The Uber-Knowledge Assumption: A Bridge to the AGM

Balthazar Bauer[1], Pooya Farshim[2,3], Patrick Harasser[4], and Markulf Kohlweiss[2,5]

[1] Université de Versailles Saint-Quentin-en-Yvelines, France
balthazar.bauer@ens.fr
[2] IOG, Singapore
{pooya.farshim, markulf.kohlweiss}@iohk.io
[3] Durham University, UK
[4] Technische Universität Darmstadt, Germany
patrick.harasser@tu-darmstadt.de
[5] University of Edinburgh, UK

Abstract. The generic-group model (GGM) and the algebraic-group model (AGM) have been immensely successful in proving the security of many classical and modern cryptosystems. These models, however, come coupled with standard-model uninstantiability results, raising the question whether the schemes analyzed under them can be based on firmer standard-model footing.

We formulate the *uber-knowledge* (UK) assumption, a standard-model assumption that naturally extends the uber-assumption family to knowledge assumptions. We justify the soundness of the UK in both the bilinear GGM and bilinear AGM. Along the way we extend these models to incorporate *hashing into groups*, an adversarial capability that is available in many concrete groups. (In contrast to standard assumptions, hashing may affect the validity of knowledge assumptions.) These results, in turn, enable a modular approach to security in GGM and AGM.

As example applications, we use the UK to prove knowledge-soundness of Groth16 and KZG polynomial commitments in the standard model, where for the former we *reuse* the existing AGM proof without hashing.

Keywords. Knowledge assumption · Standard model · Generic-group model · Algebraic-group model · Groth16 · KZG commitment

1 Introduction

1.1 Background

Security proofs in idealized models of computation or with respect to restricted classes of adversaries are a popular paradigm for studying the soundness of cryptographic constructions. Starting with the works of Fiat and Shamir [34] and Bellare and Rogaway [8], random oracles, which idealize cryptographic hash functions, have been used to justify the security of a wide range of symmetric and asymmetric systems. Subsequently, the random-permutation and the ideal-cipher

models were used to study permutation-based cryptography (e.g., SHA3 [11]) and constructions using block ciphers [16, 25, 48]. This approach was adapted to constructions involving cryptographic groups by Nachaev [58] and Shoup [66], who showed the hardness of the discrete-logarithm problem in random groups with oracle access to the group operation.

Our focus in this work is on cryptographic assumptions related to groups. We start with a high-level overview of idealization of groups as initially introduced by Nachaev and Shoup.

The generic-group model (GGM). The GGM "idealizes" the representation of group elements and the group operation. There are at least two approaches to formalizing idealized groups. One is Shoup's GGM [66], aka. the *random-representation* (RR) model [67], where group exponentiation is modeled as a random injection τ, and group operation is defined via an oracle that is compatible with τ (i.e., elements are inverted under τ, summed, and fed back to τ). Another is Maurer's GGM [56], aka. the *type-safe* (TS) model [67], where group elements are replaced by abstract "handles" that act as pointers to memory locations containing group elements. The group operation oracle works on handles, by placing the sum of the elements under two given handles under a third handle. Shoup's model has been extended to bilinear groups [17], and has been used to study a wide class of schemes, from standardized signature schemes [45] to structure-preserving signatures [1] and SNARKs [43].

The algebraic-group model (AGM). A somewhat different approach has emerged recently. Motivated by the fact that group operations are observable in the GGM, it models that an adversary can always compute a representation of new group elements that it produces based on those that it has seen so far. This model is known as the algebraic-group model (AGM) and was introduced by Fuchsbauer, Kiltz, and Loss [36], though its roots trace back to the work of Boneh and Venkatesan [19], who considered restricted adversaries that implement straight-line programs. In a sense the underlying groups in the AGM are not idealized; it is rather the adversary who is restricted/idealized. Recently there has been significant interest in using the AGM to analyze cryptosystems [37, 40, 50, 55, 61] and hardness assumptions [3, 62].

One drawback of idealized models of computation, however, is that they typically suffer from *uninstantiability results*. That is, one can construct schemes that are secure in a given idealized model, but are insecure with respect to *any* standard-model instantiation of the primitive that the model idealizes. Such uninstantiable schemes were first introduced in the seminal work of Canetti, Goldreich, and Halevi [22], and were later extended to the ideal-cipher [15] and generic-group models [29]. Arguably these uninstantiable schemes are quite "contrived" in that they are designed to fail, and as such do not disprove the security established in an idealized model of cryptosystems that follow "good crypto-

graphic practice".[6] Recently, Zhandry [67] has extended uninstantiability to the AGM, thus separating the AGM from the standard model.

Given this state of affairs, one research theme in recent years has been to identify new plausible assumptions that, although strong, facilitate proofs of security in the *standard model* in a uniform way for a range of schemes that were previously only shown to be secure in idealized models. Put differently, under such assumptions these schemes are placed outside the class of uninstantiable or contrived ones. Furthermore, if the said assumptions can be themselves justified in an idealized model, one would establish a bridge from idealized models to the standard model. Particularly successful examples of this "layered" approach to security include universal computational extractors for hash functions [5], and the uber-assumption family for cryptographic group schemes [2, 18, 32].

1.2 Contributions

We continue the above lines of work. Our focus is on identifying appropriate assumptions for lifting the security of group-based cryptosystems established in idealized computational models to the standard model.

In more detail, we are interested in *knowledge assumptions* on computational group schemes. In contrast to standard computational and decisional problems, in knowledge assumptions one demands the existence of a successful *extractor* for each adversary. Thus, these assumptions have a higher "logical complexity" and in particular are not unconditionally falsifiable; see [41,57] for further discussions.

Bridging assumptions for knowledge-type properties, such as the knowledge soundness of SNARKs, are an important and somewhat neglected area of investigation. Some schemes, e.g., Groth10 [42], Pinocchio [60], Groth–Maller [44], and Marlin [24] are proven under dedicated knowledge assumption. However, the most popular schemes are proven directly in the GGM and AGM [38, 43]. Besides SNARKs, knowledge assumptions also underlie the security of many other classical and modern cryptosystems, ranging from zero-knowledge proofs [6, 53] to plaintext-aware encryption [30], extractable collision resistant-hash functions (CRHFs) [13, 14, 52] and non-malleable codes [52].

The uber-knowledge family. To bridge this gap, we formulate the *uber-knowledge* (UK) assumption, an umbrella term for a class of assumptions formulated in both simple and bilinear groups. Roughly speaking, the UK assumption states that whenever an adversary outputs group elements that satisfy a certain polynomial relation with its inputs, it must necessarily produce these elements as a known linear combination of the group elements that it receives.

Specific instances of the UK assumption have already appeared in the literature. Examples include the knowledge-of-exponent assumptions (KEA1 and KEA3), which have been used to build efficient 3-round ZK protocols [6, 46]

[6] In more detail, these results (ab)use the fact that concrete hash functions and computational group schemes have compact representations, whereas exponentially large random objects do not.

and plaintext-aware encryption [12], the d-KEA assumption used to build extractable CRHFs, the d-PKE assumption used in [42, 60] to build SNARKs, and our novel d-KZG assumption justifying the extractability of polynomial commitments, and thus via the polynomial interactive oracle proofs (PIOPs) framework, the knowledge-soundness of a number of practical and in-use SNARKs [21, 38].

The UK assumption can be seen as an extension of the classical uber family to knowledge assumptions, and also as a *standard-model* counterpart to the "representation extractability" property that the AGM idealizes. We emphasize that the UK assumption is a standard-model assumption[7], and thus an adversary may exploit hashing and other "oblivious sampling" procedures to break it.

GGM AND AGM WITH HASHING. In continuing with the aforementioned layered approach to security, we set out to justify the soundness of the UK assumption in idealized models. The (bilinear) GGM and (bilinear) AGM are natural choices for such proofs. However, in their standard forms, the GGM and the AGM do not faithfully model the adversarial capability to hash into groups. At first this might not seem like a critical shortcoming, as hashing can be simulated by exponentiating to random exponents. This is indeed a valid approach for showing equivalence for standard computational or decisional problems in models with and without hashing. For knowledge assumptions, on the other hand, the story is somewhat different. Indeed, an extractor algorithm in the UK assumption is run with the adversary's view to output the linear representation mentioned above. This view, when run with a simulated hash oracle, includes the corresponding discrete logarithms, information that is missing when run with a hashing oracle. This, in turn, prevents an analogous equivalence to go through. Even more concretely, consider the knowledge assumption that posits that "no adversary can produce a valid group element without knowing its discrete logarithm." This assumption is trivially false when one can hash into a group, but holds in the AGM (without hashing) and also in the GGM if group representations are from a sufficiently large random set.

Accordingly, we extend the GGM and AGM with appropriate hashing oracles and call the resulting models GGM-H and AGM-H. This extension is straightforward for GGM, though different variants arise in the bilinear setting according to which groups one can hash to. Our choices here are driven by practical pairing-friendly groups [23, Definitions 2–4], where in the type-1 and type-3 settings one can hash to all groups, but in type-2 groups one can only hash to the first source group and the target group.

For the AGM-H, we follow the recent *algebraic compilation* approach of Zhandry [67], who identified a problem with the AGM related to leaking group elements one bit at a time [69]. Using the machinery of type-safe groups, where one can only operate on abstract group handles via oracles, we formalize the bilinear AGM for all three types with hashing. Besides continuing with the layered approach, we use these formalizations to study the relation between AGM-H and

[7] Note that a *standard assumption*, which roughly means falsifiable and non-interactive, is not the same as a *standard-model assumption*, with which we mean defined without idealization or setup.

GGM-H (Appendix F). Although AGM-H security implies GGM-H security for standard games, this implication fails for extractor games. This is essentially due to the fact that random representations cannot be converted into real group representations, a conversion that is needed to build a GGM-H extractor by running a given AGM-H extractor.

Layering: GGM and AGM feasibility. Given the observations above, we set out to justify the soundness of the UK in *both* the GGM-H and the AGM-H. We do this for the class of relation polynomials (the polynomials used in the winning condition) that are linear in the variables corresponding to the group elements returned by the adversary, and also have linearly independent coefficients. Linearity ensures that the winning condition can be efficiently verified in non-bilinear groups. Linear independence, on the other hand, is both necessary and sufficient for hardness.

Our GGM-H feasibility is in fact more general, and establishes hardness for a wider class of relation polynomials that contain one quadratic term in adversarial outputs. In particular these include the polynomial relation needed to study the knowledge soundness of Groth16. Our proof follows the standard Schwartz–Zippel lemma to transition to a setting where group operation oracles are implemented with respect to formal polynomials. The technical core of the analysis is identifying under which added conditions the coefficients of monomials corresponding to hashed group elements vanish. To ensure that the coefficients related to the quadratic term are zero we require that, after substituting the equalities originating from the degree-one part into the constant term, the resulting polynomial is not zero. Afterwards, linear independence of the linear terms ensures that all other coefficients are zero.

Our AGM-H proofs embed the q-power discrete logarithm (q-DLOG) problem of Fuchsbauer, Kiltz, and Loss [36] in a UK problem instance so that a representation that is nontrivial in the group elements returned by the hash oracle can be converted into a polynomial one of whose roots is the solution to the q-DLOG problem. As mentioned, we establish hardness for linear relation polynomials with linearly independent coefficients. For polynomials with quadratic terms, however, we only prove a hardness result for the assumption that is needed for the analysis of Groth16 in type-3 groups.

It may be that deciding UK hardness in general reduces to the ideal membership problem, and thus to Gröbner-basis computation, which has a double exponential complexity in the number of input variables. Despite this, for specific classes of polynomials, sufficient conditions for the hardness of the UK can be established. Generalizing the UK to a larger class of quadratic polynomials (in particular those containing multiple quadratic terms) remains an open problem.

Standard-model lifting: AGM proof reuse. The UK assumption postulates that in certain contexts standard-model adversaries, which may use local hashing or other means, are algebraic in the classical sense *without hashing*. This observation, in turn, allows us to *lift* existing AGM security proofs to the standard model. For instance, any adversary against Groth16 can be coupled with its extractor to always output representations that, under the UK assumptions,

are all-zero for hashed group elements. This means we can reuse the already existing AGM reduction to q-DLOG for Groth16 *without* hashing to establish the standard-model security of Groth16. Similar observations apply to the knowledge soundness of, for example, KZG polynomial commitments.[8] We note that the lifting is from AGM without hashing, but our assumption is justified under the "weaker" AGM with hashing.

The only other work that we are aware of that proves statements about SNARKs similar to Groth16 in the AGM with hashing is Lipmaa [54]. However, it reproves its schemes from scratch in the extended model with hashing, and does not formalize a plausible knowledge assumption for lifting the security of Groth16 to the standard model.

FUTURE WORKS. After its initial publication [18], the uber-assumption family was extended in a series of works to hardness for rational functions [63], interactive problems [3], matrix-type problems [32], and high-entropy sources [2]. A rich set of relations between notions of hardness has also been established in these works and others [3, 62]. Similar considerations and questions naturally arise when investigating knowledge assumptions. For instance, interactive knowledge assumptions are helpful in justifying the simulation soundness of certain zero-knowledge protocols [44]. Can the reach of the UK be extended to these settings while retaining soundness in the (bilinear) GGM-H and AGM-H?

PAPER OUTLINE. In Section 2 we recall basic notation. The formal definitions of the generic-group, type-safe and algebraic-group models are given in Section 3. In this section, we also formally extend these models to those with hashing. Section 4 contains the definition of the uber-knowledge assumption as well as some specific knowledge assumptions that fall under it. In Sections 5 and 6 we prove the hardness of UK in the GGM and the AGM with hashing. We conclude in Section 7 with an example application of our assumption to Groth16.

2 Preliminaries

BASIC NOTATION. We denote by $\mathbb{Z}$ and $\mathbb{N} \coloneqq \mathbb{Z}_{\geq 1}$ the sets of integers and of natural numbers, and by $\{0,1\}^*$ the set of finite-length bit-strings. For $n \in \mathbb{N}$, we let $\mathbb{Z}_n$ be the ring of integers modulo n; if $n = p$ is prime, then $\mathbb{F}_p \coloneqq \mathbb{Z}_p$ is a field. The security parameter is denoted by λ, and its unary representation is 1^λ. Sampling from a random variable $\mathcal{X}$ is denoted $x \twoheadleftarrow \mathcal{X}$; when X is a finite set, $x \twoheadleftarrow \mathsf{X}$ means sampling from the uniform distribution over X. If A and B are sets, we write $\mathrm{Inj}(\mathsf{A}, \mathsf{B})$ for the set of injective functions from A to B. Vectors are written in boldface and, unless otherwise specified or clear from the context, their entries are numbered from 1. We use the bracket notation to represent group elements: If $\gamma = (\cdot, g, p)$ is a group of order p with fixed generator g and $a \in \mathbb{Z}_p$, then $[a] \coloneqq g^a$. Similarly, if γ is a bilinear group and $a \in \mathbb{Z}_p$,

[8] Lifting is logically different to layering: the latter takes the form Model $\implies$ Assumption $\implies$ Application; the former, on the other hand, has the form (Model $\implies_R$ Application) $\implies$ (Assumption $\implies_R$ Application). Here, R is a reduction.

then $[a]_\mu \coloneqq g_\mu^a$ ($\mu \in \{1, 2, T\}$), where g_μ is the generator of the μ-th group (we omit subscripts 1 and 2 in type-1 groups, see [39]). We extend this notation to vectors of exponents: If $\boldsymbol{a} \in \mathbb{Z}_p^\ell$, then $[\boldsymbol{a}] \coloneqq (g^{\boldsymbol{a}_i})_{i=1}^\ell$, and similarly for bilinear groups with the appropriate subscripts. Note that this notation does not imply that the algorithm producing the group element knows its discrete logarithm wrt. the fixed generator. The trace (or the view) of an algorithm $\mathcal{A}$, i.e., the vector containing all its inputs, the random coins it is run on, and potential oracle replies, is denoted $\mathsf{trace}(\mathcal{A})$.

Cryptographic games [9]. We use the code-based game-playing framework of Bellare and Rogaway. A game G is an algorithm run by several parties, among others a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. The game starts with $\mathcal{C}$ generating a challenge, which is then passed on to $\mathcal{A}$, who is tasked with solving it. To model potential leakage during the game's execution, $\mathcal{C}$ may offer $\mathcal{A}$ a set of oracles that help the adversary in finding a solution. The output of $\mathcal{A}$ is then handed back to $\mathcal{C}$, who verifies the purported solution and returns a decision bit. We say that $\mathcal{A}$ wins game G if $\mathcal{C}$'s final output is 1; we then write $\mathrm{G}^{\mathcal{A}} = 1$, or simply $\mathrm{G}^{\mathcal{A}}$. Other parties may also feature in the game, according to G's description.

Let G_1 and G_2 be two games whose code is identical except for the consequent inside one if-branch, and let Bad be the event that the boolean condition in the if-statement is triggered. Then $|\Pr[\mathrm{G}_1] - \Pr[\mathrm{G}_2]| \leq \Pr[\mathsf{Bad}]$.

Group schemes [26]. A group scheme is a randomized algorithm Γ which, on input the security parameter 1^λ, returns group parameters $\gamma = (\cdot, g, p)$ (also called group), where $\cdot$ is an efficiently computable binary function, g is an element, and $2^{\lambda-1} \leq p < 2^\lambda$ is prime. Implicit in γ is the description of a set G such that $(\mathsf{G}, \cdot)$ is a cyclic group of order p with generator $g \in \mathsf{G}$.

Bilinear group schemes [39, 49, 65]. A type-3 bilinear group scheme is a randomized algorithm B which, on input the security parameter 1^λ, returns bilinear group parameters $\gamma = (\cdot_1, g_1, \cdot_2, g_2, \cdot_T, p, e)$, where $\cdot_\mu$ ($\mu \in \{1, 2, T\}$) and e are efficiently computable binary functions, g_ν ($\nu \in \{1, 2\}$) are elements, and $2^{\lambda-1} \leq p < 2^\lambda$ is prime. Implicit in γ is the description of sets G_μ such that (1) $(\mathsf{G}_\mu, \cdot_\mu)$ is a cyclic group of order p, (2) $(\mathsf{G}_\mu, \cdot_\mu)$ is generated by g_μ, and (3) $e\colon \mathsf{G}_1 \times \mathsf{G}_2 \to \mathsf{G}_T$ satisfies $e([a]_1, [b]_2) = e([1]_1, [1]_2)^{ab}$ for all $a, b \in \mathbb{Z}_p$, and $g_T \coloneqq e([1]_1, [1]_2) \neq [0]_T$. Note that $[0]_T$ is the identity element 1_{G_T} of G_T.

A type-2 bilinear group scheme is a type-3 scheme where γ also contains an efficiently computable group homomorphism $\psi\colon \mathsf{G}_2 \to \mathsf{G}_1$ satisfying $\psi(g_2) = g_1$.

A type-1 bilinear group scheme is a type-3 scheme where $\mathsf{G}_1 = \mathsf{G}_2$, $\cdot_1 = \cdot_2$ and $g_1 = g_2$. Accordingly, we will drop subscripts and repeating entries from γ. We will also often omit the index μ in $\cdot_\mu$ when no confusion arises.

Schwartz–Zippel lemma [28, 64, 70]. Below we recall the Schwartz–Zippel lemma, a simple yet powerful tool to bound the probability of finding a root of a non-zero (multivariate) polynomial when evaluating it at a random point. We present a game-based version of the lemma, similar to [2, Lemma 2]. Recall that the degree of a multivariate monomial is the sum of the exponents of the variables

Game $\mathrm{SZ}^{\mathcal{A}}_{\mathbb{F},\mathsf{S},k,\boldsymbol{d}}$:	Game q-$\mathrm{DLOG}^{\mathcal{A}}_{\Gamma}(\lambda)$:
$(P_1,\dots,P_\ell) \twoheadleftarrow \mathcal{A}$; $\boldsymbol{s} \twoheadleftarrow \mathsf{S}^k$ return $\big((\exists 1 \le i < j \le \ell)$ $(P_i(\boldsymbol{s}) = P_j(\boldsymbol{s})) \wedge (P_i \neq P_j)\big)$	$\gamma \twoheadleftarrow \Gamma(1^\lambda)$; $x \twoheadleftarrow \mathbb{Z}_p$ $\boldsymbol{x} \leftarrow (x, x^2, \dots, x^{q(\lambda)})$; $x' \twoheadleftarrow \mathcal{A}(\gamma, [\boldsymbol{x}])$ return $(x = x')$

Game (q_1,q_2)-$\mathrm{DLOG}^{\mathcal{A}}_{\mathrm{B}}(\lambda)$:

$\gamma \twoheadleftarrow \mathrm{B}(1^\lambda)$; $x \twoheadleftarrow \mathbb{Z}_p$; $\boldsymbol{x}_1 \leftarrow (x, x^2, \dots, x^{q_1(\lambda)})$; $\boldsymbol{x}_2 \leftarrow (x, x^2, \dots, x^{q_2(\lambda)})$
$x' \twoheadleftarrow \mathcal{A}(\gamma, [\boldsymbol{x}_1]_1, [\boldsymbol{x}_2]_2)$; return $(x = x')$

Fig. 1. Top left: The Schwartz–Zippel game for a field $\mathbb{F}$, a finite subset $\mathsf{S} \subseteq \mathbb{F}$, and $k, q \in \mathbb{N}$, $\boldsymbol{d} \in \mathbb{N}^q$. **Top right:** The q-DLOG game for a group scheme Γ. **Bottom:** The (q_1,q_2)-DLOG game for a type-3 bilinear group scheme B.

appearing in the monomial, and the total degree of a multivariate polynomial is the maximum degree of its monomials.

Lemma 1 (Schwartz–Zippel). *Let $k, q \in \mathbb{N}$, $\boldsymbol{d} \in \mathbb{N}^q$, $\mathbb{F}$ a field and $\mathsf{S} \subseteq \mathbb{F}$ a finite subset of $\mathbb{F}$. Consider an adversary $\mathcal{A}$ returning at most q polynomials in $\mathbb{F}[X_1,\dots,X_k]$, where the i-th polynomial has total degree at most $\boldsymbol{d}_i$. Then*

$$\mathrm{Adv}^{\mathrm{sz}}_{\mathbb{F},\mathsf{S},k,\boldsymbol{d},\mathcal{A}} \coloneqq \Pr[\mathrm{SZ}^{\mathcal{A}}_{\mathbb{F},\mathsf{S},k,\boldsymbol{d}}] \le \sum_{1 \le i < j \le q} \frac{\max(\boldsymbol{d}_i, \boldsymbol{d}_j)}{|\mathsf{S}|} \le \frac{q^2 \max(\boldsymbol{d})}{2|\mathsf{S}|}\,,$$

where the game SZ *is defined in Fig. 1 (top left).*

Bauer–Fuchsbauer–Loss lemma [3]. We also recall a technical lemma due to Bauer, Fuchsbauer, and Loss regarding the leading term of a polynomial after variable substitutions.

Lemma 2 (Bauer–Fuchsbauer–Loss). *Let $m, d \in \mathbb{N}$, $\mathbb{F}$ be a finite field, and $P \in \mathbb{F}[X_1,\dots,X_m]$ a polynomial of total degree d. Define the polynomial $Q(Z) \in (\mathbb{F}[Y_1,\dots,Y_m,V_1,\dots,V_m])(Z)$ as $Q(Z) \coloneqq P(Y_1Z+V_1,\dots,Y_mZ+V_m)$. Then the leading coefficient of Q is a polynomial in $\mathbb{F}[Y_1,\dots,Y_m]$ of degree d.*

q-DLOG [36]. Let Γ be a group scheme and $q\colon \mathbb{N} \to \mathbb{N}$ a polynomial. We define the advantage of an adversary $\mathcal{A}$ in the q-DLOG game for Γ as

$$\mathrm{Adv}^{q\text{-dlog}}_{\Gamma,\mathcal{A}}(\lambda) \coloneqq \Pr[q\text{-}\mathrm{DLOG}^{\mathcal{A}}_{\Gamma}(\lambda)]\,,$$

where the game q-DLOG is defined in Fig. 1 (top right). We say that q-DLOG holds for Γ if for every PPT adversary $\mathcal{A}$, $\mathrm{Adv}^{q\text{-dlog}}_{\Gamma,\mathcal{A}}$ is negligible. When q is the constant polynomial $q = 1$, we simply write $\mathrm{DLOG} \coloneqq 1\text{-DLOG}$.

(q_1,q_2)-DLOG [3]. Let B be a type-3 bilinear group scheme and $q_1, q_2\colon \mathbb{N} \to \mathbb{N}$ polynomials. We define the advantage of an adversary $\mathcal{A}$ in the (q_1,q_2)-DLOG game for B as

$$\mathrm{Adv}^{(q_1,q_2)\text{-dlog}}_{\mathrm{B},\mathcal{A}}(\lambda) \coloneqq \Pr[(q_1,q_2)\text{-}\mathrm{DLOG}^{\mathcal{A}}_{\mathrm{B}}(\lambda)]\,,$$

where the game (q_1, q_2)-DLOG is defined in Fig. 1 (bottom). We say that (q_1, q_2)-DLOG holds for B if for every PPT $\mathcal{A}$, $\mathrm{Adv}^{(q_1,q_2)\text{-dlog}}_{\Gamma,\mathcal{A}}$ is negligible. (q_1, q_2)-DLOG for type-2 and type-1 bilinear group schemes are defined similarly.

Berlekamp's algorithm [10]. Berlekamp's algorithm is a well-known algorithm for factoring polynomials (thus in particular for finding their roots) over finite fields. We denote by $\mathsf{BerlekampRoot}$ the algorithm which takes a prime $p \in \mathbb{N}$ and a polynomial $P \in \mathbb{Z}_p[X]$ as input, and returns the set S of roots of P in $\mathbb{Z}_p$.

3 Generic, Type-Safe, and Algebraic Groups

Proving the hardness of interesting computational problems for groups is currently out of reach. Instead, one attempts to obtain guarantees on the soundness of hardness assumptions in restricted models of computation. Shoup's generic-group model, Maurer's type-safe model, and the algebraic-group model are popular idealized/restricted models, which we recall in this section. We begin with the formal definition of the GGM.

Generic-group model (GGM) [58, 66]. Consider a prime p and a finite set $\mathsf{G} \subseteq \{0,1\}^*$ with $|\mathsf{G}| = p$. Notice that every $\tau \in \mathrm{Inj}(\mathbb{Z}_p, \mathsf{G})$ defines an associated operation $\mathsf{op}\colon (\mathbb{Z}_p \times \mathsf{G})^2 \to \mathsf{G}$ via $\mathsf{op}(a, h, b, h') := \tau(a\tau^{-1}(h) + b\tau^{-1}(h'))$.[9] Under this operation, G becomes a cyclic group of order p with generator $\tau(1)$.

The generic-group model with parameters (p, G) is a model of computation which idealizes interactions of algorithms with cyclic groups of order p: First, the challenger samples a random encoding $\tau \in \mathrm{Inj}(\mathbb{Z}_p, \mathsf{G})$. Then all parties, honest or otherwise, are run on $\tau(1)$ and potentially other application-specific inputs, and interact with the set G of labels in place of the real group. To perform group operations, algorithms are given oracle access to the operation op defined by τ.

As mentioned in the introduction, certain types of group-based extractor games can only be won given the ability to hash strings into the group, a property that many real-world groups have. To mirror this capability, we extend the GGM with an appropriate hashing oracle.

GGM with hashing (GGM-H) [4, 20]. We define the GGM-H with parameters (p, G) as the GGM above, except that besides sampling τ, the challenger also picks a random function $H\colon \{0,1\}^* \to \mathbb{Z}_p$, and offers $\mathsf{H}(m) := \tau(H(m))$ as an additional oracle to all parties.[10]

Following the approach taken for simple groups, we now recall the idealized models pertaining bilinear groups. In essence, each group is idealized independently as before, and the pairing (and potential homomorphism) is defined by the

[9] As a mathematical shorthand, we call such a term that pulls back h and h' using τ^{-1} to perform an operation and then push the result back forward using τ a *pushforward*.

[10] An alternative definition of hashing would simply pick a random r and then return $\tau(r)$. In contrast to the previous definition, this definition does not allow adversaries to *reproduce* hash values.

sampled encodings via pushforward. Just as for the GGM, we then extend these models to account for an adversary's capability to hash into any of the groups.

Generic-bilinear-group model (GBM$\{1,2,3\}$) [17,68]. Consider a prime p and finite sets G_μ ($\mu \in \{1,2,T\}$) with $|\mathsf{G}_\mu| \geq p$. Given functions $\tau_\mu \in \mathrm{Inj}(\mathbb{Z}_p, \mathsf{G}_\mu)$, one can define operations op_μ as in the GGM. Additionally, encodings τ_μ define a map $\mathsf{e}\colon \mathsf{G}_1 \times \mathsf{G}_2 \to \mathsf{G}_T$ given by $\mathsf{e}(h_1, h_2) \coloneqq \tau_T(\tau_1^{-1}(h_1)\tau_2^{-1}(h_2))$.

Similarly to the case of simple groups, the type-3 generic-bilinear-group model (GBM3) with parameters $(p, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_T)$ is a model of computation which abstracts interactions of algorithms with type-3 schemes. Concretely, the challenger first samples random encodings $\tau_\mu \in \mathrm{Inj}(\mathbb{Z}_p, \mathsf{G}_\mu)$. Then all algorithms are run on input $(\tau_1(1), \tau_2(1))$ and potentially encodings of other application-specific elements. In place of the real operations $\cdot_\mu$ and e, algorithms are given oracle access to their ideal counterparts op_μ and e to interact with labels in G_μ.

The GBM2 with parameters $(p, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_T)$ is defined analogously, except that it also idealizes ψ. More precisely, in addition to op_μ and e, in GBM2 the challenger also gives all parties oracle access to $\psi\colon \mathsf{G}_2 \to \mathsf{G}_1$ defined as $\psi(h_2) \coloneqq \tau_1(\tau_2^{-1}(h_2))$.

Likewise, GBM1 with parameters $(p, \mathsf{G}, \mathsf{G}_T)$ is defined as GBM3, but target sets G_1 and G_2 as well as encodings τ_1 and τ_2 are taken to coincide (i.e., $\mathsf{G} \coloneqq \mathsf{G}_1 = \mathsf{G}_2$ and $\tau \coloneqq \tau_1 = \tau_2$). To ease notation, we let $\mathbf{G} \coloneqq (\mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_T)$ in GBMk ($k \in \{2,3\}$), and $\mathbf{G} \coloneqq (\mathsf{G}, \mathsf{G}_T)$ in GBM1.

GBM with hashing [54]. The GBM3-H with parameters $(p, \mathbf{G})$ is defined as GBM3, but besides sampling τ_μ, $\mu \in \{1,2,T\}$, the challenger also picks *independent* $H_\mu\colon \{0,1\}^* \to \mathrm{Rng}(\tau_\mu)$ at random, and defines H_μ as in GGM-H, using encoding τ_μ and function H_μ. It then gives H_μ as an additional oracle to all parties.

The GBM2-H with parameters $(p, \mathbf{G})$ is defined as GBM3-H above, starting from GBM2, but the oracle H_2 is withheld [23].

Finally, the GBM1-H with parameters $(p, \mathbf{G})$ is defined as GBM3-H above, starting from GBM1, except that the challenger samples only one random function $H\colon \{0,1\}^* \to \mathbb{Z}_p$ for both source groups, since these coincide.

An alternative generic model of computation for groups was introduced by Maurer [56], which replaces group elements with abstract handles representing memory locations where the elements are stored. This model has recently been recast by Zhandry [67] as the type-safe model (TSM).[11] We next recall the TSM, but instead of using the language of circuits (as done in [67]), we provide an oracle-based formalization. Similarly to Shoup's GGM, we then extend the TSM to allow any party to hash strings of their choice into the idealized group.

[11] The main difference between the two models is that in the TSM, when querying their oracles, parties cannot choose the memory location (handle) where the result is stored, and they cannot access handles they are not explicitly given either at the outset or as an oracle reply. This avoids certain unnatural problems that arise when analyzing security games in Maurer's model.

TYPE-SAFE MODEL (TSM) [56, 67]. Let p be a prime. In the type-safe model with parameter p, the challenger first initializes an empty table T and a global counter $C = 1$, and sets $T[C] \coloneqq 1$. Here C plays the role of an abstract handle to a memory location where a group element is stored. Then all parties are run on input memory location 1 and potentially other application-specific handles prepared in advance. To interact with the group, the challenger offers an oracle op implemented as follows. When a party queries op on $(a, c, b, c') \in (\mathbb{Z}_p \times \mathbb{N})^2$, the challenger increments C, stores $aT[c]+bT[c']$ in $T[C]$, and returns C. Note that in contrast to Maurer's model, and in line with Zhandry's TSM, it is the challenger rather than the adversary who picks new counter values, and these values are always fresh. Additionally, all parties are given an equality oracle eq, which when queried on $(c, c') \in \mathbb{N}^2$, returns the bit $(T[c] = T[c'])$.

When calling their oracles, parties are restricted to querying only counters they have received as input or seen as response to a prior query to op. In Zhandry's circuit model, this corresponds to parties applying gates only to wires they are given (see [67]). Note that in the TSM all computation related to counters/handles is performed via oracle queries, and local computation on handles is not allowed (i.e., it does not "type-check"). As for the query complexity metrics, queries to op incur unit cost, while queries to eq are free.

TSM WITH HASHING. We define the TSM-H with parameter p as the TSM above, except that after initializing T and C, the challenger also samples a random function $H\colon \{0,1\}^* \to \mathbb{Z}_p$. In addition to oracles op and eq, the challenger then also offers all parties an oracle H defined as follows. When a party queries H on $m \in \{0,1\}^*$, the challenger increments C, stores $H(m)$ in $T[C]$ and returns C.

We now extend the TSM to the bilinear setting, and then add hashing oracles to allow an adversary to hash into the various groups. To do so, we proceed as for the GBM, but start from the TSM rather than Shoup's GGM.

BILINEAR-TYPE-SAFE MODEL (BTM$\{1,2,3\}$). Let p be a prime. In the type-3 bilinear-type-safe model (BTM3) with parameter p, the challenger first initializes tables T_1, T_2 and T_T, as well as global counters $C_1 = C_2 = C_T = 1$, and sets $T_1[C_1] \coloneqq T_2[C_2] \coloneqq T_T[C_T] \coloneqq 1$. Then all parties are run on input memory locations $(1, 1, 1)$, and potentially other application-specific handles, and are given oracles op_μ, eq_μ $(\mu \in \{1, 2, T\})$ and e. Here, op_μ and eq_μ are implemented as in the simple TSM, each using table T_μ and counter C_μ. On the other hand, when a party queries e on $(c_1, c_2) \in \mathbb{N}^2$, the challenger increments C_T, stores $T_1[c_1]T_2[c_2]$ in $T_T[C_T]$ and returns C_T.

The BTM2 with parameter p is defined analogously, except that it also offers all algorithms an oracle $\boldsymbol{\psi}$ for ψ defined as follows. When a party queries $\boldsymbol{\psi}$ on $c_2 \in \mathbb{N}$, the challenger increments C_1, stores $T_2[c_2]$ in $T_1[C_1]$, and returns C_1.

Likewise, BTM1 with parameter p is defined as BTM3, but tables T_1 and T_2 as well as counters C_1 and C_2 are taken to coincide.

In each of the models above parties are restricted to querying, for every group, only counters they got as input or seen as response to a prior query to op_μ or e.

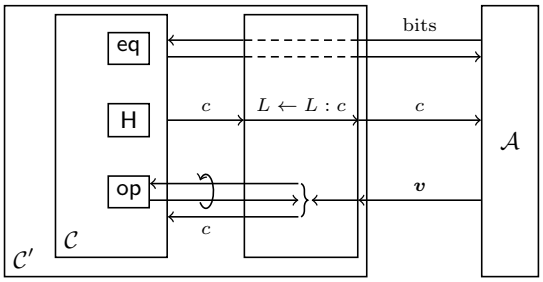

Fig. 2. Representation of the algebraic compilation of a type-safe game G. Here, $\mathcal{C}$ denotes the challenger of G, and $\mathcal{C}'$ the challenger of AC(G). The unlabeled box inside $\mathcal{C}'$ represents the compiler converting $\mathcal{A}$ into an adversary for G.

BTM WITH HASHING. The BTM3-H with parameter p is defined as BTM3, but besides initializing T_μ and C_μ, $\mu \in \{1, 2, T\}$, the challenger also samples random functions $H_\mu \colon \{0,1\}^* \to \mathbb{Z}_p$. It then additionally offers all parties an oracle H_μ defined as in TSM-H using table T_μ and counter C_μ.

The BTM2-H with parameter p is defined as BTM3-H above, starting from BTM2, but the oracle H_2 is withheld [23].

Finally, the BTM1-H with parameter p is defined as BTM3-H above, starting from BTM1, except that the challenger samples only one random function $H \colon \{0,1\}^* \to \mathbb{Z}_p$ and implements only one oracle H for both source groups.

The TSM and BTM provide an adequate setting to define the algebraic group model (AGM), where adversaries are restricted to being algebraic but have full access to the standard-model group considered in the game (rather than an idealized version as in the previous models). Algebraic algorithms, first studied in [19, 59] and later revisited in [36], are required to "explain" any group element they return in terms of elements they have received as input, either at the outset or through oracles. We follow Zhandry [67] in defining the AGM as a compiler for type-safe games, which allows to sidestep issues regarding the validity of the model (see also [69]). As usual, we then extend the AGM with a hashing oracle.

ALGEBRAIC COMPILATION. Given a game G in the TSM with parameter p, we define the algebraic compilation of G as the game AC(G) in the same model that operates as follows. Game AC(G) initializes a list $L = [\,]$ and then runs G. Whenever G outputs a counter to the adversary, AC(G) keeps track of it by first appending it to L and then forwarding it to the adversary. Whenever G expects a counter as input from the adversary, AC(G) instead takes a vector $\boldsymbol{v} \in \mathbb{Z}_p^\ell$, where $\ell = |L|$. Game AC(G) then uses the current state of the list $L = (c_1, \ldots, c_\ell)$, the vector $\boldsymbol{v}$, and the group operation oracle op of G to compute a counter value c such that $T[c] = \sum_{i=1}^{\ell} T[c_i]\boldsymbol{v}_i$, and then forwards c to G (see Fig. 2). Any output from G that is not a counter is forwarded to the adversary, and similarly any input from the adversary that is not a counter is forwarded to G. We call a game G′ *algebraic* if G′ = AC(G) for some game G in the TSM.

GROUP COMPILATION. Let G be a game in the TSM with parameter p, and $\gamma := (\cdot, g, p)$ be a group. The group compilation of G with respect to γ is the game $\mathrm{GC}(\mathrm{G}, \gamma)$ that operates as G with the following modifications. For every counter value c defined by G, $\mathrm{GC}(\mathrm{G}, \gamma)$ instead considers the group element $[T[c]]$, and whenever G queries $\mathsf{op}(c, c')$ or $\mathsf{eq}(c, c')$, $\mathrm{GC}(\mathrm{G}, \gamma)$ computes $[T[c]] \cdot [T[c']]$ and $([T[c]] = [T[c']])$, respectively. The adversary is initially provided with γ and the group elements corresponding to the counters it is given in G, and no longer receives oracles op and eq. Whenever G sends a counter value c to the adversary, $\mathrm{GC}(\mathrm{G}, \gamma)$ instead sends $[T[c]]$. (By type-safety c is an already existing counter.) Vice versa, whenever G expects a counter from the adversary, $\mathrm{GC}(\mathrm{G}, \gamma)$ accepts a group element instead. This is equivalent to picking a new counter value and forwarding to the type-safe game. Any other communication between G and the adversary is relayed as before.

ALGEBRAIC-GROUP MODEL (AGM) [19, 36, 67]. The algebraic group model is a framework to study type-safe games in the standard model. More precisely, studying a type-safe game G' in the AGM with respect to a group $\gamma := (\cdot, g, p)$ is defined as analyzing the game $\mathrm{GC}(\mathrm{AC}(\mathrm{G}'), \gamma)$. Note that with this definition, one can talk about a standard-model game G in the AGM only if G is first identified as the group compilation $\mathrm{GC}(\mathrm{G}', \gamma)$ of a type-safe game G'.

We now similarly define the AGM with hashing, which was already informally introduced by Fuchsbauer, Kiltz and Loss [36] and further studied by Lipmaa [54], in the type-safe game framework of [67].

AGM WITH HASHING. Given a game G in the TSM-H with parameter p, its algebraic compilation AC(G) is defined as before, except that for every query to oracle H, the returned counter value c is also added to the list L. Accordingly, an adversary can also use the counters obtained through the hashing oracle to construct new group elements. The group compilation $\mathrm{GC}(\mathrm{G}, \gamma)$ of G with respect to a group γ is defined as before, except that oracle H is still offered to all algorithms. Whenever any party queries H on a message $m \in \{0,1\}^*$, the game returns the group element $[H(m)]$ in place of the corresponding counter. Notice that $\mathrm{GC}(\mathrm{G}, \gamma)$ is a game in the random-oracle model.

With the definitions above, studying a TSM-H game G' in the AGM with respect to a group γ is defined as analyzing the game $\mathrm{GC}(\mathrm{AC}(\mathrm{G}'), \gamma)$. Again, one can talk about a random-oracle-model game G in the AGM only if G is first identified as $\mathrm{GC}(\mathrm{G}', \gamma)$ for a TSM-H game G'.

We conclude our overview of idealized models by defining a bilinear version of the AGM. We also add a hashing oracle for each group idealized in the model.

BILINEAR ALGEBRAIC COMPILATIONS. Let p be a prime, and consider a game G in the BTMk with parameter p. Bilinear algebraic compilation AC(G) is defined similarly to standard algebraic compilation, with the following differences.

If $k = 3$, AC(G) now maintains three initially empty lists L_1, L_2 and L_T to keep track of the counters output by the game to the adversary. Whenever G expects a value c_ν of counter C_ν ($\nu \in \{1, 2\}$) from the adversary, AC(G) instead takes a vector $\boldsymbol{v} \in \mathbb{Z}_p^{\ell_\nu}$, where $\ell_\nu = |L_\nu|$. Game AC(G) then uses the current state

of the list $L_\nu = (c_{\nu,1}, \ldots, c_{\nu,\ell_\nu})$, $\boldsymbol{v}$ and oracle op_ν to compute a counter value c_ν so that $T_\nu[c_\nu] = \sum_{i=1}^{\ell_\nu} \boldsymbol{v}_i T_\nu[c_{\nu,i}]$, and then forwards c_ν to G. Similarly, whenever G expects a value c_T of counter C_T from the adversary, AC(G) instead takes a matrix $\boldsymbol{m} \in \mathbb{Z}_p^{\ell_1 \times \ell_2}$ and a vector $\boldsymbol{v} \in \mathbb{Z}_p^{\ell_T}$. Game AC(G) then uses the current state of the lists L_μ ($\mu \in \{1, 2, T\}$), $\boldsymbol{m}$, $\boldsymbol{v}$, and oracles e and op_T to compute a counter value c_T so that $T_T[c_T] = \sum_{i=1}^{\ell_1} \sum_{j=1}^{\ell_2} \boldsymbol{m}_{ij} T_1[c_{1,i}] T_2[c_{2,j}] + \sum_{t=1}^{\ell_T} \boldsymbol{v}_t T_T[c_{T,t}]$, and then forwards c_T to G. Other inputs and outputs are relayed.

If $k = 2$, AC(G) is defined similarly, but we must account for the additional oracle ψ. Accordingly, whenever G expects a value c_1 of counter C_1 from the adversary, AC(G) instead takes vectors $\boldsymbol{v} \in \mathbb{Z}_p^{\ell_1}$ and $\boldsymbol{w} \in \mathbb{Z}_p^{\ell_2}$. Game AC(G) then uses the current state of the lists L_ν ($\nu \in \{1, 2\}$), $\boldsymbol{v}$, $\boldsymbol{w}$, and the oracles op_1 and ψ to compute a counter value c_1 so that $T_1[c_1] = \sum_{i=1}^{\ell_1} \boldsymbol{v}_i T_1[c_{1,i}] + \sum_{j=1}^{\ell_2} \boldsymbol{w}_j T_2[c_{2,j}]$, and then forwards c_1 to G. Similarly, whenever G expects a value c_T of counter C_T from the adversary, AC(G) instead takes matrices $\boldsymbol{m} \in \mathbb{Z}_p^{\ell_1 \times \ell_2}$ and $\boldsymbol{n} \in \mathbb{Z}_p^{\ell_2 \times \ell_2}$, and a vector $\boldsymbol{v} \in \mathbb{Z}_p^{\ell_T}$. Game AC(G) then uses the current state of the lists L_μ ($\mu \in \{1, 2, T\}$), $\boldsymbol{m}$, $\boldsymbol{n}$, $\boldsymbol{v}$, and oracles e, ψ and op_T to compute a counter value c_T so that $T_T[c_T] = \sum_{i=1}^{\ell_1} \sum_{j=1}^{\ell_2} \boldsymbol{m}_{ij} T_1[c_{1,i}] T_2[c_{2,j}] + \sum_{i,j=1}^{\ell_2} \boldsymbol{n}_{ij} T_2[c_{2,i}] T_2[c_{2,j}] + \sum_{t=1}^{\ell_T} \boldsymbol{v}_t T_T[c_{T,t}]$, and then forwards c_T to G.

If $k = 1$, AC(G) is defined as for $k = 3$, but now lists L_1 and L_2 coincide.

If γ is a type-k bilinear group, we define GC(G, γ) as for simple groups: Each group in G is instantiated with the corresponding parameters in γ as discussed earlier, and whenever G queries $\mathsf{e}(c_1, c_2)$ (or $\psi(c_2)$, if $k = 2$) for some counter values c_1 and c_2, GC(G, γ) computes $e([T_1[c_1]]_1, [T_2[c_2]]_2)$ (resp., $\psi([T_2[c_2]]_2)$).

Algebraic-bilinear-group model (ABM). Consider a game G′ in the BTMk with parameter p and a type-k bilinear group γ. As before, studying G′ in the ABM with respect to γ is defined as analyzing GC(AC(G′), γ).

ABM with hashing. For a game G in the BTMk-H with parameter p, its algebraic compilation AC(G) is defined as for the BTMk, except that for every query to oracle H_μ (if present), the returned counter value c_μ is also added to the list L_μ ($\mu \in \{1, 2, T\}$). The bilinear group compilation GC(G, γ) of G with respect to a type-k bilinear group γ is also defined as before, except that the challenger still offers oracles H_μ to all parties. Whenever any party queries H_μ on a message $m \in \{0,1\}^*$, the game returns the group element $[H_\mu(m)]_\mu$ in place of the corresponding counter.

With the definitions above, studying a BTMk-H game G′ in the ABM with respect to a type-k bilinear group γ is defined as analyzing GC(AC(G′), γ).

In Appendix F we study the relations between different models for standard and extractor games. Our treatment follows that of Zhandry [67] with the modifications that we consider the Turing machine model for structured type-safe games, a fixed set of group representations and the presence of a hashing oracle. There, we show that security with respect to type-safe and random-representation groups are equivalent. We also formally prove folklore belief that

AGM security implies RR security for standard (non-extractor) games. We summarize these results next.

Proposition 1 (Relations). *Let* $\mathrm{G}_{\mathsf{TS},p}$ *be a standard single-stage type-safe game with hashing for a group of size* p*. Let* G *be a group, also viewed as a representation set, be of size* p*. Let* $\mathrm{G}_{\mathsf{Alg},p,\mathsf{G}}$ *be the* G*-algebraic compilation of* $\mathrm{G}_{\mathsf{TS},p}$*, and let* $\mathrm{G}_{\mathsf{RR},p,\mathsf{G}}$ *be its compilation with respect to random representations. Then security in* $\mathrm{G}_{\mathsf{Alg},p,\mathsf{G}}$ *implies security in* $\mathrm{G}_{\mathsf{RR},p,\mathsf{G}}$*, and furthermore the latter is equivalent to security in* $\mathrm{G}_{\mathsf{RR},p}$*.*

A natural way to extend Theorem 9 (AGM-H $\implies$ GGM-H) in Appendix F to extractor games is as follows. Given a GGM-H adversary, we first convert it to an AGM-H adversary. We are now given a view in GGM-H and an AGM-H extractor (by our assumption). We need to convert the GGM-H view to an AGM-H view so as to be able to run the AGM-H extractor. This, however, does not seem possible: random group representations need to be converted to the same real group elements which the AGM-H adversary was run on.

A similar problem arises when attempting to extend Theorem 10 (GGM-H $\implies$ TSM-H) in Appendix F to extractor games. We are given a TSM-H view and need to convert it to a GGM-H view. However, in this conversion we do not have access to the random representations that the GGM-H adversary was run on.

The intuitive reason for these failures is that AGM-H adversaries have a "richer view" (the real group elements) compared to GGM-H adversaries (random group elements), and the latter cannot be converted to the former. Similarly GGM-H adversaries have a richer view compared to abstract counters that TSM-H adversaries see.

4 The Uber-Knowledge Assumption

Knowledge adversaries, sources, and extractors. A knowledge adversary is a two-stage algorithm $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, where (1) $\mathcal{A}_0$ takes group parameters $\gamma = (\cdot, g, p)$ as input and returns a DPT algorithm R and state information st, and (2) $\mathcal{A}_1$ takes a vector of group elements $[\boldsymbol{x}]$, a vector $\boldsymbol{a}$ in $\mathbb{Z}_p$, and the state of $\mathcal{A}_0$, and returns a vector of group elements $[\boldsymbol{y}]$ and a vector $\boldsymbol{b}$ in $\mathbb{Z}_p$. A knowledge source is an algorithm $\mathcal{S}$ taking as input the state returned by $\mathcal{A}_0$, and returning vectors $\boldsymbol{x}$ and $\boldsymbol{a}$ from $\mathbb{Z}_p$. A knowledge extractor (for $\mathcal{A}$) is an algorithm $\mathcal{E}$ which takes as input the trace of an execution of $\mathcal{A}$, and returns a vector (or matrix) $\boldsymbol{w}$ of elements from $\mathbb{Z}_p$.

If γ is a type-2 or type-3 bilinear group, $\mathcal{S}$ returns four vectors in $\mathbb{Z}_p$, one for each group G_μ ($\mu \in \{1, 2, T\}$) and one in the clear, and $\mathcal{A}_1$ returns three vectors of group elements, one from each G_μ. The additional inputs of $\mathcal{A}$ are adjusted accordingly. In type-1 groups, the vectors for G_1 and G_2 coincide.

Remark. The algorithm R returned by $\mathcal{A}_0$ is intended to implement the winning condition of the knowledge assumption (KA) game (see below), taking the outputs of $\mathcal{S}$, $\mathcal{A}_1$ and $\mathcal{E}$, and returning a decision bit. One could define R to take

Game $\mathrm{KA}^{\mathcal{A}}_{\Gamma,\mathcal{S},\mathcal{E}}(\lambda)$:	Game $\mathrm{UK}^{\mathcal{A}}_{\Gamma,\mathcal{S},\mathcal{E}}(\lambda)$:
$\gamma \twoheadleftarrow \Gamma(1^\lambda)$ $(\mathsf{R}, st) \twoheadleftarrow \mathcal{A}_0(\gamma; r_{\mathcal{A}})$ $(\boldsymbol{x}, \boldsymbol{a}) \twoheadleftarrow \mathcal{S}(st)$ $([\boldsymbol{y}], \boldsymbol{b}) \twoheadleftarrow \mathcal{A}_1([\boldsymbol{x}], \boldsymbol{a}, st; r_{\mathcal{A}})$ $\mathsf{trace}(\mathcal{A}) \leftarrow (r_{\mathcal{A}}, \gamma, [\boldsymbol{x}], \boldsymbol{a})$ $\boldsymbol{w} \twoheadleftarrow \mathcal{E}(\mathsf{trace}(\mathcal{A}))$ return $\mathsf{R}(\boldsymbol{x}, [\boldsymbol{y}], \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{w})$	$\gamma \twoheadleftarrow \Gamma(1^\lambda)$; $(Q, \boldsymbol{P}) \twoheadleftarrow \mathcal{A}_0(\gamma; r_{\mathcal{A}})$ $\boldsymbol{x} \twoheadleftarrow \mathcal{S}(\gamma, Q, \boldsymbol{P})$; $\boldsymbol{x}_0 \leftarrow 1$ $([\boldsymbol{y}], \boldsymbol{c}) \twoheadleftarrow \mathcal{A}_1(\gamma, Q, \boldsymbol{P}, [\boldsymbol{x}]; r_{\mathcal{A}})$ $\mathsf{trace}(\mathcal{A}) \leftarrow (r_{\mathcal{A}}, \gamma, [\boldsymbol{x}])$; $\boldsymbol{w} \twoheadleftarrow \mathcal{E}(\mathsf{trace}(\mathcal{A}))$ return $(Q(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{c}) \neq 0 \bmod p)$ $\quad \wedge ([Q(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c})] = [0])$ $\quad \wedge \big((\exists 1 \leq i \leq n)([\boldsymbol{y}_i] \neq \prod_{j=0}^{m} [\boldsymbol{w}_{ij} \boldsymbol{x}_j])\big)$
Game $\mathrm{KA}^{\mathcal{A}}_{\mathrm{B},\mathcal{S},\mathcal{E}}(\lambda)$:	Game $\mathrm{UK}^{\mathcal{A}}_{\mathrm{B},\mathcal{S},\mathcal{E}}(\lambda)$:
$\gamma \twoheadleftarrow \mathrm{B}(1^\lambda)$; $(\mathsf{R}, st) \twoheadleftarrow \mathcal{A}_0(\gamma; r_{\mathcal{A}})$ $(\boldsymbol{x}_\mu, \boldsymbol{a}) \twoheadleftarrow \mathcal{S}(st)$ $([\boldsymbol{y}_\mu]_\mu, \boldsymbol{b}) \twoheadleftarrow \mathcal{A}_1([\boldsymbol{x}_\mu]_\mu, \boldsymbol{a}, st; r_{\mathcal{A}})$ $\mathsf{trace}(\mathcal{A}) \leftarrow (r_{\mathcal{A}}, \gamma, [\boldsymbol{x}_\mu]_\mu, \boldsymbol{a})$ $\boldsymbol{w} \twoheadleftarrow \mathcal{E}(\mathsf{trace}(\mathcal{A}))$ return $\mathsf{R}(\boldsymbol{x}_\mu, [\boldsymbol{y}_\mu]_\mu, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{w})$	$\gamma \twoheadleftarrow \mathrm{B}(1^\lambda)$; $(Q, \boldsymbol{P}_\mu) \twoheadleftarrow \mathcal{A}_0(\gamma; r_{\mathcal{A}})$ $\boldsymbol{x}_\mu \twoheadleftarrow \mathcal{S}(\gamma, Q, \boldsymbol{P}_\mu)$; $\boldsymbol{x}_{\mu,0} \leftarrow 1$ $([\boldsymbol{y}_\mu]_\mu, \boldsymbol{c}) \twoheadleftarrow \mathcal{A}_1(\gamma, Q, \boldsymbol{P}_\mu, [\boldsymbol{x}_\mu]_\mu; r_{\mathcal{A}})$ $\mathsf{trace}(\mathcal{A}) \leftarrow (r_{\mathcal{A}}, \gamma, [\boldsymbol{x}_\mu]_\mu)$; $\boldsymbol{w}_\mu \twoheadleftarrow \mathcal{E}(\mathsf{trace}(\mathcal{A}))$ return $(Q(\boldsymbol{X}_\mu, \boldsymbol{Y}_\mu, \boldsymbol{c}) \neq 0 \bmod p)$ $\quad \wedge ([Q(\boldsymbol{x}_\mu, \boldsymbol{y}_\mu, \boldsymbol{c})]_T = [0]_T)$ $\quad \wedge \big((\exists \mu)(\exists i)([\boldsymbol{y}_{\mu,i}]_\mu \neq \prod_{j=0}^{m} [\boldsymbol{w}_{\mu,ij} \boldsymbol{x}_{\mu,j}]_\mu)\big)$

Fig. 3. Top left: The KA game for a group scheme Γ and source $\mathcal{S}$. **Top right:** Game defining the UK assumption for a group scheme Γ. **Bottom left:** The KA game for a type-3 bilinear group scheme B and source $\mathcal{S}$. **Bottom right:** Game defining the UK assumption for a type-3 bilinear group scheme B. In all figures, μ ranges over $\{1, 2, T\}$.

the discrete logarithms of the group elements returned by $\mathcal{A}_1$, rather than the elements themselves. Assuming that DLOG holds for Γ (resp., for some group scheme defined by B), this would in general make the KA not efficiently falsifiable [57], and one would have to distinguish between efficient and inefficient relations, and in the former case whether they are publicly or privately verifiable (i.e., whether public information is sufficient or private inputs are needed for R to be a DPT algorithm).

Knowledge assumption (KA). Let Γ be a group scheme and $\mathcal{S}$ a knowledge source. We define the advantage of an adversary $\mathcal{A}$ and an extractor $\mathcal{E}$ in the knowledge assumption (KA) game for $(\Gamma, \mathcal{S})$ as

$$\mathrm{Adv}^{\mathrm{ka}}_{\Gamma,\mathcal{S},\mathcal{A},\mathcal{E}}(\lambda) \coloneqq \Pr[\mathrm{KA}^{\mathcal{A}}_{\Gamma,\mathcal{S},\mathcal{E}}(\lambda)]\,,$$

where the game KA is defined in Fig. 3 (top left). We say that the KA holds for $(\Gamma, \mathcal{S})$ if for every PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}$ such that $\mathrm{Adv}^{\mathrm{ka}}_{\Gamma,\mathcal{S},\mathcal{A},\mathcal{E}}$ is negligible.

If B is a bilinear group scheme, the definition is adapted accordingly to accommodate for the additional inputs and outputs of $\mathcal{S}$ and $\mathcal{A}$. For example, the case of type-3 bilinear group schemes is shown in Fig. 3 (bottom left).

Remark. The definition above is framed in the asymptotic setting, but it can be readily adapted to the context of concrete security. Given a (bilinear) group scheme γ, we would then say that KA is (t, t', ϵ)-hard for $(\gamma, \mathcal{S})$ if for every

adversary $\mathcal{A}$ running in time at most t, there exists an extractor $\mathcal{E}$ running in time at most t' such that $\mathrm{Adv}^{\mathrm{ka}}_{\gamma,\mathcal{S},\mathcal{A},\mathcal{E}} \leq \epsilon$. This advantage is the winning probability in the KA game with fixed group γ (without first sampling from Γ). We also note that our extractors in idealized models do not receive any oracles. This choice, for example, ensures that justification of a knowledge problem in a model with richer oracles is stronger than one in a model with less oracles since extractors can be run without any need for oracles.

Remark. Our AGM and GGM feasibility of the UK assumptions come with universal extractors that only need black-box access to adversaries. In the standard model, such extractors do not always exist in cryptographically interesting settings: for the KEA1 assumption, if the DLOG problem is hard, adversaries that have a random exponent hard-coded in can win KEA1 while every extractor would fail.[12] However, universal extractors in other standard-model settings can exist (e.g., for sigma protocols). Finally, our definition does not allow auxiliary inputs as otherwise attacks may arise [35].

We next introduce a particular instance of the KA that will play a major role in this work, which we call the uber-knowledge (UK) assumption.

Uber-knowledge (UK) assumption. Let Γ be a group scheme. We call a knowledge adversary $\mathcal{A}$ low-degree if $\mathcal{A}_0(\gamma)$ returns a pair $(Q, \boldsymbol{P})$, where Q is a polynomial in $m + n + c + 1$ variables over $\mathbb{Z}_p$ (called relation polynomial), and $\boldsymbol{P}$ is a vector of m polynomials in k variables over $\mathbb{Z}_p$, each of total degree at most d, with $m, n, c, k, d \in \mathbb{N}$.

Let $\mathcal{S}$ be a knowledge source returning $\boldsymbol{x} \in \mathbb{Z}_p^m$. We define the advantage of a low-degree adversary $\mathcal{A}$ and an extractor $\mathcal{E}$ in the UK game for $(\Gamma, \mathcal{S})$ as

$$\mathrm{Adv}^{\mathrm{uk}}_{\Gamma,\mathcal{S},\mathcal{A},\mathcal{E}}(\lambda) \coloneqq \Pr[\mathrm{UK}^{\mathcal{A}}_{\Gamma,\mathcal{S},\mathcal{E}}(\lambda)]\,,$$

where the game UK is defined in Fig. 3 (top right). Here, $\mathcal{A}_1$ returns vectors $[\boldsymbol{y}] \in \mathsf{G}^n$ and $\boldsymbol{c} \in \mathbb{Z}_p^c$, and $\mathcal{E}$ outputs a matrix $\boldsymbol{w} \in \mathbb{Z}_p^{n\times(m+1)}$. We say that UK holds for $(\Gamma, \mathcal{S})$ if for every low-degree PPT $\mathcal{A}$ there exists a PPT $\mathcal{E}$ such that $\mathrm{Adv}^{\mathrm{uk}}_{\Gamma,\mathcal{S},\mathcal{A},\mathcal{E}}$ is negligible.

This is a special case of KA, where $\mathcal{A}_0$ returns the DPT algorithm R which checks the condition in the return statement with the given polynomial Q. An analogous definition can be formulated for bilinear group schemes, following the same blueprint, but starting from the KA for bilinear groups (for the case of type-3 bilinear group schemes, see Fig. 3 (bottom right)).

Remark. We note that whether the return condition in the UK game is efficiently verifiable depends on the degree of Q. In the case of group schemes Γ, if Q has degree at most 1 in $\boldsymbol{Y}$, the condition $(Q(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c}) = 0)$ translates into an equality involving the group elements returned by $\mathcal{A}$. For bilinear group schemes B, Q can have degree at most 2 in $\boldsymbol{Y}_\mu$ ($\mu \in \{1, 2\}$) and at most 1 in $\boldsymbol{Y}_T$, with the only

[12] Moreover, under the existence of sufficiently strong obfuscators, this negative would extend to a setting where the adversary's code is available.

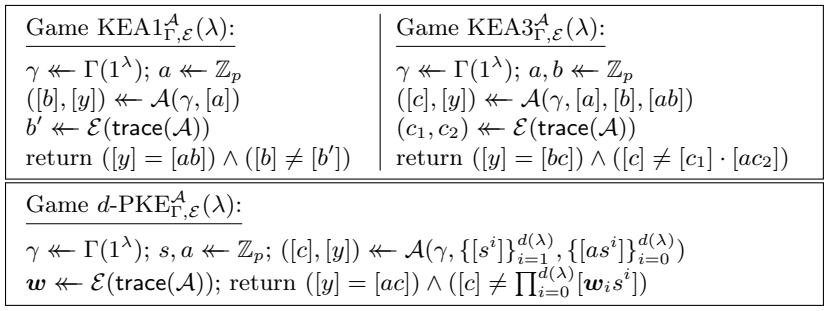

Fig. 4. Games defining the KEA1, KEA3, and d-PKE assumptions. In all figures, Γ is a group scheme and $d\colon \mathbb{N}\to\mathbb{N}$ a polynomial.

monomials of degree 2 being $\boldsymbol{Y}_{1,i}\boldsymbol{Y}_{2,j}$ (and $\boldsymbol{Y}_{2,i}\boldsymbol{Y}_{2,j}$ for type-2 group schemes). We can then use the pairing e to efficiently verify the condition above in G_T. Therefore, we will restrict our attention to polynomials Q of this type. Note that the degree in $\boldsymbol{X}$ and $\boldsymbol{C}$ can be arbitrary in both cases.

We now give a few examples of special cases of the UK assumption. We begin with examples pertaining to simple groups.

Example: KEAI, $I\in\{1,3\}$ [6, 27]. Let Γ be a group scheme. We define the advantage of an adversary $\mathcal{A}$ and an extractor $\mathcal{E}$ in the KEAI game for Γ as

$$\mathrm{Adv}^{\mathrm{kea}i}_{\Gamma,\mathcal{A},\mathcal{E}}(\lambda) \coloneqq \Pr[\mathrm{KEA}I^{\mathcal{A}}_{\Gamma,\mathcal{E}}(\lambda)]\,,$$

where the games KEAI are defined in Fig. 4 (top left and top right). Here, $\mathcal{E}$ returns an element $b'\in\mathbb{Z}_p$ (resp., $c_1,c_2\in\mathbb{Z}_p$). We say that KEAI holds for Γ if for every PPT $\mathcal{A}$ there exists a PPT $\mathcal{E}$ such that $\mathrm{Adv}^{\mathrm{kea}i}_{\Gamma,\mathcal{A},\mathcal{E}}$ is negligible.

This is a special case of UK, where $\mathcal{A}_0$ returns $Q(\boldsymbol{X}_0,\boldsymbol{X}_1,\boldsymbol{Y}_1,\boldsymbol{Y}_2)\coloneqq\boldsymbol{Y}_2-\boldsymbol{X}_1\boldsymbol{Y}_1$ and $P(S)\coloneqq S$ (resp., $Q(\boldsymbol{X}_0,\ldots,\boldsymbol{X}_3,\boldsymbol{Y}_1,\boldsymbol{Y}_2)\coloneqq\boldsymbol{Y}_2-\boldsymbol{X}_2\boldsymbol{Y}_1$ as well as $\boldsymbol{P}_1(\boldsymbol{S}_1,\boldsymbol{S}_2)\coloneqq\boldsymbol{S}_1$, $\boldsymbol{P}_2(\boldsymbol{S}_1,\boldsymbol{S}_2)\coloneqq\boldsymbol{S}_2$, $\boldsymbol{P}_3(\boldsymbol{S}_1,\boldsymbol{S}_2)\coloneqq\boldsymbol{S}_1\boldsymbol{S}_2$), $\mathcal{S}$ returns a random $a\in\mathbb{Z}_p$ (resp., samples $a,b\in\mathbb{Z}_p$ at random and returns $\boldsymbol{P}(a,b)$), $\mathcal{A}_1$ runs $\mathcal{A}$, and $\mathcal{E}$ returns the matrix $\left(\begin{smallmatrix}b' & 0\\ 0 & b'\end{smallmatrix}\right)$ (resp., $\left(\begin{smallmatrix}c_1 & c_2 & 0 & 0\\ 0 & 0 & c_1 & c_2\end{smallmatrix}\right)$).

Remark. We note that the terminology around the KEA assumptions is not well-established. For example, [7,31] refer to the notion we call KEA1 as the DHK (or DHK0) assumption, while [6] reserves the name KEA1 for the non-uniform version of the notion above. Another name for the latter version is DA-1 [47].

Example: d-PKE [42]. Let Γ be a group scheme and $d\colon\mathbb{N}\to\mathbb{N}$ a polynomial. We define the advantage of an adversary $\mathcal{A}$ and an extractor $\mathcal{E}$ in the d-PKE game for Γ as

$$\mathrm{Adv}^{d\text{-pke}}_{\Gamma,\mathcal{A},\mathcal{E}}(\lambda) \coloneqq \Pr[d\text{-PKE}^{\mathcal{A}}_{\Gamma,\mathcal{E}}(\lambda)]\,,$$

where the game d-PKE is defined in Fig. 4 (bottom). Here, $\mathcal{E}$ returns a vector $\boldsymbol{w}\in\mathbb{Z}_p^{d(\lambda)+1}$. We say that d-PKE holds for Γ if for every PPT $\mathcal{A}$ there exists a PPT $\mathcal{E}$ such that $\mathrm{Adv}^{d\text{-pke}}_{\Gamma,\mathcal{A},\mathcal{E}}$ is negligible.

This is an instance of UK where $\mathcal{A}_0$ returns $Q((\boldsymbol{X}_i)_{i=0}^{d(\lambda)}, (\boldsymbol{X}'_i)_{i=0}^{d(\lambda)}, \boldsymbol{Y}_1, \boldsymbol{Y}_2) \coloneqq \boldsymbol{Y}_2 - \boldsymbol{X}'_0\boldsymbol{Y}_1$ as well as $\boldsymbol{P}_i(\boldsymbol{S}_1, \boldsymbol{S}_2) \coloneqq \boldsymbol{S}_1^i$ $(1 \le i \le d(\lambda))$ and $\boldsymbol{P}'_i(\boldsymbol{S}_1, \boldsymbol{S}_2) \coloneqq \boldsymbol{S}_2\boldsymbol{S}_1^i$ $(0 \le i \le d(\lambda))$, $\mathcal{S}$ samples $s, a \in \mathbb{Z}_p$ at random and returns $(\boldsymbol{P}(s,a), \boldsymbol{P}'(s,a))$, $\mathcal{A}_1$ runs $\mathcal{A}$, and $\mathcal{E}$ returns $\begin{pmatrix} \boldsymbol{w} & 0^{d(\lambda)+1} \\ 0^{d(\lambda)+1} & \boldsymbol{w} \end{pmatrix}$ (here $\boldsymbol{w}$ is interpreted as a row vector).

Remark. We note that the d-PKE assumption is false if we remove the condition ($[y] = [ac]$) from the game and allow parties to hash into γ (and DLOG holds for Γ), even if we restrict the adversary to be algebraic.

The remaining examples that we consider concern bilinear groups. All assumptions are defined for type-3 bilinear group schemes, but the definitions can be readily adapted to the setting of type-2 or type-1 schemes.

Example: d-KZG [51]. Let B be a type-3 bilinear group scheme and $d\colon \mathbb{N} \to \mathbb{N}$ a polynomial. The advantage of an adversary $\mathcal{A}$ and an extractor $\mathcal{E}$ in the d-KZG game for B is

$$\mathrm{Adv}^{d\text{-kzg}}_{\mathrm{B},\mathcal{A},\mathcal{E}}(\lambda) \coloneqq \Pr[d\text{-KZG}^{\mathcal{A}}_{\mathrm{B},\mathcal{E}}(\lambda)]\,,$$

where the game d-KZG is defined in Fig. 5 (top). Here, $\mathcal{E}$ returns a vector $\boldsymbol{w} \in \mathbb{Z}_p^{d(\lambda)}$. We say that d-KZG holds for B if for every PPT $\mathcal{A}$ there exists a PPT $\mathcal{E}$ such that $\mathrm{Adv}^{d\text{-kzg}}_{\mathrm{B},\mathcal{A},\mathcal{E}}$ is negligible.

This is the instance of UK where $\mathcal{A}_0$ returns the polynomials

$$Q((\boldsymbol{X}_{1,i})_{i=0}^{d(\lambda)-1}, \boldsymbol{X}_{2,0}, \boldsymbol{X}_{2,1}, (\boldsymbol{Y}_{1,i}, \boldsymbol{C}_i)_{i=1}^{2}) \coloneqq \boldsymbol{Y}_{1,1} - \boldsymbol{C}_2 - \boldsymbol{Y}_{1,2}(\boldsymbol{X}_{2,1} - \boldsymbol{C}_1)\,,$$

as well as $\boldsymbol{P}_{1,i}(S) \coloneqq S^i$ $(1 \le i \le d(\lambda) - 1)$ and $P_2(S) \coloneqq S$, $\mathcal{S}$ samples $s \in \mathbb{Z}_p$ at random and returns $(\boldsymbol{P}_1(s), P_2(s))$, $\mathcal{A}_1$ runs $\mathcal{A}$, and $\mathcal{E}$ is as shown in the game.

Remark. The idea behind d-KZG is to allow a party to commit to a polynomial $C \in \mathbb{Z}_p[X]$ of degree at most d, and then to prove that $C(x) = y$ for certain $x, y \in \mathbb{Z}_p$. Notice that the latter means $C(X) - y = Q(X)(X - x)$ for some polynomial $Q \in \mathbb{Z}_p[X]$, which by Lemma 1 is equivalent to $c - y = q(s - x)$ with high probability, where $s \in \mathbb{Z}_p$ is random and $c = C(s)$, $q = Q(s)$. This suggests letting $[c]_1$ be the commitment to C, and $[q]_1$ the proof of the fact that $C(x) = y$. Notice that the equality above can be efficiently checked in G_T using a pairing, as in the d-KZG game. The d-KZG assumption is meant to formalize that this proof is sound, meaning that no adversary can produce group elements as above without knowing the coefficients of C.

Example: d-PKE [42]. Let B be a type-3 bilinear group scheme and $d\colon \mathbb{N} \to \mathbb{N}$ a polynomial. We define the advantage of an adversary $\mathcal{A}$ and an extractor $\mathcal{E}$ in the d-PKE game for B as

$$\mathrm{Adv}^{d\text{-pke}}_{\mathrm{B},\mathcal{A},\mathcal{E}}(\lambda) \coloneqq \Pr[d\text{-PKE}^{\mathcal{A}}_{\mathrm{B},\mathcal{E}}(\lambda)]\,,$$

where the game d-PKE is defined in Fig. 5 (center). Here, $\mathcal{E}$ returns a vector $\boldsymbol{w} \in \mathbb{Z}_p^{d(\lambda)+1}$. We say that d-PKE holds for B if for every PPT $\mathcal{A}$ there exists a PPT $\mathcal{E}$ such that $\mathrm{Adv}^{d\text{-pke}}_{\mathrm{B},\mathcal{A},\mathcal{E}}$ is negligible.

Game $d\text{-KZG}^{\mathcal{A}}_{\mathrm{B},\mathcal{E}}(\lambda)$:

$\gamma \twoheadleftarrow \mathrm{B}(1^\lambda)$; $s \twoheadleftarrow \mathbb{Z}_p$; $([c]_1, [q]_1, x, y) \twoheadleftarrow \mathcal{A}(\gamma, \{[s^i]_1\}_{i=1}^{d(\lambda)-1}, [s]_2)$; $\boldsymbol{w} \twoheadleftarrow \mathcal{E}(\mathsf{trace}(\mathcal{A}))$
return $(e([c]_1 \cdot [-y]_1, [1]_2) = e([q]_1, [s]_2 \cdot [-x]_2)) \wedge ([c]_1 \neq \prod_{i=0}^{d(\lambda)-1} [\boldsymbol{w}_i s^i]_1)$

Game $d\text{-PKE}^{\mathcal{A}}_{\mathrm{B},\mathcal{E}}(\lambda)$:

$\gamma \twoheadleftarrow \mathrm{B}(1^\lambda)$; $s, a \twoheadleftarrow \mathbb{Z}_p$; $([c]_1, [y]_1) \twoheadleftarrow \mathcal{A}(\gamma, \{[s^i]_1\}_{i=1}^{d(\lambda)}, \{[as^i]_1\}_{i=0}^{d(\lambda)}, [s]_2, [a]_2)$
$\boldsymbol{w} \twoheadleftarrow \mathcal{E}(\mathsf{trace}(\mathcal{A}))$; return $([y]_1 = [ac]_1) \wedge ([c]_1 \neq \prod_{i=0}^{d(\lambda)} [\boldsymbol{w}_i s^i]_1)$

Game $d\text{-GROTH16}^{\mathcal{A}}_{\mathrm{B},\mathcal{E}}(\lambda)$:

$\varpi \twoheadleftarrow \mathrm{B}(1^\lambda)$; $\mathsf{R} \coloneqq (\ell, (U_i, V_i, W_i)_{i=0}^m, T) \twoheadleftarrow \mathcal{A}_0(\varpi)$
$\alpha, \beta, \gamma, \delta, x \twoheadleftarrow \mathbb{Z}_p^*$; $\boldsymbol{x}_{1,0} \leftarrow 1$; $\boldsymbol{x}_{2,0} \leftarrow 1$; $\boldsymbol{x}_{T,0} \leftarrow 1$
$\boldsymbol{x}_{1,1} \leftarrow \alpha\gamma\delta$; $\boldsymbol{x}_{1,2} \leftarrow \beta\gamma\delta$; $\boldsymbol{x}_{1,3} \leftarrow \gamma\delta^2$; $\boldsymbol{x}_{2,1} \leftarrow \beta\gamma\delta$; $\boldsymbol{x}_{2,2} \leftarrow \gamma^2\delta$; $\boldsymbol{x}_{2,3} \leftarrow \gamma\delta^2$
for $i = 0$ to $d(\lambda) - 1$ do $\boldsymbol{x}_{1,4+i} \leftarrow \gamma\delta x^i$; for $i = 0$ to $d(\lambda) - 2$ do $\boldsymbol{x}_{1,d+4+i} \leftarrow \gamma x^i T(x)$
for $i = 0$ to ℓ do $\boldsymbol{x}_{1,2d+3+i} \leftarrow \beta\delta U_i(x) + \alpha\delta V_i(x) + \delta W_i(x)$
for $i = \ell + 1$ to m do $\boldsymbol{x}_{1,2d+3+i} \leftarrow \beta\gamma U_i(x) + \alpha\gamma V_i(x) + \gamma W_i(x)$
for $i = 0$ to $d(\lambda) - 1$ do $\boldsymbol{x}_{2,4+i} \leftarrow \gamma\delta x^i$
$((f_i)_{i=1}^\ell, [a]_1, [c]_1, [b]_2) \twoheadleftarrow \mathcal{A}_1(\varpi, \mathsf{R}, [\boldsymbol{x}_1]_1, [\boldsymbol{x}_2]_2)$; $(\boldsymbol{w}_i)_{i=1}^3 \twoheadleftarrow \mathcal{E}(\mathsf{trace}(\mathcal{A}))$; $f_0 \leftarrow 1$
return $\left(e([a]_1, [b]_2) = e([\boldsymbol{x}_{1,1}]_1, [\boldsymbol{x}_{2,1}]_2) \cdot \prod_{i=0}^{\ell} e([f_i \boldsymbol{x}_{1,2d+i}]_1, [\boldsymbol{x}_{2,2}]_2) \cdot e([c]_1, [\boldsymbol{x}_{2,3}]_2)\right)$
$\wedge \left(\left([a]_1 \neq \prod_{i=0}^{2d(\lambda)+m+3} [\boldsymbol{w}_{1,i} \boldsymbol{x}_{1,i}]_1\right) \vee \left([c]_1 \neq \prod_{i=0}^{2d(\lambda)+m+3} [\boldsymbol{w}_{2,i} \boldsymbol{x}_{1,i}]_1\right) \vee \left([b]_2 \neq \prod_{i=0}^{d(\lambda)+3} [\boldsymbol{w}_{3,i} \boldsymbol{x}_{2,i}]_2\right)\right)$

Fig. 5. Games defining the d-KZG, d-PKE, and d-GROTH16 assumptions. Here, B is a type-3 bilinear group scheme, and $d\colon \mathbb{N} \to \mathbb{N}$ a polynomial.

This is an instance of UK where $\mathcal{A}_0$ returns the same polynomial Q as in the case of group schemes, as well as $\boldsymbol{P}_{1,i}(\boldsymbol{S}_1, \boldsymbol{S}_2) \coloneqq \boldsymbol{S}_1^i$ $(1 \leq i \leq d(\lambda))$, $\boldsymbol{P}'_{1,i}(\boldsymbol{S}_1, \boldsymbol{S}_2) \coloneqq \boldsymbol{S}_2 \boldsymbol{S}_1^i$ $(0 \leq i \leq d(\lambda))$ and $\boldsymbol{P}_{2,1}(\boldsymbol{S}_1, \boldsymbol{S}_2) \coloneqq \boldsymbol{S}_1$, $\boldsymbol{P}_{2,2}(\boldsymbol{S}_1, \boldsymbol{S}_2) \coloneqq \boldsymbol{S}_2$. $\mathcal{S}$ samples $s, a \in \mathbb{Z}_p$ at random and returns $(\boldsymbol{P}_1(s,a), \boldsymbol{P}'_1(s,a), \boldsymbol{P}_2(s,a))$, $\mathcal{A}_1$ runs $\mathcal{A}$, and $\mathcal{E}$ returns $\begin{pmatrix} \boldsymbol{w} & 0^{d(\lambda)+1} \\ 0^{d(\lambda)+1} & \boldsymbol{w} \end{pmatrix}$ (here $\boldsymbol{w}$ is interpreted as a row vector).

Example: d-GROTH16 [43]. Let B be a type-3 bilinear group scheme, and $d\colon \mathbb{N} \to \mathbb{N}$ a polynomial. We define the advantage of an adversary $\mathcal{A}$ and an extractor $\mathcal{E}$ in the d-GROTH16 game for B as

$$\mathrm{Adv}^{d\text{-groth16}}_{\mathrm{B},\mathcal{A},\mathcal{E}}(\lambda) \coloneqq \Pr[d\text{-GROTH16}^{\mathcal{A}}_{\mathrm{B},\mathcal{E}}(\lambda)]\,,$$

where the game d-GROTH16 is defined in Fig. 5 (bottom). Here, $\mathcal{E}$ returns a vector $\boldsymbol{w} \in \mathbb{Z}_p^{m-\ell}$. We say that d-GROTH16 holds for B if for every PPT $\mathcal{A}$ there exists a PPT $\mathcal{E}$ such that $\mathrm{Adv}^{d\text{-groth16}}_{\mathrm{B},\mathcal{A},\mathcal{E}}$ is negligible.

This is a special case of UK, where $\mathcal{A}_0$ returns the polynomial Q used in the winning condition and the vector of polynomials $\boldsymbol{P}$ used to form $\boldsymbol{x}$, $\mathcal{S}$ samples $\alpha, \beta, \gamma, \delta, x \in \mathbb{Z}_p$ at random and returns $\boldsymbol{P}(\alpha, \beta, \gamma, \delta, x)$, $\mathcal{A}_1$ runs $\mathcal{A}$, and $\mathcal{E}$ is as shown in the game.

REMARK. Notice that we define a slightly modified version of the security game considered in [43], where all polynomials are multiplied by $\gamma\delta$, in order to clear the denominators and let the assumption fit the UK-framework.

5 Soundness of UK in GBM1-H

In this section we justify the soundness of the UK assumption in the GBM1-H. Our result is for a class of adversaries $\mathcal{A}$ who return a relation polynomial Q of degree at most two in the output variables, with at most one square term and linearly independent coefficients for the linear terms. The latter condition serves to avoid that $\mathcal{A}$ can satisfy the linear part of Q by hashing into the group, and then crafting other elements via exponentiation to satisfy the linear relation. The corresponding result for simple groups is included in Appendix A.

Theorem 1 (UK holds in GBM1-H). *Let $d, p \in \mathbb{N}$ with p prime, and fix G, $\mathsf{G}_T \subseteq \{0,1\}^*$ with $|\mathsf{G}| = |\mathsf{G}_T| = p$. Consider low-degree adversaries $\mathcal{A}$ and sources $\mathcal{S}$ with the following properties:*

1. *The relation polynomial Q returned by $\mathcal{A}_0$ is of the form*

$$Q(\boldsymbol{X}, \boldsymbol{X}_T, \boldsymbol{Y}, \boldsymbol{C}) = Q_{i_1 i_2}(\boldsymbol{X}, \boldsymbol{X}_T, \boldsymbol{C}) \boldsymbol{Y}_{i_1} \boldsymbol{Y}_{i_2} + \sum_{i=1}^{n} Q_i(\boldsymbol{X}, \boldsymbol{X}_T, \boldsymbol{C}) \boldsymbol{Y}_i + Q_0(\boldsymbol{X}, \boldsymbol{X}_T, \boldsymbol{C}) \,,$$

   where $m := |\boldsymbol{X}| - 1$, $m_T := |\boldsymbol{X}_T|$, $n := |\boldsymbol{Y}|$, $c := |\boldsymbol{C}|$, and $1 \le i_1, i_2 \le n$;
2. *For every Q and every $\boldsymbol{P}$, $\boldsymbol{P}_T$ returned by $\mathcal{A}_0$, and every $\boldsymbol{c} \in \mathbb{Z}_p^c$, either $\overline{Q_{i_1 i_2}} \neq 0$ and polynomials $\overline{Q_i}$ (for $i > 0$ and $i \neq i_1, i_2$) are linearly independent, or polynomials $\overline{Q_i}$ (for $i > 0$) are linearly independent;*
3. *For every Q and every $\boldsymbol{P}$, $\boldsymbol{P}_T$ returned by $\mathcal{A}_0$ such that $\overline{Q_{i_1 i_2}} \neq 0$, every $\boldsymbol{w} \in \mathbb{Z}_p^{n \times (m+1)}$, every $\boldsymbol{c} \in \mathbb{Z}_p^c$, and every $(\alpha, \beta) \in \{(i_1, i_2), (i_2, i_1)\}$, we have*

$$\Big(\sum_{j=0}^{m} \boldsymbol{w}_{\alpha j} \boldsymbol{P}_j\Big)\Big(\overline{Q_{i_1 i_2}}\Big(\sum_{j=0}^{m} \boldsymbol{w}_{\beta j} \boldsymbol{P}_j\Big) + \overline{Q_\alpha}\Big) + \sum_{\substack{i=1 \\ i \neq \alpha}}^{n} \sum_{j=0}^{m} \boldsymbol{w}_{ij} \overline{Q_i} \boldsymbol{P}_j + \overline{Q_0} \neq 0 \,;$$

4. *For every $\boldsymbol{P}$, $\boldsymbol{P}_T$ returned by $\mathcal{A}_0$, $\mathcal{S}$ samples $\boldsymbol{s} \in \mathbb{Z}_p^k$ at random and returns $(\boldsymbol{P}(\boldsymbol{s}), \boldsymbol{P}_T(\boldsymbol{s}))$.*

Then the UK *assumption holds in the* GBM1-H *with parameters (p, G) with respect to the class of adversaries above. More precisely, for every adversary $\mathcal{A}$ as above, there exists an extractor $\mathcal{E}$ such that*

$$\mathrm{Adv}^{\mathrm{uk}}_{p,\mathsf{G},\mathcal{S},\mathcal{A},\mathcal{E}} \le \big((m + 3q_{\mathsf{op}} + q_{\mathsf{H}} + 2q_{\mathsf{e}} + 1)^2 + (m + 3q_{\mathsf{op}_T} + q_{\mathsf{H}_T} + q_{\mathsf{e}})^2\big) \cdot \frac{d}{2p} \,.$$

Here q_{op}, q_{op_T}, q_{H}, q_{H_T}, and q_{e} are upper bounds on the number of queries made by $\mathcal{A}$ to the respective oracles, $\boldsymbol{P}_0(\boldsymbol{S}) := 1$, $\overline{Q_{i_1 i_2}}(\boldsymbol{S}) := Q_{i_1 i_2}(\boldsymbol{P}(\boldsymbol{S}), \boldsymbol{P}_T(\boldsymbol{S}), \boldsymbol{c})$, and similarly for $\overline{Q_i}$ and $\overline{Q_0}$.

Extractor $\mathcal{E}(\mathsf{trace}(\mathcal{A}))$:
parse $\mathsf{trace}(\mathcal{A}) = (r_{\mathcal{A}}, \gamma, \boldsymbol{x}, \boldsymbol{x}_T, h_1, \ldots, h_q)$
$(Q, \boldsymbol{P}, \boldsymbol{P}_T) \twoheadleftarrow \mathcal{A}_0(\gamma; r_{\mathcal{A}})$; $U, U_T \leftarrow [\,]$; $o, v, v_T \leftarrow 0$; $U[g] \leftarrow 1$
for $j = 1$ to m do $U[\boldsymbol{x}_j] \leftarrow \boldsymbol{P}_j(\boldsymbol{S})$; for $j = 1$ to m_T do $U_T[\boldsymbol{x}_{T,j}] \leftarrow \boldsymbol{P}_{T,j}(\boldsymbol{S})$
$(\boldsymbol{y}, \boldsymbol{c}) \twoheadleftarrow \mathcal{A}_1^{\overline{\mathsf{op}}, \overline{\mathsf{op}}_T, \overline{\mathsf{H}}, \overline{\mathsf{H}}_T, \bar{\mathsf{e}}}(\boldsymbol{x}, \boldsymbol{x}_T; r_{\mathcal{A}})$; if $\big((\exists i)(\boldsymbol{y}_i \notin \mathrm{Dom}(U))\big)$ then return 0
for $i = 1$ to n do parse $U[\boldsymbol{y}_i] = \sum_{j=1}^{m} \boldsymbol{w}_{ij} \boldsymbol{P}_j(\boldsymbol{S}) + \sum_k \boldsymbol{b}_{ik} \boldsymbol{R}_k$
return $\boldsymbol{w}$

Proc. $\overline{\mathsf{op}}(a, h_1, b, h_2)$:
$o \leftarrow o + 1$
if $(h_1 \notin \mathrm{Dom}(U))$ then
  $v \leftarrow v + 1$; $U[h_1] \leftarrow \boldsymbol{R}_v$
if $(h_2 \notin \mathrm{Dom}(U))$ then
  $v \leftarrow v + 1$; $U[h_2] \leftarrow \boldsymbol{R}_v$
if $(h_o \notin \mathrm{Dom}(U))$ then
  $U[h_o] \leftarrow aU[h_1] + bU[h_2]$
return h_o

Proc. $\overline{\mathsf{H}}(m)$:
$o \leftarrow o + 1$
if $(h_o \notin \mathrm{Dom}(U))$ then
  $v \leftarrow v + 1$
  $U[h_o] \leftarrow \boldsymbol{R}_v$
return h_o

Proc. $\bar{\mathsf{e}}(h_1, h_2)$:
$o \leftarrow o + 1$
if $(h_1 \notin \mathrm{Dom}(U))$ then
  $v \leftarrow v + 1$; $U[h_1] \leftarrow \boldsymbol{R}_v$
if $(h_2 \notin \mathrm{Dom}(U))$ then
  $v \leftarrow v + 1$; $U[h_2] \leftarrow \boldsymbol{R}_v$
if $(h_o \notin \mathrm{Dom}(U_T))$ then
  $v_T \leftarrow v_T + 1$; $U_T[h_o] \leftarrow \boldsymbol{R}_{T,v_T}$
return h_o

Fig. 6. Definition of the extractor $\mathcal{E}$ from the proof of Theorem 1. Oracles $\overline{\mathsf{op}}_T$ and $\overline{\mathsf{H}}_T$ proceed exactly as $\overline{\mathsf{op}}$ and $\overline{\mathsf{H}}$, but use table U_T and counter v_T in place of U and v, and save variables $\boldsymbol{R}_{T,v_T}$ instead of $\boldsymbol{R}_v$. The counter o is shared between all oracles.

Proof. Fix an adversary $\mathcal{A}$ in the UK game as in the statement of the theorem, and define an extractor $\mathcal{E}$ as in Fig. 6. This extractor essentially re-runs $\mathcal{A}_1$ and observes its oracle queries, keeping track of the discrete logarithms of the elements queried by $\mathcal{A}_1$. Whenever $\mathcal{E}$ is unable to "explain" an element in G (resp., G_T), it stores a fresh variable $\boldsymbol{R}_v$ (resp., $\boldsymbol{R}_{T,v_T}$) in its table for G (resp., G_T).

We claim that this extractor allows us to prove the bound in the theorem statement. To that end, consider the following sequence of games:

- G_0: This is the original UK game with respect to source $\mathcal{S}$, adversary $\mathcal{A}$ and extractor $\mathcal{E}$.
- G_1: This game proceeds as G_0, but the encodings τ and τ_T are implemented via lazy sampling. More precisely, instead of sampling τ and τ_T, the challenger of G_1 initializes tables $T \leftarrow T_T \leftarrow [\,]$. Oracles op and H (resp., op_T and H_T) are then implemented via lazy sampling using table T (resp., using table T_T). The same is done for oracle e.
- G_2: This game proceeds as G_1, but the challenger replaces the values $\boldsymbol{x}_i$ and $\boldsymbol{x}_{T,i}$ generated by $\mathcal{S}$ with the corresponding polynomials $\boldsymbol{P}_i(\boldsymbol{S})$ and $\boldsymbol{P}_{T,i}(\boldsymbol{S})$ evaluated at formal variables $\boldsymbol{S}$. Likewise, whenever it lazily samples a domain point in T (resp., T_T), it instead saves a fresh variable $\boldsymbol{R}_v$ (resp., $\boldsymbol{R}_{T,v_T}$). Notice that in this game, tables T and T_T are populated exactly as tables U and U_T in $\mathcal{E}$.

We now argue that the difference between the success probabilities in subsequent games is small.

$\mathrm{G}_0 \rightsquigarrow \mathrm{G}_1$. Notice that G_0 and G_1 have the same distribution, because the oracles given to $\mathcal{A}$ in the two games are distributed identically. In particular, this means $\Pr[\mathrm{G}_1] = \Pr[\mathrm{G}_0]$.

$\mathrm{G}_1 \rightsquigarrow \mathrm{G}_2$. Let Bad (resp., Bad_T) be the event in G_2 that there are two different polynomials in the domain of T (resp., T_T) which result in the same value when evaluating $\boldsymbol{S}$, $\boldsymbol{R}$ and $\boldsymbol{R}_T$ at random $\boldsymbol{s}$, $\boldsymbol{r}$ and $\boldsymbol{r}_T$. Notice that G_1 and G_2 are identical until Bad or Bad_T, and by the fundamental lemma of game playing we therefore have that $|\Pr[\mathrm{G}_2] - \Pr[\mathrm{G}_1]| \leq \Pr[\mathsf{Bad}] + \Pr[\mathsf{Bad}_T]$.

We bound the latter two probabilities via Lemma 1. Consider an adversary $\mathcal{B}$ in the Schwartz–Zippel game which simulates G_2 to $\mathcal{A}$ and then returns all polynomials in the domain of T. Notice that if Bad occurs, then $\mathcal{B}$ wins the SZ-game, and that T contains at most $m + 3q_{\mathsf{op}} + q_{\mathsf{H}} + 2q_{\mathsf{e}} + 1$ polynomials of degree at most d. By Lemma 1, $\Pr[\mathsf{Bad}] \leq (m + 3q_{\mathsf{op}} + q_{\mathsf{H}} + 2q_{\mathsf{e}} + 1) \cdot d/2p$. We similarly bound $\Pr[\mathsf{Bad}_T]$, noting that T_T contains at most $m_T + 3q_{\mathsf{op}_T} + q_{\mathsf{H}_T} + q_{\mathsf{e}}$ polynomials of degree at most d. Therefore,

$$\begin{aligned}|\Pr[\mathrm{G}_2] - \Pr[\mathrm{G}_1]| &\leq \Pr[\mathsf{Bad}] + \Pr[\mathsf{Bad}_T] \\ &\leq \left((m + 3q_{\mathsf{op}} + q_{\mathsf{H}} + 2q_{\mathsf{e}} + 1)^2 + (m + 3q_{\mathsf{op}_T} + q_{\mathsf{H}_T} + q_{\mathsf{e}})^2\right) \cdot \frac{d}{2p}\,.\end{aligned}$$

We conclude the proof by studying the winning probability of $\mathcal{A}$ in G_2. Notice that if the output of $\mathcal{A}$ is such that the polynomial Q is not satisfied, then $\mathcal{A}$ has trivially lost the game. So suppose that Q is satisfied. Comparing entries stored in T_T (recall that T_T implements a partial random injection), we obtain

$$\begin{aligned}\overline{Q_{i_1 i_2}}\left(\sum_{j=0}^{m} \boldsymbol{w}_{i_1 j}\boldsymbol{P}_j + \sum_k \boldsymbol{b}_{i_1 k}\boldsymbol{R}_k\right)\left(\sum_{j=0}^{m} \boldsymbol{w}_{i_2 j}\boldsymbol{P}_j + \sum_k \boldsymbol{b}_{i_2 k}\boldsymbol{R}_k\right) \\ + \sum_{i=1}^{n} \overline{Q_i}\left(\sum_{j=0}^{m} \boldsymbol{w}_{ij}\boldsymbol{P}_j + \sum_k \boldsymbol{b}_{ik}\boldsymbol{R}_k\right) + \overline{Q_0} = 0\,,\end{aligned} \tag{1}$$

as a polynomial in $\boldsymbol{S}$ and $\boldsymbol{R}$. We want to show that this implies $\boldsymbol{b}_{ik} = 0$ for all $1 \leq i \leq n$ and all k. Assume for the moment that $\overline{Q_{i_1 i_2}} \neq 0$. From the term of degree two in $\boldsymbol{R}$ we obtain

$$\overline{Q_{i_1 i_2}}\boldsymbol{b}_{i_1 r}\boldsymbol{b}_{i_2 s} = 0$$

for every r and s, which means that either $\boldsymbol{b}_{i_1 r} = 0$ or $\boldsymbol{b}_{i_2 s} = 0$ for every r and s. We claim that, in fact, $\boldsymbol{b}_{i_1 r} = \boldsymbol{b}_{i_2 s} = 0$ for every r and s. Indeed, suppose this was not the case, and let $\bar{r}$ be an index such that $\boldsymbol{b}_{i_1 \bar{r}} \neq 0$. Then $\boldsymbol{b}_{i_2 s} = 0$ for every s. The linear term in $\boldsymbol{R}_{\bar{r}}$ now becomes

$$\overline{Q_{i_1 i_2}}\boldsymbol{b}_{i_1 \bar{r}} \sum_{j=0}^{m} \boldsymbol{w}_{i_2 j}\boldsymbol{P}_j + \sum_{i=1}^{n} \overline{Q_i}\boldsymbol{b}_{i\bar{r}} = 0\,.$$

Plugging this equality into the constant term in $\boldsymbol{R}$, we obtain

$$\left(\sum_{j=0}^{m} \boldsymbol{w}_{i_2 j} \boldsymbol{P}_j\right)\left(\overline{Q_{i_1 i_2}}\left(\sum_{j=0}^{m} \boldsymbol{w}_{i_1 j} \boldsymbol{P}_j\right) + \overline{Q_{i_2}}\right) + \sum_{i=1, i\neq i_2}^{n} \sum_{j=0}^{m} \boldsymbol{w}_{ij} \overline{Q_i} \boldsymbol{P}_j + \overline{Q_0} = 0\,.$$

This, however, contradicts our assumption, from which we conclude that $\boldsymbol{b}_{i_1 r} = 0$ for every r. As a consequence, Equation (1) now becomes

$$\overline{Q_{i_1 i_2}}\left(\sum_{j=0}^{m} \boldsymbol{w}_{i_1 j} \boldsymbol{P}_j\right)\left(\sum_{j=0}^{m} \boldsymbol{w}_{i_2 j} \boldsymbol{P}_j + \sum_{k} \boldsymbol{b}_{i_2 k} \boldsymbol{R}_k\right) + \sum_{i=1}^{n} \overline{Q_i}\left(\sum_{j=0}^{m} \boldsymbol{w}_{ij} \boldsymbol{P}_j + \sum_{k} \boldsymbol{b}_{ik} \boldsymbol{R}_k\right) + \overline{Q_0} = 0\,.$$

We can similarly show that $\boldsymbol{b}_{i_2 s} = 0$ for every s. Indeed, assume for the sake of contradiction that $\boldsymbol{b}_{i_2 \bar{s}} \neq 0$ for some $\bar{s}$. The linear term in $\boldsymbol{R}_{\bar{s}}$ then is

$$\overline{Q_{i_1 i_2}} \boldsymbol{b}_{i_2 \bar{s}} \sum_{j=0}^{m} \boldsymbol{w}_{i_1 j} \boldsymbol{P}_j + \sum_{i=1}^{n} \overline{Q_i} \boldsymbol{b}_{i \bar{s}} = 0\,.$$

Plugging this equality into the constant term in $\boldsymbol{R}$, we obtain

$$\left(\sum_{j=0}^{m} \boldsymbol{w}_{i_1 j} \boldsymbol{P}_j\right)\left(\overline{Q_{i_1 i_2}}\left(\sum_{j=0}^{m} \boldsymbol{w}_{i_2 j} \boldsymbol{P}_j\right) + \overline{Q_{i_1}}\right) + \sum_{i=1, i\neq i_1}^{n} \sum_{j=0}^{m} \boldsymbol{w}_{ij} \overline{Q_i} \boldsymbol{P}_j + \overline{Q_0} = 0\,,$$

which again contradicts our assumption, and thus $\boldsymbol{b}_{i_2 s} = 0$ for every s. Consequently, Equation (1) simplifies to

$$\overline{Q_{i_1 i_2}}\left(\sum_{j=0}^{m} \boldsymbol{w}_{i_1 j} \boldsymbol{P}_j\right)\left(\sum_{j=0}^{m} \boldsymbol{w}_{i_2 j} \boldsymbol{P}_j\right) + \sum_{i=1}^{n} \overline{Q_i}\left(\sum_{j=0}^{m} \boldsymbol{w}_{ij} \boldsymbol{P}_j + \sum_{k} \boldsymbol{b}_{ik} \boldsymbol{R}_k\right) + \overline{Q_0} = 0\,.$$

Now, looking at the linear terms in $\boldsymbol{R}$, we obtain that for every k,

$$\sum_{i=1}^{n} \overline{Q_i} \boldsymbol{b}_{ik} = 0\,. \tag{2}$$

Recall that, by assumption, polynomials $\overline{Q_i}$ with $i > 0$ and $i \neq i_1, i_2$ are linearly independent, which means that $\boldsymbol{b}_{ik} = 0$ for all $1 \leq i \leq n$ and all k.

If on the other hand $\overline{Q_{i_1 i_2}} = 0$, then there are no terms of degree two in $\boldsymbol{R}$ in Equation (1). This means that we can jump directly to Equation (2) and conclude that $\boldsymbol{b}_{ik} = 0$ for all $1 \leq i \leq n$ and all k, since $\overline{Q_i}$ are linearly independent.

This in turn shows that $\mathcal{E}$ returns an accurate representation of $\boldsymbol{y}$ in terms of $\boldsymbol{x}$, which concludes the proof. □

We now show that the specific knowledge assumptions considered in the previous section all satisfy the condition stated in the theorem above (and its analogous theorem for simple groups proved under Theorem 7 in Appendix A).

Corollary 1. *Let $d, p \in \mathbb{N}$ with p prime, and fix $\mathsf{G}, \mathsf{G}_T \subseteq \{0,1\}^*$ with $|\mathsf{G}| = |\mathsf{G}_T| = p$.* KEA1, KEA3, *and* d-PKE *all hold in* GGM-H *with parameters* (p, G). d-KZG, d-PKE, *and* d-GROTH16 *all hold in* GBM1-H *with parameters* $(p, \mathbf{G})$.

Proof. The proof is straightforward for all assumptions except d-GROTH16; we cover d-KZG as an example. Clearly, the relation polynomial Q in d-KZG is of the form considered in Theorem 1, and the polynomials $\overline{Q_1}(S) = 1$ and $\overline{Q_2}(S) = -S + c$ are linearly independent for every $c \in \mathbb{Z}_p$. The third condition doesn't have to be checked because Q has no degree-two term in $\boldsymbol{Y}$, and the requirement on the source is satisfied by definition.

For d-GROTH16, the relation polynomial in the case of type-1 groups is

$$Q(\boldsymbol{X}, \boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_3, \boldsymbol{C}) = \boldsymbol{Y}_1\boldsymbol{Y}_3 - \boldsymbol{Y}_2\boldsymbol{X}_4 - \boldsymbol{X}_1\boldsymbol{X}_2 - \sum_{i=0}^{\ell} \boldsymbol{C}_i\boldsymbol{X}_{n+5+i}\boldsymbol{X}_3 \,.$$

Again, polynomial Q is of the form covered by Theorem 1, and $\overline{Q_2}(S) = -S_3S_4^2$ is non-zero and thus linearly independent. For the third condition of the theorem, we show that the constant term in $\boldsymbol{S}_5$ is not zero, thereby proving that the whole expression does not vanish for any choice of $\boldsymbol{w}$ and $\boldsymbol{c}$. To do so, observe that the terms in $\boldsymbol{S}_1\boldsymbol{S}_2\boldsymbol{S}_3^2\boldsymbol{S}_4^2$ and $\boldsymbol{S}_3^3\boldsymbol{S}_4^2$ have coefficients $\boldsymbol{w}_{1,1}\boldsymbol{w}_{3,2} + \boldsymbol{w}_{1,2}\boldsymbol{w}_{3,1} - 1$ and $\boldsymbol{w}_{1,3}\boldsymbol{w}_{3,5} + \boldsymbol{w}_{1,5}\boldsymbol{w}_{3,3} - 1$, respectively. We show that both coefficients cannot be zero at the same time, without another monomial being non-zero. Indeed if, say, $\boldsymbol{w}_{1,1} \neq 0 \neq \boldsymbol{w}_{3,2}$ and $\boldsymbol{w}_{1,3} \neq 0 \neq \boldsymbol{w}_{3,5}$, then it must be $\boldsymbol{w}_{3,3} = 0$ (because the coefficient of $\boldsymbol{S}_3^4\boldsymbol{S}_4^2$ is $\boldsymbol{w}_{1,3}\boldsymbol{w}_{3,3}$, which must be zero if the whole expression vanishes). But then the coefficient of the term $\boldsymbol{S}_2\boldsymbol{S}_3^3\boldsymbol{S}_4^2$ is $\boldsymbol{w}_{1,3}\boldsymbol{w}_{3,2} \neq 0$. The other three cases are addressed similarly. □

In Appendix E, we prove the theorem below which reduces the hardness of the UK assumption in type-2 and type-3 groups to a *compiled* UK game in type-1 groups. This confirms that the UK assumption, in an intuitive sense, is the easiest in type-1 groups (as the adversary has more power over group elements and can, for example, compute the pairing of all group elements that it is given). The compiled game simply provides all group elements in the second source group in the first source group and makes appropriate modifications to the winning conditions.

Theorem 2 (For UK: GBM1-H $\Longrightarrow$ GBM3-H**).** *For any $\mathcal{A}_3$ against a UK3 game in* GBM3-H *for any second source group (of the same size as the first), there is an adversary $\mathcal{A}_1$ against UK1, the type-1 compilation of UK3 in* GBM1-H, *such that for all extractors $\mathcal{E}_1$ for $\mathcal{A}_1$ there is an extractor $\mathcal{E}_3$ for $\mathcal{A}_3$ such that*

$$\mathrm{Adv}^{\mathrm{uk3}}_{p,\mathbf{G},\mathcal{S},\mathcal{A}_3,\mathcal{E}_3} \leq \mathrm{Adv}^{\mathrm{uk1}}_{p,\mathbf{G},\mathcal{S},\mathcal{A}_1,\mathcal{E}_1} + \frac{q_0 q_1}{p} \,,$$

where p is the size of the groups, q_1 is an upper bound on the number of G_1 elements in the game and q_0 is an upper bound on the number of locally sampled group elements in G_2. Furthermore, the extractor $\mathcal{E}_3$ makes at most q_1 queries to its first source group oracles.

As corollaries, we obtain the hardness of the UK assumption in type-2 and type-3 groups. In particular Groth16 is also hard in type-2 and type-3 groups. One caveat to these generic results is that the type-2 and type-3 extractors that we build need *oracle access* to the group operation and hashing oracles. Despite this, it is conceivable that direct proofs of security can lead to standard oracle-free extractors.

We note that a straightforward reduction also holds for simple groups as in simple groups the adversary has less power than those in type-1 groups, and furthermore our type-1 extractor makes no oracle calls.

6 Soundness of UK in ABM3-H

In this section, we justify the soundness of the UK assumption in the ABM3-H. This result complements the GBM1-H hardness of this assumption as the two models are formally incomparable for knowledge assumptions.

If we consider the classical definition of algebraic adversary [36], we can trivially build an extractor: output the scalar representation returned by the adversary in the AGM as the linear relation between the outputs and the inputs. As mentioned, this justification does not consider the effect of hashing, and thus we consider algebraic adversaries in the ABM3-H. Here the trivial extractor is no longer valid as it may output nonzero coefficients for hash outputs.

Our result here is for a class of adversaries who return a relation polynomial Q of degree one in the output variables, with linearly independent coefficients for the linear terms. In Appendix C, we include a proof of the hardness of linear UK in the AGM-H (i.e., for simple groups).

In Appendix B we prove that a simple knowledge by Bellare, Fuchsbauer, and Scafuro [4] is secure in the AGM-H. The proof there serves as a "warm-up" to the more technical proof that we present below (as well as that in Appendix D for d-GROTH16).

Theorem 3 (Linear UK **holds in** ABM3-H**).** *Let* B *be a type-3 bilinear group scheme. Consider the class of low-degree PPT adversaries $\mathcal{A}$ and sources $\mathcal{S}$ with the following properties:*

1. *The relation polynomial Q returned by $\mathcal{A}_0$ is of the form*

$$Q(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{Y}_1, \boldsymbol{Y}_2, \boldsymbol{C}) = Q_0(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{C}) + \sum_{\nu=1}^{2} \sum_{i=1}^{n_\nu} Q_{\nu,i}(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{C}) \boldsymbol{Y}_{\nu,i} \,,$$

   where $m_\nu \coloneqq |\boldsymbol{X}_\nu| - 1$, $n_\nu \coloneqq |\boldsymbol{Y}_\nu|$, and $c \coloneqq |\boldsymbol{C}|$.
2. *For every Q and every $\boldsymbol{P}$ returned by $\mathcal{A}_0$, and every $\boldsymbol{c} \in \mathbb{Z}_p^c$, the polynomials $Q_{1,i}(\boldsymbol{P}_1(\boldsymbol{S}), \boldsymbol{P}_2(\boldsymbol{S}), \boldsymbol{c})$ and $Q_{2,i}(\boldsymbol{P}_1(\boldsymbol{S}), \boldsymbol{P}_2(\boldsymbol{S}), \boldsymbol{c})$ are separately linearly independent, where we set $\boldsymbol{P}_{1,0}(\boldsymbol{S}) \coloneqq \boldsymbol{P}_{2,0}(\boldsymbol{S}) \coloneqq 1$;*
3. *For every $\boldsymbol{P}_\nu$ returned by $\mathcal{A}_0$, $\mathcal{S}$ samples $\boldsymbol{s} \twoheadleftarrow \mathbb{Z}_p^k$ and returns $\boldsymbol{P}_\nu(\boldsymbol{s})$.*

*Let $q, d \colon \mathbb{N} \to \mathbb{N}$ be polynomial upper bounds on the total degrees of the $\boldsymbol{P}_{\nu,i}$ and Q, respectively. If (q,q)-*DLOG *holds for* B*, then* UK *holds for* B *in the*

Extractor $\mathcal{E}(\mathsf{trace}(\mathcal{A}))$:
parse $\mathsf{trace}(\mathcal{A}) = (r_{\mathcal{A}}, \gamma, [\boldsymbol{x}_\nu]_\nu, [\boldsymbol{h}_\mu]_\mu)$
$m_1 \leftarrow |[\boldsymbol{x}_1]_1|$; $m_2 \leftarrow |[\boldsymbol{x}_2]_2|$; $u_1 \leftarrow u_2 \leftarrow u_T \leftarrow 0$
$(Q, \boldsymbol{P}_\nu) \twoheadleftarrow \mathcal{A}_0^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\gamma; r_{\mathcal{A}})$
$(\boldsymbol{w}_\nu, \boldsymbol{v}_\nu, \boldsymbol{c}) \twoheadleftarrow \mathcal{A}_1^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\gamma, Q, \boldsymbol{P}_\nu, [\boldsymbol{x}_\nu]_\nu; r_{\mathcal{A}})$
// $\mathcal{A}$ encodes group elements
// $[\boldsymbol{y}_{\nu,i}]_\nu = \prod_{j=0}^{m_\nu} [\boldsymbol{w}_{\nu,ij}\boldsymbol{x}_{\nu,j}]_\nu \cdot \prod_{j=1}^{u_\nu} [\boldsymbol{v}_{\nu,ij}\boldsymbol{h}_{\nu,j}]_\nu$
return $(\boldsymbol{w}_1, \boldsymbol{w}_2)$

Oracle $\mathsf{H}_\mu(m)$:
$u_\mu \leftarrow u_\mu + 1$
return $\boldsymbol{h}_{\mu,u_\mu}$

Adversary $\mathcal{B}(\gamma, [t]_1, [t^2]_1, \ldots, [t^{q(\lambda)}]_1, [t]_2, [t^2]_2, \ldots, [t^{q(\lambda)}]_2)$:
parse $\gamma = (\cdot_1, g_1, \cdot_2, g_2, p, e)$; $u_1 \leftarrow u_2 \leftarrow u_T \leftarrow 0$; $U_1 \leftarrow U_2 \leftarrow U_T \leftarrow [\,]$
$\boldsymbol{\rho} \twoheadleftarrow \mathbb{Z}_p^k$; $\boldsymbol{\sigma} \twoheadleftarrow \mathbb{Z}_p^{*k}$; $(Q, \boldsymbol{P}_\nu) \twoheadleftarrow \mathcal{A}_0^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\gamma)$
for $\nu = 1$ to 2 do for $i = 1$ to m_ν do
$\boldsymbol{P}'_{\nu,i}(T) \leftarrow \boldsymbol{P}_{\nu,i}(\boldsymbol{\rho} + \boldsymbol{\sigma}T)$; parse $\boldsymbol{P}'_{\nu,i}(T) = \sum_{j=0}^{q(\lambda)} p'_{\nu,ij} T^j$; $[\boldsymbol{x}_{\nu,i}]_\nu \leftarrow \prod_{j=0}^{q(\lambda)} [p'_{\nu,ij} t^j]_\nu$
$(\boldsymbol{w}_\nu, \boldsymbol{v}_\nu, \boldsymbol{c}) \twoheadleftarrow \mathcal{A}_1^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\gamma, Q, \boldsymbol{P}_\nu, [\boldsymbol{x}_\nu]_\nu)$; $\boldsymbol{P}'_{1,0}(T) \leftarrow \boldsymbol{P}'_{2,0}(T) \leftarrow 1$
// $\mathcal{A}$ encodes group elements $[\boldsymbol{y}_{\nu,i}]_\nu = \prod_{j=0}^{m_\nu} [\boldsymbol{w}_{\nu,ij}\boldsymbol{x}_j]_\nu \cdot \prod_{j=1}^{u_\nu} [\boldsymbol{v}_{\nu,ij}\boldsymbol{h}_{\nu,j}]_\nu$
for $\nu = 1$ to 2 do for $i = 1$ to n_ν do
$\boldsymbol{Y}'_{\nu,i}(T) \leftarrow \sum_{j=0}^{m_\nu} \boldsymbol{w}_{\nu,ij} \boldsymbol{P}'_{\nu,j}(T) + \sum_{j=1}^{u} \boldsymbol{v}_{\nu,ij} H_{\nu,j}(T)$
$Q'(T) \leftarrow Q(\boldsymbol{P}'_1(T), \boldsymbol{P}'_1(T), \boldsymbol{Y}'_1(T), \boldsymbol{Y}'_2(T), \boldsymbol{c})$
if $(Q'(T) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{BerlekampRoot}(Q', p)$ else return 0
for $t' \in \mathsf{S}$ do if $([t']_1 = [t]_1)$ then return t'
return 0

Oracle $\mathsf{H}_\nu(m)$:
if $(m \notin \mathrm{Dom}(U_\nu))$ then
$u_\nu \leftarrow u_\nu + 1$; $\alpha_{\nu,u_\nu} \twoheadleftarrow \mathbb{Z}_p$; $\beta_{\nu,u_\nu} \twoheadleftarrow \mathbb{Z}_p^*$
$H_{\nu,u_\nu}(T) \leftarrow \alpha_{\nu,u_\nu} + \beta_{\nu,u_\nu} T$
$U_\nu[m] \leftarrow [\alpha_{\nu,u_\nu} + \beta_{\nu,u_\nu} t]_\nu$
return $U_\nu[m]$

Oracle $\mathsf{H}_T(m)$:
if $(m \notin \mathrm{Dom}(U_T))$ then
$\alpha \twoheadleftarrow \mathbb{Z}_p$
$U_T[m] \leftarrow [\alpha]_T$
return $U_T[m]$

Fig. 7. Top: Extractor $\mathcal{E}$ for the algebraic adversary $\mathcal{A}$ in the UK game. **Center and bottom:** Adversary $\mathcal{B}$ against (q, q)-DLOG. In all figures, μ and ν range over $\{1, 2, T\}$ and $\{1, 2\}$, respectively.

ABM3-H *with respect to the class of adversaries above. More precisely, for every adversary $\mathcal{A}$ as above, there exist an extractor $\mathcal{E}$ and an adversary $\mathcal{B}$ against* (q, q)-DLOG*, both with approximately the same running times as $\mathcal{A}$, such that*

$$\mathrm{Adv}^{\mathrm{uk}}_{\mathrm{B},\mathcal{A},\mathcal{E}}(\lambda) \leq \left(1 - \frac{d(\lambda)q(\lambda)}{2^{\lambda-1} - 1}\right)^{-1} \cdot \mathrm{Adv}^{(q,q)\text{-dlog}}_{\mathrm{B},\mathcal{B}}(\lambda)\,. \tag{3}$$

Proof. Fix a source $\mathcal{S}$ and an algebraic adversary $\mathcal{A}$ in the UK game as in the statement of the theorem, and define an extractor $\mathcal{E}$ as in Fig. 7 (top). This extractor essentially re-runs $\mathcal{A}$ on its view to obtain $\mathcal{A}$'s output $(\boldsymbol{w}_\nu, \boldsymbol{v}_\nu, \boldsymbol{c})$. By this we mean that $\mathcal{A}$ encodes elements as $[\boldsymbol{y}_{\nu,i}]_\nu = \prod_{j=0}^{m_\nu} [\boldsymbol{w}_{\nu,ij}\boldsymbol{x}_{\nu,j}]_\nu \cdot \prod_{j=1}^{u_\nu} [\boldsymbol{v}_{ij}\boldsymbol{h}_{\nu,j}]_\nu$, where $[\boldsymbol{x}_\nu]_\nu$ and $[\boldsymbol{h}_\nu]_\nu$ are the vector of input group elements and of hash replies.

The extractor then simply ignores the coefficients $\boldsymbol{v}_\nu$ pertaining to the hash values and returns $(\boldsymbol{w}_1, \boldsymbol{w}_2)$. Extractor $\mathcal{E}$ is intended to work correctly if $\boldsymbol{v}_1 =$

$\boldsymbol{v}_2 = 0$ in the representation returned by $\mathcal{A}$. We now show that if $\mathcal{A}$ wins the UK game, this will likely be the case.

To that end, consider the adversary $\mathcal{B}$ playing the (q, q)-DLOG game for B defined in Fig. 7 (center and bottom). In essence, $\mathcal{B}$ runs $\mathcal{A}$ and simulates the UK game. When preparing the group element inputs and answering hash queries, $\mathcal{B}$ embeds the (q, q)-DLOG instance it is tasked with solving, carefully hiding it with random offsets $\boldsymbol{\rho}$ and $\boldsymbol{\alpha}$ to ensure a correct distribution. By construction, t is a root of the polynomial $Q'(T)$ that $\mathcal{B}$ defines, which means that $\mathcal{B}$ will be successful in finding t by inspecting the roots of Q' whenever $Q'(T) \neq 0$.

We now show how to use adversary $\mathcal{B}$ to prove the bound (3) for $\mathcal{A}$ and $\mathcal{E}$. To that end, consider the following sequence of games:

G_0: This is the original (q, q)-DLOG game for B played by adversary $\mathcal{B}$.

G_1: This game proceeds as G_0, but when answering queries to H_ν ($\nu \in \{1, 2\}$), tables U_ν are populated in a different way. Specifically, upon a query m to H_ν, the game samples $\alpha'_{\nu,u} \twoheadleftarrow \mathbb{Z}_p$ and $\beta_{\nu,u} \twoheadleftarrow \mathbb{Z}_p^*$, and defines $U_\nu[m] \leftarrow [\alpha'_{\nu,u}]$. Consequently, it then sets $H_{\nu,u}(T) \leftarrow \alpha'_{\nu,u} + \beta_{\nu,u}(T - t)$. Similarly, polynomials $\boldsymbol{P}'_{\nu,i}(T)$ are now defined as $\boldsymbol{P}'_{\nu,i}(T) \leftarrow \boldsymbol{P}_{\nu,i}(\boldsymbol{\rho}' + \boldsymbol{\sigma}(T - t))$ for a $\boldsymbol{\rho}' \twoheadleftarrow \mathbb{Z}_p^k$ sampled instead of $\boldsymbol{\rho}$.

G_2: This game proceeds as G_1, but we set $\boldsymbol{P}'_{\nu,i}(T) \leftarrow \boldsymbol{P}_{\nu,i}(\boldsymbol{\rho}' + \boldsymbol{\Sigma}(T - t))$, where $\boldsymbol{\Sigma}$ are new vectors of variables. Similarly, the polynomials $H_{\nu,u}$ are now defined as $H_{\nu,u}(T, \boldsymbol{B}_\nu) \leftarrow \alpha'_{\nu,u} + \boldsymbol{B}_{\nu,u}(T - t)$, where $\boldsymbol{B}_{\nu,u}$ is a fresh variable for every oracle call. Accordingly, the polynomial Q'' constructed after running $\mathcal{A}$ is now in variables T, $\boldsymbol{\Sigma}$, $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$. After defining Q'', game G_2 progresses with sampling $\boldsymbol{\sigma} \twoheadleftarrow \mathbb{Z}_p^{*k}$ and $\boldsymbol{\beta}_\nu \twoheadleftarrow \mathbb{Z}_p^{*u_\nu}$, setting $Q'(T) \leftarrow Q''(T, \boldsymbol{\sigma}, \boldsymbol{\beta}_1, \boldsymbol{\beta}_2)$, and checking if $Q'(T) = 0$. From here on, G_2 proceeds as G_1.

We now argue that subsequent games have identical success probabilities.

$\mathrm{G}_0 \rightsquigarrow \mathrm{G}_1$. Notice that the distribution of the entries $U_\nu[m]$ is the same in G_0 and G_1 (in both games they are uniformly random group elements). The same holds for the vector $\boldsymbol{x}_\nu$, which in both games are obtained by evaluating $\boldsymbol{P}_\nu$ on random inputs. Therefore, $\Pr[\mathrm{G}_0] = \Pr[\mathrm{G}_1]$. Observe that, in G_1, $\boldsymbol{P}'_\nu(t) = \boldsymbol{P}_\nu(\boldsymbol{\rho}')$ and $\mathsf{H}_\nu(m) = [\alpha'_{\nu,u}]$, which means that the vector of inputs and the hash replies are computed exactly as in the UK game.

$\mathrm{G}_1 \rightsquigarrow \mathrm{G}_2$. Notice that $\mathcal{A}$ is completely oblivious to the polynomials $H_{\nu,u}$ while being run, so the simulation of $\mathcal{A}$ is identical in both games. Afterwards, G_2 derives the polynomial Q' computed in G_1 by substituting a random $\boldsymbol{\sigma}$ and $\boldsymbol{\beta}$ into Q''. As for the change to $\boldsymbol{P}'_\nu$, observe that in both games the vectors $\boldsymbol{x}_\nu$ are computed in the same way. Therefore, $\Pr[\mathrm{G}_1] = \Pr[\mathrm{G}_2]$.

We conclude the proof by studying the winning probability in G_2. Assume that, while being run in G_2, adversary $\mathcal{A}$ wins the UK game, i.e., it returns polynomials $(Q, \boldsymbol{P}_\nu)$ and an output $(\boldsymbol{w}_\nu, \boldsymbol{v}_\nu, \boldsymbol{c})$ for which the winning condition in UK is satisfied and extractor $\mathcal{E}$ fails to compute an appropriate representation of the outputs. The latter means that $\boldsymbol{v}_1 \neq 0 \neq \boldsymbol{v}_2$, i.e., there exist $1 \leq \nu^* \leq 2, 1 \leq i^* \leq m_\nu$ and j^* such that $\boldsymbol{v}_{\nu^*,i^*j^*} \neq 0$.

We now claim that the polynomial Q'' constructed in G_2 after running $\mathcal{A}$ is not identically zero. Indeed, consider the polynomial

$$R(\boldsymbol{S}, \boldsymbol{H}_1, \boldsymbol{H}_2) \coloneqq Q\Bigg(\boldsymbol{P}_\nu(\boldsymbol{S}), \sum_{j=0}^{m_\nu} \boldsymbol{w}_{\nu,ij} \boldsymbol{P}_{\nu,j}(\boldsymbol{S}) + \sum_{j=1}^{u_\nu} \boldsymbol{v}_{\nu,ij} \boldsymbol{H}_{\nu,j}, \boldsymbol{c}\Bigg)$$
$$= \sum_{\nu=1}^{2} \sum_{i=1}^{n_\nu} Q_{\nu,i}(\boldsymbol{P}_\nu(\boldsymbol{S}), \boldsymbol{c}) \Bigg(\sum_{j=0}^{m_\nu} \boldsymbol{w}_{\nu,ij} \boldsymbol{P}_{\nu,j}(\boldsymbol{S}) + \sum_{j=1}^{u_\nu} \boldsymbol{v}_{\nu,ij} \boldsymbol{H}_{\nu,j}\Bigg) + Q_0(\boldsymbol{P}_\nu(\boldsymbol{S}), \boldsymbol{c})\,.$$

The coefficient of $\boldsymbol{H}_{\nu^*,j^*}$ is $\sum_{i=1}^{n_{\nu^*}} Q_{\nu^*,i}(\boldsymbol{P}_1(\boldsymbol{S}), \boldsymbol{P}_2(\boldsymbol{S}), \boldsymbol{c}) \boldsymbol{v}_{\nu^*,ij^*}$, which is not zero because the polynomials $Q_{\nu^*,i}(\boldsymbol{P}(\boldsymbol{S}), \boldsymbol{c})$ are linearly independent by assumption and $\boldsymbol{v}_{\nu^*,i^*j^*} \neq 0$; therefore, $R \neq 0$. Now notice that

$$Q''(T, \boldsymbol{\Sigma}, \boldsymbol{B}_1, \boldsymbol{B}_2) = R(\boldsymbol{\rho}' + \boldsymbol{\Sigma}(T-t), \boldsymbol{\alpha}_1' + \boldsymbol{B}_1(T-t), \boldsymbol{\alpha}_2' + \boldsymbol{B}_2(T-t))\,,$$

which is non-zero by Lemma 2 and of degree in T at most $d(\lambda)q(\lambda)$. Again by Lemma 2, the leading coefficient in T of $Q''(T, \boldsymbol{\Sigma}, \boldsymbol{B}_1, \boldsymbol{B}_2)$ is of total degree at most $d(\lambda)q(\lambda)$, which means that for random $\boldsymbol{\sigma} \twoheadleftarrow \mathbb{Z}_p^{*k}$ and $\boldsymbol{\beta}_\nu \twoheadleftarrow \mathbb{Z}_p^{*u_\nu}$, this leading coefficient will be zero with probability at most $d(\lambda)q(\lambda)/(2^{\lambda-1}-1)$ by Lemma 1. Thus, with probability at least $1 - d(\lambda)q(\lambda)/(2^{\lambda-1}-1)$, $Q'(T) \neq 0$. We conclude by observing that whenever this happens, game G_2 will return 1, which means

$$\mathrm{Adv}^{(q,q)\text{-dlog}}_{\mathrm{B},\mathcal{B}}(\lambda) = \Pr[\mathrm{G}_0(\lambda)] = \Pr[\mathrm{G}_2(\lambda)] \geq \left(1 - \frac{d(\lambda)q(\lambda)}{2^{\lambda-1}-1}\right) \cdot \mathrm{Adv}^{\mathrm{uk}}_{\mathrm{B},\mathcal{A},\mathcal{E}}(\lambda)\,.$$

This concludes the proof. □

Our requirements from the polynomials in the above theorem are identical to those needed for the linear case of Theorem 1 (and those needed in simple groups in Theorem 7 in Appendix A). Hence we obtain the hardness of KEA1, KEA3, d-KZG, and d-PKE assumption in the AGM-H and ABM3-H settings.

Corollary 2. *Let* Γ *be a group scheme, and* $d\colon \mathbb{N} \to \mathbb{N}$ *a polynomial. (1a) If* DLOG *holds in* Γ*, then* KEA1 *holds in* Γ *in the* AGM-H*. (1b) If* 2-DLOG *holds in* Γ*, then* KEA3 *holds in* Γ *in the* AGM-H*. (1c) If* $(d+1)$-DLOG *holds in* Γ*, then* d-PKE *holds in* Γ *in the* AGM-H*.*

Let B *be a type-3 bilinear group scheme. (2a) If* $(d-1, d-1)$-DLOG *holds in* B*, then* d-KZG *holds in* B *in the* ABM3-H*. (2b) If* $(d+1, d+1)$-DLOG *holds in* B*, then* d-PKE *holds in* B *in the* ABM3-H*.*

In Appendix D we prove the following theorem, which establishes the hardness of d-GROTH16 in the ABM3-H.

Theorem 4 (d-GROTH16 **holds in** ABM3-H**).** *Let* B *be a type-3 bilinear group scheme, and* $d\colon \mathbb{N} \to \mathbb{N}$ *a polynomial. Then for every PPT algebraic adversary* $\mathcal{A}$ *against* d-GROTH16*, there exist an extractor* $\mathcal{E}$ *and an adversary* $\mathcal{B}$

against (q,q)-DLOG*, both with approximately the same running time as* $\mathcal{A}$*, such that*

$$\mathrm{Adv}^{d\text{-groth16}}_{\mathrm{B},\mathcal{A},\mathcal{E}}(\lambda) \leq \left(1 - \frac{2q(\lambda)}{2^{\lambda-1}-1}\right)^{-1} \cdot \mathrm{Adv}^{(q,q)\text{-dlog}}_{\mathrm{B},\mathcal{B}}(\lambda)\,,$$

where $q(\lambda) \coloneqq \max(3, d(\lambda)+1, 2d(\lambda)-2)$.

7 Conclusion and Relevance to Applications

We established in Theorem 1 that the UK assumption holds in bilinear generic groups for adversaries $\mathcal{A}_0$ that return flexible Q and $\boldsymbol{P}$ polynomials. The d-PKE, d-KZG, and d-GROTH16 assumptions are instances of UK, where $\mathcal{A}_0$ returns specific Q and $\boldsymbol{P}$ polynomials. We then prove that the UK assumption for linear Q also holds in ABM3-H. This implies that the d-PKE and d-KZG assumptions are also sound with respect to algebraic groups. We proved separately that d-GROTH16 holds in ABM3-H.

We may now base the *knowledge soundness* of the modified Groth16 SNARK on the d-GROTH16 assumption as follows. For any adversary against the scheme that outputs an accepting proof, there is also an adversary that outputs the coefficient representation of the proof based only on its input elements: simply run the d-GROTH16 extractor after running the adversary. Moreover, for any such adversary, there is a reduction to q-DLOG in the standard model: run the existing AGM reduction [36, Theorem 7.2], utilizing the coefficient representation output by the extractor as the coefficient representation needed by the AGM reduction. We obtain the following corollary.

Corollary 3. *Groth16 for degree-d QAPs defined over a bilinear group γ with random generators of order p with $d^2 \leq (p-1)/8$ is knowledge sound in the standard model based on the* d-GROTH16 *and* q-DLOG *assumptions with* $q = 2d-1$.[13]

The knowledge soundness of KZG polynomial commitments in the standard model directly follows from the d-KZG assumption.

However, when applying the d-KZG assumption to lift the AGM proof of, e.g., PLONK, to the standard model, the following subtlety arises. The reduction to the soundness of PLONK's PIOP requires the extraction of the committed polynomial at the time the commitment is sent—which corresponds to hashing in the Fiat–Shamir transformed SNARK. However, our extractor is only guaranteed to succeed when provided with the full view of an adversary that also outputs a verifying polynomial evaluation proof. To address this issue one would have to truncate the view of the adversary handed to the extractor to be only up to the point in which the adversary produces the commitment.[14]

[13] Note that we multiply by $\gamma\delta$, thus $[x^{d-2}t(x)/\delta]_1$ becomes $[\gamma x^{d-2}t(x)]_1$ of degree $2d-1$, hence $q = 2d-1$.

[14] We informed the authors of [33] that the same issue might arise in their simulation-extractability result for KZG polynomial commitments, at least when considering it in the standard model.

A fascinating direction is to extend the UK to interactive settings possibly with "online" extractors to enable the layered approach for complex security notions such as simulation extractability.

Acknowledgments

Pooya Farshim was supported in part by EPSRC grant EP/V034065/1. Patrick Harasser was funded by the Deutsche Forschungsgemeinschaft (DFG) – SFB 1119 – 236615297.

References

1. M. Abe, J. Groth, K. Haralambiev, and M. Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In *CRYPTO 2011*.
2. B. Bauer, P. Farshim, P. Harasser, and A. O'Neill. Beyond uber: Instantiating generic groups via PGGs. In *TCC 2022, Part III*.
3. B. Bauer, G. Fuchsbauer, and J. Loss. A classification of computational assumptions in the algebraic group model. In *CRYPTO 2020, Part II*.
4. M. Bellare, G. Fuchsbauer, and A. Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In *ASIACRYPT 2016, Part II*.
5. M. Bellare, V. T. Hoang, and S. Keelveedhi. Cryptography from compression functions: The UCE bridge to the ROM. In *CRYPTO 2014, Part I*.
6. M. Bellare and A. Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In *CRYPTO 2004*.
7. M. Bellare and A. Palacio. Towards plaintext-aware public-key encryption without random oracles. In *ASIACRYPT 2004*.
8. M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *EUROCRYPT'94*.
9. M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *EUROCRYPT 2006*.
10. E. R. Berlekamp. Factoring polynomials over finite fields. *Bell Labs Tech. J.*, 46(8), 1967.
11. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. On the indifferentiability of the sponge construction. In *EUROCRYPT 2008*.
12. J. Birkett and A. W. Dent. Security models and proof strategies for plaintext-aware encryption. *Journal of Cryptology*, 27(1), 2014.
13. N. Bitansky, R. Canetti, A. Chiesa, S. Goldwasser, H. Lin, A. Rubinstein, and E. Tromer. The hunting of the SNARK. *Journal of Cryptology*, 30(4), 2017.
14. N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS 2012*.
15. J. Black. The ideal-cipher model, revisited: An uninstantiable blockcipher-based hash function. In *FSE 2006*.
16. J. Black, P. Rogaway, and T. Shrimpton. Black-box analysis of the block-cipher-based hash-function constructions from PGV. In *CRYPTO 2002*.
17. D. Boneh and X. Boyen. Short signatures without random oracles. In *EUROCRYPT 2004*.
18. D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT 2005*.

19. D. Boneh and R. Venkatesan. Breaking RSA may not be equivalent to factoring. In *EUROCRYPT'98*.
20. D. R. L. Brown. The exact security of ECDSA. Contributions to IEEE P1363a, 2001.
21. M. Campanelli, A. Faonio, D. Fiore, A. Querol, and H. Rodríguez. Lunar: A toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. In *ASIACRYPT 2021, Part III*.
22. R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*.
23. L. Chen, Z. Cheng, and N. P. Smart. Identity-based key agreement protocols from pairings. *Int. J. Inf. Sec.*, 6(4), 2007.
24. A. Chiesa, Y. Hu, M. Maller, P. Mishra, P. Vesely, and N. P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In *EUROCRYPT 2020, Part I*.
25. J.-S. Coron, Y. Dodis, C. Malinaud, and P. Puniya. Merkle-Damgård revisited: How to construct a hash function. In *CRYPTO 2005*.
26. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO'98*.
27. I. Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *CRYPTO'91*.
28. R. A. Demillo and R. J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4), 1978.
29. A. W. Dent. Adapting the weaknesses of the random oracle model to the generic group model. In *ASIACRYPT 2002*.
30. A. W. Dent. The Cramer-Shoup encryption scheme is plaintext aware in the standard model. In *EUROCRYPT 2006*.
31. A. W. Dent. The hardness of the DHK problem in the generic group model. Cryptology ePrint Archive, Report 2006/156, 2006.
32. A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. Villar. An algebraic framework for Diffie-Hellman assumptions. In *CRYPTO 2013, Part II*.
33. A. Faonio, D. Fiore, M. Kohlweiss, L. Russo, and M. Zajac. From polynomial iop and commitments to non-malleable zksnarks. In *TCC 2023*.
34. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86*.
35. N. Fleischhacker, V. Goyal, and A. Jain. On the existence of three round zero-knowledge proofs. In *EUROCRYPT 2018, Part III*.
36. G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In *CRYPTO 2018, Part II*.
37. G. Fuchsbauer, A. Plouviez, and Y. Seurin. Blind schnorr signatures and signed ElGamal encryption in the algebraic group model. In *EUROCRYPT 2020, Part II*.
38. A. Gabizon, Z. J. Williamson, and O. Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019.
39. S. Galbraith, K. Paterson, and N. Smart. Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165, 2006.
40. A. Ghoshal and S. Tessaro. Tight state-restoration soundness in the algebraic group model. In *CRYPTO 2021, Part III*.
41. S. Goldwasser and Y. T. Kalai. Cryptographic assumptions: A position paper. In *TCC 2016-A, Part I*.
42. J. Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT 2010*.

43. J. Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT 2016, Part II*.
44. J. Groth and M. Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In *CRYPTO 2017, Part II*.
45. J. Groth and V. Shoup. On the security of ECDSA with additive key derivation and presignatures. In *EUROCRYPT 2022, Part I*.
46. S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols. In *CRYPTO'98*.
47. S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols. Cryptology ePrint Archive, Report 1999/009, 1999.
48. T. Holenstein, R. Künzler, and S. Tessaro. The equivalence of the random oracle model and the ideal cipher model, revisited. In *43rd ACM STOC*.
49. A. Joux. A one round protocol for tripartite Diffie–Hellman. *Journal of Cryptology*, 17(4), 2004.
50. J. Kastner, J. Loss, and J. Xu. On pairing-free blind signature schemes in the algebraic group model. In *PKC 2022, Part II*.
51. A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In *ASIACRYPT 2010*.
52. A. Kiayias, F.-H. Liu, and Y. Tselekounis. Practical non-malleable codes from l-more extractable hash functions. In *ACM CCS 2016*.
53. M. Lepinski. *On the Existence of 3-Round Zero-Knowledge Proofs*. PhD thesis, Massachusetts Institute of Technology, 2002.
54. H. Lipmaa. A unified framework for non-universal SNARKs. In *PKC 2022, Part I*.
55. M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In *ACM CCS 2019*.
56. U. M. Maurer. Abstract models of computation in cryptography (invited paper). In *10th IMA International Conference on Cryptography and Coding*.
57. M. Naor. On cryptographic assumptions and challenges (invited talk). In *CRYPTO 2003*.
58. V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2), 1994.
59. P. Paillier and D. Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In *ASIACRYPT 2005*.
60. B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*.
61. C. Ràfols and A. Zapico. An algebraic framework for universal and updatable SNARKs. In *CRYPTO 2021, Part I*.
62. L. Rotem and G. Segev. Algebraic distinguishers: From discrete logarithms to decisional uber assumptions. In *TCC 2020, Part III*.
63. A. Rupp, G. Leander, E. Bangerter, A. W. Dent, and A.-R. Sadeghi. Sufficient conditions for intractability over black-box groups: Generic lower bounds for generalized DL and DH problems. In *ASIACRYPT 2008*.
64. J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. Assoc. Comput. Mach.*, 27(4), 1980.
65. H. Shacham. *New Paradigms in Signature Schemes*. PhD thesis, Stanford University, 2005.
66. V. Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT'97*.
67. M. Zhandry. To label, or not to label (in generic groups). In *CRYPTO 2022, Part III*.

68. M. Zhandry and C. Zhang. The relationship between idealized models under computationally bounded adversaries. Cryptology ePrint Archive, Report 2021/240, 2021.
69. C. Zhang, H.-S. Zhou, and J. Katz. An analysis of the algebraic group model. In *ASIACRYPT 2022, Part IV*.
70. R. Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and algebraic computation EUROSAM*. Springer, Berlin-New York, 1979.

A Soundness of Linear UK in GGM-H

Here, we give a self-contained proof of the hardness of the UK in the GGM-H for the case of linear relation polynomials.

Theorem 5 (Linear UK **holds in** GGM-H**).** *Consider $d, p \in \mathbb{N}$ with p prime, and fix $\mathsf{G} \subseteq \{0,1\}^*$ with $|\mathsf{G}| = p$. Consider low-degree adversaries $\mathcal{A}$ and sources $\mathcal{S}$ with the following properties:*

1. *The relation polynomial Q returned by $\mathcal{A}_0$ is of the form*

$$Q(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{C}) = \sum_{i=1}^{n} Q_i(\boldsymbol{X}, \boldsymbol{C})\boldsymbol{Y}_i + Q_0(\boldsymbol{X}, \boldsymbol{C}) \,,$$

   where $m \coloneqq |\boldsymbol{X}| - 1$, $n \coloneqq |\boldsymbol{Y}|$, and $c \coloneqq |\boldsymbol{C}|$;
2. *For every Q and every $\boldsymbol{P}$ returned by $\mathcal{A}_0$, and every $\boldsymbol{c} \in \mathbb{Z}_p^c$, the polynomials $\overline{Q_i}$, $1 \le i \le n$, are linearly independent;*
3. *For every $\boldsymbol{P}$ returned by $\mathcal{A}_0$, $\mathcal{S}$ samples $\boldsymbol{s} \in \mathbb{Z}_p^k$ at random and returns $\boldsymbol{P}(\boldsymbol{s})$.*

Then the UK *assumption holds in the* GGM-H *with parameters (p, G) with respect to the class of adversaries above. More precisely, for every adversary $\mathcal{A}$ as above, there exists an extractor $\mathcal{E}$ such that*

$$\mathrm{Adv}^{\mathrm{uk}}_{p,\mathsf{G},\mathcal{S},\mathcal{A},\mathcal{E}} \le (m + 3q_{\mathsf{op}} + q_{\mathsf{H}} + 1)^2 \cdot \frac{d}{2p} \,.$$

Here q_{op} and q_{H} are upper bounds on the number of queries made by $\mathcal{A}$ to the respective oracles, $\boldsymbol{P}_0(\boldsymbol{S}) \coloneqq 1$, and $\overline{Q_i}(\boldsymbol{S}) \coloneqq Q_i(\boldsymbol{P}(\boldsymbol{S}), \boldsymbol{c})$.

Proof. Fix an adversary $\mathcal{A}$ in the UK game as in the statement of the theorem, and define an extractor $\mathcal{E}$ as in Fig. 8. This extractor essentially re-runs $\mathcal{A}_1$ on its view and observes its oracle queries, keeping track of the discrete logarithms of the elements queried by $\mathcal{A}_1$. Whenever $\mathcal{E}$ is unable to "explain" an element in G, it instead stores a fresh variable $\boldsymbol{R}_v$ in its table.

We claim that this extractor allows us to prove the bound in the theorem statement. To that end, consider the following sequence of games:

G_0: This is the original UK game with respect to source $\mathcal{S}$, adversary $\mathcal{A}$ and extractor $\mathcal{E}$.

G_1: This game proceeds as G_0, but the encoding τ is implemented via lazy sampling: Instead of sampling τ, the challenger of G_1 initializes a table $T \leftarrow [\,]$. Oracles op and H are then implemented via lazy sampling using table T.

G_2: This game proceeds as G_1, but the challenger replaces the values $\boldsymbol{x}_i$ generated by $\mathcal{S}$ with the corresponding polynomials $\boldsymbol{P}_i(\boldsymbol{S})$ evaluated at formal variables $\boldsymbol{S}$. Likewise, whenever it lazily samples a domain point in T, it instead saves a fresh variable $\boldsymbol{R}_v$. Notice that in this game, table T is populated exactly as table U in $\mathcal{E}$.

We now argue that the difference between the success probabilities in subsequent games is small.

Extractor $\mathcal{E}(\mathsf{trace}(\mathcal{A}))$:
parse $\mathsf{trace}(\mathcal{A}) = (r_{\mathcal{A}}, \gamma, \boldsymbol{x}, h_1, \ldots, h_q)$
$(Q, \boldsymbol{P}) \twoheadleftarrow \mathcal{A}_0(\gamma; r_{\mathcal{A}})$; $U \leftarrow [\,]$; $U[g] \leftarrow 1$; $o, v \leftarrow 0$
for $j = 1$ to m do $U[\boldsymbol{x}_j] \leftarrow \boldsymbol{P}_j(\boldsymbol{S})$
$(\boldsymbol{y}, \boldsymbol{c}) \twoheadleftarrow \mathcal{A}^{\overline{\mathsf{op}},\overline{\mathsf{H}}}(\boldsymbol{x}; r_{\mathcal{A}})$; if $\big((\exists i)(\boldsymbol{y}_i \notin \mathrm{Dom}(U))\big)$ then return 0
for $i = 1$ to n do parse $U[\boldsymbol{y}_i] = \sum_{j=1}^{m} \boldsymbol{w}_{ij}\boldsymbol{P}_j(\boldsymbol{S}) + \sum_k \boldsymbol{b}_{ik}\boldsymbol{R}_k$
return $\boldsymbol{w}$

Proc. $\overline{\mathsf{op}}(a, h_1, b, h_2)$:
$o \leftarrow o + 1$
if $(h_1 \notin \mathrm{Dom}(U))$ then $v \leftarrow v + 1$; $U[h_1] \leftarrow \boldsymbol{R}_v$
if $(h_2 \notin \mathrm{Dom}(U))$ then $v \leftarrow v + 1$; $U[h_2] \leftarrow \boldsymbol{R}_v$
if $(h_o \notin \mathrm{Dom}(U))$ then $U[h_o] \leftarrow aU[h_1] + bU[h_2]$
return h_o

Proc. $\overline{\mathsf{H}}(m)$:
$o \leftarrow o + 1$
if $(h_o \notin \mathrm{Dom}(U))$ then
  $v \leftarrow v + 1$; $U[h_o] \leftarrow \boldsymbol{R}_v$
return h_o

Fig. 8. Definition of the extractor $\mathcal{E}$ from the proof of Theorem 5.

$\mathrm{G}_0 \rightsquigarrow \mathrm{G}_1$. Notice that G_0 and G_1 have the same distribution, because the oracles given to $\mathcal{A}$ in the two games are distributed identically. In particular, this means $\Pr[\mathrm{G}_1] = \Pr[\mathrm{G}_0]$.

$\mathrm{G}_1 \rightsquigarrow \mathrm{G}_2$. Let Bad be the event in G_2 that there are two different polynomials in the domain of T which result in the same value when evaluating $\boldsymbol{S}$ and $\boldsymbol{R}$ at random $\boldsymbol{s}$ and $\boldsymbol{r}$. Then G_1 and G_2 are identical until Bad, and by the fundamental lemma of game playing we therefore have $|\Pr[\mathrm{G}_2] - \Pr[\mathrm{G}_1]| \leq \Pr[\mathsf{Bad}]$.

We bound the latter probability via Lemma 1. Consider an adversary $\mathcal{B}$ in the Schwartz–Zippel game which simulates G_2 to $\mathcal{A}$ and then returns all polynomials in the domain of T. Notice that if Bad occurs, then $\mathcal{B}$ wins the SZ-game, and that T contains at most $m + 3q_{\mathsf{op}} + q_{\mathsf{H}} + 1$ polynomials of degree at most d. By Lemma 1, $\Pr[\mathsf{Bad}] \leq (m + 3q_{\mathsf{op}} + q_{\mathsf{H}} + 1) \cdot d/2p$.

We conclude the proof by studying the winning probability of $\mathcal{A}$ in G_2. Notice that if the output of $\mathcal{A}$ is such that the polynomial Q is not satisfied, then $\mathcal{A}$ has trivially lost the game. So suppose that Q is satisfied. Comparing entries stored in T (recall that T implements a partial random injection), we obtain

$$\sum_{i=1}^{n} \overline{Q_i}\left(\sum_{j=0}^{m} \boldsymbol{w}_{ij}\boldsymbol{P}_j + \sum_k \boldsymbol{b}_{ik}\boldsymbol{R}_k\right) + \overline{Q_0} = 0\,,$$

as a polynomial in $\boldsymbol{S}$ and $\boldsymbol{R}$. We want to show that this implies $\boldsymbol{b}_{ik} = 0$ for all $1 \leq i \leq n$ and all k. Looking at the linear terms in $\boldsymbol{R}$, we obtain that for every k,

$$\sum_{i=1}^{n} \overline{Q_i}\boldsymbol{b}_{ik} = 0\,.$$

Recall that, by assumption, polynomials $\overline{Q_i}$ are linearly independent, which means that $\boldsymbol{b}_{ik} = 0$ for all $1 \leq i \leq n$ and all k. This shows that $\mathcal{E}$ returns an accurate representation of $\boldsymbol{y}$ in terms of $\boldsymbol{x}$, which concludes the proof. □

Game $\text{DH-KE}^{\mathcal{A}}_{\text{B},\mathcal{E}}(\lambda)$:

$\gamma \twoheadleftarrow \text{B}(1^\lambda)$; $([a]_1, [b]_2, [c]_1) \twoheadleftarrow \mathcal{A}(\gamma)$; $w \twoheadleftarrow \mathcal{E}(\mathsf{trace}(\mathcal{A}))$
return $(e([a]_1, [b]_2) = e([c]_1, [1]_2)) \wedge ([w]_1 \neq [a]_1) \wedge ([w]_2 \neq [b]_2)$

Fig. 9. Game defining the DH-KE assumption. Here, B is a type-3 bilinear group scheme.

B Soundness of DH-KE in ABM3-H

In this section we prove that DH-KE, a knowledge game introduced by Bellare, Fuchsbauer, and Scafuro [4] is secure in the AGM-H. The proof here serves as a "warm-up" to the more complex reductions for the UK and d-GROTH16 assumptions presented in Section 6 and Appendix D. We first recall the definition of DH-KE.

DH-KE [4]. Let B be a type-3 bilinear group scheme. We define the advantage of an adversary $\mathcal{A}$ and an extractor $\mathcal{E}$ in the DH-KE game for B as

$$\text{Adv}^{\text{dh-ke}}_{\text{B},\mathcal{A},\mathcal{E}}(\lambda) \coloneqq \Pr[\text{DH-KE}^{\mathcal{A}}_{\text{B},\mathcal{E}}(\lambda)]\,,$$

where the game DH-KE is defined in Fig. 9. Here, $\mathcal{E}$ returns an element $w \in \mathbb{Z}_p$. We say that DH-KE holds for B if for every PPT $\mathcal{A}$ there exists a PPT $\mathcal{E}$ such that $\text{Adv}^{\text{dh-ke}}_{\text{B},\mathcal{A},\mathcal{E}}$ is negligible. DH-KE for type-2 and type-1 bilinear group schemes is defined analogously.

Remark. A similar formulation of DH-KE where $\mathcal{A}$ returns $[c]_2$ instead of $[c]_1$ (and the winning condition becomes $(e([a]_1, [b]_2) = e([1]_1, [c]_2))$) is also possible. On the other hand, note that the version where $\mathcal{A}$ returns $[c]_T$ and the game checks if $(e([a]_1, [b]_2) = [c]_T)$ is clearly false if hashing into both source groups is allowed: $\mathcal{A}$ could hash any message to get $h_1 \in \mathsf{G}_1$ and $h_2 \in \mathsf{G}_2$, set $h_T \coloneqq e(h_1, h_2)$, and return (h_1, h_2, h_T), without knowing any discrete logarithms.

Theorem 6 (DH-KE **holds in** ABM3-H**).** *Let* B *be a type-3 bilinear group scheme, such that* $(1,1)$-DLOG *holds for* B*. Then* DH-KE *holds for* B *in the* ABM3-H*. More precisely, for every PPT algebraic adversary* $\mathcal{A}$ *against* DH-KE*, there exist an extractor* $\mathcal{E}$ *and an adversary* $\mathcal{B}$ *against* $(1,1)$-DLOG*, both with approximately the same running times as* $\mathcal{A}$*, such that*

$$\text{Adv}^{\text{dh-ke}}_{\text{B},\mathcal{A},\mathcal{E}}(\lambda) \leq \left(1 - \frac{2}{2^{\lambda-1}-1}\right)^{-1} \cdot \text{Adv}^{(1,1)\text{-dlog}}_{\text{B},\mathcal{B}}(\lambda)\,. \tag{4}$$

Proof. Fix an algebraic adversary $\mathcal{A}$ in the DH-KE game, and define an extractor $\mathcal{E}$ as in Fig. 10 (top). This extractor essentially re-runs $\mathcal{A}$ on its view to obtain $\mathcal{A}$'s output $(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w})$. If all coordinates of $\boldsymbol{u}$ except the first one are zero (i.e., the first element encoded by $\mathcal{A}$ is $[\boldsymbol{u}_0]_1$), $\mathcal{E}$ returns $\boldsymbol{u}_0$, otherwise $\boldsymbol{v}_0$. Extractor $\mathcal{E}$ is intended to work if all entries but possibly the first one in either $\boldsymbol{u}$

Extractor $\mathcal{E}(\mathsf{trace}(\mathcal{A}))$:
parse $\mathsf{trace}(\mathcal{A}) = (r_{\mathcal{A}}, \gamma, \boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_T)$; $q_1 \leftarrow q_2 \leftarrow q_T \leftarrow 0$
$(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}) \twoheadleftarrow \mathcal{A}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\gamma; r_{\mathcal{A}})$; parse $\boldsymbol{u} = (\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{q_1})$
if $(\boldsymbol{u}_1 = \cdots = \boldsymbol{u}_{q_1} = 0)$ then return $\boldsymbol{u}_0$ else return $\boldsymbol{v}_0$

Oracle $\mathsf{H}_\mu(m)$:
$q_\mu \leftarrow q_\mu + 1$
return $\boldsymbol{h}_{\mu,q_\mu}$

Adversary $\mathcal{B}(\gamma, [x]_1, [x]_2)$:
parse $\gamma = (\cdot_1, g_1, \cdot_2, g_2, \cdot_T, p, e)$; $q_1 \leftarrow q_2 \leftarrow 0$; $T_1 \leftarrow T_2 \leftarrow T_T \leftarrow [\,]$
$(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}) \twoheadleftarrow \mathcal{A}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\gamma)$
$P(X) \leftarrow \big(\boldsymbol{u}_0 + \sum_{i=1}^{q_1} \boldsymbol{u}_i H_{1,i}(X)\big)\big(\boldsymbol{v}_0 + \sum_{i=1}^{q_2} \boldsymbol{v}_i H_{2,i}(X)\big) - \big(\boldsymbol{w}_0 + \sum_{i=1}^{q_1} \boldsymbol{w}_i H_{1,i}(X)\big)$
if $(P(X) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{BerlekampRoot}(P, p)$ else return 0
for $x' \in \mathsf{S}$ do if $([x']_1 = [x]_1)$ then return x'
return 0

Oracle $\mathsf{H}_\nu(m)$:
if $(m \notin \mathrm{Dom}(T_\nu))$ then
  $q_\nu \leftarrow q_\nu + 1$; $\alpha_{\nu,q_\nu} \twoheadleftarrow \mathbb{Z}_p$; $\beta_{\nu,q_\nu} \twoheadleftarrow \mathbb{Z}_p^*$
  $H_{\nu,q_\nu}(X) \leftarrow \alpha_{\nu,q_\nu} + \beta_{\nu,q_\nu} X$; $T_\nu[m] \leftarrow [\alpha_{\nu,q_\nu} + \beta_{\nu,q_\nu} x]_\nu$
return $T_\nu[m]$

Oracle $\mathsf{H}_T(m)$:
if $(m \notin \mathrm{Dom}(T_T))$ then
  $\alpha \twoheadleftarrow \mathbb{Z}_p$
  $T_T[m] \leftarrow [\alpha]_T$
return $T_T[m]$

Fig. 10. Top: Extractor $\mathcal{E}$ for the algebraic adversary $\mathcal{A}$ in the DH-KE game. **Center and bottom:** Adversary $\mathcal{B}$ against $(1,1)$-DLOG. In all figures, μ and ν range over the sets $\{1, 2, T\}$ and $\{1, 2\}$, respectively.

or $\boldsymbol{v}$ are zero. We now show that if $\mathcal{A}$ returns an output satisfying the winning condition, this will likely be the case.

To that end, consider the adversary $\mathcal{B}$ playing the $(1,1)$-DLOG game for B defined in Fig. 10 (center and bottom). In essence, $\mathcal{B}$ runs $\mathcal{A}$ and stimulates the DH-KE game. When answering hash queries, $\mathcal{B}$ embeds the discrete logarithm instances it is tasked with solving into the replies, carefully hiding them with random offsets α_{ν,q_ν} to ensure a correct distribution. By construction, x is a root of the polynomial $P(X)$ that $\mathcal{B}$ defines, which means that $\mathcal{B}$ will be successful in finding x by inspecting the roots of P whenever $P(X) \neq 0$.

We now show how to use adversary $\mathcal{B}$ to prove the bound (4) for $\mathcal{A}$ and $\mathcal{E}$. To that end, consider the following sequence of games:

G_0: This is the original $(1,1)$-DLOG game for B played by adversary $\mathcal{B}$.

G_1: This game proceeds as G_0, but when answering queries to H_ν, table T_ν is populated in a different way. Specifically, upon a query m to H_ν, the game samples $\alpha'_{\nu,q_\nu} \twoheadleftarrow \mathbb{Z}_p$ and $\beta_{\nu,q_\nu} \twoheadleftarrow \mathbb{Z}_p^*$, and defines $T_\nu[m] \leftarrow [\alpha'_{\nu,q_\nu}]$. Consequently, it then sets $H_{\nu,q_\nu}(X) \leftarrow \alpha'_{\nu,q_\nu} + \beta_{\nu,q_\nu}(X - x)$.

G_2: This game proceeds as G_1, but the polynomials H_{ν,q_ν} are now defined as $H_{\nu,q_\nu}(X, \boldsymbol{S}_\nu) \leftarrow \alpha'_{\nu,q_\nu} + \boldsymbol{S}_{\nu,q_\nu}(X - x)$, where $\boldsymbol{S}_{\nu,q_\nu}$ is a fresh variable for every oracle call. Accordingly, the polynomial P' constructed after running $\mathcal{A}$ is now in variables X, $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$. After defining P', game G_2 progresses with sampling $\boldsymbol{s}_\nu \twoheadleftarrow \mathbb{Z}_p^{*q_\nu}$, setting $P(X) \leftarrow P'(X, \boldsymbol{s}_1, \boldsymbol{s}_2)$, and checking if $P(X) = 0$. From here on, G_2 proceeds as G_1. Observe that, both in G_1 and G_2, the game simulated to $\mathcal{A}$ is exactly the DH-KE game.

We now argue that subsequent games have identical success probabilities.

$\mathrm{G}_0 \rightsquigarrow \mathrm{G}_1$. Notice that the distribution of the entries $T_\nu[m]$ is the same in G_0 and G_1 (in both games they are uniformly random group elements). This means that moving from G_0 to G_1 amounts to renaming $\alpha_{\nu,q_\nu} + \beta_{\nu,q_\nu} x$ as α'_{ν,q_ν}, and changing $H_{\nu,q_\nu}(X)$ accordingly. Therefore, $\Pr[\mathrm{G}_0] = \Pr[\mathrm{G}_1]$.

$\mathrm{G}_1 \rightsquigarrow \mathrm{G}_2$. Notice that $\mathcal{A}$ is completely oblivious to the polynomials H_{ν,q_ν} while being run in either game, so the simulation of $\mathcal{A}$ is identical in both games. Afterwards, G_2 derives the polynomial P computed in G_1 by substituting random $\boldsymbol{s}_1$ and $\boldsymbol{s}_1$ into P'. Therefore, $\Pr[\mathrm{G}_1] = \Pr[\mathrm{G}_2]$.

We conclude the proof by studying the winning probability in G_2. Assume that, while running in G_2, adversary $\mathcal{A}$ wins the DH-KE game, i.e., it returns an output $(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w})$ for which the winning condition in DH-KE is satisfied and extractor $\mathcal{E}$ fails to compute an appropriate witness. The latter means that neither $\boldsymbol{u}_1 = \cdots = \boldsymbol{u}_{q_1} = 0$, nor $\boldsymbol{v}_1 = \cdots = \boldsymbol{v}_{q_2} = 0$, where q_1 and q_2 are the number of queries made by $\mathcal{A}$ to H_1 and H_2, respectively. Therefore, the polynomial $P'(X, \boldsymbol{S}_1, \boldsymbol{S}_2)$ defined in G_2 has degree two in X, whose leading coefficient is a polynomial $Q(\boldsymbol{S}_1, \boldsymbol{S}_2)$ of degree two by Lemma 2. Now notice that, by construction, the discrete logarithm x is always a root of $P(X)$, which means that G_2 will return 1 provided that $P(X) \neq 0$. We bound the probability of this event.

By Lemma 1, $Q(\boldsymbol{s}_1, \boldsymbol{s}_2) = 0$ for random $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ with probability at most $2/(2^{\lambda-1} - 1)$, which means that with probability at least $1 - 2/(2^{\lambda-1} - 1)$, $Q(\boldsymbol{s}_1, \boldsymbol{s}_2) \neq 0$, and thus in particular $P(X) \neq 0$. Therefore,

$$\mathrm{Adv}^{(1,1)\text{-dlog}}_{\mathrm{B},\mathcal{B}}(\lambda) = \Pr[\mathrm{G}_0(\lambda)] = \Pr[\mathrm{G}_2(\lambda)] \geq \left(1 - \frac{2}{2^{\lambda-1} - 1}\right) \cdot \mathrm{Adv}^{\text{dh-ke}}_{\mathrm{B},\mathcal{A},\mathcal{E}}(\lambda)\,,$$

which concludes the proof. □

C Soundness of UK in AGM-H

In this appendix, we give a self-contained proof of the hardness of the UK in the AGM-H for the case of linear relation polynomials.

Theorem 7 (Linear UK holds in AGM-H). *Let Γ be a group scheme. Consider the class of low-degree PPT adversaries $\mathcal{A}$ and sources $\mathcal{S}$ with the following properties:*

1. *The relation polynomial Q returned by $\mathcal{A}_0$ is of the form*

$$Q(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{C}) = \sum_{i=1}^{n} Q_i(\boldsymbol{X}, \boldsymbol{C}) \boldsymbol{Y}_i + Q_0(\boldsymbol{X}, \boldsymbol{C})\,,$$

   where $m \coloneqq |\boldsymbol{X}| - 1$, $n \coloneqq |\boldsymbol{Y}|$, and $c \coloneqq |\boldsymbol{C}|$.
2. *For every Q and every $\boldsymbol{P}$ returned by $\mathcal{A}_0$, and every $\boldsymbol{c} \in \mathbb{Z}_p^c$, the polynomials $Q_i(\boldsymbol{P}(\boldsymbol{S}), \boldsymbol{c})$, $1 \leq i \leq n$, are linearly independent, where we set $\boldsymbol{P}_0(\boldsymbol{S}) \coloneqq 1$;*

Extractor $\mathcal{E}(\mathsf{trace}(\mathcal{A}))$: | Oracle $\mathsf{H}(m)$:

parse $\mathsf{trace}(\mathcal{A}) = (r_{\mathcal{A}}, \gamma, [\boldsymbol{x}], [\boldsymbol{h}])$; $m \leftarrow |\boldsymbol{x}|$; $u \leftarrow 0$
$(Q, \boldsymbol{P}) \twoheadleftarrow \mathcal{A}_0^{\mathsf{H}}(\gamma; r_{\mathcal{A}})$; $(\boldsymbol{w}, \boldsymbol{v}, \boldsymbol{c}) \twoheadleftarrow \mathcal{A}_1^{\mathsf{H}}(\gamma, Q, \boldsymbol{P}, [\boldsymbol{x}]; r_{\mathcal{A}})$
// $\mathcal{A}$ encodes group elements $[\boldsymbol{y}_i] = \prod_{j=0}^{m} [\boldsymbol{w}_{ij}\boldsymbol{x}_j] \cdot \prod_{j=1}^{u} [\boldsymbol{v}_{ij}\boldsymbol{h}_j]$
return $\boldsymbol{w}$

$u \leftarrow u + 1$
return $[\boldsymbol{h}_u]$

Adversary $\mathcal{B}(\gamma, [t], [t^2], \ldots, [t^{q(\lambda)}])$:

parse $\gamma = (\cdot, g, p)$; $u \leftarrow 0$; $U \leftarrow [\,]$; $\boldsymbol{\rho} \twoheadleftarrow \mathbb{Z}_p^k$; $\boldsymbol{\sigma} \twoheadleftarrow \mathbb{Z}_p^{*k}$; $(Q, \boldsymbol{P}) \twoheadleftarrow \mathcal{A}_0^{\mathsf{H}}(\gamma)$
for $i = 1$ to m do
  $\boldsymbol{P}_i'(T) \leftarrow \boldsymbol{P}_i(\boldsymbol{\rho} + \boldsymbol{\sigma}T)$; parse $\boldsymbol{P}_i'(T) = \sum_{j=0}^{q(\lambda)} p_{ij}' T^j$; $[\boldsymbol{x}_i] \leftarrow \prod_{j=0}^{q(\lambda)} [p_{ij}' t^j]$
$(\boldsymbol{w}, \boldsymbol{v}, \boldsymbol{c}) \twoheadleftarrow \mathcal{A}_1^{\mathsf{H}}(\gamma, Q, \boldsymbol{P}, [\boldsymbol{x}])$; $\boldsymbol{P}_0'(T) \leftarrow 1$
// $\mathcal{A}$ encodes group elements $[\boldsymbol{y}_i] = \prod_{j=0}^{m} [\boldsymbol{w}_{ij}\boldsymbol{x}_j] \cdot \prod_{j=1}^{u} [\boldsymbol{v}_{ij}\boldsymbol{h}_j]$
for $i = 1$ to n do $\boldsymbol{Y}_i'(T) \leftarrow \sum_{j=0}^{m} \boldsymbol{w}_{ij}\boldsymbol{P}_j'(T) + \sum_{j=1}^{u} \boldsymbol{v}_{ij} H_j(T)$
$Q'(T) \leftarrow Q(\boldsymbol{P}'(T), \boldsymbol{Y}'(T), \boldsymbol{c})$
if $(Q'(T) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{BerlekampRoot}(Q', p)$ else return 0
for $t' \in \mathsf{S}$ do if $([t'] = [t])$ then return t'
return 0

Oracle $\mathsf{H}(m)$:

if $(m \notin \mathrm{Dom}(U))$ then
  $u \leftarrow u + 1$; $\alpha_u \twoheadleftarrow \mathbb{Z}_p$; $\beta_u \twoheadleftarrow \mathbb{Z}_p^*$; $H_u(T) \leftarrow \alpha_u + \beta_u T$; $U[m] \leftarrow [\alpha_u + \beta_u t]$
return $U[m]$

Fig. 11. Top: Extractor $\mathcal{E}$ for the algebraic adversary $\mathcal{A}$ in the UK game. **Center and bottom:** Adversary $\mathcal{B}$ against q-DLOG.

3. For every $\boldsymbol{P}$ returned by $\mathcal{A}_0$, $\mathcal{S}$ samples $\boldsymbol{s} \in \mathbb{Z}_p^k$ at random and outputs $\boldsymbol{P}(\boldsymbol{s})$. Let $q, d\colon \mathbb{N} \to \mathbb{N}$ be polynomial upper bounds on the total degrees of the $\boldsymbol{P}_i$ and Q, respectively. If q-DLOG holds for Γ, then UK *holds for Γ in the* AGM-H *with respect to the class of adversaries above. More precisely, for every adversary $\mathcal{A}$ as above, there exist an extractor $\mathcal{E}$ and an adversary $\mathcal{B}$ against q-DLOG, both with approximately the same running times as $\mathcal{A}$, such that*

$$\mathrm{Adv}^{\mathrm{uk}}_{\Gamma,\mathcal{A},\mathcal{E}}(\lambda) \leq \left(1 - \frac{d(\lambda)q(\lambda)}{2^{\lambda-1} - 1}\right)^{-1} \cdot \mathrm{Adv}^{q\text{-dlog}}_{\Gamma,\mathcal{B}}(\lambda)\,. \tag{5}$$

Proof. Fix a source $\mathcal{S}$ and an algebraic adversary $\mathcal{A}$ in the UK game as in the statement of the theorem, and define an extractor $\mathcal{E}$ as in Fig. 11 (top). This extractor essentially re-runs $\mathcal{A}$ on its view to obtain $\mathcal{A}$'s output $(\boldsymbol{w}, \boldsymbol{v}, \boldsymbol{c})$. By this we mean that $\mathcal{A}$ encodes elements as $[\boldsymbol{y}_i] = \prod_{j=0}^{m} [\boldsymbol{w}_{ij}\boldsymbol{x}_j] \cdot \prod_{j=1}^{u} [\boldsymbol{v}_{ij}\boldsymbol{h}_j]$, where $[\boldsymbol{x}]$ and $[\boldsymbol{h}]$ are the vector of input group elements and of hash replies.

The extractor then simply ignores the coefficients $\boldsymbol{v}$ pertaining to the hash values and returns $\boldsymbol{w}$. Extractor $\mathcal{E}$ is intended to work correctly if $\boldsymbol{v} = 0$ anyway in the representation returned by $\mathcal{A}$. We now show that if $\mathcal{A}$ returns an output satisfying the winning condition for the UK game, this will likely be the case.

To that end, consider the adversary $\mathcal{B}$ playing the q-DLOG game for Γ defined in Fig. 11 (center and bottom). In essence, $\mathcal{B}$ runs $\mathcal{A}$ and simulates the UK game.

When preparing the group element inputs and answering hash queries, $\mathcal{B}$ embeds the q-DLOG instance it is tasked with solving, carefully hiding it with random offsets $\boldsymbol{\rho}$ and $\boldsymbol{\alpha}$ to ensure a correct distribution. By construction, t is a root of the polynomial $Q'(T)$ that $\mathcal{B}$ defines, which means that $\mathcal{B}$ will be successful in finding t by inspecting the roots of Q' whenever $Q'(T) \neq 0$.

We now show how to use adversary $\mathcal{B}$ to prove the bound (5) for $\mathcal{A}$ and $\mathcal{E}$. To that end, consider the following sequence of games:

G_0: This is the original q-DLOG game for Γ played by adversary $\mathcal{B}$.

G_1: This game proceeds as G_0, but when answering queries to H, table U is populated in a different way. Specifically, upon a query m to H, the game samples $\alpha'_u \twoheadleftarrow \mathbb{Z}_p$ and $\beta_u \twoheadleftarrow \mathbb{Z}_p^*$, and defines $U_\nu[m] \leftarrow [\alpha'_u]$. Consequently, it then sets $H_u(T) \leftarrow \alpha'_u + \beta_u(T-t)$. Similarly, polynomials $\boldsymbol{P}'_i(T)$ are now defined as $\boldsymbol{P}'_i(T) \leftarrow \boldsymbol{P}_i(\boldsymbol{\rho}' + \boldsymbol{\sigma}(T-t))$ for a $\boldsymbol{\rho}' \twoheadleftarrow \mathbb{Z}_p^k$ sampled instead of $\boldsymbol{\rho}$.

G_2: This game proceeds as G_1, but we set $\boldsymbol{P}'_i(T) \leftarrow \boldsymbol{P}_i(\boldsymbol{\rho}' + \boldsymbol{\Sigma}(T-t))$, where $\boldsymbol{\Sigma}$ is a new vector of variables. Similarly, the polynomials H_u are now defined as $H_u(T, \boldsymbol{B}) \leftarrow \alpha'_u + \boldsymbol{B}_u(T-t)$, where $\boldsymbol{B}_u$ is a fresh variable for every oracle call. Accordingly, the polynomial Q'' constructed after running $\mathcal{A}$ is now in variables T, $\boldsymbol{\Sigma}$ and $\boldsymbol{B}$. After defining Q'', game G_2 progresses with sampling $\boldsymbol{\sigma} \twoheadleftarrow \mathbb{Z}_p^{*k}$ and $\boldsymbol{\beta} \twoheadleftarrow \mathbb{Z}_p^{*u}$, setting $Q'(T) \leftarrow Q''(T, \boldsymbol{\sigma}, \boldsymbol{\beta})$, and checking if $Q'(T) = 0$. From here on, G_2 proceeds as G_1.

We now argue that subsequent games have identical success probabilities.

$\mathrm{G}_0 \rightsquigarrow \mathrm{G}_1$. Notice that the distribution of the entries $U[m]$ is the same in G_0 and G_1 (in both games they are uniformly random group elements). The same holds for the vector $\boldsymbol{x}$, which in both games is obtained by evaluating $\boldsymbol{P}$ on random inputs. Therefore, $\Pr[\mathrm{G}_0] = \Pr[\mathrm{G}_1]$. Observe that, in G_1, $\boldsymbol{P}'(t) = \boldsymbol{P}(\boldsymbol{\rho}')$ and $\mathsf{H}(m) = [\alpha'_u]$, which means that the vector of inputs and the hash replies are computed exactly as in the UK game.

$\mathrm{G}_1 \rightsquigarrow \mathrm{G}_2$. Notice that $\mathcal{A}$ is completely oblivious to the polynomials H_u while being run, so the simulation of $\mathcal{A}$ is identical in both games. Afterwards, G_2 derives the polynomial Q' computed in G_1 by substituting a random $\boldsymbol{\sigma}$ and $\boldsymbol{\beta}$ into Q''. As for the change to $\boldsymbol{P}'$, observe that in both games $\boldsymbol{x}$ is computed in the same way. Therefore, $\Pr[\mathrm{G}_1] = \Pr[\mathrm{G}_2]$.

We conclude the proof by studying the winning probability in G_2. Assume that, while being run in G_2, adversary $\mathcal{A}$ wins the UK game, i.e., it returns polynomials $(Q, \boldsymbol{P})$ and an output $(\boldsymbol{w}, \boldsymbol{v}, \boldsymbol{c})$ for which the winning condition in UK is satisfied and extractor $\mathcal{E}$ fails to compute an appropriate representation of the outputs. The latter means that $\boldsymbol{v} \neq 0$, i.e., there exist $1 \leq i^* \leq m$ and j^* such that $\boldsymbol{v}_{i^* j^*} \neq 0$.

We now claim that the polynomial Q'' constructed in G_2 after running $\mathcal{A}$ is not identically zero. Indeed, consider the polynomial

$$R(\boldsymbol{S}, \boldsymbol{H}) \coloneqq Q\left(\boldsymbol{P}(\boldsymbol{S}), \sum_{j=0}^{m} \boldsymbol{w}_{ij} \boldsymbol{P}_j(\boldsymbol{S}) + \sum_{j=1}^{u} \boldsymbol{v}_{ij} \boldsymbol{H}_j, \boldsymbol{c}\right)$$

$$= \sum_{i=1}^{n} Q_i(\boldsymbol{P}(\boldsymbol{S}), \boldsymbol{c}) \left(\sum_{j=0}^{m} \boldsymbol{w}_{ij} \boldsymbol{P}_j(\boldsymbol{S}) + \sum_{j=1}^{u} \boldsymbol{v}_{ij} \boldsymbol{H}_j \right) + Q_0(\boldsymbol{P}(\boldsymbol{S}), \boldsymbol{c}) \,.$$

The coefficient of $\boldsymbol{H}_{j^*}$ is $\sum_{i=1}^{n} Q_i(\boldsymbol{P}(\boldsymbol{S}), \boldsymbol{c}) \boldsymbol{v}_{ij^*}$, which is not zero because the polynomials $Q_i(\boldsymbol{P}(\boldsymbol{S}), \boldsymbol{c})$ are linearly independent by assumption and $\boldsymbol{v}_{i^*j^*} \neq 0$; therefore, $R \neq 0$. Now notice that

$$Q''(T, \boldsymbol{\Sigma}, \boldsymbol{B}) = R(\boldsymbol{\rho}' + \boldsymbol{\Sigma}(T-t), \boldsymbol{\alpha}' + \boldsymbol{B}(T-t)) \,,$$

which is non-zero by Lemma 2 and of degree in T at most $d(\lambda)q(\lambda)$. Again by Lemma 2, the leading coefficient in T of $Q''(T, \boldsymbol{\Sigma}, \boldsymbol{B})$ is of total degree at most $d(\lambda)q(\lambda)$, which means that for random $\boldsymbol{\sigma} \twoheadleftarrow \mathbb{Z}_p^{*k}$ and $\boldsymbol{\beta} \twoheadleftarrow \mathbb{Z}_p^{*u}$, this leading coefficient will be zero with probability at most $d(\lambda)q(\lambda)/(2^{\lambda-1}-1)$ by Lemma 1. Thus, with probability at least $1 - d(\lambda)q(\lambda)/(2^{\lambda-1}-1)$, $Q'(T) \neq 0$. We conclude by observing that whenever this happens, game G_2 will return 1, which means

$$\mathrm{Adv}^{q\text{-dlog}}_{\Gamma,\mathcal{B}}(\lambda) = \Pr[\mathrm{G}_0(\lambda)] = \Pr[\mathrm{G}_2(\lambda)] \geq \left(1 - \frac{d(\lambda)q(\lambda)}{2^{\lambda-1}-1}\right) \cdot \mathrm{Adv}^{\mathrm{uk}}_{\Gamma,\mathcal{A},\mathcal{E}}(\lambda) \,.$$

This concludes the proof. □

D Proof of Theorem 4

Consider the sets P_1 and P_2 of polynomials defined by adversary $\mathcal{B}$ in Fig. 13, and define the polynomial

$$\begin{aligned} P' = &- \left(\sum_{t \in \mathsf{P}_1} \boldsymbol{w}_t^{(a)} t + \sum_{t=1}^{q_1} \boldsymbol{v}_t^{(a)} h_t^{(1)} \right) \cdot \left(\sum_{t \in \mathsf{P}_2} \boldsymbol{w}_t^{(b)} t + \sum_{t=1}^{q_2} \boldsymbol{v}_t^{(b)} h_t^{(2)} \right) \\ &+ \alpha\beta\gamma^2\delta^2 + \left(\sum_{i=0}^{\ell} f_i(\beta\delta U_i(x) + \alpha\delta V_i(x) + \delta W_i(x)) \right) \cdot \gamma^2\delta \\ &+ \left(\sum_{t \in \mathsf{P}_1} \boldsymbol{w}_t^{(c)} t + \sum_{t=1}^{q_1} \boldsymbol{v}_t^{(c)} h_t^{(1)} \right) \cdot \gamma\delta^2 \,, \end{aligned} \tag{6}$$

where q_1 and q_2 denote the number of queries made by $\mathcal{A}_1$ to H_1 and H_2, respectively. Suppose that

$$P'(\alpha, \beta, \gamma, \delta, x, (h_t^{(1)})_{t \in \mathsf{P}_1}, (h_t^{(2)})_{t \in \mathsf{P}_2}) = 0 \,. \tag{7}$$

Observe that there must be $K \in \{a, b, c\}$ such that at least one of the hash elements has a nonzero coefficient, since otherwise adversary $\mathcal{A}$ is not successful against the extractor $\mathcal{E}$. Let $h_{n_K}^{(b_K)}$ be such a hash element with nonzero coefficient $\boldsymbol{v}_{n_K}^{(b_K)}$, with $b_K = 2$ if $K = c$, and $b_K = 1$ otherwise. Our proof proceeds by case distinction, depending on the value of K.

Extractor $\mathcal{E}(\mathsf{trace}(\mathcal{A}))$:	Oracle $\mathsf{H}_\mu(m)$:
parse $\mathsf{trace}(\mathcal{A}) = (r_\mathcal{A}, \varpi, [\boldsymbol{x}_1]_1, [\boldsymbol{x}_2]_2, [\boldsymbol{h}_1]_1, [\boldsymbol{h}_2]_2, [\boldsymbol{h}_T]_T)$	$u_\mu \leftarrow u_\mu + 1$
$u_1 \leftarrow u_2 \leftarrow u_T \leftarrow 0$; $\mathsf{R} \twoheadleftarrow \mathcal{A}_0^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\gamma; r_\mathcal{A})$	return $[\boldsymbol{h}_{\mu,u_\mu}]_\mu$
$(\boldsymbol{f}, \boldsymbol{w}^{(a)}, \boldsymbol{v}^{(a)}, \boldsymbol{w}^{(c)}, \boldsymbol{v}^{(c)}, \boldsymbol{w}^{(b)}, \boldsymbol{v}^{(b)}) \twoheadleftarrow$	
$\quad \twoheadleftarrow \mathcal{A}_1^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\gamma, \mathsf{R}, [\boldsymbol{x}_1]_1, [\boldsymbol{x}_2]_2; r_\mathcal{A})$	
return $(\boldsymbol{w}^{(a)}, \boldsymbol{w}^{(c)}, \boldsymbol{w}^{(b)})$	

Fig. 12. Extractor $\mathcal{E}$ for the algebraic adversary $\mathcal{A}$ in the d-GROTH16 game. Here, μ ranges in the set $\{1, 2, T\}$.

First case: $K = b$. By considering the coefficient of the monomial $h_{n_b}^{(2)}$, we deduce from (6) and (7) that

$$-\left(\sum_{t\in\mathsf{P}_1} \boldsymbol{w}_t^{(a)} t + \sum_{t=1}^{q_1} \boldsymbol{v}_t^{(a)} h_t^{(1)}\right) \boldsymbol{v}_{n_b}^{(b)} = 0\,, \tag{8}$$

and since $\boldsymbol{v}_{n_b}^{(b)} \neq 0$ by assumption, we obtain

$$\sum_{t\in\mathsf{P}_1} \boldsymbol{w}_t^{(a)} t + \sum_{t=1}^{q_1} \boldsymbol{v}_t^{(a)} h_t^{(1)} = 0\,.$$

Equality (7) then becomes

$$\left(\sum_{i=0}^{\ell} f_i(\beta\delta U_i(x) + \alpha\delta V_i(x) + \delta W_i(x))\right) \cdot \gamma^2\delta + \left(\sum_{t\in\mathsf{P}_1} \boldsymbol{w}_t^{(c)} t + \sum_{t=1}^{q_1} \boldsymbol{v}_t^{(c)} h_t^{(1)}\right) \cdot \gamma\delta^2 + \alpha\beta\gamma^2\delta^2 = 0\,.$$

Then we can divide by $\gamma\delta^2$ and consider the constant term in h_t:

$$\left(\sum_{i=0}^{\ell} f_i\left(\beta U_i(x) + \alpha V_i(x) + W_i(x)\right)\right) \cdot \gamma + \left(\sum_{t\in\mathsf{P}_1} \boldsymbol{w}_t^{(c)} t\right) + \alpha\beta\gamma = 0\,.$$

Now notice that the constant term in δ has coefficient

$$\sum_{i=0}^{\ell} f_i\left(\beta U_i(x) + \alpha V_i(x) + W_i(x)\right) \cdot \gamma + \left(\sum_{i=\ell+1}^{m} \boldsymbol{w}^{(c)}_{(\beta U_i(x)+\alpha V_i(x)+W_i(x))\gamma} \left(\beta U_i(x) + \alpha V_i(x) + W_i(x)\right)\gamma\right) + \alpha\beta\gamma = 0\,.$$

This is a contradiction, because there is only one monomial in $\alpha\beta\gamma$, which has coefficient $1 \neq 0$.

Adversary $\mathcal{B}(\varpi, [t]_1, \ldots, [t^{q(\lambda)}]_1, [t]_2, \ldots, [t^{q(\lambda)}]_2)$:

parse $\varpi = (\cdot_1, g_1, \cdot_2, g_2, \cdot_T, p, e)$; ; $q_1 \leftarrow q_2 \leftarrow 0$; $T_1 \leftarrow T_2 \leftarrow T_T \leftarrow [\,]$; $f_0 \leftarrow 1$

$\mathsf{P} \leftarrow \{\gamma\delta, \alpha\gamma\delta, \beta\gamma\delta, \gamma^2\delta, \gamma\delta^2, x\gamma\delta\}$

$\mathsf{P}_1 \leftarrow \mathsf{P} \cup \{x^{(i)}\gamma\delta\}_{i=0}^{d-1} \cup \{x^{(i)}T(x)\gamma\}_{i=0}^{d-2}$
$\quad \cup \{(\beta U_i(x) + \alpha V_i(x) + W_i(x))\delta\}_{i=0}^{\ell} \cup \{(\beta U_i(x) + \alpha V_i(x) + W_i(x))\gamma\}_{i=\ell+1}^{m}$

$\mathsf{P}_2 \leftarrow \{\beta\gamma\delta, \gamma^2\delta, \delta^2\gamma\} \cup \{x^{(i)}\delta\gamma\}_{i=0}^{d-1}$

$\quad$ // Elements of $\mathsf{P}, \mathsf{P}_1, \mathsf{P}_2$, are considered symbols, not numbers

$(\mu_t)_{t\in\mathsf{P}} \twoheadleftarrow \mathbb{Z}_p^5$; $(\lambda_t)_{t\in\mathsf{P}} \twoheadleftarrow \mathbb{Z}_p^{*5}$; $(\ell, (U_i, V_i, W_i)_{i=0}^m, T) \twoheadleftarrow \mathcal{A}_0^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\varpi)$

$Q_{\alpha\gamma\delta}(R) \leftarrow (\mu_\alpha + \lambda_\alpha R)(\mu_\delta + \lambda_\delta R)(\mu_\gamma + \lambda_\gamma R)$

$Q_{\beta\gamma\delta}(R) \leftarrow (\mu_\beta + \lambda_\beta R)(\mu_\delta + \lambda_\delta R)(\mu_\gamma + \lambda_\gamma R)$

$Q_{\delta\gamma^2}(R) \leftarrow (\mu_\delta + \lambda_\delta R)(\mu_\gamma + \lambda_\gamma R)^2$; $Q_{\delta^2\gamma}(R) \leftarrow (\mu_\delta + \lambda_\delta R)^2(\mu_\gamma + \lambda_\gamma R)$

for $i = 1$ to $d(\lambda) - 1$ do $Q_{x^i\delta\gamma}(R) \leftarrow (\mu_x + \lambda_x R)^i(\mu_\delta + \lambda_\delta R)(\mu_\gamma + \lambda_\gamma R)$

for $i = 1$ to $d(\lambda) - 2$ do $Q_{x^iT(x)\gamma}(R) \leftarrow (\mu_x + \lambda_x R)^i(\mu_\gamma + \lambda_\gamma R) \cdot T(\mu_x + \lambda_x R)$

for $i = 0$ to ℓ do

$\quad Q_{(\beta U_i + \alpha V_i + W_i)\delta}(R) \leftarrow (\mu_\delta + \lambda_\delta R)\big((\mu_\beta + \lambda_\beta R) \cdot U_i(\mu_x + \lambda_x R)$
$\qquad + (\mu_\alpha + \lambda_\alpha R) \cdot V_i(\mu_x + \lambda_x R) + W_i(\mu_x + \lambda_x R)\big)$

for $i = \ell + 1$ to m do

$\quad Q_{(\beta U_i + \alpha V_i + W_i)\gamma}(R) \leftarrow (\mu_\gamma + \lambda_\gamma R)\big((\mu_\beta + \lambda_\beta R) \cdot U_i(\mu_x + \lambda_x R)$
$\qquad + (\mu_\alpha + \lambda_\alpha R) \cdot V_i(\mu_x + \lambda_x R) + W_i(\mu_x + \lambda_x R)\big)$

for $t \in \mathsf{P}_1$ do parse $Q_t(R) = \sum_{i=0}^{q(\lambda)} \lambda_i^{(t)} R^i$; $[\boldsymbol{x}_t]_1 \leftarrow \prod_{i=0}^{q(\lambda)} [\lambda_i^{(j)} x^{(i)}]_1$

for $t \in \mathsf{P}_2$ do parse $Q_t(R) = \sum_{i=0}^{q(\lambda)} \lambda_i^{(t)} R^i$; $[\boldsymbol{x}_t]_2 \leftarrow \prod_{i=0}^{q} [\lambda_i^{(j)} x^{(i)}]_2$

$((f_i)_{i=1}^{\ell}, \boldsymbol{w}^{(a)}, \boldsymbol{v}^{(a)}, \boldsymbol{w}^{(c)}, \boldsymbol{v}^{(c)}, \boldsymbol{w}^{(b)}, \boldsymbol{v}^{(b)}) \twoheadleftarrow \mathcal{A}_1^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(([\boldsymbol{x}_t]_1)_{t\in\mathsf{P}_1}, ([\boldsymbol{x}_t]_2)_{t\in\mathsf{P}_2})$

$Q_a(R) \leftarrow \sum_{t\in\mathsf{P}_1} \boldsymbol{w}_t^{(a)} Q_t(R) + \sum_{t=1}^{q_1} \boldsymbol{v}_t^{(a)}(\mu'_{1,j} + \lambda'_{1,j} R)$

$Q_c(R) \leftarrow \sum_{t\in\mathsf{P}_1} \boldsymbol{w}_t^{(c)} Q_t(R) + \sum_{t=1}^{q_1} \boldsymbol{v}_t^{(c)}(\mu'_{1,j} + \lambda'_{1,j} R)$

$Q_b(R) \leftarrow \sum_{t\in\mathsf{P}_2} \boldsymbol{w}_t^{(b)} Q_t(R) + \sum_{t=1}^{q_2} \boldsymbol{v}_t^{(b)}(\mu'_{2,j} + \lambda'_{2,j} R)$

$$Q'(R) \leftarrow -Q_a Q_b + Q_\alpha Q_\beta + \sum_{i=0}^{\ell} f_i Q_{(\beta U_i(x) + \alpha V_i(x) + W_i(x))\delta} \cdot Q_\gamma + Q_c Q_\delta$$

if $(Q'(R) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{BerlekampRoot}(Q', p)$ else return 0

for $t' \in \mathsf{S}$ do if $([t']_1 = [t]_1)$ then return t'

return 0

Oracle $\mathsf{H}_\nu(m)$:

if $(m \notin \mathrm{Dom}(T_\nu))$ then

$\quad q_\nu \leftarrow q_\nu + 1$; $\mu'_{\nu,q_\nu} \twoheadleftarrow \mathbb{Z}_p$; $\lambda'_{\nu,q_\nu} \twoheadleftarrow \mathbb{Z}_p^*$

$\quad T_\nu[m] \leftarrow [\mu'_{\nu,q_\nu} + \lambda'_{\nu,q_\nu} x]_\nu$

return $T_\nu[m]$

Oracle $\mathsf{H}_T(m)$:

if $(m \notin \mathrm{Dom}(T_T))$ then

$\quad \mu' \twoheadleftarrow \mathbb{Z}_p$

$\quad T_T[m] \leftarrow [\mu']_T$

return $T_T[m]$

Fig. 13. Adversary $\mathcal{B}$ against (q, q)-DLOG. Here, ν ranges in the set $\{1, 2\}$.

SECOND CASE: $K = a$. By considering the coefficient of the monomial $h_{n_a}^{(1)}$, we deduce from (6) and (7) that

$$-\left(\sum_{t\in\mathsf{P}_2} \boldsymbol{w}_t^{(b)} t + \sum_{t=1}^{q_2} \boldsymbol{v}_t^{(b)} h_t^{(2)}\right) \boldsymbol{v}_{n_a}^{(a)} + \boldsymbol{v}_{n_a}^{(c)} \gamma\delta^2 = 0\,. \tag{9}$$

Recall that $\boldsymbol{v}_{n_a}^{(a)} \neq 0$, which means that by looking at the coefficient of $h_t^{(2)}$, we obtain

$$(\forall 1 \leq t \leq q_2)(\boldsymbol{v}_t^{(b)} = 0)\,, \tag{10}$$

and equation (9) then becomes

$$-\left(\sum_{t\in\mathsf{P}_2} \boldsymbol{w}_t^{(b)} t\right) \boldsymbol{v}_{n_a}^{(a)} + \boldsymbol{v}_{n_a}^{(c)} \gamma\delta^2 = 0\,.$$

Again, looking at polynomials t for $t \in \mathsf{P}_2 \setminus \{\gamma\delta^2\}$, we obtain

$$(\forall t \in \mathsf{P}_2 \setminus \{\gamma\delta^2\})(\boldsymbol{w}_t^{(b)} = 0)\,. \tag{11}$$

Combining equations (6), (7), (10) and (11), we get

$$\begin{aligned}-\left(\sum_{t\in\mathsf{P}_1} \boldsymbol{w}_t^{(a)} t + \sum_{t=1}^{q_1} \boldsymbol{v}_t^{(a)} h_t^{(1)}\right) \cdot \boldsymbol{w}_{\gamma\delta^2}^{(b)} \gamma\delta^2 + \alpha\beta\gamma^2\delta^2 \\ + \left(\sum_{i=0}^{\ell} f_i \left(\beta\delta U_i(x) + \alpha\delta V_i(x) + \delta W_i(x)\right)\right) \cdot \gamma^2\delta \\ + \left(\sum_{t\in\mathsf{P}_1} \boldsymbol{w}_t^{(c)} t + \sum_{t=1}^{q_1} \boldsymbol{v}_t^{(c)} h_t^{(1)}\right) \cdot \gamma\delta^2 = 0\,.\end{aligned}$$

We can again divide the expression above by $\gamma\delta^2$ and obtain

$$\begin{aligned}-\left(\sum_{t\in\mathsf{P}_1} \boldsymbol{w}_t^{(a)} t + \sum_{t=1}^{q_1} \boldsymbol{v}_t^{(a)} h_t^{(1)}\right) \cdot w_{\gamma\delta^2}^{(b)} + \alpha\beta\gamma \\ + \left(\sum_{i=0}^{\ell} f_i \left(\beta U_i(x) + \alpha V_i(x) + W_i(x)\right)\right) \gamma \\ + \left(\sum_{t\in\mathsf{P}_1} \boldsymbol{w}_t^{(c)} t + \sum_{t=1}^{q_1} \boldsymbol{v}_t^{(c)} h_t^{(1)}\right) = 0\,.\end{aligned}$$

This is again a contradiction, because there is only one monomial in $\alpha\beta\gamma$, which has coefficient $1 \neq 0$.

THIRD CASE: $K = c$. By considering the coefficient of the monomial $h_{n_c}^{(1)}$, we deduce from (6) and (7) that

$$-\left(\sum_{t\in\mathsf{P}_2} \boldsymbol{w}_t^{(b)} t + \sum_{t=1}^{q_2} \boldsymbol{v}_t^{(b)} h_t^{(2)}\right) \boldsymbol{v}_{n_c}^{(a)} + \boldsymbol{v}_{n_c}^{(c)} \gamma\delta^2 = 0\,. \tag{12}$$

Recall that $\boldsymbol{v}_{n_c}^{(c)} \neq 0$, which means that by looking at the coefficient of $h_t^{(2)}$, we obtain

$$(\forall 1 \leq t \leq q_2)(\boldsymbol{v}_t^{(b)} = 0)\,, \tag{13}$$

and equation (12) becomes

$$-\left(\sum_{t\in\mathsf{P}_2} \boldsymbol{w}_t^{(b)} t\right) \boldsymbol{v}_{n_c}^{(a)} + \boldsymbol{v}_{n_c}^{(c)} \gamma\delta^2 = 0\,.$$

Again, looking at polynomials t for $t \in \mathsf{P}_2 \setminus \{\gamma\delta^2\}$, we obtain

$$(\forall t \in \mathsf{P}_2 \setminus \{\gamma\delta^2\})(\boldsymbol{w}_t^{(b)} = 0)\,. \tag{14}$$

Combining equations (6), (7), (13) and (14), we get

$$\begin{aligned}-\left(\sum_{t\in\mathsf{P}_1} \boldsymbol{w}_t^{(a)} t + \sum_{t=1}^{q_1} \boldsymbol{v}_t^{(a)} h_t^{(1)}\right) \cdot \boldsymbol{w}_{\gamma\delta^2}^{(b)} \delta^2\gamma + \alpha\beta\gamma^2\delta^2 \\ + \left(\sum_{i=0}^{\ell} f_i\left(\beta\delta U_i(x) + \alpha\delta V_i(x) + \delta W_i(x)\right)\right) \cdot \gamma^2\delta \\ + \left(\sum_{t\in\mathsf{P}_1} \boldsymbol{w}_t^{(c)} t + \sum_{t=1}^{q_1} \boldsymbol{v}_t^{(c)} h_t^{(1)}\right) \cdot \gamma\delta^2 = 0\,.\end{aligned}$$

By dividing this expression by $\gamma\delta^2$, we obtain

$$\begin{aligned}-\left(\sum_{t\in\mathsf{P}_1} \boldsymbol{w}_t^{(a)} t + \sum_{t=1}^{q_1} \boldsymbol{v}_t^{(a)} h_t^{(1)}\right) \cdot \boldsymbol{w}_{\delta^2\gamma}^{(b)} + \alpha\beta\gamma \\ + \left(\sum_{i=0}^{\ell} f_i\left(\beta U_i(x) + \alpha V_i(x) + W_i(x)\right)\right) \cdot \gamma \\ + \left(\sum_{t\in\mathsf{P}_1} \boldsymbol{w}_t^{(c)} t + \sum_{t=1}^{q_1} \boldsymbol{v}_t^{(c)} h_t^{(1)}\right) = 0\,.\end{aligned}$$

This is again a contradiction, because there is only one monomial in $\alpha\beta\gamma$, which has coefficient $1 \neq 0$.

Conclusion. We can therefore use Lemma 2 to deduce that the leading coefficient of

$$Q'(R) = P'\left((\mu_i + \lambda_i R)_{i\in\{\alpha,\beta,\gamma,\delta,x\}}, \left((\mu'_{\tau,j} + \lambda'_{\tau,j} R)_{j=1}^{q_\tau}\right)_{\tau\in\{1,2\}}\right)$$

is a polynomial of degree upper bounded by $q \cdot \deg(Q)$ with indeterminates $(\lambda_i)_{1\le i\le k}$ and $\left((\lambda'_j)_{1\le j\le q_\tau}\right)_{\tau\in\{1,2\}}$. Because the lambda's are completely hidden from $\mathcal{A}$'s point of view, we can choose them *a posteriori*, and then we can use the Schwartz–Zippel lemma, and deduce that with probability lower bounded by $1 - \frac{\deg(P')}{p-1}$ this polynomial is nonzero, then the whole polynomial $Q'(R)$ is neither nonzero. In this case, the discrete logarithm t of $[t]_1$ is by construction a root of this polynomial, and the search procedure will be successful.

To conclude the proof, we upper-bound the total degree d' of P':

$$\begin{aligned}
d' &= \deg\Bigg(\left(\sum_{t\in\mathsf{P}_1} \boldsymbol{w}_t^{(a)} t + \sum_{t=1}^{q_1} \boldsymbol{v}_t^{(a)} h_t^{(1)}\right)\cdot\left(\sum_{t\in\mathsf{P}_2} \boldsymbol{w}_t^{(b)} t + \sum_{t=1}^{q_2} \boldsymbol{v}_t^{(b)} h_t^{(2)}\right)\\
&\quad + \alpha\beta\gamma^2\delta^2 + \left(\sum_{i=0}^{\ell} f_i(\beta\delta U_i(x) + \alpha\delta V_i(x) + \delta W_i(x))\right)\cdot\gamma^2\delta\\
&\quad + \left(\sum_{t\in\mathsf{P}_1} \boldsymbol{w}_t^{(c)} t + \sum_{t=1}^{q_1} \boldsymbol{v}_t^{(c)} h_t^{(1)}\right)\cdot\gamma\delta^2\Bigg)\\
&\le \max\Bigg(\deg\left(\left(\sum_{t\in\mathsf{P}_1} \boldsymbol{w}_t^{(a)} t + \sum_{t=1}^{q_1} \boldsymbol{v}_t^{(a)} h_t^{(1)}\right)\cdot\left(\sum_{t\in\mathsf{P}_2} \boldsymbol{w}_t^{(b)} t + \sum_{t=1}^{q_2} \boldsymbol{v}_t^{(b)} h_t^{(2)}\right)\right),\\
&\quad \deg(\alpha\beta\gamma^2\delta^2), \deg\left(\left(\sum_{i=0}^{\ell} f_i(\beta\delta U_i(x) + \alpha\delta V_i(x) + \delta W_i(x))\right)\cdot\gamma^2\delta\right),\\
&\quad \deg\left(\left(\sum_{t\in\mathsf{P}_1} \boldsymbol{w}_t^{(c)} t + \sum_{t=1}^{q_1} \boldsymbol{v}_t^{(c)} h_t^{(1)}\right)\cdot\gamma\delta^2\right)\Bigg)\\
&= \max\Bigg(\deg\left(\sum_{t\in\mathsf{P}_1} \boldsymbol{w}_t^{(a)} t + \sum_{t=1}^{q_1} \boldsymbol{v}_t^{(a)} h_t^{(1)}\right) + \deg\left(\sum_{t\in\mathsf{P}_2} \boldsymbol{w}_t^{(b)} t + \sum_{t=1}^{q_2} \boldsymbol{v}_t^{(b)} h_t^{(2)}\right), 6,\\
&\quad \deg\left(\sum_{i=0}^{\ell} f_i(\beta\delta U_i(x) + \alpha\delta V_i(x) + \delta W_i(x))\right) + \deg\left(\gamma^2\delta\right),\\
&\quad \deg\left(\sum_{t\in\mathsf{P}_1} t + \sum_{t=1}^{q_1} \boldsymbol{v}_t^{(c)} h_t^{(1)}\right) + \deg\left(\gamma\delta^2\right)\Bigg)\\
&\le \max\Big(\max\left(\max_{t\in\mathsf{P}_1}\deg(t), \max_{t=1}^{q_1}\deg(h_t^{(1)})\right)\\
&\quad + \max\left(\max_{t\in\mathsf{P}_2}\deg(t), \max_{t=1}^{q_2}\deg(h_t^{(2)})\right), 6,
\end{aligned}$$

$$
\begin{aligned}
&3 + \max_{i=0}^{\ell} \deg\big(\beta\delta U_i(x) + \alpha\delta V_i(x) + \delta W_i(x)\big),\\
&\max\big(\max_{t\in \mathsf{P}_1} \deg(t), \max_{t=1}^{q_1} \deg(h_t^{(1)})\big) + 3\Big)\\
\le\ &\max\Big(\max(q,1) + \max(q,1), 6,\\
&3 + \max_{i=0}^{\ell} \deg\big(\beta\delta U_i(x) + \alpha\delta V_i(x) + \delta W_i(x)\big), \max(q,1) + 3\Big)\\
\le\ &\max(2q, d+3, q+3) = 2q\,.
\end{aligned}
$$

E GBM1-H $\Longrightarrow$ GBM3-H for UK

TYPE-1 COMPILATION. Given a type-3 UK assumption in GBM3-H, define its *type-1 compilation* as a UK assumption in GBM1-H that operates as follows. It runs $\mathcal{A}_0$ and $\mathcal{S}$ as before. As input to $\mathcal{S}_1$ it now provides all group elements in the second source group in the first source group (and no group elements in the second source group are provided). In place of elements in the second source group, adversary $\mathcal{A}_1$ now outputs group elements in the first source group. Accordingly, the winning condition is changed so that second source group checks are now done in the first source group. Type-1 compilation is defined analogously for type-2 UK assumptions.

Theorem 8 (For UK: GBM1-H $\Longrightarrow$ GBM3-H**).** *For any $\mathcal{A}_3$ against a UK3 game in* GBM3-H*, there is an adversary $\mathcal{A}_1$ against UK1, the type-1 compilation of UK3 in* GBM1-H*, such that for all extractors $\mathcal{E}_1$ for $\mathcal{A}_1$ there is an extractor $\mathcal{E}_3$ for $\mathcal{A}_3$ such that*

$$
\mathrm{Adv}^{\mathrm{uk3}}_{p,\mathbf{G},\mathcal{S},\mathcal{A}_3,\mathcal{E}_3} \le \mathrm{Adv}^{\mathrm{uk1}}_{p,\mathbf{G},\mathcal{S},\mathcal{A}_1,\mathcal{E}_1} + \frac{q_0 q_1}{p}\,,
$$

where p is the size of the groups, q_1 is an upper bound on the number of G_1 elements in the game and q_0 is an upper bound on the number of locally sampled group elements in G_2. Furthermore, the extractor $\mathcal{E}_3$ makes at most q_1 queries to its first source group oracles.

Proof. Given an adversary $\mathcal{A}_3$ in GBM3-H, consider an adversary $\mathcal{A}_1$ in GMB1-H as follows.

- Set GBM3.G_1 := GBM1.G_1 =: G_1 and GBM3.G_T := GBM1.G_T =: G_T.
- Simulate GBM3.G_2 =: G_2 via random injection $\phi : \mathsf{G}_1 \to \mathsf{G}_2$.
- Fix an encoding map $\mathsf{encode} : \mathsf{G}_2 \to \mathsf{G}_1$.[15]
- Receive group elements in G_1 as input. It computes the G_2 input elements via ϕ.

[15] Looking ahead, ideally we would like set encode to the inverse of ϕ. This choice would work for standard assumptions. For extractor assumptions, however, we will need to convert a view containing G_2 elements to one with G_1 elements. Without access to ϕ this is not possible, whereas with a fixed (deterministic) encoding we can do so.

- Forward G_1 hash on m to G_1 hash as $(1, m)$.
- Relay all other G_1 and G_T queries.
- Forward G_2 hash on m to G_1 hash as $(2, m)$, apply ϕ and return.
- Convert locally sampled elements outside those obtained from group operation oracles via encode. The injection is programmed to be compatible with encode. Let Bad be the event that the output of encode for a new G_2 element is an element that was queried to ϕ. Since G_1 elements are random, $\Pr[\mathsf{Bad}] \leq q_1 q_0 / |\mathsf{G}_1|$, where q_1 is an upper bound on the number of G_1 elements in the game and q_0 is an upper bound on the number of locally sampled group elements.
- Simulate G_2 operations as follows. Extracts coefficient representations of the queries based on the G_2 group elements seen so far. There are the initial G_2 input elements, any G_2 elements received from hash elements, any sampled G_2 elements. Use G_1 operation and the corresponding input/hash/encoded G_1 group elements to compute a G_1 element and then apply ϕ and return it.
- Simulate the GBM3 pairing operation on $\mathsf{G}_1 \times \mathsf{G}_2$ by extracting the coefficient representation for the queried G_2 element, forming the corresponding G_1 element using the coefficient representation, and then using GBM1 pairing and returning the output.
- When $\mathcal{A}_3$ outputs G_1, G_2, G_T group elements, leave the G_1 and G_T elements untouched. Convert the G_2 elements to G_1 elements using their coefficient representation or encode.

Adversary $\mathcal{A}_1$ simulates the environment that $\mathcal{A}_3$ expects. Suppose that $\mathcal{A}_3$'s inputs and outputs satisfy Q with some probability. Then $\mathcal{A}_1$'s inputs and outputs satisfy Q with the same probability as 1) the distribution of exponents of inputs is the same, and 2) the distribution of DLs of hashes and locally sampled elements is uniform in both cases.

Let $\mathcal{E}_1$ be an extractor for the above $\mathcal{A}_1$. Algorithm $\mathcal{E}_1$ expects a view containing: 1) $\mathcal{A}_1$'s query-answer pairs for G_1 and G_T, including all *extra queries* made by $\mathcal{A}_1$ (as a result of running $\mathcal{A}_3$). 2) $\mathcal{A}_1$'s query-answer pairs for the pairing operation. 3) The random coins of $\mathcal{A}_1$, i.e., (parts of) ϕ used to map G_1 elements to G_2 elements.

We are given $\mathsf{trace}[\mathcal{A}_3]$, which contains: 1) The inputs, which consist of group elements in G_1 and G_2 and also possibly G_T. 2) $\mathcal{A}_3$'s query-answer pairs for G_1, G_T, and G_2 oracles. 3) $\mathcal{A}_3$'s query-answer pairs for the pairing operation. Note that this view *does not* include the extra G_1 query-answer pairs of $\mathcal{A}_1$.

We build an extractor $\mathcal{E}_3^{\mathsf{G}_1}$ for $\mathcal{A}_3$ as follows. $\mathcal{E}_3$ first converts $\mathsf{trace}[\mathcal{A}_3]$ to $\mathsf{trace}[\mathcal{A}_1]$ as follows.

- Remove G_2 query-answer pairs from $\mathsf{trace}[\mathcal{A}_3]$.
- Since the messages m that are G_2 hashed are part of the view, with access to G_1 hash, $\mathcal{E}_3$ can recompute their G_1 hash outputs by sending (2,m).
- G_1 hash view inputs are changed from m to $(1, m)$.
- Convert locally sampled G_2 elements using the encoding.
- Extracts the coefficient representations for G_2 elements and then uses G_1 operation oracles to recompute the extra G_1 operation outputs.

– Sample randomness for ϕ that maps all G_1 elements to their corresponding G_2 elements.

Extractor $\mathcal{E}_3$ now runs $\mathcal{E}_1(\mathsf{trace}[\mathcal{A}_1])$ and outputs whatever it outputs. Whenever $\mathcal{E}_1$ is successful, so will be $\mathcal{E}_3$: The compiled game checks the same relation as the original game does. Moreover, algorithm $\mathcal{A}_1$ converts all output G_2 elements to G_1 elements using the same coefficient representation. □

Analogous reductions and conversion of views hold in the GBM2-H setting with differences that 1) we need to simulate an isomorphism from G_2 to G_1. This can be done using coefficient representations and the encoding from G_2 to G_1. 2) Extractor views can also be converted using these representations and oracle access.

We also note that the hardness of the UK in GBM1-H also implies the hardness of an appropriately compiled assumption in simple groups GGM-H. This is due to the fact that in simple groups the adversary has *less* power than in type-1 groups and furthermore our GBM1-H extractors do not use any oracles.

F Relations between Models

We start by defining compilations from type-safe games to those in other models.

RR-COMPILATION. Let $\mathrm{G}_{\mathsf{TS},p}$ be a type-safe game with hashing for a group of size p. Let G be a set of representations of size p. Following Zhandry's canonical translation [67], we define the random-representation (RR) compilation $\mathrm{G}_{\mathsf{RR},p,\mathsf{G}}$ of $\mathrm{G}_{\mathsf{TS},p}$ as follows. A counter is set to 0 and a random injection with image G is picked. A table T is initialized.

- Whenever $\mathrm{G}_{\mathsf{TS},p}$ accepts counters, $\mathrm{G}_{\mathsf{RR},p,\mathsf{G}}$ instead accepts representations.
- On output a representation h from any oracle, the counter c is incremented, $T[c]$ is set to h, and c is returned to $\mathrm{G}_{\mathsf{TS},p}$.
- Queries to the labeling and hashing oracles in $\mathrm{G}_{\mathsf{TS},p}$ are forwarded to the corresponding oracles with respect to the injection.
- The counters in queries to group operation are converted to group elements via look-ups in T and then forwarded to group operation oracle with respect to the injection.
- The equality oracle on counters c_1 and c_2 is replaced by bit-string comparison for $T[c_1]$ and $T[c_2]$.
- When $\mathrm{G}_{\mathsf{TS},p}$ outputs a counter to the adversary, this counter must have been previously seen (due to type-safety). In this case the corresponding representation in T is found and forwarded to the RR adversary.
- On an incoming representation h:
  - If there is some c such that $T[c] = h$, any such c is picked and forwarded to $\mathrm{G}_{\mathsf{TS},p}$. There is no difference in which c is picked: by type-safety the only way to operate on counter values is through the oracles, where the corresponding h will be recovered irrespective of which c is chosen.

- If h does not appear in T, then counter value is increased, $T[c]$ is set to h, and c is returned to $\mathrm{G}_{\mathsf{TS},p}$.

Suppose TS advantage is defined as some function applied to the probability of winning a TS game. We define RR advantage as the same function applied to the probability of winning the compiled RR game.

We start by proving that AGM-H security implies GGM-H security for standard (non-extractor) games. We state and prove our theorem for simple groups, but note that they easily extend to bilinear groups using analogous simulations of the available extra oracles.

Theorem 9 (AGM-H $\Longrightarrow$ GGM-H)**.** *Let $\mathrm{G}_{\mathsf{TS},p}$ be a single-stage type-safe game with hashing for a group of size p, and let G be a group, also viewed as a representation set of size at least p. We assume G is decidable. Let $\mathrm{G}_{\mathsf{RR},p,\mathsf{G}}$ be the RR-compilation of $\mathrm{G}_{\mathsf{TS},p}$ with respect to G, and let $\mathrm{G}_{\mathsf{Alg},p,\mathsf{G}}$ be the G-algebraic compilation of $\mathrm{G}_{\mathsf{TS},p}$. Then* AGM-H *security with respect to $\mathrm{G}_{\mathsf{Alg},p,\mathsf{G}}$ implies* GGM-H *security with respect to $\mathrm{G}_{\mathsf{RR},p,\mathsf{G}}$. More precisely, for any adversary $\mathcal{A}_{\mathsf{RR}}$ against $\mathrm{G}_{\mathsf{RR},p,\mathsf{G}}$ there is an adversary $\mathcal{A}_{\mathsf{Alg}}$ against $\mathrm{G}_{\mathsf{Alg},p,\mathsf{G}}$ such that*

$$\mathrm{Adv}^{\mathrm{G}_{\mathsf{RR},p,\mathsf{G}}}(\mathcal{A}_{\mathsf{RR}}) \leq \mathrm{Adv}^{\mathrm{G}_{\mathsf{Alg},p,\mathsf{G}}}(\mathcal{A}_{\mathsf{Alg}}) + \frac{(q+q_0)^2}{p}\,,$$

where q is an upper bound on the combined number of queries that $\mathrm{G}_{\mathsf{RR},p,\mathsf{G}}$ and $\mathcal{A}_{\mathsf{RR}}$ make to any of the group operation or hashing oracles, and q_0 is an upper bound on the number of group elements that $\mathcal{A}_{\mathsf{RR}}$ generates. Furthermore, $\mathcal{A}_{\mathsf{Alg}}$ places at most q queries to its oracles (and its run-time increases by that needed for q_0 local group exponentiations).

Proof. Given an adversary $\mathcal{A}_{\mathsf{RR}}$ as in the statement of the theorem, we build the required adversary $\mathcal{A}_{\mathsf{Alg}}$ based on $\mathcal{A}_{\mathsf{RR}}$ as follows. Throughout, $\mathcal{A}_{\mathsf{Alg}}$ lazily samples a two-sided random injection $\sigma^{\pm}\colon \mathsf{G} \to \mathsf{G}$. (See the end of this section, where we describe a general procedure for lazy sampling of random injections with arbitrary domains and ranges.) The table for this injection needs to be kept in the memory and hence the following reduction only applies to single-stage games.

- $\mathcal{A}_{\mathsf{Alg}}$ forwards bit values from $\mathrm{G}_{\mathsf{Alg},p,\mathsf{G}}$ to $\mathcal{A}_{\mathsf{RR}}$. Conversely, it forwards bit values from $\mathcal{A}_{\mathsf{RR}}$ to $\mathrm{G}_{\mathsf{Alg},p,\mathsf{G}}$.
- Group elements from $\mathrm{G}_{\mathsf{Alg},p,\mathsf{G}}$ are converted to random representations via the injection. Observe that when the injection is composed with the group exponentiation map, the resulting map is a random injection from $\mathbb{Z}_p$ to G.
- A group element h from $\mathcal{A}_{\mathsf{RR}}$ is processed as follows.
  - If h was computed in terms of the elements seen so far, $\mathcal{A}_{\mathsf{Alg}}$ computes and forwards this representation to $\mathrm{G}_{\mathsf{Alg},p,\mathsf{G}}$, as needed in the algebraically compiled game. Note that this computation can be performed by $\mathcal{A}_{\mathsf{Alg}}$ since $\mathcal{A}_{\mathsf{Alg}}$ sees the oracle queries of $\mathcal{A}_{\mathsf{RR}}$.

- If h is a new group element in G, adversary $\mathcal{A}_{\mathsf{Alg}}$ picks a random value r, and using the real group exponentiation procedure for G computes g^r, where g is a group generator which we assume is always made available.[16] $\mathcal{A}_{\mathsf{RR}}$ then *programs* the injection at g^r to h. Element g^r is explained using r and g to $\mathrm{G}_{\mathsf{Alg},p,\mathsf{G}}$.

In the above simulation, an inconsistency may arise if g^r was already queried to the injection, or h was already output by it. These events, however, happen with probability at most $(q+q_0)^2/p$, where q, q_0, and p are as in the statement of the theorem.

We claim that running $\mathcal{A}_{\mathsf{RR}}$ within $\mathcal{A}_{\mathsf{Alg}}$ against the G-algebraic compilation of $\mathrm{G}_{\mathsf{TS},p}$ is equivalent to running $\mathcal{A}_{\mathsf{RR}}$ against the RR-compilation of $\mathrm{G}_{\mathsf{TS},p}$ wrt. G. This is due to the fact that algorithm $\mathcal{A}_{\mathsf{Alg}}$ applies the same conversions to counters that the RR-compiler does to $\mathrm{G}_{\mathsf{TS},p}$, and furthermore it faithfully converts random representations to group elements. Thus, $\mathcal{A}_{\mathsf{Alg}}$ wins $\mathrm{G}_{\mathsf{Alg},p,\mathsf{G}}$ iff $\mathcal{A}_{\mathsf{RR}}$ wins $\mathrm{G}_{\mathsf{RR},p,\mathsf{G}}$. □

In the next two theorems, we show that the relationship between AGM and GGM as initially established by Zhandry [67] extends to models with hashing for standard games.

Theorem 10 (GGM-H $\Longrightarrow$ TSM-H)**.** *Let* $\mathrm{G}_{\mathsf{TS},p}$ *be a standard single-stage type-safe game with hashing and let* $\mathrm{G}_{\mathsf{RR},p,\mathsf{G}}$ *be its RR-compilation with respect to some* G*. Then* GGM-H *security with respect to* $\mathrm{G}_{\mathsf{RR},p,\mathsf{G}}$ *implies* TSM-H *security with respect to* $\mathrm{G}_{\mathsf{TS},p}$*. More precisely, for any adversary* $\mathcal{A}_{\mathsf{TS}}$ *against* $\mathrm{G}_{\mathsf{TS},p}$ *there is an adversary* $\mathcal{A}_{\mathsf{RR}}$ *against* $\mathrm{G}_{\mathsf{RR},p,\mathsf{G}}$ *such that*

$$\mathrm{Adv}^{\mathrm{G}_{\mathsf{TS},p}}(\mathcal{A}_{\mathsf{TS}}) \leq \mathrm{Adv}^{\mathrm{G}_{\mathsf{RR},p,\mathsf{G}}}(\mathcal{A}_{\mathsf{RR}})\,.$$

Here $\mathcal{A}_{\mathsf{RR}}$ *places the same number of queries to its oracles that* $\mathcal{A}_{\mathsf{TS}}$ *does.*

Proof. The proof follows that of Zhandry [67, Theorem 3.4]. Given an adversary $\mathcal{A}_{\mathsf{TS}}$ as in the statement of the theorem we build an adversary $\mathcal{A}_{\mathsf{RR}}$ that runs $\mathcal{A}_{\mathsf{TS}}$ as follows.

- $\mathcal{A}_{\mathsf{RR}}$ forwards bit values from $\mathrm{G}_{\mathsf{RR},p,\mathsf{G}}$ to $\mathcal{A}_{\mathsf{TS}}$. Conversely, it forwards bit values from $\mathcal{A}_{\mathsf{TS}}$ to $\mathrm{G}_{\mathsf{RR},p,\mathsf{G}}$.
- $\mathcal{A}_{\mathsf{RR}}$ converts representations h from $\mathrm{G}_{\mathsf{RR},p,\mathsf{G}}$ to counters as follows. It maintains a counter c, which starts at 1 and is always incremented. It also maintains a table T and sets $T[c]$ to h.
- $\mathcal{A}_{\mathsf{RR}}$ processes the counters output by $\mathcal{A}_{\mathsf{TS}}$ as follows. Any output counter must have been previously seen as a type-safe adversary cannot generate new counters on its own. Hence $\mathcal{A}_{\mathsf{RR}}$ can look up the corresponding representations in T and output it to $\mathrm{G}_{\mathsf{RR},p,\mathsf{G}}$. In contrast to Zhandry, who used the circuit model, here we need the game to be single-stage.

[16] Here, we do not assume that there are other local sampling/hashing procedures available.

– $\mathcal{A}_{\mathsf{RR}}$ answers equality queries on two counters by looking up their corresponding representations in T and then using bit-string comparison.

We claim that running $\mathcal{A}_{\mathsf{TS}}$ within $\mathcal{A}_{\mathsf{RR}}$ against the RR-compilation of $\mathrm{G}_{\mathsf{TS},p}$ wrt. G is equivalent to running $\mathcal{A}_{\mathsf{TS}}$ directly against $\mathrm{G}_{\mathsf{TS},p}$. This is due to the fact that algorithm $\mathcal{A}_{\mathsf{TS}}$ *undoes* the RR-compilation of $\mathrm{G}_{\mathsf{TS},p}$ modulo the values used for counters. However, the actual values of counters are irrelevant to a type-safe adversary as the counters are processed via the oracles and only their underlying DL are used. The reduction above however only changes the counter value but maintains the underlying DLs (corresponding to group representations). Thus, $\mathcal{A}_{\mathsf{RR}}$ wins the RR-compilation of $\mathrm{G}_{\mathsf{TS},p}$ wrt. G iff $\mathcal{A}_{\mathsf{TS}}$ wins $\mathrm{G}_{\mathsf{TS},p}$. □

A similar implication also holds in the reverse direction. The proof follows that of Zhandry [67, Theorem 3.5]. It differs from it in that we use the Turing machine model of type-safe games and the set of group representations G is fixed.

Theorem 11 (TSM-H $\implies$ GGM-H**).** *Let $\mathrm{G}_{\mathsf{TS},p}$ be a single-stage type-safe game with hashing and let $\mathrm{G}_{\mathsf{RR},p,\mathsf{G}}$ be its RR-compilation for some decidable G. Then* TSM-H *security with respect to $\mathrm{G}_{\mathsf{TS},p}$ implies* GGM-H *security with respect to $\mathrm{G}_{\mathsf{RR},p,\mathsf{G}}$. More precisely, for any $\mathcal{A}_{\mathsf{RR}}$ against $\mathrm{G}_{\mathsf{RR},p,\mathsf{G}}$ there is an adversary $\mathcal{A}_{\mathsf{TS}}$ against $\mathrm{G}_{\mathsf{TS},p}$ such that*

$$\mathrm{Adv}^{\mathrm{G}_{\mathsf{RR},p,\mathsf{G}}}(\mathcal{A}_{\mathsf{RR}}) \leq \mathrm{Adv}^{\mathrm{G}_{\mathsf{TS},p}}(\mathcal{A}_{\mathsf{TS}}) + \frac{(q+q_0)^2}{p}\,,$$

where q is an upper bound on the combined number of queries that $\mathrm{G}_{\mathsf{RR},p,\mathsf{G}}$ and $\mathcal{A}_{\mathsf{RR}}$ make to any of the group operations or hashing oracles, and q_0 is an upper bound on the number of group elements that $\mathcal{A}_{\mathsf{RR}}$ generates. Furthermore, $\mathcal{A}_{\mathsf{TS}}$ places at most $q+q_0$ queries to its oracles.

Proof. Given an adversary $\mathcal{A}_{\mathsf{RR}}$ as in the statement of the theorem we build an adversary $\mathcal{A}_{\mathsf{TS}}$ that runs $\mathcal{A}_{\mathsf{RR}}$ as follows.

– $\mathcal{A}_{\mathsf{TS}}$ forwards bit values from $\mathrm{G}_{\mathsf{TS},p}$ to $\mathcal{A}_{\mathsf{RR}}$. Conversely, it forwards bit values from $\mathcal{A}_{\mathsf{RR}}$ to $\mathrm{G}_{\mathsf{TS},p}$.
– $\mathcal{A}_{\mathsf{TS}}$ converts counters c from $\mathrm{G}_{\mathsf{TS},p}$ to random representations via lazy sampling of an injection (as described in at the end of this section). Here the injection's range is G and its domain is lazily updated to the set of counters up to the largest c so far. $\mathcal{A}_{\mathsf{TS}}$ uses the same representation for counters that carry equal exponents. The latter property is decided using the equality oracle and applying the injection, for example, to the first element in a counter's equivalence class. Algorithm $\mathcal{A}_{\mathsf{TS}}$ stores the mapping from counters to random representations in a table T for the injection as $T[c] \leftarrow h$.
– $\mathcal{A}_{\mathsf{TS}}$ handles a representation h received from $\mathcal{A}_{\mathsf{RR}}$ as follows.
  • If there is some c such that $T[c] = h$, counter c is forwarded to $\mathrm{G}_{\mathsf{TS},p}$. Note that storing and look-ups in T restricts the reduction to single-stage games.

$\underline{\sigma(x)\text{:}}$	$\underline{\sigma^-(y)\text{:}}$	$\underline{\mathsf{op}(y_1, y_2)\text{:}}$
$x' \leftarrow \mathsf{enc}_D(x)$	$y' \leftarrow \mathsf{dec}_R(y)$	$x_1 \leftarrow \sigma^-(y_1)$; $x_2 \leftarrow \sigma^-(y_2)$
$y' \leftarrow \pi(x')$	if $(y' = \bot)$ then return $\bot$	if $(x_1 = \bot \vee x_2 = \bot)$ then
$y \leftarrow \mathsf{enc}_R(y')$	$x' \leftarrow \pi^-(y')$	return $\bot$
return y	$x \leftarrow \mathsf{dec}_D(x')$	$y \leftarrow \sigma(x_1 + x_2)$
	return x	return y

Fig. 14. Lazy sampling of a random injection using a two-sided random permutation $\pi^{\pm}$. We assume $\pi^{\pm}$ returns $\bot$ when queried on $\bot$.

- If the representation h is not in T, but is in G, algorithm $\mathcal{A}_{\mathsf{TS}}$ needs to increment the counter. It can only do this via its oracle. To this end, $\mathcal{A}_{\mathsf{TS}}$ queries its *hash oracle* on a deterministically increasing sequence of messages to get c. Entry $T[c]$ is set to h and c is forwarded to $\mathrm{G}_{\mathsf{TS},p}$. (If h is not in G, $\bot$ is used.) We note that this step increases the number of queries to the label oracle by at most the number of elements that $\mathcal{A}_{\mathsf{RR}}$ outputs.

We let Bad be the event that the underlying exponent for c coincides with any other so far. The probability of Bad is at most $(q + q_0)^2/p$, with q, q_0 and p as in the statement of the theorem,

We claim that outside event Bad, running $\mathcal{A}_{\mathsf{RR}}$ within $\mathcal{A}_{\mathsf{TS}}$ against $\mathrm{G}_{\mathsf{TS},p}$ is equivalent to running $\mathcal{A}_{\mathsf{RR}}$ against the RR-compilation of $\mathrm{G}_{\mathsf{TS},p}$ wrt. G. This is due to the fact that algorithm $\mathcal{A}_{\mathsf{TS}}$ applies the same conversions to counters that the RR-compiler does to $\mathrm{G}_{\mathsf{TS},p}$. In other words, the reduction can be seen as a conversion that is either applied to $\mathcal{A}_{\mathsf{RR}}$ or to $\mathrm{G}_{\mathsf{TS},p}$. Thus, $\mathcal{A}_{\mathsf{RR}}$ wins $\mathrm{G}_{\mathsf{TS},p,\mathsf{G}}$ exactly when algorithm $\mathcal{A}_{\mathsf{TS}}$ wins $\mathrm{G}_{\mathsf{TS},p}$. □

Lazy sampling of injections. Recall that the lazy sampling of an injection $\sigma\colon D \to R$ for decidable sets $D, R \subseteq \{0,1\}^n$ and $|R| \geq |D|$ is sufficient to lazily sample a generic group with exponent space D and image R: the operation oracle can be instantiated using $\sigma^{\pm}$.

To lazily sample an injection $\sigma\colon D \to R$, we first embed D into $\{0,1\}^n$ via any uniquely decodable encoding scheme. Similarly we embed $\{0,1\}^n$ into R.

$$\mathsf{enc}_D\colon D \to \{0,1\}^n\,, \qquad \mathsf{enc}_R\colon \{0,1\}^n \to R\,,$$
$$\mathsf{dec}_D\colon \{0,1\}^n \to D\,, \qquad \mathsf{dec}_R\colon R \to \{0,1\}^n\,.$$

This, in particular, means that $\log(|D|) \leq n \leq \log(|R|)$. When $D = \mathbb{Z}_p$, a simple candidate for such an encoding is attaching sufficiently many redundant bits to the usual binary encoding of integers modulo p.

Let $\pi^{\pm}$ be a two-sided permutation on $\{0,1\}^n$. We implement $\sigma^{\pm}$ based on $\pi^{\pm}$ and encoding schemes above as shown in Figure 14. It is not hard to see that when $\pi^{\pm}$ is sampled uniformly at random, $\sigma^{\pm}$ is also a uniform two-sided injection from D to R. Furthermore, using a standard lazy sampling procedure for $\pi^{\pm}$, the injection $\sigma^{\pm}$ can be also lazily sampled.