently been studied in [1,3,17,30]. In contrast to traditional groups, group elements are not bound to be represented by a unique bitstring (henceforth referred to as encoding). This allows to encode auxiliary information inside group elements.

Formally, a group scheme $\mathbb{H}$ consists of the algorithms $(\mathsf{GGen}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}, \mathsf{Val}_{\mathbb{H}}, \mathsf{Add}_{\mathbb{H}}, \mathsf{Eq}_{\mathbb{H}}, \mathsf{GetID}_{\mathbb{H}})$. A group generation algorithm $\mathsf{GGen}_{\mathbb{H}}$, which given 1^{λ}, samples group parameters $\mathsf{pp}_{\mathbb{H}}$. A sampling algorithm $\mathsf{Sam}_{\mathbb{H}}$, given the group parameters and an additional parameter determining the exponent of the desired group element, produces an encoding corresponding to that exponent. A validation algorithm $\mathsf{Val}_{\mathbb{H}}$, given the group parameters and a bitstring, decides whether the given bitstring is a valid encoding. The algorithm $\mathsf{Add}_{\mathbb{H}}$ implements the group operation, i.e. expects the group parameters and two encodings as input and produces an encoding of the resulting group element. Since group elements do not necessarily possess unique encodings, the equality testing algorithm $\mathsf{Eq}_{\mathbb{H}}$ enables to test whether two given encodings correspond to the same group element (with respect to the given group parameters). Note that $\mathsf{Eq}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \cdot)$ defines an equivalence relation on the set of valid bitstrings. Finally, again compensating for the non-unique encodings, a group scheme describes a "get-identifier" algorithm which given the group parameters and an encoding of a group element, produces a bitstring which is unique for all encodings of the same group element.[9] Note that $\mathsf{Eq}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, a, b)$ can be implemented using $\mathsf{GetID}_{\mathbb{H}}$ by simply comparing $\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, a)$ and $\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, b)$ as bitstrings. The "get-identifier" algorithm compensates for the potential non-uniqueness of encodings and allows to extract, for instance, symmetric keys from group elements.

For a group scheme it is required that the quotient set

$$\{a \in \{0,1\}^* \mid \mathsf{Val}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, a) = 1\}/\mathsf{Eq}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \cdot)$$

equipped with the operation defined via $\mathsf{Add}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \cdot, \cdot)$ defines a mathematical group (with overwhelming probability over the choice of $\mathsf{pp}_{\mathbb{H}} \leftarrow \mathsf{GGen}_{\mathbb{H}}(1^{\lambda})$). We say that an a is (an encoding of) a group element (relative to $\mathsf{pp}_{\mathbb{H}}$), written as $a \in \mathbb{H}$, if and only if $\mathsf{Val}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, a) = 1$.

A group scheme requires that encodings corresponding to the same group element are computationally indistinguishable as formalized by the "Switching Lemma(s)" in [1,3,17].

Due to the non-uniqueness of encodings, we henceforth use the notation $\widehat{h}$ to denote an encoding of a group element.

3.2 An Algebraic Wrapper

Given a cyclic group, an algebraic wrapper is a group scheme which equips a given group $\mathbb{G}$ with a notion of extractability while preserving its group

[9] Previous work refers to this algorithm as "extraction algorithm". However, in order not to overload the word "extraction", we rename this algorithm in this work.

structure and complexity theoretic hardness guarantees. In particular, we achieve a property which we refer to as "private extractability" with respect to a given set of group elements in the base group. More precisely, the group generation algorithm expects group parameters $\mathsf{pp}_{\mathbb{G}}$ of the base group together with a set of group elements $[\mathbf{b}]_{\mathbb{G}} \in \mathbb{G}^n$ in that base group, henceforth referred to as *basis*, and produces group parameters $\mathsf{pp}_{\mathbb{H}}$ of the wrapper group together with a corresponding trapdoor $\tau_{\mathbb{H}}$. This trapdoor enables to extract a representation with respect to the basis $[\mathbf{b}]_{\mathbb{G}}$ from every encoding. Looking ahead, this property will allow to implement proof strategies of the algebraic group model, [21].

More precisely, encodings can be seen to always carry computationally hidden representation vectors with respect to the basis $[\mathbf{b}]_{\mathbb{G}}$. The private extraction recovers this representation vector. Given the trapdoor, we require that it is possible to "privately" sample encodings which carry a specific dictated representation vector. We require that publicly sampled encodings and privately sampled encodings are computationally indistinguishable. We refer to this property as "switching". In order to preserve tightness of security reductions when implementing AGM proofs with our algebraic wrapper, we require a statistical re-randomization property. Furthermore, we require that representation vectors compose additively (in $\mathbb{Z}_p^n$) with the group operation and do not change when encodings are re-randomized.

Let $\mathcal{B}^n_{\mathsf{pp}_{\mathbb{G}}} := \{([1]_{\mathbb{G}}, [x_2]_{\mathbb{G}}, \ldots, [x_n]_{\mathbb{G}})^\intercal \in \mathbb{G}^n \mid x_2, \ldots, x_n \in \mathbb{Z}_p^\times\}$ be the set of what we call "legitimate basis vectors". Note that we require the first group element to be the generator of the group. This is necessary to allow public sampling.

Definition 7 (Algebraic wrapper for $\mathbb{G}$). *An* algebraic wrapper $\mathbb{H}$ for $\mathbb{G}$ *is a tuple of PPT algorithms* $(\mathsf{GGen}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}, \mathsf{Val}_{\mathbb{H}}, \mathsf{Add}_{\mathbb{H}}, \mathsf{Eq}_{\mathbb{H}}, \mathsf{GetID}_{\mathbb{H}}, \mathsf{Rerand}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}, \mathsf{PrivExt}_{\mathbb{H}}, \mathsf{Unwrap}_{\mathbb{H}})$ *such that* $(\mathsf{GGen}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}, \mathsf{Val}_{\mathbb{H}}, \mathsf{Add}_{\mathbb{H}}, \mathsf{Eq}_{\mathbb{H}}, \mathsf{GetID}_{\mathbb{H}})$ *constitutes a group scheme and the following properties are satisfied.*

$\mathbb{G}$-wrapping. *The algorithm* $\mathsf{Unwrap}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \cdot)$ *is deterministic and for all* $\mathsf{pp}_{\mathbb{G}} \in \mathsf{supp}(\mathsf{GGen}_{\mathbb{G}}(1^\lambda))$*, all* $[\mathbf{b}]_{\mathbb{G}} \in \mathcal{B}^n_{\mathsf{pp}_{\mathbb{G}}}$*, all* $(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \in \mathsf{supp}(\mathsf{GGen}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, [\mathbf{b}]_{\mathbb{G}}))$, $\mathsf{Unwrap}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \cdot)$ *defines a* group isomorphism *from* $\mathbb{H}$ *to* $\mathbb{G}$.

Extractability. *The algorithm* $\mathsf{PrivExt}_{\mathbb{H}}$ *is deterministic. Furthermore, for all* $\mathsf{pp}_{\mathbb{G}} \in \mathsf{supp}(\mathsf{GGen}_{\mathbb{G}}(1^\lambda))$*, all* $[\mathbf{b}]_{\mathbb{G}} \in \mathcal{B}^n_{\mathsf{pp}_{\mathbb{G}}}$*, all* $(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \in \mathsf{supp}(\mathsf{GGen}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, [\mathbf{b}]_{\mathbb{G}}))$*, all* $\widehat{h} \in \mathbb{H}$*, we require that* $\mathsf{PrivExt}_{\mathbb{H}}$ *always extracts a representation of* $[x]_{\mathbb{G}}$ *with respect to* $[\mathbf{b}]_{\mathbb{G}}$*, i.e. for* $\mathbf{z} := \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h})$, $[\mathbf{z}^\intercal \cdot \mathbf{b}]_{\mathbb{G}} = \mathsf{Unwrap}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h})$.

Correctness of extraction. *For all* $\mathsf{pp}_{\mathbb{G}} \in \mathsf{supp}(\mathsf{GGen}_{\mathbb{G}}(1^\lambda))$*, all* $[\mathbf{b}]_{\mathbb{G}} \in \mathcal{B}^n_{\mathsf{pp}_{\mathbb{G}}}$*, all* $(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \in \mathsf{supp}(\mathsf{GGen}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, [\mathbf{b}]_{\mathbb{G}}))$*, all* $\widehat{h_0}, \widehat{h_1} \in \mathbb{H}$*, we require that private extraction respects the group operation in the sense that for all* $\widehat{h_2} \in \mathsf{supp}(\mathsf{Add}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h_0}, \widehat{h_1}))$, $\mathbf{z}^{(i)} := \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h}_i)$ *satisfy* $\mathbf{z}^{(2)} = \mathbf{z}^{(0)} + \mathbf{z}^{(1)}$*. Furthermore, for all* $\mathsf{pp}_{\mathbb{G}} \in \mathsf{supp}(\mathsf{GGen}_{\mathbb{G}}(1^\lambda))$*, all* $[\mathbf{b}]_{\mathbb{G}} \in \mathcal{B}^n_{\mathsf{pp}_{\mathbb{G}}}$*, all* $(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \in \mathsf{supp}(\mathsf{GGen}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, [\mathbf{b}]_{\mathbb{G}}))$*, all* $\widehat{h} \in \mathbb{H}$*, we require that re-randomization does not interfere with private extraction in the sense that for all* $\widehat{h'} \in \mathsf{supp}(\mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h}))$, $\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h}) = \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h'})$.

Correctness of sampling. *For all* $\mathsf{pp}_{\mathbb{G}} \in \mathsf{supp}(\mathsf{GGen}_{\mathbb{G}}(1^\lambda))$*, all* $[\mathbf{b}]_{\mathbb{G}} \in \mathcal{B}^n_{\mathsf{pp}_{\mathbb{G}}}$*, all* $(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \in \mathsf{supp}(\mathsf{GGen}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, [\mathbf{b}]_{\mathbb{G}}))$*, we require that*
 – *for all* $\mathbf{v} \in \mathbb{Z}_p^n$*,* $\Pr[\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, \mathbf{v})) = \mathbf{v}] = 1$*, and*
 – *for all* $x \in \mathbb{Z}_p$*,* $\Pr[\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, x \cdot \mathbf{e_1})) = x \cdot \mathbf{e_1}] = 1$*.*

k**-Switching.** *We say a PPT adversary* $\mathcal{A}$ *is a legitimate* k*-switching adversary if on input of base group parameters* $\mathsf{pp}_{\mathbb{G}}$*,* $\mathcal{A}$ *outputs two bases* $([\mathbf{b}]^{(j)}_{\mathbb{G}})_{j\in\{0,1\}}$ *and two lists comprising* k *representation vectors* $(\mathbf{v}^{(j),(i)})_{i\in[k],j\in\{0,1\}}$ *(and an internal state st) such that* $[\mathbf{b}]^{(0)}_{\mathbb{G}}, [\mathbf{b}]^{(1)}_{\mathbb{G}} \in \mathcal{B}^n_{\mathsf{pp}_{\mathbb{G}}}$ *and* $\mathbf{v}^{(0),(i)}, \mathbf{v}^{(1),(i)} \in \mathbb{Z}_p^n$ *for some* $n \in \mathbb{N}$ *and all* $i \in [k]$ *and* $\left[(\mathbf{v}^{(0),(i)})^\intercal \cdot \mathbf{b}^{(0)}\right]_{\mathbb{G}} = \left[(\mathbf{v}^{(1),(i)})^\intercal \cdot \mathbf{b}^{(1)}\right]_{\mathbb{G}}$ *for all* $i \in [k]$*.*
For all legitimate k*-switching PPT adversaries* $\mathcal{A}$*,*

$$\mathrm{Adv}^{k\text{-}\mathtt{switching}}_{\mathbb{H},\mathcal{A}}(\lambda) := \left|\Pr[\mathrm{Exp}^{k\text{-}\mathtt{switching}}_{\mathbb{H},\mathcal{A},0}(\lambda) = 1] - \Pr[\mathrm{Exp}^{k\text{-}\mathtt{switching}}_{\mathbb{H},\mathcal{A},1}(\lambda) = 1]\right|$$

is negligible, where $\mathrm{Exp}^{k\text{-}\mathtt{switching}}_{\mathbb{H},\mathcal{A},b}(\lambda)$ *(for* $b \in \{0,1\}$*) is defined in Fig. 1.*

Statistically re-randomizable. *We say an unbounded adversary* $\mathcal{A}$ *is a legitimate re-randomization adversary if on input of base group parameters* $\mathsf{pp}_{\mathbb{G}}$*,* $\mathcal{A}$ *outputs* $[\mathbf{b}]_{\mathbb{G}}$ *and a state st such that* $[\mathbf{b}]_{\mathbb{G}} \in \mathcal{B}^n_{\mathsf{pp}_{\mathbb{G}}}$ *and, in a second phase,* $\mathcal{A}$ *on input of* $(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}, st)$ *outputs two valid encodings* $\widehat{h_0}, \widehat{h_1}$ *(and a state st) such that* $\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h_0}) = \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h_1})$*.*
For all unbounded legitimate re-randomization adversaries $\mathcal{A}$*,*

$$\mathrm{Adv}^{\mathtt{rerand}}_{\mathbb{H},\mathcal{A}}(\lambda) := \left|\Pr[\mathrm{Exp}^{\mathtt{rerand}}_{\mathbb{H},\mathcal{A},0}(\lambda) = 1] - \Pr[\mathrm{Exp}^{\mathtt{rerand}}_{\mathbb{H},\mathcal{A},1}(\lambda) = 1]\right| \leq \frac{1}{2^\lambda},$$

where $\mathrm{Exp}^{\mathtt{rerand}}_{\mathbb{H},\mathcal{A},b}(\lambda)$ *(for* $b \in \{0,1\}$*) is defined in Fig. 1.*

```
$\mathrm{Exp}^{\mathtt{rerand}}_{\mathbb{H},\mathcal{A},b}(\lambda)$
$\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{GGen}_{\mathbb{G}}(1^\lambda)$
$([\mathbf{b}]_{\mathbb{G}}, st) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp}_{\mathbb{G}})$
$(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{GGen}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, [\mathbf{b}]_{\mathbb{G}})$
$(\widehat{h_0}, \widehat{h_1}, st) \leftarrow \mathcal{A}(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}, st)$
$\widehat{h} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h_b})$
return $\mathcal{A}(\widehat{h}, st)$
```

```
$\mathrm{Exp}^{k\text{-}\mathtt{switching}}_{\mathbb{H},\mathcal{A},b}(\lambda)$
$\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{GGen}_{\mathbb{G}}(1^\lambda)$
$\big(([\mathbf{b}]^{(j)}_{\mathbb{G}})_{j\in\{0,1\}},$
  $(\mathbf{v}^{(j),(i)})_{i\in[k],j\in\{0,1\}}, st\big) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp}_{\mathbb{G}})$
$(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{GGen}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, [\mathbf{b}]^{(b)}_{\mathbb{G}})$
$\widehat{h^*_i} \leftarrow \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, \mathbf{v}^{(b),(i)})$
return $\mathcal{A}(\mathsf{pp}_{\mathbb{H}}, (\widehat{h^*_i})_{i\in[k]}, st)$
```

Fig. 1. The re-randomization and k-switching games.

For simplicity we require that encodings are always in $\{0,1\}^{p_{\mathsf{enc}}(\lambda)}$ for a fixed polynomial $p_{\mathsf{enc}}(\lambda)$.

The k-switching property allows to simultaneously switch the representation vectors of multiple group element encodings. It is necessary to switch all encodings simultaneously since private sampling can only be simulated knowing the trapdoor $\tau_{\mathbb{H}}$ which is not the case in $\mathrm{Exp}^{k\text{-}\mathtt{switching}}_{\mathbb{H},\mathcal{A},b}(\lambda)$.

3.3 Construction

Our construction follows the ideas from [1,3,17]. Let $\mathsf{GGen}_{\mathbb{G}}$ be a group generator for a cyclic group $\mathbb{G}$. Let $\mathcal{TD}$ be a family of hard subset membership problems. Let $\mathsf{FHE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}, \mathsf{Rerand})$ be a perfectly correct and perfectly re-randomizable fully homomorphic public-key encryption scheme. Let $\mathsf{pp}_{\mathbb{G}}$ be group parameters for $\mathbb{G}$ and $[\mathbf{\Omega}]_{\mathbb{G}} \in \mathbb{G}^n$ for some $n \in \mathbb{N}$. Let $\mathtt{TD} \subseteq X$ be a subset membership problem from $\mathcal{TD}$ and $y \leftarrow X \setminus \mathtt{TD}$ and pk be a public key for FHE. For ease of notation, we define $\mathsf{pars} := (\mathsf{pp}_{\mathbb{G}}, \mathtt{TD}, y, pk, [\mathbf{\Omega}]_{\mathbb{G}})$. Let $\Pi := (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{HSetup}, \mathsf{Ext})$ be a perfectly complete, perfectly sound and perfectly witness-indistinguishable dual-mode NIZK proof system for the language

$$\mathcal{L} := \left\{ y := (\mathsf{pars}, [x]_{\mathbb{G}}, C) \;\middle|\; \exists w\colon (y, w) \in \mathcal{R} := \mathcal{R}_1 \vee \mathcal{R}_2 \vee \mathcal{R}_3 \right\}.$$

The relations $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3$ are defined as follows.

$$\mathcal{R}_1 = \left\{ \Big((\mathsf{pars}, [x]_{\mathbb{G}}, C), (sk, \mathbf{v}) \Big) \;\middle|\; \begin{array}{rc} & \mathsf{KGen}(1^\lambda; sk) = (pk, sk) \\ \wedge & \mathsf{Dec}(sk, C) = \mathbf{v} \\ \wedge & [\mathbf{\Omega}^\intercal \cdot \mathbf{v}]_{\mathbb{G}} = [x]_{\mathbb{G}} \end{array} \right\}$$

$$\mathcal{R}_2 = \left\{ \Big((\mathsf{pars}, [x]_{\mathbb{G}}, C), (r, \mathbf{v}) \Big) \;\middle|\; \begin{array}{rc} & \mathsf{Enc}(pk, \mathbf{v}; r) = C \\ \wedge & [\mathbf{\Omega}^\intercal \cdot \mathbf{v}]_{\mathbb{G}} = [x]_{\mathbb{G}} \end{array} \right\}$$

$$\mathcal{R}_3 = \left\{ \Big((\mathsf{pars}, [x]_{\mathbb{G}}, C), (w_y) \Big) \;\middle|\; (y, w_y) \in R_{\mathtt{TD}} \right\}$$

With $m'(\lambda)$ we denote a polynomial upper bound on the number of random bits $\mathsf{FHE.Rerand}(1^\lambda, \cdot, \cdot)$ expects and with $m''(\lambda)$ we denote a polynomial upper bound on the number of random bits $\Pi.\mathsf{Prove}(1^\lambda, \cdot, \cdot, \cdot)$ expects. Let $\ell(\lambda) := m'(\lambda) + m''(\lambda) + 2(\lambda + 1) + 3$. Let piO be a pIO scheme for the class of samplers $\mathcal{S}^{X\text{-ind}}$ and let $\mathsf{piO}^\star_\ell$ be an ℓ-expanding pIO scheme for the class of samplers $\mathcal{S}_\ell^{X\text{-}(\star)\text{-ind}}$. Further, let $p_{\mathsf{add}}(\lambda)$ denote a polynomial upper bound on the size of addition circuits and $p_{\mathsf{rerand}}(\lambda)$ denote a polynomial upper bound on the size of re-randomization circuits which are used during the proof, see the full version [2] for details.

Our algebraic wrapper $\mathbb{H}$ is composed of the PPT algorithms $(\mathsf{GGen}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}, \mathsf{Val}_{\mathbb{H}}, \mathsf{Add}_{\mathbb{H}}, \mathsf{Eq}_{\mathbb{H}}, \mathsf{Rerand}_{\mathbb{H}}, \mathsf{PrivExt}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}, \mathsf{GetID}_{\mathbb{H}}, \mathsf{Unwrap}_{\mathbb{H}})$ which are defined in Figs. 2a and 2b. We note that the algorithm $\mathsf{Val}_{\mathbb{H}}$ which is evaluated inside C_{Add} and C_{rerand} only requires a certain part of the public parameters as input. In particular, $\mathsf{Val}_{\mathbb{H}}$ does not depend on Λ_{Add} and $\Lambda_{\mathsf{rerand}}$.

During "honest" use of our algebraic wrapper, encodings carry proofs produced for relation $\mathcal{R}_1$ or relation $\mathcal{R}_2$. Relation $\mathcal{R}_2$ enables sampling without knowledge of any trapdoors. Re-randomized encodings always carry proofs for relation $\mathcal{R}_1$. Relation $\mathcal{R}_3$ is a trapdoor branch enabling simulation. Note that during "honest" use of the algebraic wrapper $y \notin \mathtt{TD}$ and, hence, due to perfect soundness of Π, there exists no proof for relation $\mathcal{R}_3$.

```
$\mathsf{GGen}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, [\mathbf{b}]_{\mathbb{G}} = [(b_1, \ldots, b_n)^\intercal]_{\mathbb{G}})$
$\alpha_1 := 1, \alpha_2, \ldots, \alpha_n \leftarrow \mathbb{Z}_p^\times$
$[\mathbf{\Omega}]_{\mathbb{G}} := ([b_1]_{\mathbb{G}}^{\alpha_1}, \ldots, [b_n]_{\mathbb{G}}^{\alpha_n})^\intercal$
$(pk, sk) \leftarrow \mathsf{FHE.KGen}(1^\lambda)$
$\mathsf{crs} \leftarrow \Pi.\mathsf{Setup}(1^\lambda), \mathtt{TD} \leftarrow \mathcal{TD}, y \leftarrow \overline{\mathtt{TD}}$
$\Lambda_{\mathsf{Add}} \leftarrow \mathsf{piO}(1^{p_{\mathsf{add}}(\lambda)}, C_{\mathsf{Add}})$
$\Lambda_{\mathsf{rerand}} \leftarrow \mathsf{piO}_\ell^\star(1^{p_{\mathsf{rerand}}(\lambda)}, C_{\mathsf{rerand}})$
$\mathsf{pars} := (\mathsf{pp}_{\mathbb{G}}, \mathtt{TD}, y, pk, [\mathbf{\Omega}]_{\mathbb{G}})$
$\mathsf{pp}_{\mathbb{H}} := (\mathsf{crs}, \mathsf{pars}, \Lambda_{\mathsf{Add}}, \Lambda_{\mathsf{rerand}})$
$\tau_{\mathbb{H}} := (\mathsf{pp}_{\mathbb{H}}, sk, \alpha_1, \ldots, \alpha_n, [\mathbf{b}]_{\mathbb{G}})$
return $(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}})$

$\mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathbf{v} \in \mathbb{Z}_p^n)$
$C = \mathsf{Enc}(pk, \mathbf{v}; r)$
$[x]_{\mathbb{G}} := [\mathbf{\Omega}^\intercal \cdot \mathbf{v}]_{\mathbb{G}}$
$\pi = \mathsf{Prove}(\mathsf{crs}, (\mathsf{pars}, [x]_{\mathbb{G}}, C), (r, \mathbf{v}))$
return $\widehat{h} := ([x]_{\mathbb{G}}, C, \pi)_{\mathbb{H}}$

$\mathsf{Val}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h})$
parse $\widehat{x} =: ([x]_{\mathbb{G}}, C, \pi)_{\mathbb{H}}$
return $\Pi.\mathsf{Verify}(\mathsf{crs}, (\mathsf{pars}, [x]_{\mathbb{G}}, C), \pi)$

$\mathsf{Unwrap}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h})$
if $\neg \mathsf{Val}_{\mathbb{H}}((\mathsf{crs}, \mathsf{pars}), \widehat{h})$ then
  return $\perp$
parse $\widehat{h} =: ([x]_{\mathbb{G}}, C, \pi)_{\mathbb{H}}$
return $[x]_{\mathbb{G}}$
```

```
$\mathsf{Eq}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h_1}, \widehat{h_2})$
if $\exists j \in [2] \colon \neg \mathsf{Val}_{\mathbb{H}}((\mathsf{crs}, \mathsf{pars}), \widehat{h_j})$ then
  return $\perp$
parse $\widehat{h_i} =: ([x_i]_{\mathbb{G}}, C_i, \pi_i)_{\mathbb{H}}$
return $[x_1]_{\mathbb{G}} = [x_2]_{\mathbb{G}}$

$\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h})$
if $\neg \mathsf{Val}_{\mathbb{H}}((\mathsf{crs}, \mathsf{pars}), \widehat{h})$ then
  return $\perp$
parse $\widehat{h} =: ([x]_{\mathbb{G}}, C, \pi)_{\mathbb{H}}$
return $[x]_{\mathbb{G}}$

$\mathsf{Add}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h_1}, \widehat{h_2})$
return $\Lambda_{\mathsf{Add}}(\widehat{h_1}, \widehat{h_2})$

$C_{\mathsf{Add}}[\mathsf{pars}, \mathsf{crs}, sk](\widehat{h_1}, \widehat{h_2}; r)$
if $\exists j \in [2] \colon \neg \mathsf{Val}_{\mathbb{H}}((\mathsf{crs}, \mathsf{pars}), \widehat{h_j})$ then
  return $\perp$
parse $\widehat{h_i} =: ([x_i]_{\mathbb{G}}, C_i, \pi_i)_{\mathbb{H}}$
$[x_{\mathsf{out}}]_{\mathbb{G}} := [x_1]_{\mathbb{G}} \cdot [x_2]_{\mathbb{G}}$
$C_{\mathsf{out}} \leftarrow \mathsf{FHE.Eval}(pk, C^{(+)}[\mathbb{Z}_p^n], C_1, C_2)$
// $C^{(+)}[\mathbb{Z}_p^n]$ computes addition in $\mathbb{Z}_p^n$
$\mathbf{v}_i \leftarrow \mathsf{Dec}(sk, C_i)$
$\mathbf{v}_{\mathsf{out}} := \mathbf{v}_1 + \mathbf{v}_2$
$\pi_{\mathsf{out}} \leftarrow \mathsf{Prove}(\mathsf{crs},$
    $(\mathsf{pars}, [x_{\mathsf{out}}]_{\mathbb{G}}, C_{\mathsf{out}}), (sk, \mathbf{v}_{\mathsf{out}}))$
return $\widehat{h_{\mathsf{out}}} := ([x_{\mathsf{out}}]_{\mathbb{G}}, C_{\mathsf{out}}, \pi_{\mathsf{out}})$
```

(a) Definition of the algorithms $\mathsf{GGen}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}, \mathsf{Val}_{\mathbb{H}}, \mathsf{Eq}_{\mathbb{H}}, \mathsf{GetID}_{\mathbb{H}}, \mathsf{Add}_{\mathbb{H}}, \mathsf{Unwrap}_{\mathbb{H}}$ and the circuit C_{Add}.

```
$\mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, \mathbf{v} \in \mathbb{Z}_p^n)$
$\mathbf{v}^* := (v_1 \cdot \alpha_1^{-1}, \ldots, v_n \cdot \alpha_n^{-1})^\intercal$
$[x]_{\mathbb{G}} := [\mathbf{b}^\intercal \cdot \mathbf{v}]_{\mathbb{G}} = [\mathbf{\Omega}^\intercal \cdot \mathbf{v}^*]_{\mathbb{G}}$
$C = \mathsf{Enc}(pk, \mathbf{v}^*; r)$
$\pi = \mathsf{Prove}(\mathsf{crs}, (\mathsf{pars}, [x]_{\mathbb{G}}, C), (sk, \mathbf{v}^*))$
return $([x]_{\mathbb{G}}, C, \pi)_{\mathbb{H}}$

$\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h})$
if $\neg \mathsf{Val}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h})$ then
  return $\perp$
parse $\widehat{h} =: ([x]_{\mathbb{G}}, C, \pi)_{\mathbb{H}}$
$(v_1, \ldots, v_n)^\intercal =: \mathbf{v} = \mathsf{Dec}(sk, C)$
return $(v_1 \cdot \alpha_1, \ldots, v_n \cdot \alpha_n)^\intercal$
```

```
$\mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h})$
$u \leftarrow \{0,1\}^{\ell(\lambda)}$
return $\Lambda_{\mathsf{rerand}}(\widehat{h}, u)$

$C_{\mathsf{rerand}}[\mathsf{pars}, \mathsf{crs}, sk](\widehat{h}; r_1, r_2)$
if $\neg \mathsf{Val}_{\mathbb{H}}((\mathsf{crs}, \mathsf{pars}), \widehat{h})$ then
  return $\perp$
parse $\widehat{h} =: ([x]_{\mathbb{G}}, C, \pi)_{\mathbb{H}}$
$\mathbf{v} := \mathsf{Dec}(sk, C)$
$C_{\mathsf{out}} := \mathsf{FHE.Rerand}(pk, C; r_1)$
$\pi_{\mathsf{out}} \leftarrow \mathsf{Prove}(\mathsf{crs},$
    $(\mathsf{pars}, [x]_{\mathbb{G}}, C_{\mathsf{out}}), (sk, \mathbf{v}); r_2)$
return $\widehat{h_{\mathsf{out}}} := ([x]_{\mathbb{G}}, C_{\mathsf{out}}, \pi_{\mathsf{out}})_{\mathbb{H}}$
```

(b) Definition of the algorithms $\mathsf{PrivSam}_{\mathbb{H}}, \mathsf{PrivExt}_{\mathbb{H}}, \mathsf{Rerand}_{\mathbb{H}}$ and the circuit C_{rerand}.

Fig. 2. Algorithms of our algebraic wrapper construction.

Differences to [1,3,17]**.** [3,17] introduce similar constructions of a group scheme featuring a multilinear map and of a graded encoding scheme, respectively. More precisely, [3,17] equip a base group with encodings carrying auxiliary information which can be used (in an obfuscated circuit) to "multiply in the exponent". We observe that these constructions already *wrap* a given base group in the sense that "unwrapping" encodings yields a group isomorphism to the base group.

Our construction builds upon these group schemes. In order to enable extractability with respect to a dynamically chosen basis[10], our group parameters must be generated depending on that basis.

This modification, however, comes at the cost of the multilinear map functionality. This is because any implementation of a multilinear map requires knowledge of discrete logarithms of each group element encoding to a fixed generator. This is undesirable for our purposes, since we want to be able to use sets of group elements as basis which we do not know discrete logarithms of (for instance group elements provided by a reduction). Thus, we have to give up the multiplication functionality.

Furthermore, looking ahead, we crucially require that the basis can be altered via computational game hops during proofs. We solve this problem by linearly perturbing the given basis $[\mathbf{b}]_{\mathbb{G}}$ (except for its first entry to enable meaningful public sampling). We refer to this perturbed basis as $[\mathbf{\Omega}]_{\mathbb{G}}$. Our group element encodings are defined to carry representation vectors with respect to $[\mathbf{\Omega}]_{\mathbb{G}}$. By construction of C_{Add}, these representation vectors are treated homomorphically by the group operation.

To preserve tightness of security reductions, we additionally introduce a statistical re-randomization mechanism.

As opposed to [1,3,17] uses a quite different approach. In [1], the group scheme is constructed from scratch, meaning there is no necessity for an underlying group. The consequences are twofold. On one hand, very strong decisional assumptions can be proven to hold in the resulting group scheme. On the other hand, however, the group scheme from [1] lacks a $\mathsf{GetID}_{\mathbb{H}}$ algorithm limiting its applicability.

Theorem 2. *Let (i)* $\mathsf{GGen}_{\mathbb{G}}$ *be a group generator for a cyclic group* $\mathbb{G}$*, (ii)* $\mathcal{TD}$ *be a family of hard subset membership problems, (iii)* $\mathsf{FHE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}, \mathsf{Rerand})$ *be a perfectly correct and perfectly re-randomizable fully homomorphic public-key encryption scheme, (iv)* $\Pi := (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{HSetup}, \mathsf{Ext})$ *be a perfectly complete, perfectly sound and perfectly witness-indistinguishable dual-mode NIZK proof system for the language* $\mathcal{L}$*, (v)* piO *be a pIO scheme for the class of samplers* $\mathcal{S}^{X\text{-ind}}$ *and (vi)* $\mathsf{piO}^{\star}_{\ell}$ *be an* ℓ*-expanding pIO scheme for the class of samplers* $\mathcal{S}_{\ell}^{X\text{-}(\star)\text{-ind}}$*. Then,* $\mathbb{H}$ *defined in Figs. 2a and 2b is an algebraic wrapper.*

Here we provide a formal proof of the statistical re-randomization property and a high-level idea for the remaining properties. For a formal analysis of the remaining properties, we refer the reader to the full version [2].

[10] With basis we mean a set of group elements in the base group.

Proof (sketch). Since piO is support respecting, the algorithms defined in Fig. 2a equip the base group $\mathbb{G}$ with non-unique encodings but respect its group structure. Thus, the tuple $(\mathsf{GGen}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}, \mathsf{Val}_{\mathbb{H}}, \mathsf{Eq}_{\mathbb{H}}, \mathsf{Add}_{\mathbb{H}}, \mathsf{GetID}_{\mathbb{H}})$ forms a group scheme such that $\mathsf{Unwrap}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \cdot)$ defines a group isomorphism from $\mathbb{H}$ to $\mathbb{G}$. Therefore, $\mathbb{H}$ satisfies $\mathbb{G}$-wrapping. Extractability follows (more or less) directly by the soundness of the consistency proof and correctness of FHE. Correctness of extraction follows by construction and the correctness of FHE and the fact that piO and $\mathsf{piO}^{\star}_{\ell}$ are support respecting. Correctness of sampling follows directly by correctness of FHE.

Since our construction builds upon techniques developed in [3], we also employ similar strategies to remove information about the secret decryption key from the public group parameters $\mathsf{pp}_{\mathbb{H}}$. To prove k-switching, we next use the IND-CPA security of FHE to remove all information about the basis from the group element encodings. Finally, the only remaining information about the basis used to setup the group parameters resides in $[\mathbf{\Omega}]_{\mathbb{G}}$ which thus looks uniformly random to even an unbounded adversary.

A crucial technical difference to previous work [1,3,17] is the ability to statistically re-randomize encodings. The key ingredient enabling this is our statistically correct pIO scheme due to Theorem 1.

Lemma 1. *The group scheme $\mathbb{H}$ defined in Figs. 2a and 2b satisfies statistical re-randomizability.*

Proof (of Lemma 1). The circuit C_{rerand} takes inputs from $\{0,1\}^{p_{\mathsf{enc}}(\lambda)}$ and expects a randomness from $\{0,1\}^{m'(\lambda)} \times \{0,1\}^{m''(\lambda)}$. We recall that $\mathsf{piO}^{\star}_{\ell}$ is an ℓ-expanding pIO scheme for $\ell(\lambda) = m'(\lambda) + m''(\lambda) + 2(\lambda+1) + 3$. Since for every distribution X_1 over $\{0,1\}^{p_{\mathsf{enc}}(\lambda)}$, $\widetilde{\mathrm{H}}_{\infty}(U_{\ell(\lambda)} \mid X_1) = \ell(\lambda) > m'(\lambda) + m''(\lambda) + 2(\lambda+1) + 2$, the statistical distance between

$$\begin{aligned} &\left\{\Lambda_{\mathsf{rerand}} \leftarrow \mathsf{piO}^{\star}_{\ell}(C_{\mathsf{rerand}})\colon (\Lambda_{\mathsf{rerand}}, \Lambda_{\mathsf{rerand}}(X_1, X_2))\right\} \\ \text{and } &\left\{\Lambda_{\mathsf{rerand}} \leftarrow \mathsf{piO}^{\star}_{\ell}(C_{\mathsf{rerand}})\colon (\Lambda_{\mathsf{rerand}}, C_{\mathsf{rerand}}(X_1; U_{m'(\lambda)+m''(\lambda)}))\right\} \end{aligned}$$

is at most $2^{-(\lambda+1)}$.

Let $\widehat{h_0} =: ([x_0]_{\mathbb{G}}, C_0, \pi_0)_{\mathbb{H}}, \widehat{h_1} =: ([x_1]_{\mathbb{G}}, C_1, \pi_1)_{\mathbb{H}} \in \mathbb{H}$ be the encodings chosen by the adversary $\mathcal{A}$. Since $\mathcal{A}$ is a legitimate re-randomization adversary, $\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h_0}) = \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h_1})$. Due to perfect correctness of FHE and since $\alpha_1, \ldots, \alpha_n \in \mathbb{Z}_p^{\times}$ are invertible, $\mathsf{Dec}(sk, C_0) = \mathsf{Dec}(sk, C_1)$. Due to perfect re-randomizability of FHE, the ciphertexts produced by $C_{\mathsf{rerand}}(\widehat{h_0})$ and $C_{\mathsf{rerand}}(\widehat{h_1})$ are identically distributed. Furthermore, since $C_{\mathsf{rerand}}(\widehat{h_b})$ produces the consistency proof using the witness $(sk, \mathsf{Dec}(sk, C_b))$, the distributions produced by $C_{\mathsf{rerand}}(\widehat{h_0})$ and $C_{\mathsf{rerand}}(\widehat{h_1})$ are identical. Therefore, $\mathrm{Adv}^{\mathtt{rerand}}_{\mathbb{H},\mathcal{A}}(\lambda) \leq 2 \cdot 2^{-(\lambda+1)} = 2^{-\lambda}$.

Note that since $\mathbb{G}$ has unique encodings, $\mathcal{A}$ is unable to extract auxiliary information from the encodings of $\mathsf{Unwrap}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h})$. This is crucial since such auxiliary information may be used to distinguish whether $\widehat{h_0}$ or $\widehat{h_1}$ was used to derive $\widehat{h}$. $\square$

4 How to Use Algebraic Wrappers – Implementing Proofs from the AGM

In the following, we demonstrate how proof techniques from the algebraic group model can be implemented with our algebraic wrapper. Mainly, we want to use the extracted representation provided by the algebraic wrapper in a similar way as in AGM proofs. We adapt the proofs of Diffie-Hellman assumptions from [21] in Sect. 4.1 as well as the proof for the EUF-CMA security of Schnorr signatures from [22] in Sect. 4.2. Before we demonstrate how to use the algebraic wrapper, we sketch two modifications which will be necessary when we replace the AGM with the algebraic wrapper.

The symmetrization technique. Information-theoretically, the basis[11] the algebraic wrapper enables extraction for, as well as the representation vectors inside group element encodings are known to the adversary. However, several security reductions in [21] employ case distinctions where different reduction algorithms embed their challenge in different group elements. For instance, in the CDH game, the discrete logarithm challenge Z can be embedded either in $[x]_{\mathbb{H}}$ or $[y]_{\mathbb{H}}$, leading to two different security reductions. Due to the ideal properties of the AGM, both reductions simulate identically distributed games.

However, transferring this strategy directly using algebraic wrappers fails, since the two reductions are information-theoretically distinguishable depending on the choice of basis. An unbounded adversary who knows which game he is playing could therefore influence the representation of his output in such a way that it always becomes impossible for the reduction to use the representation to compute the discrete logarithm. We call such a situation a *bad case*. It is necessary that the different reduction subroutines have mutually exclusive bad cases, so that extraction is always possible in at least one game type. Thus, we need find a way that even these representations (and the basis used to generate $\mathsf{pp}_{\mathbb{H}}$) are identically distributed.

We therefore introduce a proof technique which we call *symmetrization*. We extend the basis and group element representations in such a way that the games played by different reduction subroutines are identically distributed (as they would be in the AGM). This is done by choosing additional base elements to which the reduction knows the discrete logarithm (or partial logarithms), so that these additional base elements do not add any unknowns when solving for the discrete logarithm. With this technique, we achieve that the games defined by the different reduction algorithms are identically distributed but entail different mutually-exclusive bad cases. For the CDH reduction, this means that both challenge elements $[x]_{\mathbb{H}}$ and $[y]_{\mathbb{H}}$ are contained in the basis, so that it is not known to the adversary which one is the reduction's discrete logarithm challenge. This allows to adopt the proofs from AGM.

[11] With *basis* we mean the set of group elements in the base group to which we can extract.

The origin element trick. Applying the algebraic wrapper to AGM proofs where an oracle (e.g. a random oracle or a signing oracle) is present, entails the need to change the representation vectors of all oracle responses. One possibility to realize this is to apply Q-`switching`, where Q denotes a polynomial upper bound on the number of oracle queries. However, as the switching property only provides computational guarantees, this naive approach results in a non-tight reduction. Since we are interested in preserving the tightness of AGM proofs when applying the algebraic wrapper, we use so-called *origin elements* from which we construct the oracle responses using the group operation. This enables to use n-`switching` for a constant number n of origin elements instead of Q-`switching` for Q oracle responses.

Limitations of our techniques. While our algebraic wrapper provides an extraction property that is useful for many proofs in the AGM, it also has its limitations. Mainly, the base elements to which the $\mathsf{PrivExt}$ algorithm can extract need to be fixed at the time of group parameter generation. Therefore, we cannot mimic reductions to assumptions with a variable amount of challenge elements, where extraction needs to be possible with respect to all these challenge elements. For instance, q-type assumptions which are used in [21] to prove CCA1-security of ElGamal and the knowledge-soundness of Groth's ZK-SNARK.

Furthermore, there are security proofs in the AGM that rely on the representation used by the reduction being information-theoretically hidden from the adversary. An example for this is the tight reduction for the BLS scheme from [21]. As the reduction can forge a signature for any message, it relies on the representations provided by the adversary being different from what the reduction could have computed on its own. In the AGM, it is highly unlikely that the adversary computes the forged signature in the exact same way as the reduction simulates the signing oracle, because the reduction does not provide the adversary with an algebraic representation. However, since we need to be able to extract privately from group element encodings, the group elements output by the reduction information theoretically contain algebraic representations. Therefore, information-theoretically, an adversary sees how the reduction simulates hash responses and signatures, and thus could provide signatures with a representation that is useless to the reduction.

This problem is circumvented in the Schnorr proof in Sect. 4.2 due to the representation provided by the adversary already being fixed by the time it receives a challenge through the Random Oracle. We leave it as an open problem to transfer the BLS proof to the algebraic wrapper.

Another limitation is that due to the reduction being private, we cannot use the extraction in reductions between problems in the same group. That is, our wrapper does not allow for "multi-step" reductions as in the AGM.

4.1 Diffie-Hellman Assumptions

We show how to adapt the security reductions for Diffie-Hellman problems from [21] to our algebraic wrapper (see Fig. 3 for the definitions). The main proof idea, namely to use the representation of the adversary's output to compute the discrete logarithm, stays the same; however, due to the nature of our wrapper, we need to apply the symmetrization technique to achieve the same distributions as in the AGM.

```
cdh
$x, y \leftarrow \mathbb{Z}_p$
$s \leftarrow \mathcal{A}([1]_{\mathbb{G}}, [x]_{\mathbb{G}}, [y]_{\mathbb{G}})$
return $s = [xy]_{\mathbb{G}}$
```

```
sqdh
$x \leftarrow \mathbb{Z}_p$
$s \leftarrow \mathcal{A}([1]_{\mathbb{G}}, [x]_{\mathbb{G}})$
return $s = \; x^2 \;_{\mathbb{G}}$
```

```
lcdh
$x, y \leftarrow \mathbb{Z}_p$
$u, v, w, s \leftarrow \mathcal{A}([1]_{\mathbb{G}}, [x]_{\mathbb{G}}, [y]_{\mathbb{G}})$
return $s = \; u \cdot x^2 + v \cdot xy + w \cdot y^2]_{\mathbb{G}}$
```

Fig. 3. The different types of Diffie-Hellman games shown in [21]

Theorem 3. *Let $\mathbb{G}$ be a group where the discrete logarithm is hard. Then, the computational Diffie-Hellman assumption holds in an algebraic wrapper $\mathbb{H}$ for $\mathbb{G}$ of dimension ≥ 3.*

We sketch the proof here and refer the reader to the full version [2] for the full proof.

```
$G_0$
$\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{GGen}_{\mathbb{G}}(1^\lambda)$
$\beta_2, \beta_3 \leftarrow \mathbb{Z}_p$
$(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{GGen}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, ([1]_{\mathbb{G}}, [\beta_2]_{\mathbb{G}}, [\beta_3]_{\mathbb{G}})^\intercal)$
$x, y \leftarrow \mathbb{Z}_p$
$\widehat{1} = \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1))$
$\widehat{x} = \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, x))$
$\widehat{y} = \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, y))$
$s \leftarrow \mathcal{A}(\mathsf{pp}_{\mathbb{H}}, \widehat{1}, \widehat{x}, \widehat{y})$
return $\mathsf{Eq}_{\mathbb{H}}(\widehat{x}^y, s)$
```

```
$G_1$
$\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{GGen}_{\mathbb{G}}(1^\lambda)$
$X \leftarrow \mathbb{G}$
$z \leftarrow \mathbb{Z}_p$
$(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{GGen}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, ([1]_{\mathbb{G}}, [x]_{\mathbb{G}}, [y]_{\mathbb{G}})^\intercal)$
$\widehat{1} = \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1))$
$\widehat{x} = \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0, 1, 0)^\intercal))$
$\widehat{y} = \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0, 0, 1)^\intercal))$
$s \leftarrow \mathcal{A}(\mathsf{pp}_{\mathbb{H}}, \widehat{1}, \widehat{x}, \widehat{y})$
return $\mathsf{Eq}_{\mathbb{H}}([xy]_{\mathbb{G}}, s)$
```

Fig. 4. The CDH games used in the security proof. G_0 corresponds to the honest CDH-game. Games of type G_1 allow the reduction to embed its discrete logarithm challenge and extract a useful representation.

Proof (sketch). We use an algebraic wrapper with basis $[1]_{\mathbb{G}}, [x]_{\mathbb{G}}, [y]_{\mathbb{G}}$. Initially, we perform game hops starting from the CDH game (where every encoding carries representation vectors in the first component), see G_0 in Fig. 4 and reach a game, where the encodings produced as CDH challenge carry representation vectors of $\mathbf{e_1}, \mathbf{e_2}$ and $\mathbf{e_3}$, respectively, see G_1 in Fig. 4. These game hops are justified by 2-`switching` and `rerand`.

The reduction flips a coin whether to embed the DLOG challenge Z as $[x]_{\mathbb{G}}$ or $[y]_{\mathbb{G}}$, i.e. it applies the symmetrization technique. In both cases, the view of the CDH adversary is identical. When the CDH adversary outputs a solution, the reduction is able to compute the discrete logarithm of the embedded DLOG challenge from the representation vector extracted from the solution. □

We additionally show the following in the full version [2].

Theorem 4. *Let $\mathbb{G}$ be a group where the discrete logarithm is hard. Then, the square Diffie-Hellman assumption holds in an algebraic wrapper $\mathbb{H}$ of dimension ≥ 2 for $\mathbb{G}$.*

Theorem 5. *Let $\mathbb{G}$ be a group where DLOG is hard and $\mathbb{H}$ be an algebraic wrapper of dimension ≥ 3 for $\mathbb{G}$. Then, the linear-combination Diffie-Hellman problem is hard in $\mathbb{H}$.*

4.2 Schnorr Signatures

We apply the algebraic wrapper to mimic the proof of tight EUF-CMA security of Schnorr Signatures from [22].

Theorem 6. *Let $\mathsf{GGen}_{\mathbb{G}}$ be a group generator for a cyclic group $\mathbb{G}$ such that DLOG is hard relative to $\mathsf{GGen}_{\mathbb{G}}$ and let $\mathbb{H}$ be an algebraic wrapper of dimension ≥ 2 for $\mathbb{G}$. Then, the Schnorr signature scheme in $\mathbb{H}$ (Fig. 5) is tightly EUF-CMA secure in the random oracle model.*

More precisely, for all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ and a legitimate switching adversary $\mathcal{A}''$ both running in time $T(\mathcal{B}) \approx T(\mathcal{A}) + (q_s + q_h) \cdot \mathsf{poly}(\lambda)$ and $T(\mathcal{A}'') \approx T(\mathcal{A}) + (q_s + q_h) \cdot \mathsf{poly}(\lambda)$ such that

$$\mathrm{Adv}^{\mathsf{euf\text{-}cma}}_{\Sigma_{\mathsf{schnorr}},\mathcal{A}}(\lambda) \leq \mathrm{Adv}^{DLOG}_{\mathcal{B},\mathbb{G}}(\lambda) + \mathrm{Adv}^{1\text{-}\mathtt{switching}}_{\mathcal{A}'',\mathbb{H}}(\lambda) + \frac{O(q_s(q_s + q_h))}{2^{\lambda}},$$

where q_h is a polynomial upper bound on the number of random oracle queries, q_s is a polynomial upper bound on the number of signing queries and poly is a polynomial independent of q_s and q_h.

```
$\mathsf{KGen}(\mathsf{pp}_{\mathbb{H}})$
$x \leftarrow \mathbb{Z}_p$
$\widehat{1} := \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1))$
$\widehat{X} := \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, x))$
$pk := (\mathsf{pp}_{\mathbb{H}}, \widehat{1}, \widehat{X})$
$sk := (pk, x)$
$\mathbf{return}\ (pk, sk)$
```

```
$\mathsf{Sign}(sk, m)$
$r \leftarrow \mathbb{Z}_p$
$\widehat{R} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, r))$
$c := H(\widehat{R}, m)$
$s := r + c \cdot x \mod p$
$\mathbf{return}\ \sigma := (\widehat{R}, s)$
```

```
$\mathsf{Ver}(pk = (\mathsf{pp}_{\mathbb{H}}, \widehat{1}, \widehat{X}), m, \sigma = (\widehat{R}, s))$
$c := H(\widehat{R}, m)$
$\mathbf{return}\ \mathsf{Eq}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, s), \widehat{R} \cdot \widehat{X}^c)$
```

Fig. 5. The Schnorr signature scheme $\Sigma_{\mathtt{schnorr}}$. Note that to compensate for the non-uniqueness of group element encodings, the (random oracle) hash value of a group element encoding is computed for the unique identifier produced by $\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \cdot)$.

```
$\mathrm{Exp}^{\mathsf{euf\text{-}cma}}_{\Sigma_{\mathtt{schnorr}},\mathcal{A}}(\lambda)$
$\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{GGen}_{\mathbb{G}}(1^\lambda)$
$(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{GGen}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, ([1]_{\mathbb{G}}, [\beta_2]_{\mathbb{G}})^\top)$
$x \leftarrow \mathbb{Z}_p$
$\xi_1 \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1))$
$\xi_2 \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, x))$
$pk := (\mathsf{pp}_{\mathbb{H}}, \xi_1, \xi_2)$
$Q := \varnothing, T := []$
$(m^*, \widehat{R^*}, s^*) \leftarrow \mathcal{A}^{H,\mathsf{Sign}}(1^\lambda, pk)$
$\mathbf{if}\ m^* \in Q\ \mathbf{then\ return}\ 0$
$c^* = H(\widehat{R^*}, m^*)$
$\mathbf{return}\ \mathsf{Eq}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, s^*), \widehat{R^*} \cdot \xi_2^{c^*})$
```

```
$H(\widehat{R}, m)$
$\mathbf{if}\ T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)] = \perp\ \mathbf{then}$
    $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)] \leftarrow \mathbb{Z}_p$
$\mathbf{return}\ T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)]$
```

```
$\mathsf{Sign}(m)$
$r \leftarrow \mathbb{Z}_p$
$\widehat{R} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, r))$
$c := H(\widehat{R}, m)$
$s := r + cx$
$Q := Q \cup \{m\}$
$\mathbf{return}\ (\widehat{R}, s)$
```

Fig. 6. The EUF-CMA game for Schnorr signatures. Note that β_2 can be chosen arbitrarily.

Proof. We use the origin element trick to avoid using q_s-`switching` (see Definition 7) which would compromise tightness of the reduction. Figure 6 shows the EUF-CMA game with Schnorr signatures instantiated with the algebraic wrapper. We note that for groups with non-unique encodings, the hash function hashes the unique identifier returned by $\mathsf{GetID}_{\mathbb{H}}$, hence, encodings corresponding to the same group element are mapped to the same hash value. The reduction uses a table T to keep track of previously made hash queries and their responses, as well as a set Q to keep track of the messages the adversary has requested signatures for.

```
$\boxed{G_1}$ $G_2$ $G_3$ $G_4$ $G_5$
$\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{GGen}_{\mathbb{G}}(1^\lambda)$
$(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{GGen}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, ([1]_{\mathbb{G}},$
$\boxed{[\beta_2]_{\mathbb{G}}}$ $[x]_{\mathbb{G}}$ $)^\top)$
$x \leftarrow \mathbb{Z}_p$
$\xi_1 \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1))$
$\xi_2 \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}},$ $\boxed{\mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, x)}$
$\mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (x, 0))$ $)$
$pk := (\mathsf{pp}_{\mathbb{H}}, \xi_1, \xi_2)$
$Q := \varnothing, T := []$ $U := []$
$(m^*, \widehat{R^*}, s^*) \leftarrow \mathcal{A}^{H,\mathsf{Sign}}(1^\lambda, pk)$
if $m^* \in Q$ then return $0$
if $U[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R^*}), m^*)] \neq \perp$ then
    $(\gamma^*, \zeta^*) := U[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R^*}), m^*)]$
    if $\zeta^* = -T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R^*}), m^*)]$ then return $0$
$c^* = H(\widehat{R^*}, m^*)$
return $\mathsf{Eq}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, s), \widehat{R^*} \cdot \xi_2^{c^*})$
```

```
$H(\widehat{R}, m)$
if $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)] = \perp$ then
    $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)] \leftarrow \mathbb{Z}_p$
    $U[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)] = \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{R})$
return $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)]$

$\mathsf{Sign}(m)$
$\boxed{r}$ $c, s$ $\leftarrow \mathbb{Z}_p$
$\boxed{\widehat{R_1} \leftarrow \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, r)}$
$\boxed{c := H(\widehat{R_1}, m)}$
$\boxed{s := r + cx}$

$\boxed{\widehat{R_2} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, s - cx))}$
$\widehat{R_2} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \xi_1^s \cdot \xi_2^{-c})$
$\widehat{R_2} = \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \xi_1^s \cdot \xi_2^{-c})$
if $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R_2}), m)] = \perp$ then
    $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R_2}), m)] := c$
else
    abort

$Q := Q \cup \{m\}$
return $(\boxed{\widehat{R_2}}, s)$
```

Fig. 7. Games G_1, G_2, G_3. Boxed content happens in the corresponding games and following games if no replacement is defined. The randomness for signatures is drawn using an x-component in G_1. G_1 is identically distributed to $\mathrm{Exp}^{\mathsf{euf\text{-}cma}}_{\Sigma_{\mathtt{schnorr}},\mathcal{A}}(\lambda)$. In G_2, the second origin element is sampled through private sampling and the random part of the signatures is generated through origin elements. G_2 is statistically close to G_1 due to re-randomizability. In G_3, we switch the basis and representation of ξ_2; this hop is justified by 1-`switching`.

Game hop from $\mathrm{Exp}^{\mathsf{euf\text{-}cma}}_{\Sigma_{\mathtt{schnorr}},\mathcal{A}}(\lambda) \rightsquigarrow G_1$**.** Since $r = s - cx \bmod p$ and hence $\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R_1}) = \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R_2})$, these two games are identically distributed.

Game hop $G_1 \rightsquigarrow G_2$**.** In G_2 (see Fig. 7), we construct $\widehat{R_2}$ from origin elements through the group operation instead of sampling. This game hop is justified by the re-randomizability of the algebraic wrapper. A reduction to this property works as a series of q_s+1 hybrids where H_0 is G_1, where q_s denotes a polynomial upper bound on the number of signing queries. In H_i, the first i signature queries are answered as in G_2 and the $i+1$-th to q_s-th signature queries are answered as in G_1. In the last hybrid, the public key is also changed to private sampling. If there is an (unbounded) adversary that distinguishes H_i and H_{i+1}, the reduction $\mathcal{A}'$ uses this adversary to attack the re-randomizability as follows. On input of base group parameters $\mathsf{pp}_{\mathbb{G}}$, $\mathcal{A}'$ picks a basis $([1]_{\mathbb{G}}, [\beta_2]_{\mathbb{G}})$ and gives it to the `rerand` challenger. It receives public parameters and the trapdoor. Then, it simulates H_i to the adversary for the first i signature queries, i.e. it samples $\widehat{R_{2,j}} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \xi_1^{s_j} \cdot \xi_2^{-c_j})$ for $j < i$. For the $i+1$-th signature query, $\mathcal{A}'$ sends the two elements $\widehat{h_0} = \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, s_{i+1} - c_{i+1} \cdot x)$ and $\widehat{h_1} = \xi_1^{s_{i+1}} \cdot \xi_2^{-c_{i+1}}$ to the challenger and receives a challenge $\widehat{C}$. It uses this challenge $\widehat{C}$ as $\widehat{R_{2,i+1}}$ to answer the $i+1$-th hash query and responds to the remaining queries as in H_{i+1}, i.e. it samples $\widehat{R_j} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, s_j - c_j \cdot x))$ for $j > i+1$. Depending on the challenge encoding $\widehat{C}$, $\mathcal{A}'$ either simulates H_i or H_{i+1} perfectly and outputs the output of the corresponding game.

In hybrid H_{q_s}, all signature queries are answered as in game G_2. The last step to game $H_{q_s+1} = G_2$ changes how ξ_2 (which is part of the public key) is sampled. An adversary distinguishing H_{q_s} and H_{q_s+1} can be used to build an adversary $\mathcal{A}'$ in `rerand` similarly as above. More precisely, $\mathcal{A}'$ outputs the encodings $\widehat{h_0} \leftarrow \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, x)$ and $\widehat{h_1} \leftarrow \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, x)$ (note that $\tau_{\mathbb{H}}$ is known during the `rerand` game) and uses the challenge encoding from the `rerand` challenger as ξ_2. We note that this last game hop paves the way to apply `1-switching`.

Due to correctness of sampling and correctness of extraction, the representation vectors of the elements used in the `rerand` game are identical and hence $\mathcal{A}'$ is a legitimate adversary in the `rerand` game and its advantage is upper bounded by $\frac{1}{2^\lambda}$. Therefore,

$$|\Pr[out_1 = 1] - \Pr[out_2 = 1]| \leq \frac{q_s + 1}{2^\lambda}.$$

Game hop $G_2 \rightsquigarrow G_3$. In game G_3 (see Fig. 7) we switch the basis and the representation of the origin element ξ_2. This game hop is justified by `1-switching`. Let $\mathcal{A}$ be an adversary distinguishing G_2 and G_3. We construct an adversary $\mathcal{A}''$ on `1-switching` as follows. Initially, $\mathcal{A}''$ on input of $\mathsf{pp}_{\mathbb{G}}$, outputs $[\mathbf{b}]_{\mathbb{G}}^{(G_2)} = [(1, \beta_2)^\intercal]_{\mathbb{G}}$ and $[\mathbf{b}]_{\mathbb{G}}^{(G_3)} = [(1, x)^\intercal]_{\mathbb{G}}$ and the representation vectors $\mathbf{v}^{(\mathbf{G_2})} := (x, 0)^\intercal$ and $\mathbf{v}^{(\mathbf{G_3})} := (0, 1)^\intercal$. In return, $\mathcal{A}''$ receives public parameters $\mathsf{pp}_{\mathbb{H}}$ and an encoding $\widehat{C}$ and samples $\xi_2 \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{C})$. The trapdoor $\tau_{\mathbb{H}}$ is not necessary to simulate G_3 and G_4 (except for sampling ξ_2). Hence, $\mathcal{A}''$ perfectly simulates G_3 or G_4 for $\mathcal{A}$ depending on the challenge provided by the `1-switching` challenger. Thus, $|\Pr[out_3 = 1] - \Pr[out_2 = 1]| \leq \mathrm{Adv}_{\mathbb{H},\mathcal{A}''}^{\texttt{1-switching}}(\lambda)$. Note that $\mathcal{A}''$ is a legitimate switching adversary since $[(1, \beta_2)]_{\mathbb{G}} \cdot (x, 0)^\intercal = [x]_{\mathbb{G}} = [(1, x)]_{\mathbb{G}} \cdot (0, 1)^\intercal$ and hence $\mathrm{Adv}_{\mathbb{H},\mathcal{A}''}^{\texttt{1-switching}}(\lambda)$ is negligible.

Game hop $G_3 \rightsquigarrow G_4$. In G_4 (see Fig. 7), we introduce a list U to keep track of the representations of group elements used in Random Oracle queries. The games G_3 and G_4 differ in the fact that G_4 extracts the representation vectors contained in the encoding of a group element when this group element message tuple is queried for the first time and stores this representation in a list. Furthermore, G_4 introduces an abort condition which is triggered if the representation of $\widehat{R^*}$ originally used to query the random oracle on $(\widehat{R^*}, m^*)$ already contained the response in the second component ζ^*. This corresponds to the game hop from G_0 to G_1 in [22]. The game only aborts if the hash $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R^*}), m^*)]$ is the same as the second component ζ^* of the representation extracted from $\widehat{R^*}$. Since the hash $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R^*}), m^*)]$ is chosen uniformly at random *after* the representation (γ^*, ζ^*) is fixed, the probability that an unbounded adversary can find such an $(\widehat{R^*}, m^*)$ is upper bounded by $\frac{q_h}{p} \leq \frac{q_h}{2^\lambda}$, where q_h denotes a polynomial upper bound on the number of random oracle queries. Hence, $|\Pr[out_4 = 1] - \Pr[out_3 = 1]| \leq \frac{q_h}{2^\lambda}$.

Game hop $G_4 \rightsquigarrow G_5$. In game G_5 (see Fig. 7), we change how signature queries are answered such that it is not necessary anymore to know the discrete logarithm of the public key. This game hop corresponds to the hop from G_1 to G_2 in [22]. On one hand, since $\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R_1}) = \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R_2})$, replacing $\widehat{R_1}$ with $\widehat{R_2}$ does not change the distribution. On the other hand, as we are only able to answer a signing query if we can program the random oracle at $(\widehat{R_2}, m)$ (for randomly chosen $\widehat{R_2}$), the signing oracle has to abort in case the hash was already queried before. Since $\widehat{R_2}$ is a independently sampled uniformly random group element, this happens only with probability $\frac{1}{p} \leq \frac{1}{2^\lambda}$. Hence, by a union bound, this abort occurs at most with probability $\frac{q_s(q_s+q_h)}{2^\lambda}$ cases, where q_s denotes a polynomial upper bound on the number of signing queries and q_h denotes a polynomial upper bound on the number of random oracle queries. Conditioned on the event that no abort occurs, G_4 and G_5 are distributed identically. Hence, by the Difference Lemma due to Shoup [39], we have $|\Pr[out_5 = 1] - \Pr[out_4 = 1]| \leq \frac{q_s(q_s+q_h)}{2^\lambda}$. As in [22], on extraction of the initial representation (γ^*, ζ^*) of $\widehat{R^*}$ from a valid signature $(\widehat{R^*}, s^*)$ output by the adversary, the reduction can use that $\widehat{R^*} = [\gamma^*]_{\mathbb{H}} \cdot [\zeta^* \cdot z]_{\mathbb{H}} = [s^* - c^* \cdot z]_{\mathbb{H}}$. Therefore,

$$z = \frac{s^* - \gamma^*}{\zeta^* - c^*}.$$

Due to the added check in G_4, an adversary can only win G_4 or G_5 when $\zeta^* - c^* \neq 0$ which concludes the proof. □

Acknowledgments. We would like to thank the anonymous reviewers of EC20 for many helpful comments and for pointing out an error in previous versions of Lemma 1 and the proof of the switching property.

References

1. Agrikola, T., Hofheinz, D.: Interactively secure groups from obfuscation. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 341–370. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76581-5_12
2. Agrikola, T., Hofheinz, D., Kastner, J.: On instantiating the algebraic group model from falsifiable assumptions. Cryptology ePrint Archive, Report 2020/070 (2020). https://eprint.iacr.org/2020/070

3. 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, Part I. LNCS, vol. 9562, pp. 446–473. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49096-9_19
4. Bellare, M., Palacio, A.: The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 273–289. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28628-8_17
5. Bitansky, N., Canetti, R., Paneth, O., Rosen, A.: On the existence of extractable one-way functions. In: Shmoys, D.B. (ed.) 46th ACM STOC, pp. 505–514. ACM Press, May/June 2014
6. Boneh, D., Lipton, R.J.: Algorithms for black-box fields and their application to cryptography. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 283–297. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68697-5_22
7. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. J. Cryptol. **17**(4), 297–319 (2004). https://doi.org/10.1007/s00145-004-0314-9
8. Boneh, D., Segev, G., Waters, B.: Targeted malleability: homomorphic encryption for restricted computations. In: Goldwasser, S. (ed.) ITCS 2012, pp. 350–366. ACM, January 2012
9. Boneh, D., Venkatesan, R.: Breaking RSA may not be equivalent to factoring. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 59–71. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0054117
10. Boyen, X.: The uber-assumption family. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 39–56. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85538-5_3
11. Canetti, R., Lin, H., Tessaro, S., Vaikuntanathan, V.: Obfuscation of probabilistic circuits and applications. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 468–497. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_19
12. Coron, J.-S.: On the exact security of full domain hash. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 229–235. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44598-6_14
13. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_4
14. Damgård, I.: Towards practical public key systems secure against chosen ciphertext attacks. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 445–456. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_36
15. Dent, A.W.: Adapting the weaknesses of the random oracle model to the generic group model. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 100–109. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36178-2_6
16. Dent, A.W.: The cramer-shoup encryption scheme is plaintext aware in the standard model. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 289–307. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_18
17. Farshim, P., Hesse, J., Hofheinz, D., Larraia, E.: Graded encoding schemes from obfuscation. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 371–400. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76581-5_13
18. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: 22nd ACM STOC, pp. 416–426. ACM Press, May 1990

19. Fleischhacker, N., Jager, T., Schröder, D.: On tight security proofs for Schnorr signatures. J. Cryptol. **32**(2), 566–599 (2019). https://doi.org/10.1007/s00145-019-09311-5
20. Freire, E.S.V., Hofheinz, D., Paterson, K.G., Striecks, C.: Programmable hash functions in the multilinear setting. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 513–530. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_28
21. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 33–62. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_2
22. Fuchsbauer, G., Plouviez, A., Seurin, Y.: Blind Schnorr signatures in the algebraic group model. Cryptology ePrint Archive, Report 2019/877 (2019). http://eprint.iacr.org/2019/877
23. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1–17. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_1
24. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_24
25. Hada, S., Tanaka, T.: On the existence of 3-round zero-knowledge protocols. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 408–423. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0055744
26. Hofheinz, D., Kiltz, E.: Programmable hash functions and their applications. J. Cryptol. **25**(3), 484–527 (2012). https://doi.org/10.1007/s00145-011-9102-5
27. Hofheinz, D., Ursu, B.: Dual-mode NIZKs from obfuscation. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11921, pp. 311–341. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34578-5_12. https://eprint.iacr.org/2019/475
28. Hohenberger, S., Sahai, A., Waters, B.: Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 494–512. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_27
29. Hohenberger, S., Sahai, A., Waters, B.: Replacing a random oracle: full domain hash from indistinguishability obfuscation. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 201–220. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_12
30. Kastner, J., Pan, J.: Towards instantiating the algebraic group model. Cryptology ePrint Archive, Report 2019/1018 (2019). https://eprint.iacr.org/2019/1018
31. Maurer, U.: Abstract models of computation in cryptography. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 1–12. Springer, Heidelberg (2005). https://doi.org/10.1007/11586821_1
32. Maurer, U., Wolf, S.: Lower bounds on generic algorithms in groups. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 72–84. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0054118
33. Naor, M.: On cryptographic assumptions and challenges. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 96–109. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_6
34. Nechaev, V.I.: Complexity of a determinate algorithm for the discrete logarithm. Math. Notes **55**(2), 165–172 (1994). https://doi.org/10.1007/BF02113297

35. Paillier, P., Vergnaud, D.: Discrete-log-based signatures may not be equivalent to discrete log. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 1–20. Springer, Heidelberg (2005). https://doi.org/10.1007/11593447_1
36. Peikert, C., Shiehian, S.: Noninteractive zero knowledge for NP from (plain) learning with errors. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 89–114. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26948-7_4
37. Schnorr, C.P.: Efficient signature generation by smart cards. J. Cryptol. **4**(3), 161–174 (1991). https://doi.org/10.1007/BF00196725
38. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_18
39. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332 (2004). http://eprint.iacr.org/2004/332