IACR Communications in Cryptology
ISSN 3006-5496, Vol. 1, No. 3, 66 pages.

https://doi.org/10.62056/anr-zoja5

The Uber-Knowledge Assumption: A Bridge to the AGM

Balthazar Bauer[1], Pooya Farshim[2,3], Patrick Harasser[4] and Markulf Kohlweiss[5,6]

[1] Université de Versailles Saint-Quentin-en-Yvelines, France
[2] IOG, Switzerland
[3] Durham University, United Kingdom
[4] Technical University of Darmstadt, Germany
[5] University of Edinburgh, United Kingdom
[6] IOG, United Kingdom

Abstract. The generic-group model (GGM) and the algebraic-group model (AGM) have been exceptionally successful in proving the security of many classical and modern cryptosystems. These models, however, come with standard-model uninstantiability results, raising the question of 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-type problems. We justify the soundness of UK in both the bilinear GGM and the bilinear AGM. Along the way we extend these models to account for *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 the GGM and the AGM.

As example applications, we use the UK assumption to prove knowledge soundness of Groth's zero-knowledge SNARK (EUROCRYPT 2016) and of KZG polynomial commitments (ASIACRYPT 2010) in the standard model, where for the former we reuse the existing proof in the AGM without hashing.

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

1 Introduction

1.1 Background

Idealized models. 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 [FS87] and Bellare and Rogaway [BR93], random oracles, which idealize cryptographic hash functions, have been used to justify the security of a wide range of symmetric and asymmetric schemes. Subsequently, the random-permutation and the ideal-cipher models were used to study permutation-based cryptography (e.g., SHA3 [BDPV08]) and constructions using block ciphers [BRS02, CDMP05, HKT11]. This approach was also adapted to the setting of cryptographic groups by Nechaev [Nec94] and Shoup [Sho97], who showed the hardness of the discrete-logarithm problem in random groups with oracle access to the group operation.

E-mail: balthazar.bauer@ens.fr (Balthazar Bauer), pooya.farshim@gmail.com (Pooya Farshim), patrick.harasser@tu-darmstadt.de (Patrick Harasser), mkohlwei@ed.ac.uk (Markulf Kohlweiss)

Received: 2024-07-08 Accepted: 2024-09-02

Our focus in this work is on cryptographic assumptions related to groups. We start with a high-level overview of idealization of groups as put forward by Nechaev 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 [Sho97], aka. the random-representation (RR) model [Zha22], where group exponentiation is modeled as a random injection τ, and the group operation is defined via an oracle that is compatible with τ (i.e., elements are inverted under τ, added up, and fed back to τ). Another is Maurer's GGM [Mau05], aka. the type-safe (TS) model [Zha22], where group elements are replaced by abstract "handles" containing their corresponding discrete logarithms. The group operation oracle works on handles by placing the sum of the discrete logarithms under two given handles behind a third handle. Shoup's model has been extended to bilinear groups [BB04] and has been used to study a wide class of schemes, from standardized signature schemes [GS22] to structure-preserving signatures [AGHO11] and SNARKs [Gro16].

The algebraic-group model (AGM). An alternative approach towards modeling groups has emerged in more recent work. Motivated by the fact that group operations are observable in the GGM, it posits that adversaries always compute a representation of the group elements that they output in terms of those that they have seen thus far. This model is known as the AGM and was introduced by Fuchsbauer, Kiltz, and Loss [FKL18], though its roots trace back to the work of Boneh and Venkatesan [BV98], who considered restricted adversaries that implement straight-line programs. In a sense, the underlying groups are not idealized in the AGM; it is rather the adversary who is restricted and must "explain" its outputs in terms of its inputs. Recently, there has been significant interest in using the AGM to study cryptosystems [MBKM19, GT21, KLX22, FPS20, RZ21] and hardness assumptions [BFL20, RS20].

Uninstantiability results. 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 presented for the random-oracle model in the seminal work of Canetti, Goldreich, and Halevi [CGH98], which was later extended to the ideal-cipher [Bla06] and generic-group [Den02] models. Recently, Zhandry [Zha22] proved an analogous result for the AGM, thus separating the AGM from the standard model. We note that the schemes presented in these works are arguably "contrived" in that they are designed to fail, and as such do not disprove the security established in an idealized model of real-world cryptosystems that follow "good cryptographic practice" [KM07].[1]

Standard-model security. 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. As a result, under such assumptions, these constructions are placed outside the class of uninstantiable schemes. Moreover, if said assumptions can themselves be justified in an idealized model, one would gain additional assurance of their soundness, and at the same time establish a bridge from the idealized model to the standard model. Particularly successful examples of this "layered" approach to security include universal computational extractors for hash functions [BHK14], and the uber-assumption family for cryptographic group schemes [BBG05, EHK$^+$13, BFHO22].

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

1.2 Contributions

In this paper we continue the above line of work. We seek to identify assumptions to lift security of group-based cryptosystems proven in idealized models to the standard model.

Knowledge assumptions. In more detail, we are interested in knowledge assumptions for group schemes. In contrast to standard unpredictability and decisional problems, in knowledge assumptions one demands that for every adversary there exists a successful extractor. Thus, these assumptions have a higher "logical complexity"[2] and are not unconditionally falsifiable; see [Nao03, GK16] for further discussions.

Bridging assumptions for knowledge-type properties, such as the knowledge soundness of SNARKs, is an important and somewhat neglected area of investigation. Some schemes, e.g., Groth10 [Gro10], Pinocchio [PHGR13], Groth–Maller [GM17], and Marlin [CHM+20] are proven under dedicated knowledge assumptions. However, most popular schemes are proven directly in the GGM or AGM [Gro16, GWC19]. Besides SNARKs, knowledge assumptions also underlie the security of many other cryptosystems, from zero-knowledge proofs [Lep02,BP04a] to plaintext-aware encryption [Den06a], extractable collision resistant-hash functions (CRHFs) [BCCT12, BCC+17, KLT16] and non-malleable codes [KLT16].

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

Specific assumptions implied by UK have already appeared in the literature. Examples include the knowledge-of-exponent assumptions KEA1 and KEA3, which have been used to construct efficient three-round zero-knowledge protocols [HT98, BP04a] and plaintext-aware encryption [BD14], the d-KEA assumption utilized to build extractable CRHFs [BCCT12], the d-PKE assumption used in [Gro10,PHGR13] to build SNARKs, and our novel d-KZG assumption justifying the extractability of polynomial commitments, and thus the knowledge soundness of a number of practical in-use SNARKs [GWC19, CFF+21] via the framework of polynomial interactive oracle proofs (PIOPs) [BFS20].

We prove the implications above assuming hardness of the q-power discrete logarithm (q-DL) problem of Fuchsbauer, Kiltz, and Loss [FKL18]. To do so we must construct, for any adversary $\mathcal{A}$ in the considered notions, a corresponding extractor $\mathcal{E}$. This is done by transforming $\mathcal{A}$ into a UK adversary $\mathcal{B}$, for which there exists an extractor $\mathcal{F}$ by the assumed hardness of UK. Unfortunately, we cannot directly set $\mathcal{E} := \mathcal{F}$, because UK usually gives $\mathcal{F}$ more freedom in representing the outputs of $\mathcal{B}$ than $\mathcal{E}$ has for $\mathcal{A}$. We can, however, show via a reduction to q-DL that the additional coefficients that $\mathcal{F}$ can use will likely be zero, so that $\mathcal{E}$ can return the remaining output of $\mathcal{F}$. The reduction to q-DL (which we emphasize is in the standard model) follows an AGM-type strategy and embeds a q-DL challenge x into the inputs of $\mathcal{B}$ and $\mathcal{F}$. If the extra coefficients that $\mathcal{F}$ can use are not zero, we obtain a nontrivial polynomial equation involving x and can solve for x using a polynomial root-finding algorithm, e.g., Berlekamp's algorithm [Ber67].

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 requires. We emphasize that the UK assumption is a standard-model assumption[3], and thus an adversary may exploit hashing and other procedures to "obliviously sample" group elements to break it.

[2] Here we mean quantifier complexity: Knowledge assumptions are typically of the form "$(\forall\mathcal{A})(\exists\mathcal{E})(\ldots)$," while conventional assumptions take the form "$(\forall\mathcal{A})(\ldots)$," with no additional existential quantification.

[3] 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 **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 the (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 a critical shortcoming, as hashing can be simulated by exponentiating the group generator to random powers. This is indeed a valid approach for showing equivalence for standard unpredictability or decisional problems in models with and without hashing. On the other hand, the situation is different for knowledge assumptions. Indeed, observe that an extractor algorithm in the UK game is run on the adversary's view. When the hash oracle is simulated, this view contains the discrete logarithms of the hash outputs, information that is missing when hashing is done via an external oracle, where the view only contains the hash outputs themselves. This discrepancy 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 set.

Accordingly, we extend the GGM and the AGM with appropriate hashing oracles and call the resulting models GGM-H and AGM-H. This extension is straightforward for the 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 [CCS07, Definitions 2–4], where in type-1 and type-3 groups one can hash to all groups, but in the type-2 setting 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 [Zha22], who identified a problem with the original definition of the AGM related to leaking group elements one bit at a time [ZZK22]. Using the machinery of type-safe groups, where one can only operate on abstract group handles via oracles, we formalize the bilinear AGM with hashing for all three types of groups.

Layering: GGM **and** AGM **feasibility.** Given the observations above, we set out to justify the soundness of the UK assumption in both the GGM-H and the AGM-H. We do this for the class of relation polynomials (the polynomial in the winning condition that adversary inputs and outputs must satisfy) 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, while linear independence 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 the adversarial outputs. This class includes the polynomial needed to study the knowledge soundness of Groth16 [Gro16]. Our proof uses the Schwartz–Zippel lemma to transition to a setting where group operation oracles are implemented with respect to formal polynomials. The 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 vanish.

Our AGM-H proofs embed an instance of the q-DL problem into 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 that has the solution of the q-DL problem as one of its roots. As mentioned, we establish hardness for linear relation polynomials with linearly independent coefficients. For polynomials with quadratic terms, we directly prove hardness for the assumption that is needed in 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, we show that for specific classes of polynomials, sufficient conditions for the hardness of UK can be established. We believe that the restrictions we have identified are meaningful in the sense that they are sufficient to capture a number of knowledge assumptions in the literature, and also base the security of existing schemes on the UK. Generalizing UK hardness in GGM-H or AGM-H to a larger class of polynomials (e.g., quadratic polynomials with multiple degree-two terms in the adversary output variables) 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 assumption, ignore the hashed group elements. We can then reuse the already existing AGM reduction to q-DL 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.[4] We note that the lifting is from AGM without hashing, but our assumption is justified under the "weaker" AGM with hashing.

Related work. The only other work that we are aware of that proves statements about SNARKs in the AGM with hashing is that of Lipmaa [Lip22]. However, [Lip22] reproves security from scratch in the extended model with hashing, and does not formulate a plausible knowledge assumption for lifting the security of Groth16 to the standard model.

In concurrent and independent work, Lipmaa, Parisella and Siim [LPS23] introduce the AGM with oblivious sampling, an extension of AGM where adversaries can sample group elements obliviously via an oracle. Roughly speaking, in this model parties can query an oracle on admissible distributions $\mathcal{S}$ over $\mathbb{Z}_p$ and admissible encodings $E\colon \mathbb{Z}_p \to \mathsf{G}$. The oracle then samples $s \twoheadleftarrow \mathcal{S}$ and returns $(s, E(s))$. Our work is technically and conceptually incomparable to [LPS23]. Indeed, oblivious sampling cannot be replicated (due to the random choice of s), whereas hashing can be, both by the honest and adversarial parties. Also, we do not investigate general encodings and instead consider standard encoding via exponentiation. Finally, we emphasize that UK is a *standard-model* assumption, on which one can base the standard-model security of schemes. Adversaries against UK may hash or obliviously sample elements in arbitrary ways. Our feasibility results in idealized models (with hashing) provide supporting evidence for the soundness of UK. In contrast, the results of [LPS23] hold in the AGM with oblivious sampling, which is an idealized setting.

Future work. After its initial publication [BBG05], the uber-assumption family was extended in a series of works to hardness for rational functions [RLB+08], interactive problems [BFL20], matrix-type problems [EHK+13], and high-entropy sources [BFHO22]. A rich set of relations between notions of hardness has also been established in these works and others [BFL20, RS20]. 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 [GM17]. Can the reach of UK be extended to these settings while retaining soundness in the (bilinear) GGM-H and AGM-H?

[4] Lifting is logically different from layering: The former takes the form (Model $\Longrightarrow$ Application) $\Longrightarrow$ (Assumption $\Longrightarrow$ Application), while the latter Model $\Longrightarrow$ Assumption $\Longrightarrow$ Application.

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, where we also extend these models to include hashing. Section 4 contains the definition of the uber-knowledge assumption as well as some specific knowledge assumptions implied by UK. As a "warm-up" for the analysis of UK in idealized models, we study in Section 5 the soundness of the Diffie–Hellman knowledge of exponent assumption in the bilinear GGM and the bilinear AGM with hashing. In Sections 6 and 7, we then prove hardness of UK in these models. We conclude in Section 8 with an example application to Groth16.

2 Preliminaries

Basic notation. We denote by $\mathbb{Z}$ and $\mathbb{N} := \mathbb{Z}_{\geq 1}$ the sets of integers and of natural numbers, and by $\{0,1\}^*$ the set of finite-length bitstrings. For $n \in \mathbb{N}$, we let $\mathbb{Z}_n$ be the ring of integers modulo n; if $n = p$ is prime, then $\mathbb{F}_p := \mathbb{Z}_p$ is a field. The security parameter is denoted by λ, with unary representation 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. A table T is a set of pairs (x, y) without collisions in the first coordinate, and we write $T[x] \leftarrow y$ to mean that any pair $(x, \cdot)$ is removed from T, and the pair (x, y) is added to T. We let $\mathrm{Dom}(T)$ denote the set of all values x such that $(x, y) \in T$ for some y, and similarly $\mathrm{Rng}(T)$ denotes the set of all values y such that $(x, y) \in T$ for some x. Vectors are written in boldface and, depending on the context, their entries are indexed starting from 0 or 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 $[a] := g^a$. Similarly, if γ is a bilinear group and $a \in \mathbb{Z}_p$, then $[a]_\mu := g_\mu^a$ ($\mu \in \{1, 2, T\}$), where g_μ is the generator of the μ-th group. We extend this notation to vectors of exponents: If $\boldsymbol{a} \in \mathbb{Z}_p^\ell$, then $[\boldsymbol{a}] := (g^{\boldsymbol{a}_i})_{i=1}^\ell$, and similarly for bilinear groups with the appropriate subscripts. Note that this notation does not mean that the algorithm producing the group element "knows" its discrete logarithm wrt. the fixed generator. For an algorithm $\mathcal{A}$, we denote by $\mathcal{R}_\mathcal{A}(\lambda)$ the random variable returning random coins for $\mathcal{A}$ when run on security parameter λ. The trace (or view) of $\mathcal{A}$, 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 [BR06]**.** We use the code-based game-playing framework of Bellare and Rogaway. A game G is an algorithm run together with several parties, among which there is an adversary $\mathcal{A}$. The game starts by 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, G 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 G, who verifies the purported solution and returns a decision bit. We say that $\mathcal{A}$ wins game G if the final output of the game is 1; we then write $\mathrm{G}^\mathcal{A} = 1$, and let $\Pr[\mathrm{G}^\mathcal{A}] := \Pr[\mathrm{G}^\mathcal{A} = 1]$. Other parties may also feature in the game, according to its description.

Let G_1 and G_2 be two games whose code is identical except for the consequent in one if-branch, let $\mathcal{A}$ be an adversary interacting with either game, and let Bad be the event that the boolean condition in the if-statement is triggered when $\mathcal{A}$ is run with either game. Then $\left|\Pr[\mathrm{G}_1^\mathcal{A}] - \Pr[\mathrm{G}_2^\mathcal{A}]\right| \leq \Pr[\mathsf{Bad}]$.

Group schemes [CS98]**.** 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 $g \in \mathsf{G}$.

```
Game $\mathrm{SZ}^{\mathcal{A}}_{\mathbb{F},\mathsf{S},k,\boldsymbol{d}}$:
$(P_1,\ldots,P_\ell) \twoheadleftarrow \mathcal{A}$; $\boldsymbol{s} \twoheadleftarrow \mathsf{S}^k$
return $(\exists 1 \le i < j \le \ell)$
    $\big((P_i \ne P_j) \wedge (P_i(\boldsymbol{s}) = P_j(\boldsymbol{s}))\big)$
```

```
Game $q\text{-}\mathrm{DL}^{\mathcal{A}}_{\Gamma}(\lambda)$:
$\gamma \twoheadleftarrow \Gamma(1^\lambda)$
$x \twoheadleftarrow \mathbb{Z}_p$; $\boldsymbol{x} \leftarrow (x,\ldots,x^{q(\lambda)})$
$x' \twoheadleftarrow \mathcal{A}(\gamma,[\boldsymbol{x}])$; return $(x = x')$
```

Figure 1: *Left:* The Schwartz–Zippel game for a field $\mathbb{F}$, a finite subset $\mathsf{S} \subseteq \mathbb{F}$, and $k, q \in \mathbb{N}$, $\boldsymbol{d} \in \mathbb{N}^q$. *Right:* The q-DL game for a group scheme Γ.

Bilinear group schemes [Jou04, GPS06, Sha05]**.** 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 $2^{\lambda-1} \le p < 2^\lambda$ 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) $(\mathsf{G}_\nu, \cdot_\nu)$ is generated by g_ν, 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 $g_T \coloneqq e([1]_1,[1]_2) \ne [0]_T$. Here $[0]_T$ 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 drop subscripts and repeating entries from γ. In general, we will also omit the index μ in $\cdot_\mu$ when no confusion arises.

Schwartz–Zippel lemma [Sch80, Zip79, DL78]**.** We next 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 [BFHO22]. Recall that the degree of a multivariate monomial is the sum of the exponents of the variables appearing in the monomial, and the total degree of a multivariate polynomial is the maximum degree of its monomials.

Lemma 1. *Let $k, q \in \mathbb{N}$, $\boldsymbol{d} \in \mathbb{N}^q$, $\mathbb{F}$ be a field and $\mathsf{S} \subseteq \mathbb{F}$ a finite subset of $\mathbb{F}$. 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 Figure 1 (left). Here, q is an upper bound on the number of polynomials in $\mathbb{F}[X_1,\ldots,X_k]$ returned by $\mathcal{A}$, where the i-th polynomial has total degree at most $\boldsymbol{d}_i$.*

Bauer–Fuchsbauer–Loss lemma [BFL20]**.** We also recall a technical lemma due to Bauer, Fuchsbauer, and Loss regarding the leading term of a polynomial after variable substitutions.

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

q-DL [FKL18]**.** 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-DL game for Γ as

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

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

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

Figure 2: The (q_1, q_2)-DL game for a type-3 bilinear group scheme B.

(q_1, q_2)-DL [BFL20]. 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)-DL game for B as

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

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

Berlekamp's algorithm [Ber67]**.** Berlekamp's algorithm is a well-known method for factoring polynomials over finite fields, thus in particular for finding their roots. We denote by Berlekamp 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 of roots of P in $\mathbb{Z}_p$.

3 Generic-Group, Type-Safe, and Algebraic-Group Models

Unconditionally proving the hardness of interesting computational problems pertaining to groups appears to be currently out of reach. As a valid alternative, one instead 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 and restricted models often used to establish such results. We recall them in this section, and begin with the formal definition of the GGM.

Generic-group model (GGM) [Nec94, Sho97]**.** Consider a prime p and a finite set $\mathsf{G} \subseteq \{0,1\}^*$ with $|\mathsf{G}| = p$. Notice that every encoding $\tau \in \mathrm{Inj}(\mathbb{Z}_p, \mathsf{G})$ defines an associated operation $\mathsf{op}\colon \mathsf{G}^2 \to \mathsf{G}$ via $\mathsf{op}(h_1, h_2) \coloneqq \tau\big(\tau^{-1}(h_1) + \tau^{-1}(h_2)\big)$.[5] 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: A game in the GGM first samples a random encoding $\tau \in \mathrm{Inj}(\mathbb{Z}_p, \mathsf{G})$. Then the game and all parties it operates with are run on input $\tau(1)$, and interact with the labels in G in place of a real group. To perform group operations, the game offers all algorithms oracle access to the operation op defined by τ.

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

[5] As a mathematical shorthand, we call the action of pulling back h_1 and h_2 using τ^{-1}, adding up the preimages, and then pushing the result back forward using τ, a *pushforward*.

GGM **with hashing** (GGM-H) [Bro01, BFS16]**.** We define the GGM-H with parameters (p, G) as the GGM above, except that besides sampling τ, the game also (lazily) samples a function $H\colon \{0,1\}^* \to \mathbb{Z}_p$ at random, and offers $\mathsf{H}\colon \{0,1\}^* \to \mathsf{G}$ given by $\mathsf{H}(m) \coloneqq \tau(H(m))$ as an additional oracle to all algorithms.[6]

Following the approach taken for simple groups, we now recall the idealized models pertaining to bilinear groups. In essence, each group is idealized independently as before, and the pairing (and homomorphism, if applicable) is defined by the 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 (GBMk**,** $k \in \{1,2,3\}$**)** [BB04, ZZ23]**.** Consider a prime p and finite sets G_μ $(\mu \in \{1,2,T\})$ with $|\mathsf{G}_\mu| = p$. Given functions $\tau_\mu \in \mathrm{Inj}(\mathbb{Z}_p, \mathsf{G}_\mu)$, one can define operations op_μ on G_μ as in the GGM. Additionally, encodings τ_μ define a map $\mathsf{e}\colon \mathsf{G}_1 \times \mathsf{G}_2 \to \mathsf{G}_T$ via $\mathsf{e}(h_1, h_2) \coloneqq \tau_T\big(\tau_1^{-1}(h_1)\tau_2^{-1}(h_2)\big)$.

The type-3 generic-bilinear-group model 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 bilinear groups of order p: A game in the GBM3 first samples random encodings $\tau_\mu \in \mathrm{Inj}(\mathbb{Z}_p, \mathsf{G}_\mu)$. Then the game and all parties it operates with are run on input $(\tau_1(1), \tau_2(1))$, and interact with the labels in G_μ in place of a real type-3 bilinear group. To operate on labels, the game gives all algorithms oracle access to the operations op_μ and pairing e defined by τ_μ.

The GBM2 with parameters $(p, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_T)$ is defined analogously, except that it also idealizes the group homomorphism provided by a type-2 bilinear group. More precisely, in addition to op_μ and e, a game in the GBM2 also gives all algorithms oracle access to the function $\psi\colon \mathsf{G}_2 \to \mathsf{G}_1$ given by $\psi(h_2) \coloneqq \tau_1(\tau_2^{-1}(h_2))$.

Likewise, GBM1 with parameters $(p, \mathsf{G}, \mathsf{G}_T)$ is defined as the GBM3, but the target sets G_1 and G_2 as well as the 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 the GBMk for $k \in \{2,3\}$, and $\mathbf{G} \coloneqq (\mathsf{G}, \mathsf{G}_T)$ in the GBM1.

GBM **with hashing** [Lip22]**.** The GBM3-H with parameters $(p, \mathbf{G})$ is defined as GBM3, except that besides sampling τ_μ $(\mu \in \{1,2,T\})$, the game also (lazily) samples functions $H_\mu\colon \{0,1\}^* \to \mathbb{Z}_p$ independently at random. It then offers all algorithms access to oracles H_μ defined as in GGM-H, each using H_μ and τ_μ.

The GBM2-H with parameters $(p, \mathbf{G})$ is defined as the GBM3-H with parameters $(p, \mathbf{G})$, starting from GBM2, but the oracle H_2 is withheld [CCS07].

The GBM1-H with parameters $(p, \mathbf{G})$ is defined as the GBM3-H, starting from GBM1, except that the game (lazily) 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 [Mau05], which replaces group elements with abstract handles. This model has recently been recast by Zhandry [Zha22] as the type-safe model (TSM).[7] We next recall the TSM, but instead of using the language of circuits (as done by Zhandry [Zha22]), 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.

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

[7] The main difference between the two models is that in the TSM, when querying their oracles, parties cannot choose the 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 games in Maurer's model.

Type-safe model (TSM**)** [Mau05,Zha22]**.** Let p be a prime. In the type-safe model with parameter p, group elements are replaced by abstract handles, which we denote by $\{x\}$ with $x \in \mathbb{Z}_p$. These are tokens issued to algorithms in place of group elements, and x is meant to be the discrete logarithm of the group element represented by $\{x\}$. A handle $\{x\}$ hides its argument x from any party except the game.

In the TSM, a game and all parties it operates with are run on input handle $\{1\}$, and interact with handles in place of a real group. To operate on handles, the game offers all algorithms an oracle op defined as $\mathsf{op}(\{x_1\}, \{x_2\}) \coloneqq \{x_1 + x_2\}$. Note that in contrast to Maurer's model, and in line with Zhandry's TSM, handles are never overwritten and always fresh. Additionally, all algorithms are given an equality oracle eq and a copy oracle cp defined as $\mathsf{eq}(\{x_1\}, \{x_2\}) \coloneqq (x_1 = x_2)$ and $\mathsf{cp}(\{x\}) \coloneqq (\{x\}, \{x\})$.

Handles all look identical from the outside, and all computation related to handles is performed via the oracles above (i.e., local computation on handles is not allowed, as it does not "type-check"). In particular, when calling their oracles, algorithms are restricted to querying handles they have received as input or as response to a prior query. (In [Zha22], this corresponds to them applying gates only to wires they possess.) As for the query complexity metrics, queries to op incur unit cost, while queries to eq and cp are free.

TSM **with hashing.** We define the TSM-H with parameter p as the TSM above, except that the game also (lazily) samples a random function $H \colon \{0,1\}^* \to \mathbb{Z}_p$. In addition to oracles op, eq and cp, the game offers all algorithms an oracle H given by $\mathsf{H}(m) \coloneqq \{H(m)\}$.

We now extend the TSM to the bilinear setting, and then add hashing oracles to allow an adversary to hash into the various groups. We proceed as for the GBM, but start from the TSM rather than Shoup's GGM. To account for different groups in the bilinear setting, we denote handles representing elements in group $\mu \in \{1, 2, T\}$ by $\{x\}_\mu$, with $x \in \mathbb{Z}_p$.

Bilinear-type-safe model (BTMk**,** $k \in \{1,2,3\}$**).** Let p be a prime. In the type-3 bilinear-type-safe model (BTM3) with parameter p, a game and all parties it operates with are run on input $(\{1\}_1, \{1\}_2)$, and interact with handles in place of a real type-3 bilinear group. To operate on handles, the game offers all algorithms oracles op_μ, eq_μ, cp_μ ($\mu \in \{1, 2, T\}$) and e. Here, op_μ, eq_μ and cp_μ are implemented as in the TSM, each using handles for group μ, and oracle e is defined as $\mathsf{e}(\{x_1\}_1, \{x_2\}_2) \coloneqq \{x_1 x_2\}_T$.

The BTM2 with parameter p is defined analogously, except that it also offers all algorithms an oracle ψ which idealizes the group homomorphism provided by a type-2 bilinear group. Oracle ψ is defined as $\psi(\{x\}_2) \coloneqq \{x\}_1$.

Likewise, the BTM1 with parameter p is defined as the BTM3, but handles for the left and the right source group are taken to coincide. In each of the models above parties are restricted to querying, for every group, only handles they received as input or have seen as response to a prior query to op_μ, cp_μ or e.

BTM **with hashing.** The BTM3-H with parameter p is defined as the BTM3, except that the game also (lazily) samples functions $H_\mu \colon \{0,1\}^* \to \mathbb{Z}_p$ ($\mu \in \{1, 2, T\}$) independently at random. It then additionally offers all algorithms oracles H_μ defined as in TSM-H, each using function H_μ and returning handles for group μ.

The BTM2-H with parameter p is defined as the BTM3-H with parameter p, starting from BTM2, but oracle H_2 is withheld [CCS07].

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

The TSM and BTM provide an adequate setting for the algebraic-group model (AGM), where adversaries are restricted to being algebraic but have full access to the real group considered in a given game (rather than an idealized version as in the previous models).

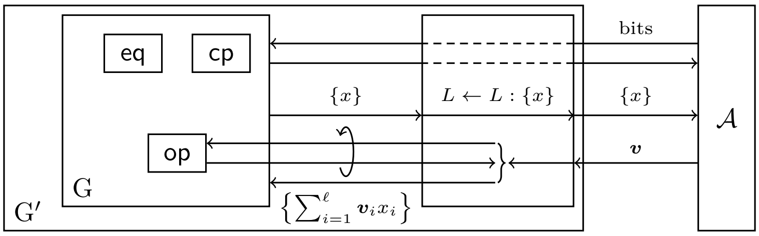

Figure 3: Representation of the algebraic compilation $\mathrm{G}' = \mathrm{AC}(\mathrm{G})$ of a type-safe game G. The unlabeled box inside G′ represents the compiler converting an adversary $\mathcal{A}$ against G′ into an adversary for G.

Algebraic algorithms, first studied in [BV98, PV05] and later revisited in [FKL18], 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 [Zha22] in defining the AGM as a compiler for type-safe games, which allows sidestepping issues regarding the validity of the model (see also [ZZK22]). 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 an empty list L and then runs G. Whenever G outputs a handle to the adversary, AC(G) keeps track of it by first appending a copy to L and then forwarding it to the adversary. Whenever G takes a handle 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 $L = (\{x_1\}, \ldots, \{x_\ell\})$, the vector $\boldsymbol{v}$, and the group operation oracle op of G to compute the handle $\left\{\sum_{i=1}^{\ell} \boldsymbol{v}_i x_i\right\}$, and forwards it to G (see Figure 3). Any output from G that is not a handle is forwarded to the adversary, and similarly any input from the adversary that is not a handle is forwarded to G. We call a game G′ in the TSM *algebraic* if $\mathrm{G}' = \mathrm{AC}(\mathrm{G})$ for some game G in the TSM.

Group compilation. Let $\mathrm{G} = \{\mathrm{G}_p\}_p$ be a family of games, each in the TSM with parameter p, and let Γ be a group scheme. The group compilation of G with respect to Γ is the standard-model game $\mathrm{GC}(\mathrm{G}, \Gamma)$ defined as follows: On security parameter λ, it first runs $\gamma = (\cdot, g, p) \twoheadleftarrow \Gamma(1^\lambda)$, and then operates as G_p with the following modifications. All parties are run on input γ, and no longer receive oracles op, eq and cp. Whenever G_p sends a handle to (resp., receives a handle from) any party, $\mathrm{GC}(\mathrm{G}, \Gamma)$ instead sends a group element to (resp., receives a group element from) the same party. The elements sent (resp., received) by $\mathrm{GC}(\mathrm{G}, \Gamma)$ are obtained (resp., operated on) by performing the same computations on group elements as G_p does on handles. This is possible because, by type safety, game G_p acts on handles only through the TSM oracles op, eq and cp. Therefore, whenever G_p queries $\mathsf{op}(\{x_1\}, \{x_2\})$, $\mathsf{eq}(\{x_1\}, \{x_2\})$ or $\mathsf{cp}(\{x\})$, $\mathrm{GC}(\mathrm{G}, \Gamma)$ can locally compute $h_1 \cdot h_2$, $(h_1 = h_2)$, and (h, h), respectively. Here, h_i and h are the group elements considered by the compiled game in place of the handles $\{x_i\}$ and $\{x\}$ considered by G_p. Any other communication between $\mathrm{GC}(\mathrm{G}, \Gamma)$ and the parties is processed as in G_p.

Algebraic-group model (AGM) **[BV98, FKL18, Zha22].** The algebraic-group model is a framework to study type-safe games in the standard model. More precisely, studying a family $\mathrm{G} = \{\mathrm{G}_p\}_p$ of type-safe games in the AGM with respect to a group scheme Γ is defined as analyzing the game $\mathrm{GC}(\mathrm{G}', \Gamma)$, where $\mathrm{G}' := \{\mathrm{AC}(\mathrm{G}_p)\}_p$. 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 family of type-safe games G′.

We now similarly define the AGM with hashing, which was already informally introduced by Fuchsbauer, Kiltz and Loss [FKL18] and further studied by Lipmaa [Lip22], using the type-safe framework of Zhandry [Zha22].

AGM **with hashing.** Given a game G in the TSM-H with parameter p, its algebraic compilation AC(G) is defined as for TSM games, except that for every query to oracle H, the returned handle is also copied into list L. Accordingly, an adversary can now also use handles obtained through H to specify group elements.

The group compilation $\mathrm{GC}(\mathrm{G}, \Gamma)$ of a family of games $\mathrm{G} = \{\mathrm{G}_p\}_p$, each in the TSM-H with parameter p, with respect to a group scheme Γ is defined as before, except that oracle H is still offered to all algorithms. Notice that $\mathrm{GC}(\mathrm{G}, \Gamma)$ is therefore a game in the random-oracle model.

With the definitions above, studying a family of TSM-H games $\mathrm{G} = \{\mathrm{G}_p\}_p$ in the AGM with respect to a group scheme Γ is defined as analyzing the game $\mathrm{GC}(\mathrm{G}', \Gamma)$, where $\mathrm{G}' \coloneqq \{\mathrm{AC}(\mathrm{G}_p)\}_p$. 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 family 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 considered in the model.

Bilinear algebraic compilations. Let p be a prime, $k \in \{1, 2, 3\}$, and G a game in the BTMk with parameter p. The bilinear algebraic compilation AC(G) of G is defined similarly to standard algebraic compilation, with the following differences.

If $k = 3$, AC(G) now maintains three initially empty lists L_μ, $\mu \in \{1, 2, T\}$, to keep track of the handles returned by game G to the adversary in the three groups. Whenever G takes a handle $\{x\}_\nu$, $\nu \in \{1, 2\}$, 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 $L_\nu = (\{x_{\nu,1}\}_\nu, \ldots, \{x_{\nu,\ell}\}_\nu)$, $\boldsymbol{v}$ and oracle op_ν to compute the handle $\left\{\sum_{i=1}^{\ell_\nu} \boldsymbol{v}_i x_{\nu,i}\right\}_\nu$, and then forwards it to G. Similarly, whenever G takes a handle $\{x\}_T$ 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, $\boldsymbol{m}$, $\boldsymbol{v}$, and oracles e and op_T to compute the handle $\left\{\sum_{i=1}^{\ell_1} \sum_{j=1}^{\ell_2} \boldsymbol{m}_{ij} x_{1,i} x_{2,j} + \sum_{t=1}^{\ell_T} \boldsymbol{v}_t x_{T,t}\right\}_T$, and forwards it to G. Any output of G or input from the adversary that is not a handle is relayed.

If $k = 2$, AC(G) is defined similarly, but we must account for the additional oracle ψ. Accordingly, whenever G takes a handle $\{x\}_1$ from the adversary, AC(G) instead takes vectors $(\boldsymbol{v}, \boldsymbol{w}) \in \mathbb{Z}_p^{\ell_1} \times \mathbb{Z}_p^{\ell_2}$. Game AC(G) then uses the current state of the lists L_ν, $\boldsymbol{v}$, $\boldsymbol{w}$, and the oracles op_1 and ψ to compute the handle $\left\{\sum_{i=1}^{\ell_1} \boldsymbol{v}_i x_{1,i} + \sum_{j=1}^{\ell_2} \boldsymbol{w}_j x_{2,j}\right\}_1$, and then forwards it to G. Handles $\{x\}_2$ are processed as above. Finally, whenever G takes a handle $\{x\}_T$, AC(G) instead takes matrices $(\boldsymbol{m}, \boldsymbol{n}) \in \mathbb{Z}_p^{\ell_1 \times \ell_2} \times \mathbb{Z}_p^{\ell_2 \times \ell_2}$ and a vector $\boldsymbol{v} \in \mathbb{Z}_p^{\ell_T}$. Game AC(G) then uses the current state of the lists L_μ, $\boldsymbol{m}$, $\boldsymbol{n}$, $\boldsymbol{v}$, and oracles e, ψ and op_T to compute $\left\{\sum_{i=1}^{\ell_1} \sum_{j=1}^{\ell_2} \boldsymbol{m}_{ij} x_{1,i} x_{2,j} + \sum_{i,j=1}^{\ell_2} \boldsymbol{n}_{ij} x_{2,i} x_{2,j} + \sum_{t=1}^{\ell_T} \boldsymbol{v}_t x_{T,t}\right\}_T$, and forwards it to G. Again, any inputs to or outputs of G that are not handles are relayed.

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

If $\mathrm{G} = \{\mathrm{G}_p\}_p$ is a family of games, each in the BTMk with parameter p, and B is a type-k bilinear group scheme, we define GC(G, B) as for simple groups: Each group in G is instantiated with the corresponding parameters in γ as discussed earlier, and whenever G queries $\mathsf{e}(\{x_1\}_1, \{x_2\}_2)$ (or $\psi(\{x\}_2)$, if $k = 2$) for handles $\{x_1\}_1$ and $\{x_2\}_2$ (and $\{x\}_2$), GC(G, B) computes $e([x_1]_1, [x_2]_2)$ (and $\psi([x]_2)$).

Algebraic-bilinear-group model (ABM)**.** Let $k \in \{1, 2, 3\}$, $\mathrm{G} = \{\mathrm{G}_p\}_p$ be a family of games, each in the BTMk with parameter p, and B a type-k bilinear group scheme. Studying G in the ABM w.r.t. B is defined as analyzing $\mathrm{GC}(\mathrm{G}', \mathrm{B})$, where $\mathrm{G}' \coloneqq \{\mathrm{AC}(\mathrm{G}_p)\}_p$.

ABM **with hashing.** For a prime p, $k \in \{1, 2, 3\}$, and a game G in the BTMk-H with parameter p, the algebraic compilation AC(G) of G is defined as for the BTMk, except that for every query to oracle H_μ (if present), the returned handle is also added to the list L_μ ($\mu \in \{1, 2, T\}$).

The bilinear group compilation GC(G, B) of a family of games $\mathrm{G} = \{\mathrm{G}_p\}_p$, each in the BTM$k$-H with parameter p, with respect to a type-k bilinear group scheme B is also defined as before, except that the game still offers oracles H_μ (if present) to all parties.

With the definitions above, studying a family of BTMk-H games $\mathrm{G} = \{\mathrm{G}_p\}_p$ in the ABM with respect to a type-k bilinear group scheme B is defined as analyzing $\mathrm{GC}(\mathrm{G}', \mathrm{B})$, where $\mathrm{G}' := \{\mathrm{AC}(\mathrm{G}_p)\}_p$.

In Appendix C, we study the relations between different models for standard games. Our treatment follows Zhandry [Zha22], with the difference that we consider the Turing machine model for type-safe games, a fixed set of group representations, and include a hashing oracle. We show that security with respect to type-safe and random-representation groups are equivalent. This result is summarized below.

Proposition 1. *Let p be a prime, and $\mathsf{G} \subseteq \{0, 1\}^*$ a finite set with $|\mathsf{G}| = p$. Let* G *be a single-stage game in the* TSM-H *with parameter p, and* $\mathrm{G}' := \mathrm{RR}(\mathrm{G}, \mathsf{G})$ *the RR-compilation of* G *with respect to* G. *Then* G *is secure if and only if* G' *is secure.*

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 $[\boldsymbol{x}]$ and a vector $\boldsymbol{a}$ in $\mathbb{Z}_p$, and returns a vector of group elements $[\boldsymbol{y}]$ and a vector $\boldsymbol{b}$ in $\mathbb{Z}_p$. Note that the two stages of $\mathcal{A}$ have access to shared randomness. A knowledge source is an algorithm $\mathcal{S}$ taking as input the state returned by $\mathcal{A}_0$, and returning vectors $\boldsymbol{x}$ and $\boldsymbol{a}$ in $\mathbb{Z}_p$. 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 in $\mathbb{Z}_p$.

If γ is a type-2 or type-3 bilinear group, $\mathcal{S}$ returns four vectors in $\mathbb{Z}_p$, three to define elements in 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 the discrete logarithms of the group elements returned by $\mathcal{A}_1$, rather than the elements themselves. Assuming that DL holds for Γ (resp., for some group scheme defined by B), this would in general make the KA not efficiently falsifiable [Nao03], 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 DPT).

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

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

where the game KA is defined in Figure 4 (top left). For a class of PPT algorithms $\mathfrak{A}$ we say that the KA holds for $(\Gamma, \mathcal{S}, \mathfrak{A})$ if for every PPT adversary $\mathcal{A}$ with $\mathcal{A}_0 \in \mathfrak{A}$, there exists a PPT extractor $\mathcal{E}$ such that $\mathrm{Adv}^{\mathrm{ka}}_{\Gamma,\mathcal{S},\mathcal{A},\mathcal{E}}$ is negligible.

```
Game $\mathrm{KA}^{\mathcal{A}}_{\Gamma,\mathcal{S},\mathcal{E}}(\lambda)$:
$\gamma \twoheadleftarrow \Gamma(1^\lambda)$; $r_{\mathcal{A}} \twoheadleftarrow \mathcal{R}_{\mathcal{A}}(\lambda)$
$(\mathsf{R}, st) \leftarrow \mathcal{A}_0(\gamma; r_{\mathcal{A}})$
$(\boldsymbol{x}, \boldsymbol{a}) \twoheadleftarrow \mathcal{S}(st)$
$([\boldsymbol{y}], \boldsymbol{b}) \leftarrow \mathcal{A}_1([\boldsymbol{x}], \boldsymbol{a}; r_{\mathcal{A}})$
$\mathsf{trace}(\mathcal{A}) \leftarrow (r_{\mathcal{A}}, \gamma, [\boldsymbol{x}], \boldsymbol{a})$
$\boldsymbol{w} \twoheadleftarrow \mathcal{E}(\mathsf{trace}(\mathcal{A}))$
return $\mathsf{R}(\boldsymbol{x}, [\boldsymbol{y}], \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{w})$
```

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

```
Game $\mathrm{KA}^{\mathcal{A}}_{\mathrm{B},\mathcal{S},\mathcal{E}}(\lambda)$:
$\gamma \twoheadleftarrow \mathrm{B}(1^\lambda)$; $r_{\mathcal{A}} \twoheadleftarrow \mathcal{R}_{\mathcal{A}}(\lambda)$
$(\mathsf{R}, st) \leftarrow \mathcal{A}_0(\gamma; r_{\mathcal{A}})$
$(\boldsymbol{x}_\mu, \boldsymbol{a}) \twoheadleftarrow \mathcal{S}(st)$
$([\boldsymbol{y}_\mu]_\mu, \boldsymbol{b}) \leftarrow \mathcal{A}_1([\boldsymbol{x}_\mu]_\mu, \boldsymbol{a}; r_{\mathcal{A}})$
$\mathsf{trace}(\mathcal{A}) \leftarrow (r_{\mathcal{A}}, \gamma, [\boldsymbol{x}_\mu]_\mu, \boldsymbol{a})$
$\boldsymbol{w} \twoheadleftarrow \mathcal{E}(\mathsf{trace}(\mathcal{A}))$
return $\mathsf{R}(\boldsymbol{x}_\mu, [\boldsymbol{y}_\mu]_\mu, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{w})$
```

```
Game $\mathrm{UK}^{\mathcal{A}}_{\mathrm{B},\mathcal{S},\mathcal{E}}(\lambda)$:
$\gamma \twoheadleftarrow \mathrm{B}(1^\lambda)$; $r_{\mathcal{A}} \twoheadleftarrow \mathcal{R}_{\mathcal{A}}(\lambda)$
$(Q, \boldsymbol{P}_\mu) \leftarrow \mathcal{A}_0(\gamma; r_{\mathcal{A}})$; $\boldsymbol{x}_\mu \twoheadleftarrow \mathcal{S}(\gamma, Q, \boldsymbol{P}_\mu)$
$([\boldsymbol{y}_\mu]_\mu, \boldsymbol{c}) \leftarrow \mathcal{A}_1(\gamma, [\boldsymbol{x}_\mu]_\mu; r_{\mathcal{A}})$
$\mathsf{trace}(\mathcal{A}) \leftarrow (r_{\mathcal{A}}, \gamma, [\boldsymbol{x}_\mu]_\mu)$
$(\boldsymbol{w}_\mu, \boldsymbol{z}) \twoheadleftarrow \mathcal{E}(\mathsf{trace}(\mathcal{A}))$; $\boldsymbol{x}_{1,0}, \boldsymbol{x}_{2,0} \leftarrow 1$
return $(Q(\boldsymbol{X}_\mu, \boldsymbol{Y}_\mu, \boldsymbol{c}) \neq 0) \wedge ([Q(\boldsymbol{x}_\mu, \boldsymbol{y}_\mu, \boldsymbol{c})]_T = [0]_T)$
    $\wedge \Big((\exists \nu)(\exists i)\big([\boldsymbol{y}_{\nu,i}]_\nu \neq \prod_{j=0}^{m_\nu} [\boldsymbol{w}_{\nu,ij}\boldsymbol{x}_{\nu,j}]_\nu\big) \vee$
        $(\exists i)\big([\boldsymbol{y}_{T,i}]_T \neq \prod_{j=0}^{m_1} \prod_{k=0}^{m_2} [\boldsymbol{z}_{ijk}\boldsymbol{x}_{1,j}\boldsymbol{x}_{2,k}]_T \cdot$
            $\prod_{t=1}^{m_T} [\boldsymbol{w}_{T,it}\boldsymbol{x}_{T,t}]_T\big)\Big)$
```

Figure 4: *Left:* The KA games for a group scheme Γ (resp., a type-3 bilinear group scheme B) and source $\mathcal{S}$. *Right:* Games defining the UK assumption for Γ (resp., B). In all figures, μ and ν are indices ranging over $\{1, 2, T\}$ and $\{1, 2\}$, respectively.

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 Figure 4 (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 $(\gamma, \mathcal{S}, \mathfrak{A})$ if for every adversary $\mathcal{A}$ running in time at most t with $\mathcal{A}_0 \in \mathfrak{A}$, 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 or B). We also note that our extractors in idealized models do not use the oracles they receive. This choice ensures, for example, that justification of a knowledge problem in a model with richer oracles is stronger than one in a model with fewer 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 DL problem is hard, adversaries that have a random exponent hard-coded can win KEA1 while every extractor would fail.[8] 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 [FGJ18].

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.

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

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 Figure 4 (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 $\boldsymbol{w} \in \mathbb{Z}_p^{n\times(m+1)}$.[9] For a class of PPT algorithms $\mathfrak{A}$ we say that UK holds for $(\Gamma, \mathcal{S}, \mathfrak{A})$ if for every low-degree PPT $\mathcal{A}$ with $\mathcal{A}_0 \in \mathfrak{A}$ 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 Figure 4 (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 $\boldsymbol{Y}_\nu$ ($\nu \in \{1, 2\}$) and at most 1 in $\boldsymbol{Y}_T$, with the only 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 (and isomorphism ψ) to efficiently verify the winning condition in G_T. To ensure that verification does not require private information, we will restrict our attention to polynomials Q of this type. Note that the degree of Q in both $\boldsymbol{X}$ and $\boldsymbol{C}$ can be arbitrary.

Remark. One could also envision formulating umbrella knowledge assumptions taking different forms. We are motivated by bridging the AGM to the standard model, which the definition above allows us to do. Interestingly, a number of classical knowledge assumptions (KEA1, KEA3, d-PKE, etc.) fall under the reach of the UK assumption formulated above.

We now give a few example assumptions implied by the UK assumption. We first state the assumptions individually, and then show in Proposition 2 that they are indeed implied by UK. Examples in the bilinear setting are defined for type-3 bilinear group schemes, but the definitions can be readily adapted to type-1 or type-2 schemes.

Example: KEAI, $I \in \{1, 3\}$ [Dam92, BP04a]. 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 Figure 5 (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 KEA$I$ 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.

Remark. We note that the terminology around the KEA assumptions is not well established. For example, [Den06b, BP04b] refer to the notion we call KEA1 as the DHK (or DHK0) assumption, while [BP04a] reserves the name KEA1 for the non-uniform version of the notion above. Another name for the latter version is DA-1 [HT99].

[9] To simplify notation, we will sometimes allow parties in the UK game to return outputs with slightly different formats.

```
Game $\mathrm{KEA1}^{\mathcal{A}}_{\Gamma,\mathcal{E}}(\lambda)$:
    $\gamma \twoheadleftarrow \Gamma(1^\lambda)$; $a \twoheadleftarrow \mathbb{Z}_p$
    $([b],[y]) \twoheadleftarrow \mathcal{A}(\gamma,[a])$
    $b' \twoheadleftarrow \mathcal{E}(\mathsf{trace}(\mathcal{A}))$
    return $([y]=[ab]) \wedge ([b] \neq [b'])$
```

```
Game $\mathrm{KEA3}^{\mathcal{A}}_{\Gamma,\mathcal{E}}(\lambda)$:
    $\gamma \twoheadleftarrow \Gamma(1^\lambda)$; $a, b \twoheadleftarrow \mathbb{Z}_p$
    $([c],[y]) \twoheadleftarrow \mathcal{A}(\gamma,[a],[b],[ab])$
    $(c_1,c_2) \twoheadleftarrow \mathcal{E}(\mathsf{trace}(\mathcal{A}))$
    return $([y]=[bc]) \wedge ([c] \neq [c_1]\cdot[ac_2])$
```

```
Game $d\text{-}\mathrm{PKE}^{\mathcal{A}}_{\Gamma,\mathcal{E}}(\lambda)$:
    $\gamma \twoheadleftarrow \Gamma(1^\lambda)$; $s, a \twoheadleftarrow \mathbb{Z}_p$; $([c],[y]) \twoheadleftarrow \mathcal{A}(\gamma, ([s^i])_{i=1}^{d(\lambda)}, ([as^i])_{i=0}^{d(\lambda)})$; $\boldsymbol{w} \twoheadleftarrow \mathcal{E}(\mathsf{trace}(\mathcal{A}))$
    return $([y]=[ac]) \wedge \big([c] \neq \prod_{i=0}^{d(\lambda)}[\boldsymbol{w}_i s^i]\big)$
```

Figure 5: 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.

Example: d-PKE [Gro10]. 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

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

where the game d-PKE is defined in Figure 5 (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.

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 DL holds for Γ), even if we restrict the adversary to be algebraic.

Example: d-KZG [KZG10]. 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

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

where the game d-KZG is defined in Figure 6 (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.

Remark. The idea behind d-KZG is allowing 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 [Gro10]. 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

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

```
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 $(e([c]_1 \cdot [-y]_1, [1]_2) = e([q]_1, [s]_2 \cdot [-x]_2)) \wedge \left([c]_1 \neq \prod_{i=0}^{d(\lambda)-1} [\boldsymbol{w}_i s^i]_1\right)$
```

```
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 $([y]_1 = [ac]_1) \wedge \left([c]_1 \neq \prod_{i=0}^{d(\lambda)} [\boldsymbol{w}_i s^i]_1\right)$
```

```
Game $d\text{-GROTH16}^{\mathcal{A}}_{\mathrm{B},\mathcal{E}}(\lambda)$:
    $\varpi \twoheadleftarrow \mathrm{B}(1^\lambda)$; $\mathsf{R} := (\ell, (U_i, V_i, W_i)_{i=0}^m, T) \twoheadleftarrow \mathcal{A}_0(\varpi)$; $\alpha, \beta, \gamma, \delta, x \twoheadleftarrow \mathbb{Z}_p^*$
    $\boldsymbol{x}_{1,0} \leftarrow 1$; $\boldsymbol{x}_{1,1} \leftarrow \alpha\gamma\delta$; $\boldsymbol{x}_{1,2} \leftarrow \beta\gamma\delta$; $\boldsymbol{x}_{1,3} \leftarrow \gamma\delta^2$
    $\boldsymbol{x}_{2,0} \leftarrow 1$; $\boldsymbol{x}_{2,1} \leftarrow \beta\gamma\delta$; $\boldsymbol{x}_{2,2} \leftarrow \gamma^2\delta$; $\boldsymbol{x}_{2,3} \leftarrow \gamma\delta^2$
    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 $\ell$ 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 $\left(e([a]_1, [b]_2) = e([\boldsymbol{x}_{1,1}]_1, [\boldsymbol{x}_{2,1}]_2) \cdot \prod_{i=0}^{\ell} e([f_i \boldsymbol{x}_{1,2d+3+i}]_1, [\boldsymbol{x}_{2,2}]_2) \cdot e([c]_1, [\boldsymbol{x}_{2,3}]_2)\right)$
        $\wedge \left(\left([a]_1 \neq \prod_{i=0}^{2d(\lambda)+m+3} [\boldsymbol{w}_{1,i} \boldsymbol{x}_{1,i}]_1\right) \vee \left([c]_1 \neq \prod_{i=0}^{2d(\lambda)+m+3} [\boldsymbol{w}_{2,i} \boldsymbol{x}_{1,i}]_1\right) \vee \left([b]_2 \neq \prod_{i=0}^{d(\lambda)+3} [\boldsymbol{w}_{3,i} \boldsymbol{x}_{2,i}]_2\right)\right)$
```

Figure 6: Games defining the d-KZG, d-PKE, and d-GROTH16 assumptions. In all figures, B is a type-3 bilinear group scheme and $d\colon \mathbb{N} \to \mathbb{N}$ a polynomial.

where the game d-PKE is defined in Figure 6 (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.

Example: d-GROTH16 [Gro16]**.** 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

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

where the game d-GROTH16 is defined in Figure 6 (bottom). Here, $\mathcal{E}$ returns a vector $\boldsymbol{w} \in \mathbb{Z}_p^{m-\ell}$. For a class of PPT algorithms $\mathfrak{A}$ we say that d-GROTH16 holds for $(\mathrm{B}, \mathfrak{A})$ if for every PPT $\mathcal{A}$ with $\mathcal{A}_0 \in \mathfrak{A}$, there exists a PPT $\mathcal{E}$ such that $\mathrm{Adv}^{d\text{-groth16}}_{\mathrm{B},\mathcal{A},\mathcal{E}}$ is negligible.

Remark. Notice that we define a slightly modified version of the security game considered in [Gro16], where all polynomials are multiplied by $\gamma\delta$, in order to clear the denominators and let the assumption fit the UK-framework.

We now prove that all examples above follow from the UK assumption. Jumping ahead, when we give a modular proof that these example assumptions hold in the GGM-H (resp., GBM3-H, see Corollary 1) via our GGM-H hardness result (resp., GBM3-H hardness, see Theorems 3 and 6) of UK, we will have to check that the requirements of Proposition 2 are satisfied by these theorems.

Source $\mathcal{S}(\gamma, Q, \boldsymbol{P})$:	Source $\mathcal{S}(\gamma, Q, \boldsymbol{P}_\mu)$:
$\boldsymbol{s} \twoheadleftarrow \mathbb{Z}_p^k$; return $\boldsymbol{P}(\boldsymbol{s})$	$\boldsymbol{s} \twoheadleftarrow \mathbb{Z}_p^k$; return $\boldsymbol{P}_\mu(\boldsymbol{s})$

Figure 7: Knowledge sources for which we require the UK assumption to hold for $(\Gamma, \mathcal{S}, \mathfrak{B})$ (resp., $(\mathrm{B}, \mathcal{S}, \mathfrak{B})$) in Proposition 2. Here, k is an upper bound on the number of variables appearing in any polynomial in $\boldsymbol{P}$ (resp., $\boldsymbol{P}_\mu$), and $\mu \in \{1, 2, T\}$.

Proposition 2. *Let* Γ *be a group scheme,* $\mathcal{S}$ *the knowledge source given in Figure 7 (left),* $\mathfrak{B}$ *a class of PPT algorithms such that* UK *holds for* $(\Gamma, \mathcal{S}, \mathfrak{B})$*, and* $d\colon \mathbb{N} \to \mathbb{N}$ *a polynomial. (1a) If* $\mathcal{B}_0 \in \mathfrak{B}$ *for* $\mathcal{B}_0$ *given in Figure 9 (left) and* DL *holds for* Γ*, then* KEA1 *holds for* Γ*. (1b) If* $\mathcal{B}_0 \in \mathfrak{B}$ *for* $\mathcal{B}_0$ *given in Figure 9 (right) and* 2-DL *holds for* Γ*, then* KEA3 *holds for* Γ*. (1c) If* $\mathcal{B}_0 \in \mathfrak{B}$ *for* $\mathcal{B}_0$ *given in Figure 8 (top) and* $(d{+}1)$-DL *holds for* Γ*, then* d-PKE *holds for* Γ*.*

Let B *be a type-3 bilinear group scheme,* $\mathcal{S}$ *the knowledge source given in Figure 7 (right), and* $\mathfrak{A}$ *and* $\mathfrak{B}$ *classes of PPT algorithms such that* UK *holds for* $(\mathrm{B}, \mathcal{S}, \mathfrak{B})$*. (2a) If* $\mathcal{B}_0 \in \mathfrak{B}$ *for* $\mathcal{B}_0$ *given in Figure 10 (top) and* $(d+1, 1)$-DL *holds for* B*, then* d-PKE *holds for* B*. (2b) If* $\mathcal{B}_0 \in \mathfrak{B}$ *for* $\mathcal{B}_0$ *given in Figure 11, then* d-KZG *holds for* B*. (2c) If* $\mathcal{B}_0 \in \mathfrak{B}$ *for every* $\mathcal{A}_0 \in \mathfrak{A}$*, where* $\mathcal{B}_0$ *is given in Figure 12, then* d-GROTH16 *holds for* $(\mathrm{B}, \mathfrak{A})$*.*

Proof overview. Given an adversary $\mathcal{A}$ against any of the considered notions, we transform it into a UK adversary $\mathcal{B}$ against $(\Gamma, \mathcal{S})$ (resp., $(\mathrm{B}, \mathcal{S})$) with $\mathcal{B}_0 \in \mathfrak{B}$, for which there must exist a UK extractor $\mathcal{F}$ by hardness of UK. We then turn $\mathcal{F}$ into an extractor $\mathcal{E}$ for $\mathcal{A}$ by returning only some of the coefficients computed by $\mathcal{F}$, since $\mathcal{E}$ has to represent (some of) the outputs of $\mathcal{A}$ in terms of only a subset of its inputs. To ensure that this representation is correct (i.e., that the coefficients omitted by $\mathcal{E}$ were equal to zero in the first place), we carry out a reduction to power-DL. We show how a reduction $\mathcal{C}$ can embed the power-DL-challenge x into the inputs of $\mathcal{A}$, and then obtain a non-trivial polynomial equation $T(x) = 0$ involving x if one of the coefficients that $\mathcal{E}$ omits from $\mathcal{F}$ is non-zero. Adversary $\mathcal{C}$ can then recover x by computing the roots of T, using Berlekamp's algorithm.

For d-KZG and d-GROTH16, the last step is not needed since extractor $\mathcal{E}$ is allowed to use all input elements to $\mathcal{A}$, so that we simply set $\mathcal{E} := \mathcal{F}$.

Proof. We prove our claims separately.

(1c) d-PKE **(simple groups).** Given a d-PKE adversary $\mathcal{A}$, let $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$ be the UK adversary where $\mathcal{B}_0$ is given in Figure 8 (top), and $\mathcal{B}_1$ runs $\mathcal{A}$ and returns its output. Let $\mathcal{F}$ be a UK extractor for $\mathcal{B}$ (as per hardness of UK for $(\Gamma, \mathcal{S}, \mathfrak{B})$) that outputs $(\boldsymbol{w}, \boldsymbol{w}') = \left(\begin{smallmatrix} \boldsymbol{w}_{10} & \cdots & \boldsymbol{w}_{1,d(\lambda)} & \boldsymbol{w}'_{10} & \cdots & \boldsymbol{w}'_{1,d(\lambda)} \\ \boldsymbol{w}_{20} & \cdots & \boldsymbol{w}_{2,d(\lambda)} & \boldsymbol{w}'_{20} & \cdots & \boldsymbol{w}'_{2,d(\lambda)} \end{smallmatrix}\right)$. Define a d-PKE extractor $\mathcal{E}$ for $\mathcal{A}$ that runs $\mathcal{F}$ and outputs $(\boldsymbol{w}_{10}, \ldots, \boldsymbol{w}_{1,d(\lambda)})$. We claim that $\mathrm{Adv}^{d\text{-pke}}_{\Gamma,\mathcal{A},\mathcal{E}}$ is negligible, proving that d-PKE holds for Γ. To that end, consider the following sequence of games (the formal description of which can be found in Figure 8 (second from bottom)):

- G_0: This is the original d-PKE game for Γ run with adversary $\mathcal{A}$ and extractor $\mathcal{E}$. We reformulate the winning condition by letting the game immediately return 0 if $[y] \neq [ac]$, and then checking $\big([c] = \prod_{i=0}^{d(\lambda)} [\boldsymbol{w}_{1,i} s^i]\big)$.
- G_1: This game proceeds as G_0, but additionally returns 0 if $(\boldsymbol{w}, \boldsymbol{w}')$ is not a correct representation of all outputs of $\mathcal{A}$ in terms of all its (group element) inputs.
- G_2: This game proceeds as G_1, but additionally returns 0 if the representation $(\boldsymbol{w}, \boldsymbol{w}')$ is not of the form $\left(\begin{smallmatrix} \boldsymbol{w}_{10} & \cdots & \boldsymbol{w}_{1,d(\lambda)} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \boldsymbol{w}_{10} & \cdots & \boldsymbol{w}_{1,d(\lambda)} \end{smallmatrix}\right)$.

We now bound the difference between the success probabilities in subsequent games.

```
Adversary $\mathcal{B}_0(\gamma)$:
$Q((\boldsymbol{X}_i)_{i=0}^{d(\lambda)}, (\boldsymbol{X}'_i)_{i=0}^{d(\lambda)}, \boldsymbol{Y}_1, \boldsymbol{Y}_2) \leftarrow \boldsymbol{Y}_2 - \boldsymbol{X}'_0 \boldsymbol{Y}_1$
for $i = 1$ to $d(\lambda)$ do $\boldsymbol{P}_i(\boldsymbol{S}_1, \boldsymbol{S}_2) \leftarrow \boldsymbol{S}_1^i$; $\boldsymbol{P}'_i(\boldsymbol{S}_1, \boldsymbol{S}_2) \leftarrow \boldsymbol{S}_2 \boldsymbol{S}_1^i$
return $(Q, \boldsymbol{P}, \boldsymbol{P}')$
```

```
Adversary $\mathcal{C}(\gamma, [x], \dots, [x^{d(\lambda)}], [x^{d(\lambda)+1}])$:
$\beta_1, \beta_2 \twoheadleftarrow \mathbb{Z}_p^*$; $\alpha_1, \alpha_2 \twoheadleftarrow \mathbb{Z}_p$; $(Q, \boldsymbol{P}, \boldsymbol{P}') \twoheadleftarrow \mathcal{B}_0(\gamma)$; $\mathsf{S} \leftarrow \emptyset$
$([c], [y]) \twoheadleftarrow \mathcal{A}(\gamma, ([(\beta_1 x + \alpha_1)^i])_{i=1}^{d(\lambda)}, ([(\beta_2 x + \alpha_2)(\beta_1 x + \alpha_1)^i])_{i=0}^{d(\lambda)})$
$\begin{pmatrix} \boldsymbol{w}_{10} & \cdots & \boldsymbol{w}_{1,d(\lambda)} & \boldsymbol{w}'_{10} & \cdots & \boldsymbol{w}'_{1,d(\lambda)} \\ \boldsymbol{w}_{20} & \cdots & \boldsymbol{w}_{2,d(\lambda)} & \boldsymbol{w}'_{20} & \cdots & \boldsymbol{w}'_{2,d(\lambda)} \end{pmatrix} \twoheadleftarrow \mathcal{F}(\mathsf{trace}(\mathcal{A}))$; $\boldsymbol{P}_0 \leftarrow 1$
for $j = 0$ to $d(\lambda)$ do
    $\boldsymbol{X}_j \leftarrow \boldsymbol{P}_j(\beta_1 X + \alpha_1, \beta_2 X + \alpha_2)$; $\boldsymbol{X}'_j \leftarrow \boldsymbol{P}'_j(\beta_1 X + \alpha_1, \beta_2 X + \alpha_2)$
for $i = 1$ to 2 do $\boldsymbol{Y}_i \leftarrow \sum_{j=0}^{d(\lambda)} \boldsymbol{w}_{ij} \boldsymbol{X}_j + \boldsymbol{w}'_{ij} \boldsymbol{X}'_j$
$T(X) \leftarrow Q(\boldsymbol{X}_0, \dots, \boldsymbol{X}_{d(\lambda)}, \boldsymbol{X}'_0, \dots, \boldsymbol{X}'_{d(\lambda)}, \boldsymbol{Y}_1, \boldsymbol{Y}_2)$
if $(T(X) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{Berlekamp}(T, p)$
for $x' \in \mathsf{S}$ do if $([x'] = [x])$ then return $x'$
return 0
```

```
Game $\mathrm{G}_0(\lambda)$:
$\gamma \twoheadleftarrow \Gamma(1^\lambda)$; $s, a \twoheadleftarrow \mathbb{Z}_p$
$([c], [y]) \twoheadleftarrow$
    $\mathcal{A}(\gamma, ([s^i])_{i=1}^d, ([as^i])_{i=0}^d)$
$(\boldsymbol{w}, \boldsymbol{w}') \twoheadleftarrow \mathcal{F}(\mathsf{trace}(\mathcal{A}))$
if $([y] \neq [ac])$ then return 0
return
    $\big([c] \neq \prod_{i=0}^d [\boldsymbol{w}_{1,i} s^i]\big)$
```

```
Game $\mathrm{G}_1(\lambda)$:
$\gamma \twoheadleftarrow \Gamma(1^\lambda)$; $s, a \twoheadleftarrow \mathbb{Z}_p$
$([c], [y]) \twoheadleftarrow$
    $\mathcal{A}(\gamma, ([s^i])_{i=1}^d, ([as^i])_{i=0}^d)$
$(\boldsymbol{w}, \boldsymbol{w}') \twoheadleftarrow \mathcal{F}(\mathsf{trace}(\mathcal{A}))$
if $([y] \neq [ac])$ then return 0
if $([c] \neq \prod_{i=0}^d [\boldsymbol{w}_{1,i} s^i][\boldsymbol{w}'_{1,i} as^i])$
    $\vee$
    $([y] \neq \prod_{i=0}^d [\boldsymbol{w}_{2,i} s^i][\boldsymbol{w}'_{2,i} as^i])$
    then return 0
return $\big([c] \neq \prod_{i=0}^d [\boldsymbol{w}_{1,i} s^i]\big)$
```

```
Game $\mathrm{G}_2(\lambda)$:
$\gamma \twoheadleftarrow \Gamma(1^\lambda)$; $s, a \twoheadleftarrow \mathbb{Z}_p$
$([c], [y]) \twoheadleftarrow$
    $\mathcal{A}(\gamma, ([s^i])_{i=1}^d, ([as^i])_{i=0}^d)$
$(\boldsymbol{w}, \boldsymbol{w}') \twoheadleftarrow \mathcal{F}(\mathsf{trace}(\mathcal{A}))$
if $([y] \neq [ac])$ then return 0
if $([c] \neq \prod_{i=0}^d [\boldsymbol{w}_{1,i} s^i][\boldsymbol{w}'_{1,i} as^i]) \vee$
    $([y] \neq \prod_{i=0}^d [\boldsymbol{w}_{2,i} s^i][\boldsymbol{w}_{2,i} as^i])$
    then return 0
if $(\boldsymbol{w}'_{10} \neq 0) \vee \cdots \vee (\boldsymbol{w}'_{1,d} \neq 0) \vee$
    $(\boldsymbol{w}_{20} \neq 0) \vee \cdots \vee (\boldsymbol{w}_{2,d} \neq 0) \vee$
    $(\boldsymbol{w}_{10} \neq \boldsymbol{w}'_{20}) \vee \cdots \vee$
    $(\boldsymbol{w}_{1,d} \neq \boldsymbol{w}'_{2,d})$ then return 0
return $\big([c] \neq \prod_{i=0}^d [\boldsymbol{w}_{1,i} s^i]\big)$
```

```
Game $\mathrm{G}'(\lambda)$:
$\gamma \twoheadleftarrow \Gamma(1^\lambda)$; $r_1, r_2 \twoheadleftarrow \mathbb{Z}_p$; $(Q, \boldsymbol{P}, \boldsymbol{P}') \twoheadleftarrow \mathcal{B}_0(\gamma)$; $\mathsf{S} \leftarrow \emptyset$; $([c], [y]) \twoheadleftarrow \mathcal{A}(\gamma, ([r_1^i])_{i=1}^{d(\lambda)}, ([r_2 r_1^i])_{i=0}^{d(\lambda)})$
$(\boldsymbol{w}, \boldsymbol{w}') \twoheadleftarrow \mathcal{F}(\mathsf{trace}(\mathcal{A}))$; $x \twoheadleftarrow \mathbb{Z}_p$; $\beta_1, \beta_2 \twoheadleftarrow \mathbb{Z}_p^*$; $\alpha_1 \leftarrow r_1 - \beta_1 x$; $\alpha_2 \leftarrow r_2 - \beta_2 x$; $\boldsymbol{P}_0 \leftarrow 1$
for $j = 0$ to $d(\lambda)$ do $\boldsymbol{X}_j \leftarrow \boldsymbol{P}_j(\beta_1 X + \alpha_1, \beta_2 X + \alpha_2)$; $\boldsymbol{X}'_j \leftarrow \boldsymbol{P}'_j(\beta_1 X + \alpha_1, \beta_2 X + \alpha_2)$
for $i = 1$ to 2 do $\boldsymbol{Y}_i \leftarrow \sum_{j=0}^{d(\lambda)} \boldsymbol{w}_{ij} \boldsymbol{X}_j + \boldsymbol{w}'_{ij} \boldsymbol{X}'_j$
$T(X) \leftarrow Q(\boldsymbol{X}_0, \dots, \boldsymbol{X}_{d(\lambda)}, \boldsymbol{X}'_0, \dots, \boldsymbol{X}'_{d(\lambda)}, \boldsymbol{Y}_1, \boldsymbol{Y}_2)$; if $(T(X) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{Berlekamp}(T, p)$
$x' \leftarrow 0$; for $z \in \mathsf{S}$ do if $([t] = [x])$ then return $x' \leftarrow z$; break
return $(x = x')$
```

Figure 8: *Top:* First-stage UK adversary $\mathcal{B}_0$ from the proof that UK implies d-PKE for simple groups. *Second from top:* Adversary $\mathcal{C}$ against $(d+1)$-DL from the proof that UK implies d-PKE for simple groups. *Second from bottom and bottom:* Code of the intermediate games in the proof that UK implies d-PKE for simple groups.

$\mathrm{G}_0 \rightsquigarrow \mathrm{G}_1$. Notice that G_0 and G_1 are identical until Bad, where Bad is the event in the d-PKE game for Γ played by $(\mathcal{A}, \mathcal{E})$ that $[y] = [ac]$ and $(\boldsymbol{w}, \boldsymbol{w}')$ is not a correct representation of $([c], [y])$ in terms of $([1], \dots, [s^{d(\lambda)}], [a], \dots, [as^{d(\lambda)}])$. By definition of $\mathcal{S}$, $\mathcal{B}$ and $\mathcal{F}$, this corresponds to the event that $(\mathcal{B}, \mathcal{F})$ win the UK game for $(\Gamma, \mathcal{S})$. By the fundamental lemma of game playing we therefore have

$$|\Pr[\mathrm{G}_1] - \Pr[\mathrm{G}_0]| \leq \Pr[\mathsf{Bad}] = \mathrm{Adv}^{\mathrm{uk}}_{\Gamma,\mathcal{S},\mathcal{B},\mathcal{F}}(\lambda)\,.$$

$\mathrm{G}_1 \rightsquigarrow \mathrm{G}_2$. Observe that G_1 and G_2 are identical until Bad', where Bad' is the event in the d-PKE game for Γ played by $(\mathcal{A}, \mathcal{E})$ that $([c], [y])$ and $(\boldsymbol{w}, \boldsymbol{w}')$ are correct, but not of the form $\left(\begin{smallmatrix} \boldsymbol{w}_{10} & \cdots & \boldsymbol{w}_{1,d(\lambda)} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \boldsymbol{w}_{10} & \cdots & \boldsymbol{w}_{1,d(\lambda)} \end{smallmatrix}\right)$. Again by the fundamental lemma of game playing we have $|\Pr[\mathrm{G}_2] - \Pr[\mathrm{G}_1]| \le \Pr[\mathsf{Bad}']$.

Collecting all terms above, we obtain

$$\mathrm{Adv}^{d\text{-pke}}_{\Gamma,\mathcal{A},\mathcal{E}}(\lambda) = \Pr[\mathrm{G}_0] \le \Pr[\mathrm{G}_2] + \mathrm{Adv}^{\mathrm{uk}}_{\Gamma,\mathcal{S},\mathcal{B},\mathcal{F}}(\lambda) + \Pr[\mathsf{Bad}'] = \mathrm{Adv}^{\mathrm{uk}}_{\Gamma,\mathcal{S},\mathcal{B},\mathcal{F}}(\lambda) + \Pr[\mathsf{Bad}'] ,$$

where the last equality holds because $\Pr[\mathrm{G}_2] = 0$, since the return statement introduced in G_2 ensures that $[c] = [\boldsymbol{w}_{10}] \cdots [\boldsymbol{w}_{1,d(\lambda)} s^{d(\lambda)}]$, while the winning condition is $[c] \neq [\boldsymbol{w}_{10}] \cdots [\boldsymbol{w}_{1,d(\lambda)} s^{d(\lambda)}]$.

We are now left with bounding $\Pr[\mathsf{Bad}']$. To that end, consider the adversary $\mathcal{C}$ against $(d+1)$-DL for Γ defined in Figure 8 (second from top); we will bound $\Pr[\mathsf{Bad}']$ in terms of the advantage of $\mathcal{C}$. To do so, we show that if Bad' occurs, then the polynomial T constructed by $\mathcal{C}$ is non-zero with overwhelming probability. Whenever that is the case, $\mathcal{C}$ will succeed in winning the $(d+1)$-DL game for Γ, because it can recover x by finding the correct root of T using Berlekamp's algorithm.

Starting from $(d+1)\text{-DL}^{\mathcal{C}}_{\Gamma}$, we transition to a game G' (see Figure 8 (bottom)) where $\mathcal{A}$ is given group elements $([r_1], \ldots, [r_1^{d(\lambda)}], [r_2], \ldots, [r_2 r_1^{d(\lambda)}])$ for $r_1, r_2 \twoheadleftarrow \mathbb{Z}_p$ and then, only after $\mathcal{F}$ is run, G' samples $x \twoheadleftarrow \mathbb{Z}_p$, $\beta_1, \beta_2 \twoheadleftarrow \mathbb{Z}_p^*$, and then sets $\alpha_1 \leftarrow r_1 - \beta_1 x$ and $\alpha_2 \leftarrow r_2 - \beta_2 x$. Then observe that $\Pr[(d+1)\text{-DL}^{\mathcal{C}}_{\Gamma}] = \Pr[\mathrm{G}']$, because the inputs of $\mathcal{A}$ are equally distributed in both games. Now write

$$\mathsf{Bad}' = \mathsf{Bad}'_{d(\lambda)+2} \vee \cdots \vee \mathsf{Bad}'_0 ,$$

where

$$\begin{aligned}
\mathsf{Bad}'_{d+2} &:= \mathsf{Bad}' \wedge (\boldsymbol{w}'_{1,d} \neq 0) \\
\mathsf{Bad}'_{d+1} &:= \mathsf{Bad}' \wedge \neg\mathsf{Bad}'_{d+2} \wedge ((\boldsymbol{w}'_{1,d-1} \neq 0) \vee (\boldsymbol{w}_{1,d} \neq \boldsymbol{w}'_{2,d})) \\
\mathsf{Bad}'_{d} &:= \mathsf{Bad}' \wedge \neg\mathsf{Bad}'_{d+2} \wedge \neg\mathsf{Bad}'_{d+1} \wedge ((\boldsymbol{w}'_{1,d-2} \neq 0) \vee (\boldsymbol{w}_{2,d} \neq 0) \vee (\boldsymbol{w}_{1,d-1} \neq \boldsymbol{w}'_{2,d-1})) \\
&\ \ \vdots \\
\mathsf{Bad}'_{2} &:= \mathsf{Bad}' \wedge \neg\mathsf{Bad}'_{d+2} \wedge \cdots \wedge \neg\mathsf{Bad}'_{3} \wedge ((\boldsymbol{w}'_{10} \neq 0) \vee (\boldsymbol{w}_{22} \neq 0) \vee (\boldsymbol{w}_{11} \neq \boldsymbol{w}'_{21})) \\
\mathsf{Bad}'_{1} &:= \mathsf{Bad}' \wedge \neg\mathsf{Bad}'_{d+2} \wedge \cdots \wedge \neg\mathsf{Bad}'_{2} \wedge ((\boldsymbol{w}_{21} \neq 0) \vee (\boldsymbol{w}_{10} \neq \boldsymbol{w}'_{20})) \\
\mathsf{Bad}'_{0} &:= \mathsf{Bad}' \wedge \neg\mathsf{Bad}'_{d+2} \wedge \cdots \wedge \neg\mathsf{Bad}'_{1} \wedge (\boldsymbol{w}_{20} \neq 0) .
\end{aligned}$$

Here, Bad'_i is the event that the coefficient of degree i in T is non-zero as a polynomial in β_1 and β_2, but every coefficient of higher degree is zero. (Note that $\Pr[\mathsf{Bad}'_0] = 0$, because if Bad' occurs, then $T(x) = 0$, so it cannot be that the constant term is the only non-zero term of T.) Then

$$\begin{aligned}
\Pr[(d+1)\text{-DL}^{\mathcal{C}}_{\Gamma}(\lambda)] = \Pr[\mathrm{G}'] \ge \Pr[\mathrm{G}' \wedge \mathsf{Bad}'] &= \sum\nolimits_{i=1}^{d(\lambda)+2} \Pr[\mathrm{G}' \mid \mathsf{Bad}'_i] \Pr[\mathsf{Bad}'_i] \\
&\ge \left(1 - \frac{2}{2^{\lambda-1} - 1}\right) \left(\sum\nolimits_{i=1}^{d(\lambda)+2} \Pr[\mathsf{Bad}'_i]\right) = \left(1 - \frac{2}{2^{\lambda-1} - 1}\right) \Pr[\mathsf{Bad}'] .
\end{aligned}$$

Here, the last inequality holds because of the Schwartz–Zippel lemma (Lemma 1). Indeed, given that Bad'_i occurs, the coefficient of degree i in T is a non-zero polynomial of degree 2 in β_1 and β_2, which for $\beta_1, \beta_2 \twoheadleftarrow \mathbb{Z}_p^*$ will vanish with probability at most $2/(2^{\lambda-1} - 1)$.

(1a) KEA1 **and** (1b) KEA3**.** Both statements directly follow from result (1c) above, by observing that KEA1 = 0-PKE and KEA3 = 1-PKE.

```
Adversary $\mathcal{B}_0(\gamma)$:
    $Q(\boldsymbol{X}_0, \boldsymbol{X}_1, \boldsymbol{Y}_1, \boldsymbol{Y}_2) \leftarrow \boldsymbol{Y}_2 - \boldsymbol{X}_1\boldsymbol{Y}_1$
    $P(S) \leftarrow S$; return $(Q, P)$
```

```
Adversary $\mathcal{B}_0(\gamma)$:
    $Q(\boldsymbol{X}_0, \dots, \boldsymbol{X}_3, \boldsymbol{Y}_1, \boldsymbol{Y}_2) \leftarrow \boldsymbol{Y}_2 - \boldsymbol{X}_2\boldsymbol{Y}_1$
    $\boldsymbol{P}_1(\boldsymbol{S}_1, \boldsymbol{S}_2) \leftarrow \boldsymbol{S}_1$; $\boldsymbol{P}_2(\boldsymbol{S}_1, \boldsymbol{S}_2) \leftarrow \boldsymbol{S}_2$
    $\boldsymbol{P}_3(\boldsymbol{S}_1, \boldsymbol{S}_2) \leftarrow \boldsymbol{S}_1\boldsymbol{S}_2$; return $(Q, \boldsymbol{P})$
```

Figure 9: First-stage UK adversaries $\mathcal{B}_0$ from the proof that UK implies KEA1 and KEA3.

(2a) d-PKE **(type-3 groups).** The proof strongly resembles the one given above for simple groups, but we nonetheless provide all details. Given a d-PKE adversary $\mathcal{A}$, let $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$ be the UK adversary where $\mathcal{B}_0$ is given in Figure 10 (top), and $\mathcal{B}_1$ runs $\mathcal{A}$ and returns its output. Let $\mathcal{F}$ be a UK extractor for $\mathcal{B}$ (as per hardness of UK for $(\mathrm{B}, \mathcal{S}, \mathfrak{B})$) that outputs $(\boldsymbol{w}, \boldsymbol{w}') = \begin{pmatrix} \boldsymbol{w}_{10} & \cdots & \boldsymbol{w}_{1,d(\lambda)} & \boldsymbol{w}'_{10} & \cdots & \boldsymbol{w}'_{1,d(\lambda)} \\ \boldsymbol{w}_{20} & \cdots & \boldsymbol{w}_{2,d(\lambda)} & \boldsymbol{w}'_{20} & \cdots & \boldsymbol{w}'_{2,d(\lambda)} \end{pmatrix}$. Define a d-PKE extractor $\mathcal{E}$ for $\mathcal{A}$ that runs $\mathcal{F}$ and outputs $(\boldsymbol{w}_{10}, \dots, \boldsymbol{w}_{1,d(\lambda)})$. We claim that $\mathrm{Adv}^{d\text{-pke}}_{\mathrm{B},\mathcal{A},\mathcal{E}}$ is negligible, proving that d-PKE holds for B. To that end, consider the following sequence of games (the formal description of which can be found in Figure 10 (second from bottom)):

G_0: This is the original d-PKE game for B run with adversary $\mathcal{A}$ and extractor $\mathcal{E}$. We reformulate the winning condition by letting the game immediately return 0 if $[y]_1 \neq [ac]_1$, and then checking $\big([c]_1 = \prod_{i=0}^{d(\lambda)} [\boldsymbol{w}_{1,i} s^i]_1\big)$.

G_1: This game proceeds as G_0, but additionally returns 0 if $(\boldsymbol{w}, \boldsymbol{w}')$ is not a correct representation of all outputs of $\mathcal{A}$ in terms of all its (group element) inputs.

G_2: This game proceeds as G_1, but additionally returns 0 if the representation $(\boldsymbol{w}, \boldsymbol{w}')$ is not of the form $\begin{pmatrix} \boldsymbol{w}_{10} & \cdots & \boldsymbol{w}_{1,d(\lambda)} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \boldsymbol{w}_{10} & \cdots & \boldsymbol{w}_{1,d(\lambda)} \end{pmatrix}$.

We now bound the difference between the success probabilities in subsequent games.

$\mathrm{G}_0 \rightsquigarrow \mathrm{G}_1$. Notice that G_0 and G_1 are identical until Bad, where Bad is the event in the d-PKE game for B played by $(\mathcal{A}, \mathcal{E})$ that $[y]_1 = [ac]_1$ and $(\boldsymbol{w}, \boldsymbol{w}')$ is not a correct representation of $([c]_1, [y]_1)$ in terms of $([1]_1, \dots, [s^{d(\lambda)}]_1, [a]_1, \dots, [as^{d(\lambda)}]_1)$. By definition of $\mathcal{S}$, $\mathcal{B}$ and $\mathcal{F}$, this corresponds to the event that $(\mathcal{B}, \mathcal{F})$ win the UK game for $(\mathrm{B}, \mathcal{S})$. By the fundamental lemma of game playing we therefore have

$$|\Pr[\mathrm{G}_1] - \Pr[\mathrm{G}_0]| \leq \Pr[\mathsf{Bad}] = \mathrm{Adv}^{\mathrm{uk}}_{\mathrm{B},\mathcal{S},\mathcal{B},\mathcal{F}}(\lambda)\,.$$

$\mathrm{G}_1 \rightsquigarrow \mathrm{G}_2$. Observe that G_1 and G_2 are identical until Bad', where Bad' is the event in the d-PKE game for B played by $(\mathcal{A}, \mathcal{E})$ that $([c]_1, [y]_1)$ and $(\boldsymbol{w}, \boldsymbol{w}')$ are correct, but not of the form $\begin{pmatrix} \boldsymbol{w}_{10} & \cdots & \boldsymbol{w}_{1,d(\lambda)} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \boldsymbol{w}_{10} & \cdots & \boldsymbol{w}_{1,d(\lambda)} \end{pmatrix}$. Again by the fundamental lemma of game playing we have $|\Pr[\mathrm{G}_2] - \Pr[\mathrm{G}_1]| \leq \Pr[\mathsf{Bad}']$.

Collecting all terms above, we obtain

$$\mathrm{Adv}^{d\text{-pke}}_{\mathrm{B},\mathcal{A},\mathcal{E}}(\lambda) = \Pr[\mathrm{G}_0] \leq \Pr[\mathrm{G}_2] + \mathrm{Adv}^{\mathrm{uk}}_{\mathrm{B},\mathcal{S},\mathcal{B},\mathcal{F}}(\lambda) + \Pr[\mathsf{Bad}'] = \mathrm{Adv}^{\mathrm{uk}}_{\mathrm{B},\mathcal{S},\mathcal{B},\mathcal{F}}(\lambda) + \Pr[\mathsf{Bad}']\,,$$

where the last equality holds because $\Pr[\mathrm{G}_2] = 0$, since the return statement introduced in G_2 ensures that $[c]_1 = [\boldsymbol{w}_{10}]_1 \cdots [\boldsymbol{w}_{1,d(\lambda)} s^{d(\lambda)}]_1$, while the winning condition is exactly $[c]_1 \neq [\boldsymbol{w}_{10}]_1 \cdots [\boldsymbol{w}_{1,d(\lambda)} s^{d(\lambda)}]_1$.

We are now left with bounding $\Pr[\mathsf{Bad}']$. To that end, consider the adversary $\mathcal{C}$ against $(d+1, 1)$-DL for B defined in Figure 10 (second from top); we will bound $\Pr[\mathsf{Bad}']$ in terms of the advantage of $\mathcal{C}$. To do so, we show that if Bad' occurs, then the polynomial T constructed by $\mathcal{C}$ is non-zero with overwhelming probability. Whenever that is the case, $\mathcal{C}$ will succeed in winning the $(d+1, 1)$-DL game for B, because it can recover x by finding the correct root of T using Berlekamp's algorithm.

```
Adversary $\mathcal{B}_0(\gamma)$:
$Q((\boldsymbol{X}_{1,i})_{i=0}^{d(\lambda)}, (\boldsymbol{X}'_{1,i})_{i=0}^{d(\lambda)}, \boldsymbol{X}_{2,0}, \boldsymbol{X}_{2,1}, \boldsymbol{X}_{2,2}, \boldsymbol{Y}_{1,1}, \boldsymbol{Y}_{1,2}) \leftarrow \boldsymbol{Y}_{1,2} - \boldsymbol{X}'_{1,0}\boldsymbol{Y}_{1,1}$
for $i = 1$ to $d(\lambda)$ do $\boldsymbol{P}_{1,i}(\boldsymbol{S}_1, \boldsymbol{S}_2) \leftarrow \boldsymbol{S}_1^i$
for $i = 0$ to $d(\lambda)$ do $\boldsymbol{P}'_{1,i}(\boldsymbol{S}_1, \boldsymbol{S}_2) \leftarrow \boldsymbol{S}_2\boldsymbol{S}_1^i$
$\boldsymbol{P}_{2,1}(\boldsymbol{S}_1, \boldsymbol{S}_2) \leftarrow \boldsymbol{S}_1$; $\boldsymbol{P}_{2,2}(\boldsymbol{S}_1, \boldsymbol{S}_2) \leftarrow \boldsymbol{S}_2$; return $(Q, \boldsymbol{P}_1, \boldsymbol{P}'_1, \boldsymbol{P}_2)$
```

```
Adversary $\mathcal{C}(\gamma, [x]_1, \ldots, [x^{d(\lambda)}]_1, [x^{d(\lambda)+1}]_1, [x]_2)$:
$\beta_1, \beta_2 \twoheadleftarrow \mathbb{Z}_p^*$; $\alpha_1, \alpha_2 \twoheadleftarrow \mathbb{Z}_p$; $(Q, \boldsymbol{P}_1, \boldsymbol{P}'_1, \boldsymbol{P}_2) \twoheadleftarrow \mathcal{B}_0(\gamma)$; $\mathsf{S} \leftarrow \emptyset$
$([c]_1, [y]_1) \twoheadleftarrow$
    $\mathcal{A}(\gamma, ([(\beta_1 x + \alpha_1)^i]_1)_{i=1}^{d(\lambda)}, ([(\beta_2 x + \alpha_2)(\beta_1 x + \alpha_1)^i]_1)_{i=0}^{d(\lambda)}, [\beta_1 x + \alpha_1]_2, [\beta_2 x + \alpha_2]_2)$
$\begin{pmatrix} \boldsymbol{w}_{10} & \cdots & \boldsymbol{w}_{1,d(\lambda)} & \boldsymbol{w}'_{10} & \cdots & \boldsymbol{w}'_{1,d(\lambda)} \\ \boldsymbol{w}_{20} & \cdots & \boldsymbol{w}_{2,d(\lambda)} & \boldsymbol{w}'_{20} & \cdots & \boldsymbol{w}'_{2,d(\lambda)} \end{pmatrix} \twoheadleftarrow \mathcal{F}(\mathsf{trace}(\mathcal{A}))$; $\boldsymbol{P}_{1,0}, \boldsymbol{P}_{2,0} \leftarrow 1$
for $j = 0$ to $d(\lambda)$ do
    $\boldsymbol{X}_{1,j} \leftarrow \boldsymbol{P}_{1,j}(\beta_1 X + \alpha_1, \beta_2 X + \alpha_2)$; $\boldsymbol{X}'_{1,j} \leftarrow \boldsymbol{P}'_{1,j}(\beta_1 X + \alpha_1, \beta_2 X + \alpha_2)$
for $j = 0$ to $2$ do $\boldsymbol{X}_{2,j} \leftarrow \boldsymbol{P}_{2,j}(\beta_1 X + \alpha_1, \beta_2 X + \alpha_2)$
for $i = 1$ to $2$ do $\boldsymbol{Y}_{1,i} \leftarrow \sum_{j=0}^{d(\lambda)} \boldsymbol{w}_{ij}\boldsymbol{X}_{1,j} + \boldsymbol{w}'_{ij}\boldsymbol{X}'_{1,j}$
$T(X) \leftarrow Q(\boldsymbol{X}_{1,0}, \ldots, \boldsymbol{X}'_{1,d(\lambda)}, \boldsymbol{X}_{2,0}, \boldsymbol{X}_{2,1}, \boldsymbol{X}_{2,2}, \boldsymbol{Y}_{1,1}, \boldsymbol{Y}_{1,2})$
if $(T(X) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{Berlekamp}(T, p)$
for $x' \in \mathsf{S}$ do if $([x']_1 = [x]_1)$ then return $x'$
return 0
```

```
Game $\mathrm{G}_0(\lambda)$:
$\gamma \twoheadleftarrow \mathrm{B}(1^\lambda)$
$s, a \twoheadleftarrow \mathbb{Z}_p$
$([c]_1, [y]_1) \twoheadleftarrow \mathcal{A}(\gamma,$
    $([s^i]_1)_{i=1}^d,$
    $([as^i]_1)_{i=0}^d,$
    $[s]_2, [a]_2)$
$(\boldsymbol{w}, \boldsymbol{w}') \twoheadleftarrow$
    $\mathcal{F}(\mathsf{trace}(\mathcal{A}))$
if $([y]_1 \neq [ac]_1)$ then
    return 0
return $([c]_1 \neq$
    $\prod_{i=0}^d [\boldsymbol{w}_{1,i} s^i]_1)$
```

```
Game $\mathrm{G}_1(\lambda)$:
$\gamma \twoheadleftarrow \mathrm{B}(1^\lambda)$; $s, a \twoheadleftarrow \mathbb{Z}_p$
$([c]_1, [y]_1) \twoheadleftarrow$
    $\mathcal{A}(\gamma, ([s^i]_1)_{i=1}^d, ([as^i]_1)_{i=0}^d,$
    $[s]_2, [a]_2)$
$(\boldsymbol{w}, \boldsymbol{w}') \twoheadleftarrow \mathcal{F}(\mathsf{trace}(\mathcal{A}))$
if $([y]_1 \neq [ac]_1)$ then return 0
if $([c]_1 \neq \prod_{i=0}^d [\boldsymbol{w}_{1,i} s^i]_1 [\boldsymbol{w}'_{1,i} as^i]_1)$
    $\vee$
    $([y]_1 \neq \prod_{i=0}^d [\boldsymbol{w}_{2,i} s^i]_1 [\boldsymbol{w}'_{2,i} as^i]_1)$
    then return 0
return $([c]_1 \neq \prod_{i=0}^d [\boldsymbol{w}_{1,i} s^i]_1)$
```

```
Game $\mathrm{G}_2(\lambda)$:
$\gamma \twoheadleftarrow \mathrm{B}(1^\lambda)$; $s, a \twoheadleftarrow \mathbb{Z}_p$
$([c]_1, [y]_1) \twoheadleftarrow$
    $\mathcal{A}(\gamma, ([s^i]_1)_{i=1}^d, ([as^i]_1)_{i=0}^d,$
    $[s]_2, [a]_2)$
$(\boldsymbol{w}, \boldsymbol{w}') \twoheadleftarrow \mathcal{F}(\mathsf{trace}(\mathcal{A}))$
if $([y]_1 \neq [ac]_1)$ then return 0
if $([c]_1 \neq \prod_{i=0}^d [\boldsymbol{w}_{1,i} s^i]_1 [\boldsymbol{w}'_{1,i} as^i]_1) \vee$
    $([y]_1 \neq \prod_{i=0}^d [\boldsymbol{w}_{2,i} s^i]_1 [\boldsymbol{w}'_{2,i} as^i]_1)$
    then return 0
if $(\boldsymbol{w}'_{10} \neq 0) \vee \cdots \vee (\boldsymbol{w}'_{1,d} \neq 0) \vee$
    $(\boldsymbol{w}_{20} \neq 0) \vee \cdots \vee (\boldsymbol{w}_{2,d} \neq 0) \vee$
    $(\boldsymbol{w}_{10} \neq \boldsymbol{w}'_{20}) \vee \cdots \vee$
    $(\boldsymbol{w}_{1,d} \neq \boldsymbol{w}'_{2,d})$ then return 0
return $([c]_1 \neq \prod_{i=0}^d [\boldsymbol{w}_{1,i} s^i]_1)$
```

```
Game $\mathrm{G}'(\lambda)$:
$\gamma \twoheadleftarrow \mathrm{B}(1^\lambda)$; $r_1, r_2 \twoheadleftarrow \mathbb{Z}_p$ $(Q, \boldsymbol{P}_1, \boldsymbol{P}'_1, \boldsymbol{P}_2) \twoheadleftarrow \mathcal{B}_0(\gamma)$; $\mathsf{S} \leftarrow \emptyset$
$([c]_1, [y]_1) \twoheadleftarrow \mathcal{A}(\gamma, ([r_1^i]_1)_{i=1}^{d(\lambda)}, ([r_2 r_1^i]_1)_{i=0}^{d(\lambda)}, [r_1]_2, [r_2]_2)$
$(\boldsymbol{w}, \boldsymbol{w}') \twoheadleftarrow \mathcal{F}(\mathsf{trace}(\mathcal{A}))$; $x \twoheadleftarrow \mathbb{Z}_p$; $\beta_1, \beta_2 \twoheadleftarrow \mathbb{Z}_p^*$; $\alpha_1 \leftarrow r_1 - \beta_1 x$; $\alpha_2 \leftarrow r_2 - \beta_2 x$; $\boldsymbol{P}_{1,0}, \boldsymbol{P}_{2,0} \leftarrow 1$
for $j = 0$ to $d(\lambda)$ do $\boldsymbol{X}_{1,j} \leftarrow \boldsymbol{P}_{1,j}(\beta_1 X + \alpha_1, \beta_2 X + \alpha_2)$; $\boldsymbol{X}'_{1,j} \leftarrow \boldsymbol{P}'_{1,j}(\beta_1 X + \alpha_1, \beta_2 X + \alpha_2)$
for $j = 0$ to $2$ do $\boldsymbol{X}_{2,j} \leftarrow \boldsymbol{P}_{2,j}(\beta_1 X + \alpha_1, \beta_2 X + \alpha_2)$
for $i = 1$ to $2$ do $\boldsymbol{Y}_{1,i} \leftarrow \sum_{j=0}^{d(\lambda)} \boldsymbol{w}_{ij}\boldsymbol{X}_{1,j} + \boldsymbol{w}'_{ij}\boldsymbol{X}'_{1,j}$
$T(X) \leftarrow Q(\boldsymbol{X}_{1,0}, \ldots, \boldsymbol{X}'_{1,d(\lambda)}, \boldsymbol{X}_{2,0}, \boldsymbol{X}_{2,1}, \boldsymbol{X}_{2,2}, \boldsymbol{Y}_{1,1}, \boldsymbol{Y}_{1,2})$
if $(T(X) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{Berlekamp}(T, p)$
$x' \leftarrow 0$; for $z \in \mathsf{S}$ do if $([z]_1 = [x]_1)$ then return $x' \leftarrow z$; break
return $(x = x')$
```

Figure 10: *Top:* First-stage UK adversary $\mathcal{B}_0$ from the proof that UK implies d-PKE for type-3 bilinear group schemes. *Second from top:* Adversary $\mathcal{C}$ against $(d+1,1)$-DL from the proof that UK implies d-PKE for type-3 bilinear group schemes. *Second from bottom and bottom:* Code of the intermediate games in the proof that UK implies d-PKE for type-3 bilinear group schemes.

```
Adversary $\mathcal{B}_0(\gamma)$:
$Q((\boldsymbol{X}_{1,i})_{i=0}^{d(\lambda)-1}, \boldsymbol{X}_{2,0}, \boldsymbol{X}_{2,1}, (\boldsymbol{Y}_{1,i}, \boldsymbol{C}_i)_{i=1}^{2}) \leftarrow \boldsymbol{Y}_{1,1} - \boldsymbol{C}_2 - \boldsymbol{Y}_{1,2}(\boldsymbol{X}_{2,1} - \boldsymbol{C}_1)$
for $i = 1$ to $d(\lambda) - 1$ do $\boldsymbol{P}_{1,i}(S) \leftarrow S^i$
$P_2(S) \leftarrow S$; return $(Q, \boldsymbol{P}_1, P_2)$
```

Figure 11: First-stage UK adversary $\mathcal{B}_0$ from the proof that UK implies d-KZG for type-3 bilinear group schemes.

Starting from $(d+1,1)\text{-DL}_{\mathrm{B}}^{\mathcal{C}}$, we transition to a game G' (see Figure 10 (bottom)) where $\mathcal{A}$ is given group elements $([r_1]_1, \ldots, [r_1^{d(\lambda)}]_1, [r_2]_1, \ldots, [r_2 r_1^{d(\lambda)}]_1, [r_1]_2, [r_2]_2)$ for $r_1, r_2 \twoheadleftarrow \mathbb{Z}_p$ and then, only after $\mathcal{F}$ is run, G' samples $x \twoheadleftarrow \mathbb{Z}_p$, $\beta_1, \beta_2 \twoheadleftarrow \mathbb{Z}_p^*$, and then sets $\alpha_1 \leftarrow r_1 - \beta_1 x$ and $\alpha_2 \leftarrow r_2 - \beta_2 x$. Observe that $\Pr[(d+1,1)\text{-DL}_{\mathrm{B}}^{\mathcal{C}}] = \Pr[\mathrm{G}']$, because the inputs of $\mathcal{A}$ are equally distributed in both games. Now write

$$\mathsf{Bad}' = \mathsf{Bad}'_{d(\lambda)+2} \vee \cdots \vee \mathsf{Bad}'_0\,,$$

where

$$\begin{aligned}
\mathsf{Bad}'_{d+2} &\coloneqq \mathsf{Bad}' \wedge (\boldsymbol{w}'_{1,d} \neq 0)\\
\mathsf{Bad}'_{d+1} &\coloneqq \mathsf{Bad}' \wedge \neg\mathsf{Bad}'_{d+2} \wedge ((\boldsymbol{w}'_{1,d-1} \neq 0) \vee (\boldsymbol{w}_{1,d} \neq \boldsymbol{w}'_{2,d}))\\
\mathsf{Bad}'_{d} &\coloneqq \mathsf{Bad}' \wedge \neg\mathsf{Bad}'_{d+2} \wedge \neg\mathsf{Bad}'_{d+1} \wedge ((\boldsymbol{w}'_{1,d-2} \neq 0) \vee (\boldsymbol{w}_{2,d} \neq 0) \vee (\boldsymbol{w}_{1,d-1} \neq \boldsymbol{w}'_{2,d-1}))\\
&\;\;\vdots\\
\mathsf{Bad}'_{2} &\coloneqq \mathsf{Bad}' \wedge \neg\mathsf{Bad}'_{d+2} \wedge \cdots \wedge \neg\mathsf{Bad}'_{3} \wedge ((\boldsymbol{w}'_{10} \neq 0) \vee (\boldsymbol{w}_{22} \neq 0) \vee (\boldsymbol{w}_{11} \neq \boldsymbol{w}'_{21}))\\
\mathsf{Bad}'_{1} &\coloneqq \mathsf{Bad}' \wedge \neg\mathsf{Bad}'_{d+2} \wedge \cdots \wedge \neg\mathsf{Bad}'_{2} \wedge ((\boldsymbol{w}_{21} \neq 0) \vee (\boldsymbol{w}_{10} \neq \boldsymbol{w}'_{20}))\\
\mathsf{Bad}'_{0} &\coloneqq \mathsf{Bad}' \wedge \neg\mathsf{Bad}'_{d+2} \wedge \cdots \wedge \neg\mathsf{Bad}'_{1} \wedge (\boldsymbol{w}_{20} \neq 0)\,.
\end{aligned}$$

Here, Bad'_i is the event that the coefficient of degree i in T is non-zero as a polynomial in β_1 and β_2, but every coefficient of higher degree is zero. (Note that $\Pr[\mathsf{Bad}'_0] = 0$, because if Bad' occurs, then $T(x) = 0$, so it cannot be that the constant term is the only non-zero term of T.) Then

$$\begin{aligned}
\Pr[(d+1,1)\text{-DL}_{\mathrm{B}}^{\mathcal{C}}(\lambda)] = \Pr[\mathrm{G}'] \geq \Pr[\mathrm{G}' \wedge \mathsf{Bad}'] &= \sum\nolimits_{i=1}^{d(\lambda)+2} \Pr[\mathrm{G}' \mid \mathsf{Bad}'_i] \Pr[\mathsf{Bad}'_i]\\
&\geq \left(1 - \frac{2}{2^{\lambda-1}-1}\right)\left(\sum\nolimits_{i=1}^{d(\lambda)+2} \Pr[\mathsf{Bad}'_i]\right) = \left(1 - \frac{2}{2^{\lambda-1}-1}\right)\Pr[\mathsf{Bad}']\,.
\end{aligned}$$

Here, the last inequality holds because of the Schwartz–Zippel lemma (Lemma 1). Indeed, given that Bad'_i occurs, the coefficient of degree i in T is a non-zero polynomial of degree 2 in β_1 and β_2, which for $\beta_1, \beta_2 \twoheadleftarrow \mathbb{Z}_p^*$ will vanish with probability at most $2/(2^{\lambda-1} - 1)$.

(2b) d-KZG**.** Given a d-KZG adversary $\mathcal{A}$, let $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$ be the UK adversary where $\mathcal{B}_0$ is given in Figure 11, and $\mathcal{B}_1$ runs $\mathcal{A}$ and returns its output. Let $\mathcal{F}$ be a UK extractor for $\mathcal{B}$ (as per hardness of UK for $(\mathrm{B}, \mathcal{S}, \mathfrak{B})$). Define a d-KZG extractor $\mathcal{E}$ for $\mathcal{A}$ that runs $\mathcal{F}$ and returns the representation of the first output of $\mathcal{B}_1$, i.e., the first row of the output of $\mathcal{F}$. Then clearly $\mathrm{Adv}_{\mathrm{B},\mathcal{A},\mathcal{E}}^{d\text{-kzg}}$ is negligible, proving that d-KZG holds for B, since any extractor for $\mathcal{A}$ is permitted to use all the inputs of $\mathcal{A}$ (from the first group) in its representation, just as $\mathcal{F}$ itself. In particular, no reduction is needed to show that some inputs are not used.

```
Adversary $\mathcal{B}_0(\varpi)$:
$(\ell, (U_i, V_i, W_i)_{i=0}^{m}, T) \twoheadleftarrow \mathcal{A}_0(\varpi)$
$Q((\boldsymbol{X}_{1,i})_{i=0}^{2d(\lambda)+m+3}, (\boldsymbol{X}_{2,i})_{i=0}^{d(\lambda)+3}, (\boldsymbol{Y}_{1,i})_{i=1}^{2}, Y_2, (\boldsymbol{C}_i)_{i=0}^{\ell})$
    $\leftarrow \boldsymbol{Y}_{1,1} Y_2 - \boldsymbol{Y}_{1,2} \boldsymbol{X}_{2,3} - \boldsymbol{X}_{1,1} \boldsymbol{X}_{2,1} - \sum_{i=0}^{\ell} \boldsymbol{C}_i \boldsymbol{X}_{1,2d(\lambda)+3+i} \boldsymbol{X}_{2,2}$
$\boldsymbol{P}_{1,1}(\boldsymbol{S}) \leftarrow \boldsymbol{S}_1 \boldsymbol{S}_3 \boldsymbol{S}_4$; $\boldsymbol{P}_{1,2}(\boldsymbol{S}) \leftarrow \boldsymbol{S}_2 \boldsymbol{S}_3 \boldsymbol{S}_4$; $\boldsymbol{P}_{1,3}(\boldsymbol{S}) \leftarrow \boldsymbol{S}_3 \boldsymbol{S}_4^2$
for $i = 0$ to $d(\lambda) - 1$ do $\boldsymbol{P}_{1,4+i}(\boldsymbol{S}) \leftarrow \boldsymbol{S}_3 \boldsymbol{S}_4 \boldsymbol{S}_5^i$
for $i = 0$ to $d(\lambda) - 2$ do $\boldsymbol{P}_{1,d+4+i}(\boldsymbol{S}) \leftarrow \boldsymbol{S}_3 \boldsymbol{S}_5^i T(\boldsymbol{S}_5)$
for $i = 0$ to $\ell$ do $\boldsymbol{P}_{1,2d+3+i}(\boldsymbol{S}) \leftarrow \boldsymbol{S}_2 \boldsymbol{S}_4 U_i(\boldsymbol{S}_5) + \boldsymbol{S}_1 \boldsymbol{S}_4 V_i(\boldsymbol{S}_5) + \boldsymbol{S}_4 W_i(\boldsymbol{S}_5)$
for $i = \ell + 1$ to $m$ do $\boldsymbol{P}_{1,2d+3+i}(\boldsymbol{S}) \leftarrow \boldsymbol{S}_2 \boldsymbol{S}_3 U_i(\boldsymbol{S}_5) + \boldsymbol{S}_1 \boldsymbol{S}_3 V_i(\boldsymbol{S}_5) + \boldsymbol{S}_3 W_i(\boldsymbol{S}_5)$
$\boldsymbol{P}_{2,1}(\boldsymbol{S}) \leftarrow \boldsymbol{S}_2 \boldsymbol{S}_3 \boldsymbol{S}_4$; $\boldsymbol{P}_{2,2}(\boldsymbol{S}) \leftarrow \boldsymbol{S}_3^2 \boldsymbol{S}_4$; $\boldsymbol{P}_{2,3}(\boldsymbol{S}) \leftarrow \boldsymbol{S}_3 \boldsymbol{S}_4^2$
for $i = 0$ to $d(\lambda) - 1$ do $\boldsymbol{P}_{2,4+i}(\boldsymbol{S}) \leftarrow \boldsymbol{S}_3 \boldsymbol{S}_4 \boldsymbol{S}_5^i$
return $(Q, \boldsymbol{P}_1, \boldsymbol{P}_2)$
```

Figure 12: First-stage UK adversary $\mathcal{B}_0$ from the proof that UK implies d-GROTH16 for type-3 bilinear group schemes. Here, all polynomials in $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ are in variables $\boldsymbol{S} = (\boldsymbol{S}_1, \ldots, \boldsymbol{S}_5)$.

(2c) d-GROTH16**.** Given a d-GROTH16 adversary $\mathcal{A}$ with $\mathcal{A}_0 \in \mathfrak{A}$, let $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$ be the UK adversary where $\mathcal{B}_0$ is given in Figure 12, and $\mathcal{B}_1$ runs $\mathcal{A}$ and returns its output. Let $\mathcal{F}$ be a UK extractor for $\mathcal{B}$ (as per hardness of UK for $(\mathrm{B}, \mathcal{S}, \mathfrak{B})$). Define a d-GROTH16 extractor $\mathcal{E}$ for $\mathcal{A}$ as $\mathcal{E} := \mathcal{F}$. Then clearly $\mathrm{Adv}_{\mathrm{B},\mathcal{A},\mathcal{E}}^{d\text{-groth16}}$ is negligible, proving that d-GROTH16 holds for B, since any extractor for $\mathcal{A}$ is permitted to use all the inputs of $\mathcal{A}$ (separately in each group) in its representation, just as $\mathcal{F}$ itself. In particular, no additional reduction is required to show that some inputs are not used. □

5 Soundness of DH-KE

In this section, we study the soundness of DH-KE, a simple knowledge assumption introduced by Bellare, Fuchsbauer, and Scafuro [BFS16]. Following the blueprint given in [BFS16], we prove that DH-KE holds in the GBM3-H, and then show that it holds in the ABM3-H. These results serve as a "warm-up" to the more complex soundness proofs for the UK assumption presented in Sections 6 and 7. We first recall the definition of DH-KE.

DH-KE [BFS16]**.** 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

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

where the game DH-KE is defined in Figure 13 (top). 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 $\mathrm{Adv}_{\mathrm{B},\mathcal{A},\mathcal{E}}^{\text{dh-ke}}$ is negligible. DH-KE for type-2 and type-1 bilinear group schemes is defined analogously.

Remark. A 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, the version where $\mathcal{A}$ returns $[c]_T$ and the game checks if $(e([a]_1, [b]_2) = [c]_T)$ is 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 $h_T := e(h_1, h_2)$, and return (h_1, h_2, h_T), without "knowing" any discrete logarithms.

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

```
Extractor $\mathcal{E}^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(\mathsf{trace}(\mathcal{A}))$:
  parse $\mathsf{trace}(\mathcal{A}) = (r_{\mathcal{A}}, u_1, u_2, \boldsymbol{h})$; $o, v \leftarrow 0$
  $U_{\tau_1}, U_{\tau_2}, U_{\tau_T}, U_{H_1}, U_{H_2}, U_{H_T} \leftarrow [\,]$
  $U_{\tau_1}[1] \leftarrow u_1$; $U_{\tau_2}[1] \leftarrow u_2$
  $\boldsymbol{v} \leftarrow \mathcal{A}^{\overline{\mathsf{op}}_1,\overline{\mathsf{op}}_2,\overline{\mathsf{op}}_T,\overline{\mathsf{H}}_1,\overline{\mathsf{H}}_2,\overline{\mathsf{H}}_T,\overline{\mathsf{e}}}(u_1, u_2; r_{\mathcal{A}})$
  for $\nu = 1$ to $2$ do
    if $(\boldsymbol{v}_\nu \notin \mathrm{Rng}(U_{\tau_\nu}))$ then
      $v \leftarrow v + 1$; $U_{\tau_\nu}[\boldsymbol{R}_v] \leftarrow \boldsymbol{v}_\nu$
    parse $U_{\tau_\nu}^{-1}[\boldsymbol{v}_\nu] = w_\nu + \sum_l \boldsymbol{b}_{\nu l} \boldsymbol{R}_l$
  if $(\boldsymbol{b}_1 = 0)$ then return $w_1$
  return $w_2$
```

```
Proc. $\overline{\mathsf{op}}_\nu(h_1, h_2)$:
  for $i = 1$ to $2$ do
    if $(h_i \notin \mathrm{Rng}(U_{\tau_\nu}))$
    then
      $v \leftarrow v + 1$
      $U_{\tau_\nu}[\boldsymbol{R}_v] \leftarrow h_i$
    $x_i \leftarrow U_{\tau_\nu}^{-1}[h_i]$
  $x \leftarrow x_1 + x_2$; $o \leftarrow o + 1$
  if $(x \notin \mathrm{Dom}(U_{\tau_\nu}))$
  then
    $U_{\tau_\nu}[x] \leftarrow \boldsymbol{h}_o$
  return $U_{\tau_\nu}[x]$
```

```
Proc. $\overline{\mathsf{op}}_T(h_1, h_2)$:
  for $i = 1$ to $2$ do
    if $(h_i \notin \mathrm{Rng}(U_{\tau_T}))$
    then
      $x_i \leftarrow \mathbb{Z}_p \setminus \mathrm{Dom}(U_{\tau_T})$
      $U_{\tau_T}[x_i] \leftarrow h_i$
    $x_i \leftarrow U_{\tau_T}^{-1}[h_i]$
  $x \leftarrow x_1 + x_2$; $o \leftarrow o + 1$
  if $(x \notin \mathrm{Dom}(U_{\tau_T}))$
  then
    $U_{\tau_T}[x] \leftarrow \boldsymbol{h}_o$
  return $U_{\tau_T}[x]$
```

```
Proc. $\overline{\mathsf{H}}_\nu(m)$:
  if $(m \notin \mathrm{Dom}(U_{H_\nu}))$ then
    $v \leftarrow v + 1$; $U_{H_\nu}[m] \leftarrow \boldsymbol{R}_v$
  $r \leftarrow U_{H_\nu}[m]$; $o \leftarrow o + 1$
  if $(r \notin \mathrm{Dom}(U_{\tau_\nu}))$ then
    $U_{\tau_\nu}[r] \leftarrow \boldsymbol{h}_o$
  return $U_{\tau_\nu}[r]$
```

```
Proc. $\overline{\mathsf{H}}_T(m)$:
  if $(m \notin \mathrm{Dom}(U_{H_T}))$ then
    $r \twoheadleftarrow \mathbb{Z}_p$; $U_{H_T}[m] \leftarrow r$
  $r \leftarrow U_{H_T}[m]$; $o \leftarrow o + 1$
  if $(r \notin \mathrm{Dom}(U_{\tau_T}))$ then
    $U_{\tau_T}[r] \leftarrow \boldsymbol{h}_o$
  return $U_{\tau_T}[r]$
```

```
Proc. $\overline{\mathsf{e}}(h_1, h_2)$:
  for $\nu = 1$ to $2$ do
    if $(h_\nu \notin \mathrm{Rng}(U_{\tau_\nu}))$ then
      $v \leftarrow v + 1$; $U_{\tau_\nu}[\boldsymbol{R}_v] \leftarrow h_\nu$
    $x_\nu \leftarrow U_{\tau_\nu}^{-1}[h_\nu]$
  $x \leftarrow x_1 x_2$; $o \leftarrow o + 1$
  if $(x \notin \mathrm{Dom}(U_{\tau_T}))$ then
    $U_{\tau_T}[x] \leftarrow \boldsymbol{h}_o$
  return $U_{\tau_T}[x]$
```

Figure 13: *Top:* Game defining the DH-KE assumption. Here, B is a type-3 bilinear group scheme. *Bottom:* Definition of the extractor $\mathcal{E}$ from the proof of Theorem 1. Counters o and v are shared between all oracles, and ν is an index ranging over $\{1, 2\}$.

5.1 Soundness of DH-KE in GBM3-H

Theorem 1 (DH-KE holds in GBM3-H)**.** *Let $p \in \mathbb{N}$ be prime, and fix $\mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_T \subseteq \{0,1\}^*$ with $|\mathsf{G}_1| = |\mathsf{G}_2| = |\mathsf{G}_T| = p$. Then the* DH-KE *assumption holds in the* GBM3-H *with parameters $(p, \mathbf{G})$. More precisely, for every adversary $\mathcal{A}$ in the* DH-KE *game in the* GBM3-H *with parameters $(p, \mathbf{G})$, there exists an extractor $\mathcal{E}$ such that*

$$\mathrm{Adv}^{\text{dh-ke}}_{p,\mathbf{G},\mathcal{A},\mathcal{E}} \leq \mathcal{O}\left(\frac{(q_{\mathsf{op}} + q_{\mathsf{H}} + q_{\mathsf{e}})^2}{p}\right). \tag{1}$$

Here, q_{op}, q_{H}, and q_{e} are upper bounds on the number of queries made by $\mathcal{A}$ to the respective oracles.

Proof. Fix an adversary $\mathcal{A}$ in the DH-KE game, and define an extractor $\mathcal{E}$ as in Figure 13 (bottom). This extractor essentially re-runs $\mathcal{A}$ on its view and observes its oracle queries, keeping track of the discrete logarithms of the elements queried by $\mathcal{A}$ via tables U_{τ_μ}, $\mu \in \{1, 2, T\}$. Whenever $\mathcal{E}$ is unable to "explain" an element in G_ν, $\nu \in \{1, 2\}$, it instead stores a fresh variable $\boldsymbol{R}_v$ in U_{τ_ν}. On the other hand, oracles pertaining to G_T are implemented via lazy sampling with no further modifications.

We claim that this extractor allows proving Inequality (1). To that end, consider the following sequence of games (the formal description of which can be found in Figure 14):

G_0: This is the original DH-KE game in the GBM3-H with parameters $(p, \mathbf{G})$ run with adversary $\mathcal{A}$ and extractor $\mathcal{E}$. We reformulate the winning condition by not applying τ_μ in the winning clauses, which results in an equivalent game since they are all injective. The operation, hashing and pairing oracles are augmented to construct the view of $\mathcal{A}$ along the way.

G_1: This game proceeds as G_0, but the encodings τ_μ are implemented via lazy sampling. More precisely, instead of sampling τ_μ, G_1 initializes tables $T_{\tau_\mu} \leftarrow [\,]$. Oracles op_μ and H_μ are then implemented via lazy sampling from G_μ using table T_{τ_μ}. The same is done for oracle e, using tables T_{τ_ν}.

G_2: This game proceeds as G_1, but whenever it lazily samples a domain point in T_{τ_ν}, G_2 instead saves a fresh variable $\boldsymbol{R}_\upsilon$. (Note that this is only done for oracles pertaining to G_ν; oracles for G_T are as in G_1.) Only after $\mathcal{A}$ and $\mathcal{E}$ are run, G_2 samples random $\boldsymbol{r}$ of the appropriate length, evaluates the output of $\mathcal{A}$ at this point, and checks the winning condition as in G_1. Notice that in this game, tables T_{τ_μ} are populated exactly as tables U_{τ_μ} compiled by $\mathcal{E}$.

G_3: This game proceeds as G_2, but we omit the sampling of $\boldsymbol{r}$, and instead regard the winning condition as a set of (in)equalities between polynomials in $\boldsymbol{R}$.

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_μ be the events in G_2 that there are two distinct polynomials in $\mathrm{Dom}(T_{\tau_\mu})$ which result in the same value when evaluating $\boldsymbol{R}$ at random $\boldsymbol{r}$. Notice that G_1 and G_2 are identical until Bad_1 or Bad_2 or Bad_T, and by the fundamental lemma of game playing we therefore have that $|\Pr[\mathrm{G}_2] - \Pr[\mathrm{G}_1]| \leq \Pr[\mathsf{Bad}_1] + \Pr[\mathsf{Bad}_2] + \Pr[\mathsf{Bad}_T]$.

We bound the latter probabilities via Lemma 1. Consider the adversary $\mathcal{B}_1$ in the Schwartz–Zippel game defined in Figure 15. Here, $\mathcal{B}_1$ simulates G_2 to $\mathcal{A}$ and then returns all entries in $\mathrm{Dom}(T_{\tau_1})$. Notice that if Bad_1 occurs, then $\mathcal{B}_1$ wins the SZ-game, and that T_{τ_1} contains at most $3q_{\mathsf{op}_1} + q_{\mathsf{H}_1} + q_{\mathsf{e}} + 3$ polynomials of degree at most 1. By Lemma 1, $\Pr[\mathsf{Bad}] \leq (3q_{\mathsf{op}_1} + q_{\mathsf{H}_1} + q_{\mathsf{e}} + 3)^2/2p$. We similarly bound $\Pr[\mathsf{Bad}_2]$ and $\Pr[\mathsf{Bad}_T]$ using adversaries $\mathcal{B}_2$ and $\mathcal{B}_T$ in the Schwartz–Zippel game defined in Figure 15, noting that T_{τ_2} and T_{τ_T} contain at most $3q_{\mathsf{op}_2} + q_{\mathsf{H}_2} + q_{\mathsf{e}} + 2$ polynomials of degree at most 1, and at most $3q_{\mathsf{op}_T} + q_{\mathsf{H}_T} + q_{\mathsf{e}}$ polynomials of degree at most 2, respectively. Therefore,

$$|\Pr[\mathrm{G}_2] - \Pr[\mathrm{G}_1]| \leq \Pr[\mathsf{Bad}_1] + \Pr[\mathsf{Bad}_2] + \Pr[\mathsf{Bad}_T] \leq \frac{3(3q_{\mathsf{op}} + q_{\mathsf{H}} + q_{\mathsf{e}} + 3)^2}{2p}\,.$$

$\mathrm{G}_2 \rightsquigarrow \mathrm{G}_3$. Let Bad' be the event in G_3 that $\boldsymbol{y}_1\boldsymbol{y}_2 \neq \boldsymbol{y}_3$ or $\boldsymbol{y}_1 \neq w$ or $\boldsymbol{y}_2 \neq w$, but the corresponding equality holds when evaluating $\boldsymbol{R}$ at a random $\boldsymbol{r}$. Then G_2 and G_3 are identical until Bad', and by the fundamental lemma of game playing we therefore have $|\Pr[\mathrm{G}_3] - \Pr[\mathrm{G}_2]| \leq \Pr[\mathsf{Bad}']$.

We again bound the latter probability via Lemma 1. Consider the adversaries $\mathcal{B}'$ and $\mathcal{B}'_\nu$ in the Schwartz–Zippel game defined in Figure 15. Here, $\mathcal{B}'$ and $\mathcal{B}'_\nu$ simulate G_3 to $\mathcal{A}$ and then return $(\boldsymbol{y}_1\boldsymbol{y}_2 - \boldsymbol{y}_3, 0)$ and $(\boldsymbol{y}_\nu - w, 0)$, respectively. Notice that if Bad' occurs, then $\mathcal{B}'$ or $\mathcal{B}'_\nu$ win the SZ-game, and that the polynomials returned by $\mathcal{B}'$ and $\mathcal{B}'_\nu$ have total degree at most 2 and 1, respectively. By Lemma 1, $\Pr[\mathsf{Bad}'] \leq 2/p + 2 \cdot 1/p = 4/p$.

We conclude the proof by showing that the winning probability of $\mathcal{A}$ in G_3 is zero. Notice that if the output of $\mathcal{A}$ is such that $\boldsymbol{y}_1\boldsymbol{y}_2 \neq \boldsymbol{y}_3$, then $\mathcal{A}$ has trivially lost the game.

```
Game G_0:
$\tau_1 \twoheadleftarrow \mathrm{Inj}(\mathbb{Z}_p, \mathsf{G}_1)$; $\tau_2 \twoheadleftarrow \mathrm{Inj}(\mathbb{Z}_p, \mathsf{G}_2)$; $\tau_T \twoheadleftarrow \mathrm{Inj}(\mathbb{Z}_p, \mathsf{G}_T)$; $T_{H_1}, T_{H_2}, T_{H_T} \leftarrow [\,]$
$o \leftarrow 0$; $u_1 \leftarrow \tau_1(1)$; $u_2 \leftarrow \tau_2(1)$; $r_{\mathcal{A}} \twoheadleftarrow \mathcal{R}_{\mathcal{A}}$; $\boldsymbol{v} \leftarrow \mathcal{A}^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(u_1, u_2; r_{\mathcal{A}})$
$\mathsf{trace}(\mathcal{A}) \leftarrow (r_{\mathcal{A}}, u_1, u_2, \boldsymbol{h})$; $w \twoheadleftarrow \mathcal{E}^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(\mathsf{trace}(\mathcal{A}))$
$\boldsymbol{y}_1 \leftarrow \tau_1^{-1}(\boldsymbol{v}_1)$; $\boldsymbol{y}_2 \leftarrow \tau_2^{-1}(\boldsymbol{v}_2)$; $\boldsymbol{y}_3 \leftarrow \tau_1^{-1}(\boldsymbol{v}_3)$; return $(\boldsymbol{y}_1\boldsymbol{y}_2 = \boldsymbol{y}_3) \wedge (w \neq \boldsymbol{y}_1) \wedge (w \neq \boldsymbol{y}_2)$
```

```
Proc. $\mathsf{op}_\mu(h_1, h_2)$:
$x_1 \leftarrow \tau_\mu^{-1}(h_1)$
$x_2 \leftarrow \tau_\mu^{-1}(h_2)$
$o \leftarrow o+1$; $\boldsymbol{h}_o \leftarrow \tau_\mu(x_1+x_2)$
return $\boldsymbol{h}_o$
```

```
Proc. $\mathsf{H}_\mu(m)$:
if $m \notin \mathrm{Dom}(T_{H_\mu})$ then
   $r \twoheadleftarrow \mathbb{Z}_p$; $T_{H_\mu}[m] \leftarrow r$
$r \leftarrow T_{H_\mu}[m]$; $o \leftarrow o + 1$; $\boldsymbol{h}_o \leftarrow \tau_\mu(r)$
return $\boldsymbol{h}_o$
```

```
Proc. $\mathsf{e}(h_1, h_2)$:
$x_1 \leftarrow \tau_1^{-1}(h_1)$
$x_2 \leftarrow \tau_2^{-1}(h_2)$
$o \leftarrow o + 1$; $\boldsymbol{h}_o \leftarrow \tau_T(x_1x_2)$
return $\boldsymbol{h}_o$
```

```
Game G_1:
$T_{\tau_1}, T_{\tau_2}, T_{\tau_T}, T_{H_1}, T_{H_2}, T_{H_T} \leftarrow [\,]$; $o \leftarrow 0$
$u_1 \twoheadleftarrow \mathsf{G}_1$; $u_2 \twoheadleftarrow \mathsf{G}_2$; $T_{\tau_1}[1] \leftarrow u_1$; $T_{\tau_2}[1] \leftarrow u_2$
$r_{\mathcal{A}} \twoheadleftarrow \mathcal{R}_{\mathcal{A}}$; $\boldsymbol{v} \leftarrow \mathcal{A}^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(u_1, u_2; r_{\mathcal{A}})$
$\mathsf{trace}(\mathcal{A}) \leftarrow (r_{\mathcal{A}}, u_1, u_2, \boldsymbol{h})$
$w \twoheadleftarrow \mathcal{E}^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(\mathsf{trace}(\mathcal{A}))$
if $(\boldsymbol{v}_1 \notin \mathrm{Rng}(T_{\tau_1}))$ then $\boldsymbol{y}_1 \twoheadleftarrow \mathbb{Z}_p \setminus \mathrm{Dom}(T_{\tau_1})$; $T_{\tau_1}[\boldsymbol{y}_1] \leftarrow \boldsymbol{v}_1$
if $(\boldsymbol{v}_2 \notin \mathrm{Rng}(T_{\tau_2}))$ then $\boldsymbol{y}_2 \twoheadleftarrow \mathbb{Z}_p \setminus \mathrm{Dom}(T_{\tau_2})$; $T_{\tau_2}[\boldsymbol{y}_2] \leftarrow \boldsymbol{v}_2$
if $(\boldsymbol{v}_3 \notin \mathrm{Rng}(T_{\tau_1}))$ then $\boldsymbol{y}_3 \twoheadleftarrow \mathbb{Z}_p \setminus \mathrm{Dom}(T_{\tau_1})$; $T_{\tau_1}[\boldsymbol{y}_3] \leftarrow \boldsymbol{v}_3$
$\boldsymbol{y}_1 \leftarrow T_{\tau_1}^{-1}[\boldsymbol{v}_1]$; $\boldsymbol{y}_2 \leftarrow T_{\tau_2}^{-1}[\boldsymbol{v}_2]$; $\boldsymbol{y}_3 \leftarrow T_{\tau_1}^{-1}[\boldsymbol{v}_3]$
return $(\boldsymbol{y}_1\boldsymbol{y}_2 = \boldsymbol{y}_3) \wedge (w \neq \boldsymbol{y}_1) \wedge (w \neq \boldsymbol{y}_2)$
```

```
Proc. $\mathsf{op}_\mu(h_1, h_2)$:
for $i = 1$ to $2$ do
   if $(h_i \notin \mathrm{Rng}(T_{\tau_\mu}))$ then
      $x_i \leftarrow \mathbb{Z}_p \setminus \mathrm{Dom}(T_{\tau_\mu})$
      $T_{\tau_\mu}[x_i] \leftarrow h_i$
   $x_i \leftarrow T_{\tau_\mu}^{-1}[h_i]$
$x \leftarrow x_1 + x_2$
if $(x \notin \mathrm{Dom}(T_{\tau_\mu}))$ then
   $h \twoheadleftarrow \mathsf{G}_\mu \setminus \mathrm{Rng}(T_{\tau_\mu})$; $T_{\tau_\mu}[x] \leftarrow h$
$o \leftarrow o + 1$; $\boldsymbol{h}_o \leftarrow T_{\tau_\mu}[x]$; return $\boldsymbol{h}_o$
```

```
Proc. $\mathsf{H}_\mu(m)$:
if $(m \notin \mathrm{Dom}(T_{H_\mu}))$ then
   $r \twoheadleftarrow \mathbb{Z}_p$; $T_{H_\mu}[m] \leftarrow r$
$r \leftarrow T_{H_\mu}[m]$
if $(r \notin \mathrm{Dom}(T_{\tau_\mu}))$ then
   $h \twoheadleftarrow \mathsf{G}_\mu \setminus \mathrm{Rng}(T_{\tau_\mu})$; $T_{\tau_\mu}[r] \leftarrow h$
$o \leftarrow o + 1$; $\boldsymbol{h}_o \leftarrow T_{\tau_\mu}[r]$; return $\boldsymbol{h}_o$
```

```
Proc. $\mathsf{e}(h_1, h_2)$:
for $\nu = 1$ to $2$ do
   if $(h_\nu \notin \mathrm{Rng}(T_{\tau_\nu}))$ then $x_\nu \leftarrow \mathbb{Z}_p \setminus \mathrm{Dom}(T_{\tau_\nu})$; $T_{\tau_\nu}[x_\nu] \leftarrow h_\nu$
   $x_\nu \leftarrow T_{\tau_\nu}^{-1}[h_\nu]$
$x \leftarrow x_1x_2$
if $(x \notin \mathrm{Dom}(T_{\tau_T}))$ then $h \twoheadleftarrow \mathsf{G}_T \setminus \mathrm{Rng}(T_{\tau_T})$; $T_{\tau_T}[x] \leftarrow h$
$o \leftarrow o + 1$; $\boldsymbol{h}_o \leftarrow T_{\tau_T}[x]$; return $\boldsymbol{h}_o$
```

```
Game G_2:
$T_{\tau_1}, T_{\tau_2}, T_{\tau_T}, T_{H_1}, T_{H_2}, T_{H_T} \leftarrow [\,]$; $o, v \leftarrow 0$
$u_1 \twoheadleftarrow \mathsf{G}_1$; $u_2 \twoheadleftarrow \mathsf{G}_2$; $T_{\tau_1}[1] \leftarrow u_1$; $T_{\tau_2}[1] \leftarrow u_2$
$r_{\mathcal{A}} \twoheadleftarrow \mathcal{R}_{\mathcal{A}}$
$\boldsymbol{v} \leftarrow \mathcal{A}^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(u_1, u_2; r_{\mathcal{A}})$
$\mathsf{trace}(\mathcal{A}) \leftarrow (r_{\mathcal{A}}, u_1, u_2, \boldsymbol{h})$
$w \twoheadleftarrow \mathcal{E}^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(\mathsf{trace}(\mathcal{A}))$
if $(\boldsymbol{v}_1 \notin \mathrm{Rng}(T_{\tau_1}))$ then $v \leftarrow v+1$; $T_{\tau_1}[\boldsymbol{R}_v] \leftarrow \boldsymbol{v}_1$
if $(\boldsymbol{v}_2 \notin \mathrm{Rng}(T_{\tau_2}))$ then $v \leftarrow v+1$; $T_{\tau_2}[\boldsymbol{R}_v] \leftarrow \boldsymbol{v}_2$
if $(\boldsymbol{v}_3 \notin \mathrm{Rng}(T_{\tau_1}))$ then $v \leftarrow v+1$; $T_{\tau_1}[\boldsymbol{R}_v] \leftarrow \boldsymbol{v}_3$
$\boldsymbol{y}_1 \leftarrow T_{\tau_1}^{-1}[\boldsymbol{v}_1]$; $\boldsymbol{y}_2 \leftarrow T_{\tau_2}^{-1}[\boldsymbol{v}_2]$; $\boldsymbol{y}_3 \leftarrow T_{\tau_1}^{-1}[\boldsymbol{v}_3]$
$\boldsymbol{r} \twoheadleftarrow \mathbb{Z}_p^{2q_{\mathsf{op}}+q_{\mathsf{H}}+2q_{\mathsf{e}}+3}$
for $i = 1$ to $3$ do $\boldsymbol{y}_i \leftarrow \boldsymbol{y}_i(\boldsymbol{r})$
return $(\boldsymbol{y}_1\boldsymbol{y}_2 = \boldsymbol{y}_3) \wedge (w \neq \boldsymbol{y}_1) \wedge (w \neq \boldsymbol{y}_2)$
```

```
Game G_3:
$T_{\tau_1}, T_{\tau_2}, T_{\tau_T}, T_{H_1}, T_{H_2}, T_{H_T} \leftarrow [\,]$; $o, v \leftarrow 0$
$u_1 \twoheadleftarrow \mathsf{G}_1$; $u_2 \twoheadleftarrow \mathsf{G}_2$
$T_{\tau_1}[1] \leftarrow u_1$; $T_{\tau_2}[1] \leftarrow u_2$; $r_{\mathcal{A}} \twoheadleftarrow \mathcal{R}_{\mathcal{A}}$
$\boldsymbol{v} \leftarrow \mathcal{A}^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(u_1, u_2; r_{\mathcal{A}})$
$\mathsf{trace}(\mathcal{A}) \leftarrow (r_{\mathcal{A}}, u_1, u_2, \boldsymbol{h})$
$w \twoheadleftarrow \mathcal{E}^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(\mathsf{trace}(\mathcal{A}))$
if $(\boldsymbol{v}_1 \notin \mathrm{Rng}(T_{\tau_1}))$ then $v \leftarrow v+1$; $T_{\tau_1}[\boldsymbol{R}_v] \leftarrow \boldsymbol{v}_1$
if $(\boldsymbol{v}_2 \notin \mathrm{Rng}(T_{\tau_2}))$ then $v \leftarrow v+1$; $T_{\tau_2}[\boldsymbol{R}_v] \leftarrow \boldsymbol{v}_2$
if $(\boldsymbol{v}_3 \notin \mathrm{Rng}(T_{\tau_1}))$ then $v \leftarrow v+1$; $T_{\tau_1}[\boldsymbol{R}_v] \leftarrow \boldsymbol{v}_3$
$\boldsymbol{y}_1 \leftarrow T_{\tau_1}^{-1}[\boldsymbol{v}_1]$; $\boldsymbol{y}_2 \leftarrow T_{\tau_2}^{-1}[\boldsymbol{v}_2]$; $\boldsymbol{y}_3 \leftarrow T_{\tau_1}^{-1}[\boldsymbol{v}_3]$
return $(\boldsymbol{y}_1\boldsymbol{y}_2 = \boldsymbol{y}_3) \wedge (w \neq \boldsymbol{y}_1) \wedge (w \neq \boldsymbol{y}_2)$
```

```
Proc. $\mathsf{op}_\nu(h_1, h_2)$:
for $i = 1$ to $2$ do
   if $(h_i \notin \mathrm{Rng}(T_{\tau_\nu}))$ then
      $v \leftarrow v + 1$; $T_{\tau_\nu}[\boldsymbol{R}_v] \leftarrow h_i$
   $x_i \leftarrow T_{\tau_\nu}^{-1}[h_i]$
$x \leftarrow x_1 + x_2$
if $(x \notin \mathrm{Dom}(T_{\tau_\nu}))$ then
   $h \twoheadleftarrow \mathsf{G}_\nu \setminus \mathrm{Rng}(T_{\tau_\nu})$
   $T_{\tau_\nu}[x] \leftarrow h$
$o \leftarrow o + 1$; $\boldsymbol{h}_o \leftarrow T_{\tau_\nu}[x]$
return $\boldsymbol{h}_o$
```

```
Proc. $\mathsf{H}_\nu(m)$:
if $(m \notin \mathrm{Dom}(T_{H_\nu}))$ then
   $v \leftarrow v + 1$; $T_{H_\nu}[m] \leftarrow \boldsymbol{R}_v$
$r \leftarrow T_{H_\nu}[m]$
if $(r \notin \mathrm{Dom}(T_{\tau_\nu}))$ then
   $h \twoheadleftarrow \mathsf{G}_\nu \setminus \mathrm{Rng}(T_{\tau_\nu})$
$T_{\tau_\nu}[r] \leftarrow h$
$o \leftarrow o + 1$; $\boldsymbol{h}_o \leftarrow T_{\tau_\nu}[r]$
return $\boldsymbol{h}_o$
```

```
Proc. $\mathsf{e}(h_1, h_2)$:
for $\nu = 1$ to $2$ do
   if $(h_\nu \notin \mathrm{Rng}(T_{\tau_\nu}))$ then
      $v \leftarrow v+1$; $T_{\tau_\nu}[\boldsymbol{R}_v] \leftarrow h_\nu$
   $x_\nu \leftarrow T_{\tau_\nu}^{-1}[h_\nu]$
$x \leftarrow x_1x_2$
if $(x \notin \mathrm{Dom}(T_{\tau_T}))$ then
   $h \twoheadleftarrow \mathsf{G}_T \setminus \mathrm{Rng}(T_{\tau_T})$
   $T_{\tau_T}[x] \leftarrow h$
$o \leftarrow o + 1$; $\boldsymbol{h}_o \leftarrow T_{\tau_T}[x]$
return $\boldsymbol{h}_o$
```

Figure 14: Code of the intermediate games in the proof of Inequality (1). For games G_2 and G_3, oracles op_T and H_T are as in G_1. In all figures, μ and ν are indices ranging over $\{1, 2, T\}$ and $\{1, 2\}$, respectively.

```
Adversaries $\mathcal{B}_\mu/\mathcal{B}'/\mathcal{B}'_\nu$:
$T_{\tau_1}, T_{\tau_2}, T_{\tau_T}, T_{H_1}, T_{H_2}, T_{H_T} \leftarrow []$; $o, v \leftarrow 0$
$u_1 \twoheadleftarrow \mathsf{G}_1$; $u_2 \twoheadleftarrow \mathsf{G}_2$; $T_{\tau_1}[1] \leftarrow u_1$; $T_{\tau_2}[1] \leftarrow u_2$
$r_\mathcal{A} \twoheadleftarrow \mathcal{R}_\mathcal{A}$; $\boldsymbol{v} \leftarrow \mathcal{A}^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(u_1, u_2; r_\mathcal{A})$
$\mathsf{trace}(\mathcal{A}) \leftarrow (r_\mathcal{A}, u_1, u_2, \boldsymbol{h})$; $w \twoheadleftarrow \mathcal{E}^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(\mathsf{trace}(\mathcal{A}))$
if $(\boldsymbol{v}_1 \notin \mathrm{Rng}(T_{\tau_1}))$ then $v \leftarrow v+1$; $T_{\tau_1}[\boldsymbol{R}_v] \leftarrow \boldsymbol{v}_1$
if $(\boldsymbol{v}_2 \notin \mathrm{Rng}(T_{\tau_2}))$ then $v \leftarrow v+1$; $T_{\tau_2}[\boldsymbol{R}_v] \leftarrow \boldsymbol{v}_2$
if $(\boldsymbol{v}_3 \notin \mathrm{Rng}(T_{\tau_1}))$ then $v \leftarrow v+1$; $T_{\tau_1}[\boldsymbol{R}_v] \leftarrow \boldsymbol{v}_3$
$\boldsymbol{y}_1 \leftarrow T_{\tau_1}^{-1}[\boldsymbol{v}_1]$; $\boldsymbol{y}_2 \leftarrow T_{\tau_2}^{-1}[\boldsymbol{v}_2]$; $\boldsymbol{y}_3 \leftarrow T_{\tau_1}^{-1}[\boldsymbol{v}_3]$
$\mathcal{B}_\mu$: return $\mathrm{Dom}(T_{\tau_\mu})$          $\mathcal{B}'$: return $(\boldsymbol{y}_1\boldsymbol{y}_2 - \boldsymbol{y}_3, 0)$          $\mathcal{B}'_\nu$: return $(\boldsymbol{y}_\nu - w, 0)$
```

Figure 15: Definition of the adversaries $\mathcal{B}_\mu$, $\mathcal{B}'$ and $\mathcal{B}'_\nu$ from the proof of Theorem 1. In all cases, oracles op_1, op_2, op_T, H_1, H_2, H_T and e are defined as in Figure 14 (bottom), and μ and ν are indices ranging over $\{1, 2, T\}$ and $\{1, 2\}$, respectively.

If on the other hand $\boldsymbol{y}_1\boldsymbol{y}_2 = \boldsymbol{y}_3$, we obtain

$$\Big(w_1 + \sum_l \boldsymbol{b}_{1l}\boldsymbol{R}_l\Big)\Big(w_2 + \sum_l \boldsymbol{b}_{2l}\boldsymbol{R}_l\Big) - \Big(w_3 + \sum_l \boldsymbol{b}_{3l}\boldsymbol{R}_l\Big) = 0\,, \tag{2}$$

as a polynomial in $\boldsymbol{R}$. We want to show that this implies either $\boldsymbol{b}_{1l} = 0$ for all l or $\boldsymbol{b}_{2l} = 0$ for all l, since the representation returned by $\mathcal{E}$ will be correct if that is the case. Indeed, expanding Equation (2) gives

$$w_1w_2 - w_3 + \sum_l (w_1\boldsymbol{b}_{2l} + w_2\boldsymbol{b}_{1l} - \boldsymbol{b}_{3l})\boldsymbol{R}_l \\ + \sum_{l<l'} \big(\boldsymbol{b}_{1l}\boldsymbol{b}_{2l'} + \boldsymbol{b}_{1l'}\boldsymbol{b}_{2l}\big)\boldsymbol{R}_l\boldsymbol{R}_{l'} + \sum_l \boldsymbol{b}_{1l}\boldsymbol{b}_{2l}\boldsymbol{R}_l^2 = 0\,,$$

that is, in particular, (1) $\boldsymbol{b}_{1l}\boldsymbol{b}_{2l} = 0$ for all l, and (2) $\boldsymbol{b}_{1l}\boldsymbol{b}_{2l'} + \boldsymbol{b}_{1l'}\boldsymbol{b}_{2l} = 0$ for all $l < l'$. Now assume that there exists $\bar{l}$ such that $\boldsymbol{b}_{1\bar{l}} \neq 0$. Then from (1) we obtain $\boldsymbol{b}_{2\bar{l}} = 0$, and from (2) that $\boldsymbol{b}_{2\tilde{l}} = 0$ for all $\tilde{l} \neq \bar{l}$ by either setting $l = \bar{l}$ and l' any other index larger than $\bar{l}$, or $l' = \tilde{l}$ and l any other index smaller than $\tilde{l}$. This in turn means $\boldsymbol{b}_2 = 0$, and a similar argument shows that if $\boldsymbol{b}_2 \neq 0$, then it must be $\boldsymbol{b}_1 = 0$.

This proves that if $\mathcal{A}$ returns a valid output, then $\mathcal{E}$ returns an accurate representation of either $\boldsymbol{v}_1$ or $\boldsymbol{v}_2$ in terms of the generator u, which means that $\Pr[\mathrm{G}_3] = 0$. Collecting all the terms above, we obtain

$$\mathrm{Adv}^{\text{dh-ke}}_{p,\mathbf{G},\mathcal{A},\mathcal{E}} \leq \frac{3(3q_{\mathsf{op}} + q_{\mathsf{H}} + q_{\mathsf{e}} + 3)^2}{2p} + \frac{4}{p} \leq \mathcal{O}\bigg(\frac{(q_{\mathsf{op}} + q_{\mathsf{H}} + q_{\mathsf{e}})^2}{p}\bigg)\,. \qquad \square$$

5.2 Soundness of DH-KE in ABM3-H

Theorem 2 (DH-KE holds in ABM3-H)**.** *Let* B *be a type-3 bilinear group scheme. If* $(1,1)$-DL *holds for* B*, then* DH-KE *holds for* B *in the* ABM3-H*. More precisely, for every PPT algebraic adversary $\mathcal{A}$ in the* DH-KE *game, there exist an extractor $\mathcal{E}$ and an adversary $\mathcal{B}$ against* $(1,1)$-DL*, both with approximately the same running time as $\mathcal{A}$, such that*

$$\mathrm{Adv}^{\text{dh-ke}}_{\mathrm{B},\mathcal{A},\mathcal{E}}(\lambda) \leq \bigg(1 - \frac{2}{2^{\lambda-1} - 1}\bigg)^{-1} \cdot \mathrm{Adv}^{(1,1)\text{-dl}}_{\mathrm{B},\mathcal{B}}(\lambda)\,. \tag{3}$$

```
Extractor $\mathcal{E}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\mathsf{trace}(\mathcal{A}))$:
parse $\mathsf{trace}(\mathcal{A}) = (r_{\mathcal{A}}, \gamma, [\boldsymbol{h}_\mu]_\mu)$; $o_1, o_2, o_T \leftarrow 0$
$(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}) \leftarrow \mathcal{A}^{\overline{\mathsf{H}}_1,\overline{\mathsf{H}}_2,\overline{\mathsf{H}}_T}(\gamma; r_{\mathcal{A}})$
if $(\boldsymbol{u}_1 = \cdots = \boldsymbol{u}_{o_1} = 0)$ then return $\boldsymbol{u}_0$ else return $\boldsymbol{v}_0$
```

```
Oracle $\overline{\mathsf{H}}_\mu(m)$:
$o_\mu \leftarrow o_\mu + 1$
return $[\boldsymbol{h}_{\mu,o_\mu}]_\mu$
```

```
Adversary $\mathcal{B}(\gamma, [t]_1, [t]_2)$:
$o_1, o_2 \leftarrow 0$; $U_1, U_2, U_T \leftarrow [\,]$; $\mathsf{S} \leftarrow \emptyset$; $(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}) \twoheadleftarrow \mathcal{A}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\gamma)$
$Q'(T) \leftarrow \big(\boldsymbol{u}_0 + \sum_{i=1}^{o_1} \boldsymbol{u}_i H_{1,i}\big)\big(\boldsymbol{v}_0 + \sum_{j=1}^{o_2} \boldsymbol{v}_j H_{2,j}\big) - \big(\boldsymbol{w}_0 + \sum_{i=1}^{o_1} \boldsymbol{w}_i H_{1,i}\big)$
if $(Q'(T) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{Berlekamp}(Q', p)$
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
    $o_\nu \leftarrow o_\nu + 1$; $\alpha_{\nu,o_\nu} \twoheadleftarrow \mathbb{Z}_p$; $\beta_{\nu,o_\nu} \twoheadleftarrow \mathbb{Z}_p^*$
    $H_{\nu,o_\nu}(T) \leftarrow \alpha_{\nu,o_\nu} + \beta_{\nu,o_\nu} T$; $U_\nu[m] \leftarrow [H_{\nu,o_\nu}(t)]_\nu$
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 $U_T[m]$
```

Figure 16: *Top:* Extractor $\mathcal{E}$ for the algebraic adversary $\mathcal{A}$ in the DH-KE game. *Bottom:* Adversary $\mathcal{B}$ against $(1,1)$-DL. In all figures, μ and ν range over the sets $\{1, 2, T\}$ and $\{1, 2\}$, respectively.

Proof. Fix an adversary $\mathcal{A}$ in the DH-KE game as in the statement of the theorem, and define an extractor $\mathcal{E}$ as in Figure 16 (top). This extractor essentially re-runs $\mathcal{A}$ on its view to obtain $\mathcal{A}$'s output $(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w})$. Recall that this means that $\mathcal{A}$ encodes group elements $[a]_1 = [\boldsymbol{u}_0]_1 \cdot \prod_{l \geq 1} [\boldsymbol{u}_l \boldsymbol{h}_{1l}]_1$, where $[\boldsymbol{h}_1]_1$ is the vector of hash replies in G_1, and similarly for $[b]_2$ and $[c]_1$ using vectors $\boldsymbol{v}$ and $\boldsymbol{w}$. If all coordinates of $\boldsymbol{u}$ except possibly $\boldsymbol{u}_0$ are zero (i.e., the first element encoded by $\mathcal{A}$ is $[\boldsymbol{u}_0]_1$), then $\mathcal{E}$ returns $\boldsymbol{u}_0$, and otherwise $\boldsymbol{v}_0$. Clearly, $\mathcal{E}$ will be correct if all entries but possibly the first one in either $\boldsymbol{u}$ or $\boldsymbol{v}$ vanish.

We now show that if $\mathcal{A}$ returns a valid output and $(1,1)$-DL holds for B, this will likely be the case. To that end, consider the adversary $\mathcal{B}$ playing the $(1,1)$-DL game for B defined in Figure 16 (bottom). In essence, $\mathcal{B}$ runs $\mathcal{A}$ and simulates the DH-KE game. When answering hash queries, $\mathcal{B}$ embeds the $(1,1)$-DL instance it is tasked with solving into the replies. By construction, if $\mathcal{A}$ returns an output that satisfies the relation polynomial of DH-KE, then t is a root of the polynomial $Q'(T)$ defined by $\mathcal{B}$. This means that $\mathcal{B}$ will be able to find t by inspecting the roots of Q' whenever $Q'(T) \neq 0$. We prove that the latter happens with overwhelming probability if $\boldsymbol{u}_{i^*} \neq 0$ and $\boldsymbol{v}_{j^*} \neq 0$ for some $i^*, j^* > 0$, which means that this cannot happen if $(1,1)$-DL holds for B.

We now show in detail how to use adversary $\mathcal{B}$ to prove Inequality (3) for $\mathcal{A}$ and $\mathcal{E}$. To that end, consider the following sequence of games (the formal description of which can be found in Figure 17):

G_0: This is the original $(1,1)$-DL game for B run with adversary $\mathcal{B}$.

G_1: This game proceeds as G_0, but performs variable substitutions $\alpha'_{\nu,l} = \alpha_{\nu,l} + \beta_{\nu,l} t$ and $\beta'_{\nu,l} = \beta_{\nu,l}$ in polynomials $H_{\nu,l}$. More precisely, upon a query m to H_ν, game G_2 samples random $\alpha'_{\nu,l}$ and invertible $\beta'_{\nu,l}$, and sets $H_{\nu,l}(T) \leftarrow \alpha'_{\nu,l} + \beta'_{\nu,l}(T - t)$. Hash replies are still computed as $[H_{\nu,l}(t)]_\nu = [\alpha'_{\nu,l}]_\nu$.

G_2: This game proceeds as G_1, but polynomials $H_{\nu,l}$ are now defined as $H_{\nu,l}(T, \boldsymbol{B}'_\nu) \leftarrow \alpha'_{\nu,l} + \boldsymbol{B}'_{\nu,l}(T - t)$, where $\boldsymbol{B}'_{\nu,l}$ is a fresh variable for every oracle call. Accordingly, the polynomial Q'' constructed after running $\mathcal{A}$ is now in variables T, $\boldsymbol{B}'_1$ and $\boldsymbol{B}'_2$. After defining Q'', game G_2 samples invertible $\boldsymbol{\beta}'_1$ and $\boldsymbol{\beta}'_2$, sets $Q'(T) \leftarrow Q''(T, \boldsymbol{\beta}'_1, \boldsymbol{\beta}'_2)$, and checks if $Q'(T) = 0$. From here on, game G_2 proceeds as G_1.

```
Game $G_0(\lambda)$:
$\gamma \twoheadleftarrow B(1^\lambda)$; $t \twoheadleftarrow \mathbb{Z}_p$; $o_1, o_2 \leftarrow 0$; $U_1, U_2, U_T \leftarrow [\,]$
$\mathsf{S} \leftarrow \emptyset$; $(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}) \twoheadleftarrow \mathcal{A}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\gamma)$
$Q'(T) \leftarrow \big(\boldsymbol{u}_0+\sum_{i=1}^{o_1} \boldsymbol{u}_i H_{1,i}\big)\big(\boldsymbol{v}_0+\sum_{j=1}^{o_2} \boldsymbol{v}_j H_{2,j}\big) - \big(\boldsymbol{w}_0+\sum_{i=1}^{o_1} \boldsymbol{w}_i H_{1,i}\big)$
if $(Q'(T) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{Berlekamp}(Q', p)$
$t' \leftarrow 0$
for $z \in \mathsf{S}$ do if $([z]_1 = [t]_1)$ then return $t' \leftarrow z$; break
return $(t = t')$

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 $U_T[m]$

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

```
Game $G_1(\lambda)$:
$\gamma \twoheadleftarrow B(1^\lambda)$; $t \twoheadleftarrow \mathbb{Z}_p$; $o_1, o_2 \leftarrow 0$; $U_1, U_2, U_T \leftarrow [\,]$
$\mathsf{S} \leftarrow \emptyset$; $(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}) \twoheadleftarrow \mathcal{A}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\gamma)$
$Q'(T) \leftarrow \big(\boldsymbol{u}_0+\sum_{i=1}^{o_1} \boldsymbol{u}_i H_{1,i}\big)\big(\boldsymbol{v}_0+\sum_{j=1}^{o_2} \boldsymbol{v}_j H_{2,j}\big) - \big(\boldsymbol{w}_0+\sum_{i=1}^{o_1} \boldsymbol{w}_i H_{1,i}\big)$
if $(Q'(T) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{Berlekamp}(Q', p)$
$t' \leftarrow 0$
for $z \in \mathsf{S}$ do if $([z]_1 = [t]_1)$ then return $t' \leftarrow z$; break
return $(t = t')$

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 $U_T[m]$

Oracle $\mathsf{H}_\nu(m)$:
if $(m \notin \mathrm{Dom}(U_\nu))$ then
  $o_\nu \leftarrow o_\nu + 1$; $\alpha'_{\nu,o_\nu} \twoheadleftarrow \mathbb{Z}_p$; $\beta'_{\nu,o_\nu} \twoheadleftarrow \mathbb{Z}_p^*$; $H_{\nu,o_\nu}(T) \leftarrow \alpha'_{\nu,o_\nu} + \beta'_{\nu,o_\nu}(T - t)$; $U_\nu[m] \leftarrow [H_{\nu,o_\nu}(t)]_\nu$
return $U_\nu[m]$
```

```
Game $G_2(\lambda)$:
$\gamma \twoheadleftarrow B(1^\lambda)$; $t \twoheadleftarrow \mathbb{Z}_p$; $o_1, o_2 \leftarrow 0$; $U_1, U_2, U_T \leftarrow [\,]$
$\mathsf{S} \leftarrow \emptyset$; $(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}) \twoheadleftarrow \mathcal{A}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\gamma)$
$Q''(T, \boldsymbol{B}'_1, \boldsymbol{B}'_2) \leftarrow$
  $\big(\boldsymbol{u}_0 + \sum_{i=1}^{o_1} \boldsymbol{u}_i H_{1,i}\big)\big(\boldsymbol{v}_0 + \sum_{j=1}^{o_2} \boldsymbol{v}_j H_{2,j}\big) - \big(\boldsymbol{w}_0 + \sum_{i=1}^{o_1} \boldsymbol{w}_i H_{1,i}\big)$
$\boldsymbol{\beta}'_1 \twoheadleftarrow \mathbb{Z}_p^{*o_1}$; $\boldsymbol{\beta}'_2 \twoheadleftarrow \mathbb{Z}_p^{*o_2}$; $Q'(T) \leftarrow Q''(T, \boldsymbol{\beta}'_1, \boldsymbol{\beta}'_2)$
if $(Q'(T) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{Berlekamp}(Q', p)$
$t' \leftarrow 0$
for $z \in \mathsf{S}$ do if $([z]_1 = [t]_1)$ then return $t' \leftarrow z$; break
return $(t = t')$

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 $U_T[m]$

Oracle $\mathsf{H}_\nu(m)$:
if $(m \notin \mathrm{Dom}(U_\nu))$ then
  $o_\nu \leftarrow o_\nu + 1$; $\alpha'_{\nu,o_\nu} \twoheadleftarrow \mathbb{Z}_p$; $H_{\nu,o_\nu}(T, \boldsymbol{B}'_\nu) \leftarrow \alpha'_{\nu,o_\nu} + \boldsymbol{B}'_{\nu,o_\nu}(T - t)$; $U_\nu[m] \leftarrow [H_{\nu,o_\nu}(t, \boldsymbol{B}'_\nu)]_\nu$
return $U_\nu[m]$
```

Figure 17: Code of the intermediate games in the proof of Theorem 2. In all figures, ν is an index ranging over $\{1, 2\}$.

We now argue that subsequent games have identical success probabilities.

$G_0 \rightsquigarrow G_1$. Observe that for every fixed $\lambda \in \mathbb{N}$, γ returned by $B(1^\lambda)$, $t \in \mathbb{Z}_p$, and randomness $r_\mathcal{A}$ returned by $\mathcal{R}_\mathcal{A}(\lambda)$, the random variates $\alpha'_{\nu,l}$ and $\beta'_{\nu,l}$ in G_1 are related to the random variates $\alpha_{\nu,l}$ and $\beta_{\nu,l}$ in G_0 via the transformation $\mathrm{diag}(\begin{smallmatrix} 1 & t \\ 0 & 1 \end{smallmatrix})$, which is invertible. Consequently, $\Pr[G_0] = \Pr[G_1]$, since there is a one-to-one correspondence between the random variables in the two games.

$G_1 \rightsquigarrow G_2$. Notice that $\mathcal{A}$ is oblivious to the changes to polynomials $H_{\nu,l}$, so the simulation of $\mathcal{A}$ is identical in both games. Indeed, in both games the hash replies are computed in the same way. After running $\mathcal{A}$, G_2 derives the same polynomial Q' computed in G_1 by substituting random $\boldsymbol{\beta}'_1$ and $\boldsymbol{\beta}'_2$ into Q'', so the winning condition is again the same in both games. Therefore, $\Pr[G_1] = \Pr[G_2]$.

We conclude the proof by studying the winning probability in G_2. First, notice that in this game adversary $\mathcal{A}$ plays the DH-KE game, since the hash replies are random group elements. Now for any $\lambda \in \mathbb{N}$, γ returned by $\mathrm{B}(1^\lambda)$, $t \in \mathbb{Z}_p$, randomness $r_\mathcal{A}$ returned by $\mathcal{R}_\mathcal{A}(\lambda)$, and vectors $\boldsymbol{\alpha}'_\nu$ and $\boldsymbol{\alpha}$ in $\mathbb{Z}_p$, denote by $\mathrm{G}' \coloneqq \mathrm{G}'(\lambda, \gamma, t, r_\mathcal{A}, \boldsymbol{\alpha}'_\nu, \boldsymbol{\alpha})$ the game $\mathrm{G}_2(\lambda)$ with these random choices fixed. Then we have

$$\Pr[\mathrm{G}_2(\lambda)] = \sum_{(\gamma,t,r_\mathcal{A},\boldsymbol{\alpha}'_\nu,\boldsymbol{\alpha})} \Pr[\mathrm{G}'] \Pr[\gamma, t, r_\mathcal{A}, \boldsymbol{\alpha}'_\nu, \boldsymbol{\alpha}]\,,$$

where $\Pr[\gamma, t, r_\mathcal{A}, \boldsymbol{\alpha}'_\nu, \boldsymbol{\alpha}]$ denotes the probability that such a tuple is drawn in $\mathrm{G}_2(\lambda)$, and the sum extends over all $(\gamma, t, r_\mathcal{A}, \boldsymbol{\alpha}'_\nu, \boldsymbol{\alpha})$ such that $\Pr[\gamma, t, r_\mathcal{A}, \boldsymbol{\alpha}'_\nu, \boldsymbol{\alpha}] \neq 0$.

Now consider the set X of all $(\gamma, t, r_\mathcal{A}, \boldsymbol{\alpha}'_\nu, \boldsymbol{\alpha})$ in the sum above such that $\mathcal{A}$ returns $(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w})$ for which the relation polynomial in DH-KE is satisfied and extractor $\mathcal{E}$ fails to compute a correct representation of the outputs. Notice that

$$\sum_{(\gamma,t,r_\mathcal{A},\boldsymbol{\alpha}'_\nu,\boldsymbol{\alpha})\in\mathsf{X}} \Pr[\gamma, t, r_\mathcal{A}, \boldsymbol{\alpha}'_\nu, \boldsymbol{\alpha}] = \mathrm{Adv}^{\text{dh-ke}}_{\mathrm{B},\mathcal{A},\mathcal{E}}(\lambda)\,.$$

We claim that for any $(\gamma, t, r_\mathcal{A}, \boldsymbol{\alpha}'_\nu, \boldsymbol{\alpha}) \in \mathsf{X}$, $\Pr[\mathrm{G}'] \geq 1 - 2/(2^{\lambda-1} - 1)$. Indeed, fix any $(\gamma, t, r_\mathcal{A}, \boldsymbol{\alpha}'_\nu, \boldsymbol{\alpha}) \in \mathsf{X}$. Since $\mathcal{E}$ fails to return a correct representation of the output of $\mathcal{A}$, neither $\boldsymbol{u}_1 = \cdots = \boldsymbol{u}_{o_1} = 0$ nor $\boldsymbol{v}_1 = \cdots = \boldsymbol{v}_{o_2} = 0$, where o_1 and o_2 are the number of queries made by $\mathcal{A}$ to H_1 and H_2, respectively. This means that there exist $1 \leq i^* \leq o_1$ and $1 \leq j^* \leq o_2$ such that $\boldsymbol{u}_{i^*} \neq 0$ and $\boldsymbol{v}_{j^*} \neq 0$. Then observe that the polynomial $Q''(T, \boldsymbol{B}'_1, \boldsymbol{B}'_2)$ constructed in G_2 after running $\mathcal{A}$ is not identically zero, because the coefficient of $\boldsymbol{B}'_{1,i^*}\boldsymbol{B}'_{2,j^*}$ is $(T-t)^2\boldsymbol{u}_{i^*}\boldsymbol{u}_{j^*} \neq 0$. Moreover, the leading coefficient in T of $Q''(T, \boldsymbol{B}'_1, \boldsymbol{B}'_2)$ is a polynomial in $\boldsymbol{B}'_1$ and $\boldsymbol{B}'_2$ of total degree at most 2, which for random invertible $\boldsymbol{\beta}'_1$ and $\boldsymbol{\beta}'_2$ will be zero with probability at most $2/(p-1) \leq 2/(2^{\lambda-1}-1)$ by Lemma 1. Thus, with probability at least $1 - 2/(2^{\lambda-1} - 1)$, $Q'(T) \neq 0$ in G'. We conclude by observing that whenever this happens, game G' will return 1, because t is a root of $Q'(T)$ by construction, and will therefore be found by inspecting its roots. This means

$$\begin{aligned}\mathrm{Adv}^{(1,1)\text{-dl}}_{\mathrm{B},\mathcal{B}}(\lambda) &= \Pr[\mathrm{G}_0(\lambda)] = \Pr[\mathrm{G}_2(\lambda)] = \sum_{(\gamma,t,r_\mathcal{A},\boldsymbol{\alpha}'_\nu,\boldsymbol{\alpha})} \Pr[\mathrm{G}'] \Pr[\gamma, t, r_\mathcal{A}, \boldsymbol{\alpha}'_\nu, \boldsymbol{\alpha}] \\ &\geq \sum_{(\gamma,t,r_\mathcal{A},\boldsymbol{\alpha}'_\nu,\boldsymbol{\alpha})\in\mathsf{X}} \Pr[\mathrm{G}'] \Pr[\gamma, t, r_\mathcal{A}, \boldsymbol{\alpha}'_\nu, \boldsymbol{\alpha}] \geq \left(1 - \frac{2}{2^{\lambda-1}-1}\right) \cdot \mathrm{Adv}^{\text{uk}}_{\mathrm{B},\mathcal{A},\mathcal{E}}(\lambda)\,,\end{aligned}$$

which concludes the proof. □

6 Soundness of UK in GBM3-H

In this section we justify the soundness of the UK assumption in the GBM3-H. Our result is for a class of adversaries $\mathcal{A}$ where $\mathcal{A}_0$ returns a relation polynomial Q of degree at most two in the output variables and no output variable for the target group, with at most one degree-two 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 3 (UK holds in GBM3-H)**.** *Let $p \in \mathbb{N}$ be prime, and fix $\mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_T \subseteq \{0,1\}^*$ with $|\mathsf{G}_1| = |\mathsf{G}_2| = |\mathsf{G}_T| = p$. Consider the class of algorithms $\mathfrak{A}$ and the source $\mathcal{S}$ defined as follows:*

1. *For every $\mathcal{A}_0 \in \mathfrak{A}$, the relation polynomial Q returned by $\mathcal{A}_0$ is of the form*

$$Q(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_T, \boldsymbol{Y}_1, \boldsymbol{Y}_2, \boldsymbol{C}) = Q_{i_1 i_2}(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_T, \boldsymbol{C}) \boldsymbol{Y}_{1,i_1} \boldsymbol{Y}_{2,i_2} \\ + \sum_{\nu=1}^{2} \sum_{i=1}^{|\boldsymbol{Y}_\nu|} Q_{\nu,i}(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_T, \boldsymbol{C}) \boldsymbol{Y}_{\nu,i} + Q_0(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_T, \boldsymbol{C}) \,,$$

   where $1 \leq i_1 \leq |\boldsymbol{Y}_1|$ and $1 \leq i_2 \leq |\boldsymbol{Y}_2|$;
2. *For every $\mathcal{A}_0 \in \mathfrak{A}$, every $(Q, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_T)$ returned by $\mathcal{A}_0$, and every $\boldsymbol{c} \in \mathbb{Z}_p^{|\boldsymbol{C}|}$, the polynomials $\overline{Q_{\nu,i}}$ (for $1 \leq \nu \leq 2$ and $1 \leq i \leq |\boldsymbol{Y}_\nu|$) are linearly independent;*
3. *For every $\mathcal{A}_0 \in \mathfrak{A}$, every $(Q, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_T)$ returned by $\mathcal{A}_0$ and every $\boldsymbol{c} \in \mathbb{Z}_p^{|\boldsymbol{C}|}$, if $\overline{Q_{i_1 i_2}} \neq 0$ then $\overline{Q_0}$ does not lie in the linear span of*

$$\{\overline{Q_{\nu,i}} \boldsymbol{P}_{\nu',j} \mid 1 \leq \nu, \nu' \leq 2, 1 \leq i \leq |\boldsymbol{Y}_\nu|, 1 \leq j \leq |\boldsymbol{X}_{\nu'}|\} \,;$$

4. *For every $\mathcal{A}_0 \in \mathfrak{A}$ and every $(Q, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_T)$ returned by $\mathcal{A}_0$, $\mathcal{S}$ samples $\boldsymbol{s} \in \mathbb{Z}_p^k$ at random and returns $(\boldsymbol{P}_1(\boldsymbol{s}), \boldsymbol{P}_2(\boldsymbol{s}), \boldsymbol{P}_T(\boldsymbol{s}))$.*

Then the UK *assumption holds in the* GBM3-H *with parameters $(p, \mathbf{G})$ with respect to the class of first-stage adversaries $\mathfrak{A}$ and source $\mathcal{S}$ above. More precisely, for every low-degree adversary $\mathcal{A}$ with $\mathcal{A}_0 \in \mathfrak{A}$, there exists an extractor $\mathcal{E}$ such that*

$$\mathrm{Adv}^{\mathrm{uk}}_{p,\mathsf{G},\mathcal{S},\mathcal{A},\mathcal{E}} \leq \mathcal{O}\left(\frac{(m + n + q_{\mathsf{op}} + q_{\mathsf{H}} + q_{\mathsf{e}} + d_Q)^2 \cdot d_P}{p} \right) . \tag{4}$$

Here, d_Q is an upper bound on the total degree of Q, d_P and k are upper bounds on the total degree and the number of variables of every polynomial P in $\boldsymbol{P}_\mu$, m and n are upper bounds on $|\boldsymbol{X}_\mu|$ and $|\boldsymbol{Y}_\nu|$, q_{op}, q_{H} and q_{e} are upper bounds on the number of queries made by $\mathcal{A}$ to the respective oracles, and we let $\boldsymbol{P}_{1,0}(\boldsymbol{S}) := \boldsymbol{P}_{2,0}(\boldsymbol{S}) := 1$ in $\overline{Q_{i_1 i_2}}(\boldsymbol{S}) := Q_{i_1 i_2}(\boldsymbol{P}_1(\boldsymbol{S}), \boldsymbol{P}_2(\boldsymbol{S}), \boldsymbol{P}_T(\boldsymbol{S}), \boldsymbol{c})$ and $\overline{Q_{\nu,i}}(\boldsymbol{S}) := Q_{\nu,i}(\boldsymbol{P}_1(\boldsymbol{S}), \boldsymbol{P}_2(\boldsymbol{S}), \boldsymbol{P}_T(\boldsymbol{S}), \boldsymbol{c})$.

Proof overview. Fix an adversary $\mathcal{A}$ in the UK game as above, and define the extractor $\mathcal{E}$ as in Figure 18. This extractor essentially re-runs $\mathcal{A}$ on its view and observes its oracle queries, keeping track of the discrete logarithms of the elements queried by $\mathcal{A}$ via appropriate tables. Whenever $\mathcal{E}$ is unable to "explain" an element in G_ν, $\nu \in \{1,2\}$, it instead stores a fresh variable $\boldsymbol{R}_v$ in the corresponding table. Oracles for G_T are instead implemented via plain lazy sampling. Eventually, $\mathcal{E}$ returns the representation of the outputs of $\mathcal{A}$ it has constructed while observing $\mathcal{A}$, but ignoring the parts pertaining to the variables $\boldsymbol{R}_v$.

To show that $\mathcal{E}$ correctly represents the outputs of $\mathcal{A}$, we must prove that it is unlikely that these outputs satisfy the relation polynomial, and yet use group elements not obtained through op_ν in a non-trivial way. To that end, we first apply the Schwartz–Zippel lemma and transition to a setting where the game replaces all values it samples at random with formal variables. Accordingly, equality $Q(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_T, \boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{c}) = 0$ in the winning condition is now an equality between polynomials, with elements not obtained through op_ν corresponding to the variables $\boldsymbol{R}_v$ above. We must then show that the coefficients $\boldsymbol{b}_{\nu,i_\nu l}$ of these variables in the representation of $\mathcal{E}$ are all zero. First, we prove that the coefficients $\boldsymbol{b}_{1,i_1 l_1} \boldsymbol{b}_{2,i_2 l_2}$ of the square terms $\boldsymbol{R}_{l_1} \boldsymbol{R}_{l_2}$ in $\boldsymbol{R}$ are separately zero. Indeed, if that was not the case and, say, $\boldsymbol{b}_{1,i_1 \bar{l}} \neq 0$, we can use the linear term in $\boldsymbol{R}_{\bar{l}}$ to express all terms involving $\overline{Q_{i_1 i_2}}$ as a linear combination of the $\overline{Q_{\nu,i}}$. Plugging that into the constant term in $\boldsymbol{R}$, we obtain a linear representation of $\overline{Q_0}$ in terms of polynomials $\overline{Q_{\nu,i}} \boldsymbol{P}_{\nu',j}$, which contradicts our assumption. Once the coefficients of all square terms are shown to be zero, for each l the linear term in $\boldsymbol{R}_l$ is a linear combination of the $\overline{Q_{\nu,i}}$, weighted with $\boldsymbol{b}_{\nu,i_\nu l}$. By linear independence of the $\overline{Q_{\nu,i}}$, we conclude that $\boldsymbol{b}_{\nu,i_\nu l} = 0$ for all ν, i_ν, and l.

```
Extractor $\mathcal{E}^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(\mathsf{trace}(\mathcal{A}))$:
parse $\mathsf{trace}(\mathcal{A}) = (r_{\mathcal{A}}, \boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_T, \boldsymbol{h})$; $o, v \leftarrow 0$
$U_{\tau_1}, U_{\tau_2}, U_{\tau_T}, U_{H_1}, U_{H_2}, U_{H_T} \leftarrow [\,]$
$U_{\tau_1}[1] \leftarrow \boldsymbol{u}_{1,0}$; $U_{\tau_2}[1] \leftarrow \boldsymbol{u}_{2,0}$
$(Q, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_T) \leftarrow \mathcal{A}_0^{\overline{\mathsf{op}}_1,\overline{\mathsf{op}}_2,\overline{\mathsf{op}}_T,\overline{\mathsf{H}}_1,\overline{\mathsf{H}}_2,\overline{\mathsf{H}}_T,\overline{\mathsf{e}}}(\boldsymbol{u}_{1,0}, \boldsymbol{u}_{2,0}; r_{\mathcal{A}})$
for $\mu \in \{1, 2, T\}$ do
   for $j = 1$ to $|\boldsymbol{P}_\mu|$ do $U_{\tau_\mu}[\boldsymbol{P}_{\mu,j}(\boldsymbol{S})] \leftarrow \boldsymbol{u}_{\mu,j}$
$(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{c}) \leftarrow \mathcal{A}_1^{\overline{\mathsf{op}}_1,\overline{\mathsf{op}}_2,\overline{\mathsf{op}}_T,\overline{\mathsf{H}}_1,\overline{\mathsf{H}}_2,\overline{\mathsf{H}}_T,\overline{\mathsf{e}}}(\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_T; r_{\mathcal{A}})$
$\boldsymbol{P}_{1,0}(\boldsymbol{S}), \boldsymbol{P}_{2,0}(\boldsymbol{S}) \leftarrow 1$
for $\nu = 1$ to $2$ do for $i = 1$ to $|\boldsymbol{Y}_\nu|$ do
   if $(\boldsymbol{v}_{\nu,i} \notin \mathrm{Rng}(U_{\tau_\nu}))$ then $v \leftarrow v + 1$; $U_{\tau_\nu}[\boldsymbol{R}_v] \leftarrow \boldsymbol{v}_i$
   parse $U_{\tau_\nu}^{-1}[\boldsymbol{v}_{\nu,i}] =$
      $\sum_{j=0}^{|\boldsymbol{X}_\nu|-1} \boldsymbol{w}_{\nu,ij} \boldsymbol{P}_{\nu,j}(\boldsymbol{S}) + \sum_l \boldsymbol{b}_{\nu,il} \boldsymbol{R}_l$
return $(\boldsymbol{w}_1, \boldsymbol{w}_2)$
```

```
Proc. $\overline{\mathsf{op}}_\nu(h_1, h_2)$:
for $i = 1$ to $2$ do
   if $(h_i \notin \mathrm{Rng}(U_{\tau_\nu}))$
   then
      $v \leftarrow v + 1$
      $U_{\tau_\nu}[\boldsymbol{R}_v] \leftarrow h_i$
   $x_i \leftarrow U_{\tau_\nu}^{-1}[h_i]$
$x \leftarrow x_1 + x_2$; $o \leftarrow o + 1$
if $(x \notin \mathrm{Dom}(U_{\tau_\nu}))$
   then $U_{\tau_\nu}[x] \leftarrow \boldsymbol{h}_o$
return $U_{\tau_\nu}[x]$
```

```
Proc. $\overline{\mathsf{op}}_T(h_1, h_2)$:
for $i = 1$ to $2$ do
   if $(h_i \notin \mathrm{Rng}(U_{\tau_T}))$
   then
      $x_i \leftarrow \mathbb{Z}_p \setminus \mathrm{Dom}(U_{\tau_T})$
      $U_{\tau_T}[x_i] \leftarrow h_i$
   $x_i \leftarrow U_{\tau_T}^{-1}[h_i]$
$x \leftarrow x_1 + x_2$; $o \leftarrow o + 1$
if $(x \notin \mathrm{Dom}(U_{\tau_T}))$ then
   $U_{\tau_T}[x] \leftarrow \boldsymbol{h}_o$
return $U_{\tau_T}[x]$
```

```
Proc. $\overline{\mathsf{H}}_\nu(m)$:
if $(m \notin \mathrm{Dom}(U_{H_\nu}))$ then
   $v \leftarrow v + 1$; $U_{H_\nu}[m] \leftarrow \boldsymbol{R}_v$
$r \leftarrow U_{H_\nu}[m]$; $o \leftarrow o + 1$
if $(r \notin \mathrm{Dom}(U_{\tau_\nu}))$ then
   $U_{\tau_\nu}[r] \leftarrow \boldsymbol{h}_o$
return $U_{\tau_\nu}[r]$
```

```
Proc. $\overline{\mathsf{H}}_T(m)$:
if $(m \notin \mathrm{Dom}(U_{H_T}))$ then
   $r \twoheadleftarrow \mathbb{Z}_p$; $U_{H_T}[m] \leftarrow r$
$r \leftarrow U_{H_T}[m]$; $o \leftarrow o + 1$
if $(r \notin \mathrm{Dom}(U_{\tau_T}))$ then
   $U_{\tau_T}[r] \leftarrow \boldsymbol{h}_o$
return $U_{\tau_T}[r]$
```

```
Proc. $\overline{\mathsf{e}}(h_1, h_2)$:
for $\nu = 1$ to $2$ do
   if $(h_\nu \notin \mathrm{Rng}(U_{\tau_\nu}))$ then
      $v \leftarrow v + 1$
      $U_{\tau_\nu}[\boldsymbol{R}_v] \leftarrow h_\nu$
   $x_\nu \leftarrow U_{\tau_\nu}^{-1}[h_\nu]$
$x \leftarrow x_1 x_2$; $o \leftarrow o + 1$
if $(x \notin \mathrm{Dom}(U_{\tau_T}))$ then
   $U_{\tau_T}[x] \leftarrow \boldsymbol{h}_o$
return $U_{\tau_T}[x]$
```

Figure 18: Definition of the extractor $\mathcal{E}$ from the proof of Theorem 3. Counters o and v are shared between all oracles, and ν is an index ranging over $\{1, 2\}$.

Proof. We now formally implement the intuition presented in the proof overview above. Fix an adversary $\mathcal{A}$ as above, and define an extractor $\mathcal{E}$ as shown in Figure 18. We claim that this extractor allows proving Inequality (4). To that end, consider the sequence of games below (the formal description of which can be found in Figures 19 and 20). For brevity, we define the predicate $\mathsf{R}(Q, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_T, \boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{c}, \boldsymbol{w}_1, \boldsymbol{w}_2)$ to return 1 if and only if

$$\big(Q(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_T, \boldsymbol{Y}_1, \boldsymbol{Y}_2, \boldsymbol{c}) \neq 0\big) \wedge \big(Q(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_T, \boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{c}) = 0\big) \wedge \\ \left((\exists \nu)(\exists i)\left(\boldsymbol{y}_{\nu,i} \neq \sum\nolimits_{j=0}^{|\boldsymbol{X}_\nu|-1} \boldsymbol{w}_{\nu,ij} \boldsymbol{x}_{\nu,j}\right)\right).$$

- G_0: This is the original UK game in the GBM3-H with parameters $(p, \mathbf{G})$ and source $\mathcal{S}$, run with adversary $\mathcal{A}$ and extractor $\mathcal{E}$. We omit repeated invocations of op_μ to create the inputs of $\mathcal{A}_1$, and instead compute $\tau_\mu(\boldsymbol{x}_\mu)$ directly. We also reformulate the winning condition by not applying τ_μ in the last two clauses, which results in an equivalent game since they are all injective. The operation, hashing and pairing oracles are augmented to construct the view of $\mathcal{A}$ along the way.
- G_1: This game proceeds as G_0, but the encodings τ_μ are implemented via lazy sampling. More precisely, instead of sampling τ_μ, G_1 initializes tables $T_{\tau_\mu} \leftarrow [\,]$. Oracles op_μ and H_μ are then implemented via lazy sampling from G_μ using table T_{τ_μ}. The same is done for oracle e, using tables T_ν.

```
Game $\mathrm{G}_0$:
$\tau_1 \twoheadleftarrow \mathrm{Inj}(\mathbb{Z}_p, \mathsf{G}_1)$; $\tau_2 \twoheadleftarrow \mathrm{Inj}(\mathbb{Z}_p, \mathsf{G}_2)$; $\tau_T \twoheadleftarrow \mathrm{Inj}(\mathbb{Z}_p, \mathsf{G}_T)$; $T_{H_1}, T_{H_2}, T_{H_T} \leftarrow [\,]$; $o \leftarrow 0$; $r_{\mathcal{A}} \twoheadleftarrow \mathcal{R}_{\mathcal{A}}$
$\boldsymbol{u}_{1,0} \leftarrow \tau_1(1)$; $\boldsymbol{u}_{2,0} \leftarrow \tau_2(1)$; $(Q, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_T) \leftarrow \mathcal{A}_0^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(\boldsymbol{u}_{1,0}, \boldsymbol{u}_{2,0}; r_{\mathcal{A}})$
$\boldsymbol{s} \twoheadleftarrow \mathbb{Z}_p^k$; $\boldsymbol{x}_1 \leftarrow \boldsymbol{P}_1(\boldsymbol{s})$; $\boldsymbol{x}_2 \leftarrow \boldsymbol{P}_2(\boldsymbol{s})$; $\boldsymbol{x}_T \leftarrow \boldsymbol{P}_T(\boldsymbol{s})$; $\boldsymbol{x}_{1,0}, \boldsymbol{x}_{2,0} \leftarrow 1$
$\boldsymbol{u}_1 \leftarrow \tau_1(\boldsymbol{x}_1)$; $\boldsymbol{u}_2 \leftarrow \tau_2(\boldsymbol{x}_2)$; $\boldsymbol{u}_T \leftarrow \tau_T(\boldsymbol{x}_T)$; $(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{c}) \leftarrow \mathcal{A}_1^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_T; r_{\mathcal{A}})$
$\mathsf{trace}(\mathcal{A}) \leftarrow (r_{\mathcal{A}}, \boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_T, \boldsymbol{h})$; $(\boldsymbol{w}_1, \boldsymbol{w}_2) \twoheadleftarrow \mathcal{E}^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(\mathsf{trace}(\mathcal{A}))$
$\boldsymbol{y}_1 \leftarrow \tau_1^{-1}(\boldsymbol{v}_1)$; $\boldsymbol{y}_2 \leftarrow \tau_2^{-1}(\boldsymbol{v}_2)$; return $\mathsf{R}(Q, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_T, \boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{c}, \boldsymbol{w}_1, \boldsymbol{w}_2)$
```

```
Proc. $\mathsf{op}_\mu(h_1, h_2)$:
$x_1 \leftarrow \tau_\mu^{-1}(h_1)$
$x_2 \leftarrow \tau_\mu^{-1}(h_2)$
$o \leftarrow o+1$; $\boldsymbol{h}_o \leftarrow \tau_\mu(x_1+x_2)$
return $\boldsymbol{h}_o$
```

```
Proc. $\mathsf{H}_\mu(m)$:
if $m \notin \mathrm{Dom}(T_{H_\mu})$ then
    $r \twoheadleftarrow \mathbb{Z}_p$; $T_{H_\mu}[m] \leftarrow r$
$r \leftarrow T_{H_\mu}[m]$; $o \leftarrow o+1$; $\boldsymbol{h}_o \leftarrow \tau_\mu(r)$
return $\boldsymbol{h}_o$
```

```
Proc. $\mathsf{e}(h_1, h_2)$:
$x_1 \leftarrow \tau_1^{-1}(h_1)$
$x_2 \leftarrow \tau_2^{-1}(h_2)$
$o \leftarrow o+1$; $\boldsymbol{h}_o \leftarrow \tau_T(x_1 x_2)$
return $\boldsymbol{h}_o$
```

```
Game $\mathrm{G}_1$:
$T_{\tau_1}, T_{\tau_2}, T_{\tau_T}, T_{H_1}, T_{H_2}, T_{H_T} \leftarrow [\,]$
$o \leftarrow 0$; $r_{\mathcal{A}} \twoheadleftarrow \mathcal{R}_{\mathcal{A}}$
$\boldsymbol{u}_{1,0} \twoheadleftarrow \mathsf{G}_1$; $\boldsymbol{u}_{2,0} \twoheadleftarrow \mathsf{G}_2$
$T_{\tau_1}[1] \leftarrow \boldsymbol{u}_{1,0}$; $T_{\tau_2}[1] \leftarrow \boldsymbol{u}_{2,0}$
$(Q, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_T) \leftarrow$
    $\mathcal{A}_0^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(\boldsymbol{u}_{1,0}, \boldsymbol{u}_{2,0}; r_{\mathcal{A}})$
$\boldsymbol{s} \twoheadleftarrow \mathbb{Z}_p^k$; $\boldsymbol{x}_{1,0}, \boldsymbol{x}_{2,0} \leftarrow 1$
$\boldsymbol{x}_1 \leftarrow \boldsymbol{P}_1(\boldsymbol{s})$; $\boldsymbol{x}_2 \leftarrow \boldsymbol{P}_2(\boldsymbol{s})$; $\boldsymbol{x}_T \leftarrow \boldsymbol{P}_T(\boldsymbol{s})$
for $\mu \in \{1, 2, T\}$ do for $j = 1$ to $|\boldsymbol{P}_\mu|$ do
    if $(\boldsymbol{x}_{\mu,j} \notin \mathrm{Dom}(T_{\tau_\mu}))$ then
        $\boldsymbol{u}_{\mu,j} \twoheadleftarrow \mathsf{G}_\mu \setminus \mathrm{Rng}(T_{\tau_\mu})$; $T_{\tau_\mu}[\boldsymbol{x}_{\mu,j}] \leftarrow \boldsymbol{u}_{\mu,j}$
    $\boldsymbol{u}_{\mu,j} \leftarrow T_{\tau_\mu}[\boldsymbol{x}_{\mu,j}]$
$(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{c}) \leftarrow$
    $\mathcal{A}_1^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_T; r_{\mathcal{A}})$
$\mathsf{trace}(\mathcal{A}) \leftarrow (r_{\mathcal{A}}, \boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_T, \boldsymbol{h})$
$(\boldsymbol{w}_1, \boldsymbol{w}_2) \twoheadleftarrow \mathcal{E}^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(\mathsf{trace}(\mathcal{A}))$
for $\nu = 1$ to $2$ do for $i = 1$ to $|\boldsymbol{Y}_\nu|$ do
    if $(\boldsymbol{v}_{\nu,i} \notin \mathrm{Rng}(T_{\tau_\nu}))$ then
        $\boldsymbol{y}_{\nu,i} \twoheadleftarrow \mathbb{Z}_p \setminus \mathrm{Dom}(T_{\tau_\nu})$; $T_{\tau_\nu}[\boldsymbol{y}_{\nu,i}] \leftarrow \boldsymbol{v}_{\nu,i}$
    $\boldsymbol{y}_{\nu,i} \leftarrow T_{\tau_\nu}^{-1}[\boldsymbol{v}_{\nu,i}]$
return $\mathsf{R}(Q, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_T, \boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{c}, \boldsymbol{w}_1, \boldsymbol{w}_2)$
```

```
Proc. $\mathsf{op}_\mu(h_1, h_2)$:
for $i = 1$ to $2$ do
    if $(h_i \notin \mathrm{Rng}(T_{\tau_\mu}))$ then
        $x_i \leftarrow \mathbb{Z}_p \setminus \mathrm{Dom}(T_{\tau_\mu})$; $T_{\tau_\mu}[x_i] \leftarrow h_i$
    $x_i \leftarrow T_{\tau_\mu}^{-1}[h_i]$
$x \leftarrow x_1 + x_2$
if $(x \notin \mathrm{Dom}(T_{\tau_\mu}))$ then
    $h \twoheadleftarrow \mathsf{G}_\mu \setminus \mathrm{Rng}(T_{\tau_\mu})$; $T_{\tau_\mu}[x] \leftarrow h$
$o \leftarrow o + 1$; $\boldsymbol{h}_o \leftarrow T_{\tau_\mu}[x]$; return $\boldsymbol{h}_o$
```

```
Proc. $\mathsf{H}_\mu(m)$:
if $(m \notin \mathrm{Dom}(T_{H_\mu}))$ then $r \twoheadleftarrow \mathbb{Z}_p$; $T_{H_\mu}[m] \leftarrow r$
$r \leftarrow T_{H_\mu}[m]$
if $(r \notin \mathrm{Dom}(T_{\tau_\mu}))$ then
    $h \twoheadleftarrow \mathsf{G}_\mu \setminus \mathrm{Rng}(T_{\tau_\mu})$; $T_{\tau_\mu}[r] \leftarrow h$
$o \leftarrow o + 1$; $\boldsymbol{h}_o \leftarrow T_{\tau_\mu}[r]$; return $\boldsymbol{h}_o$
```

```
Proc. $\mathsf{e}(h_1, h_2)$:
for $\nu = 1$ to $2$ do
    if $(h_\nu \notin \mathrm{Rng}(T_{\tau_\nu}))$ then
        $x_\nu \leftarrow \mathbb{Z}_p \setminus \mathrm{Dom}(T_{\tau_\nu})$; $T_{\tau_\nu}[x_\nu] \leftarrow h_\nu$
    $x_\nu \leftarrow T_{\tau_\nu}^{-1}[h_\nu]$
$x \leftarrow x_1 x_2$
if $(x \notin \mathrm{Dom}(T_{\tau_T}))$ then
    $h \twoheadleftarrow \mathsf{G}_T \setminus \mathrm{Rng}(T_{\tau_T})$; $T_{\tau_T}[x] \leftarrow h$
$o \leftarrow o + 1$; $\boldsymbol{h}_o \leftarrow T_{\tau_T}[x]$; return $\boldsymbol{h}_o$
```

Figure 19: Code of the intermediate games in the proof of Inequality (4). Here, μ is an index ranging over $\{1, 2, T\}$.

- G_2: This game proceeds as G_1, but it replaces the values $\boldsymbol{x}_\mu$ generated by $\mathcal{S}$ with the corresponding polynomials $\boldsymbol{P}_\mu(\boldsymbol{S})$ evaluated at formal variables $\boldsymbol{S}$. Likewise, whenever it lazily samples a domain point in T_{τ_ν}, it instead saves a fresh variable $\boldsymbol{R}_\nu$. (Note that this is only done for oracles pertaining to G_ν; oracles for G_T are as in G_1.) Only after $\mathcal{A}$ and $\mathcal{E}$ are run, G_2 samples random $\boldsymbol{s}$ and $\boldsymbol{r}$ of the appropriate length, evaluates the inputs and outputs of $\mathcal{A}$ at these points, and checks the winning condition as in G_1. Notice that in this game, tables T_{τ_μ} are populated as tables U_{τ_μ} compiled by $\mathcal{E}$.
- G_3: This game proceeds as G_2, but we omit the sampling of $\boldsymbol{s}$ and $\boldsymbol{r}$, and instead regard the winning condition as a set of (in)equalities between polynomials in $\boldsymbol{S}$ and $\boldsymbol{R}$.

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 $\Pr[\mathrm{G}_1] = \Pr[\mathrm{G}_0]$.

```
Game G_2:
$T_{\tau_1}, T_{\tau_2}, T_{\tau_T}, T_{H_1}, T_{H_2}, T_{H_T} \leftarrow []$
$o, v \leftarrow 0$; $r_{\mathcal{A}} \twoheadleftarrow \mathcal{R}_{\mathcal{A}}$; $\boldsymbol{u}_{1,0} \twoheadleftarrow \mathsf{G}_1$; $\boldsymbol{u}_{2,0} \twoheadleftarrow \mathsf{G}_2$
$T_{\tau_1}[1] \leftarrow \boldsymbol{u}_{1,0}$; $T_{\tau_2}[1] \leftarrow \boldsymbol{u}_{2,0}$
$(Q, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_T) \leftarrow$
  $\mathcal{A}_0^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(\boldsymbol{u}_{1,0}, \boldsymbol{u}_{2,0}; r_{\mathcal{A}})$
$\boldsymbol{x}_1 \leftarrow \boldsymbol{P}_1(\boldsymbol{S})$; $\boldsymbol{x}_2 \leftarrow \boldsymbol{P}_2(\boldsymbol{S})$; $\boldsymbol{x}_T \leftarrow \boldsymbol{P}_T(\boldsymbol{S})$
$\boldsymbol{x}_{1,0}, \boldsymbol{x}_{2,0} \leftarrow 1$
for $\mu \in \{1, 2, T\}$ do for $j = 1$ to $|\boldsymbol{P}_\mu|$ do
  if $(\boldsymbol{x}_{\mu,j} \notin \mathrm{Dom}(T_{\tau_\mu}))$ then
    $\boldsymbol{u}_{\mu,j} \twoheadleftarrow \mathsf{G}_\mu \setminus \mathrm{Rng}(T_{\tau_\mu})$; $T_{\tau_\mu}[\boldsymbol{x}_{\mu,j}] \leftarrow \boldsymbol{u}_{\mu,j}$
  $\boldsymbol{u}_{\mu,j} \leftarrow T_{\tau_\mu}[\boldsymbol{x}_{\mu,j}]$
$(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{c}) \leftarrow$
  $\mathcal{A}_1^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_T; r_{\mathcal{A}})$
$\mathsf{trace}(\mathcal{A}) \leftarrow (r_{\mathcal{A}}, \boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_T, \boldsymbol{h})$
$(\boldsymbol{w}_1, \boldsymbol{w}_2) \twoheadleftarrow \mathcal{E}^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(\mathsf{trace}(\mathcal{A}))$
$\boldsymbol{s} \twoheadleftarrow \mathbb{Z}_p^k$; $\boldsymbol{r} \twoheadleftarrow \mathbb{Z}_p^{2q_{\mathsf{op}}+q_{\mathsf{H}}+2q_{\mathsf{e}}+2n}$
for $\nu = 1$ to 2 do for $i = 1$ to $|\boldsymbol{Y}_\nu|$ do
  if $(\boldsymbol{v}_{\nu,i} \notin \mathrm{Rng}(T_{\tau_\nu}))$ then
    $v \leftarrow v + 1$; $T_{\tau_\nu}[\boldsymbol{R}_v] \leftarrow \boldsymbol{v}_{\nu,i}$
  $\boldsymbol{y}_{\nu,i} \leftarrow T_{\tau_\nu}^{-1}[\boldsymbol{v}_{\nu,i}]$; $\boldsymbol{y}_{\nu,i} \leftarrow \boldsymbol{y}_{\nu,i}(\boldsymbol{s}, \boldsymbol{r})$
return $\mathsf{R}(Q, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_T, \boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{c}, \boldsymbol{w}_1, \boldsymbol{w}_2)$
```

```
Game G_3:
$T_{\tau_1}, T_{\tau_2}, T_{\tau_T}, T_{H_1}, T_{H_2}, T_{H_T} \leftarrow []$
$o, v \leftarrow 0$; $r_{\mathcal{A}} \twoheadleftarrow \mathcal{R}_{\mathcal{A}}$; $\boldsymbol{u}_{1,0} \twoheadleftarrow \mathsf{G}_1$; $\boldsymbol{u}_{2,0} \twoheadleftarrow \mathsf{G}_2$
$T_{\tau_1}[1] \leftarrow \boldsymbol{u}_{1,0}$; $T_{\tau_2}[1] \leftarrow \boldsymbol{u}_{2,0}$
$(Q, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_T) \leftarrow$
  $\mathcal{A}_0^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(\boldsymbol{u}_{1,0}, \boldsymbol{u}_{2,0}; r_{\mathcal{A}})$
$\boldsymbol{x}_1 \leftarrow \boldsymbol{P}_1(\boldsymbol{S})$; $\boldsymbol{x}_2 \leftarrow \boldsymbol{P}_2(\boldsymbol{S})$; $\boldsymbol{x}_T \leftarrow \boldsymbol{P}_T(\boldsymbol{S})$
$\boldsymbol{x}_{1,0}, \boldsymbol{x}_{2,0} \leftarrow 1$
for $\mu \in \{1, 2, T\}$ do for $j = 1$ to $|\boldsymbol{P}_\mu|$ do
  if $(\boldsymbol{x}_{\mu,j} \notin \mathrm{Dom}(T_{\tau_\mu}))$ then
    $\boldsymbol{u}_{\mu,j} \twoheadleftarrow \mathsf{G}_\mu \setminus \mathrm{Rng}(T_{\tau_\mu})$; $T_{\tau_\mu}[\boldsymbol{x}_{\mu,j}] \leftarrow \boldsymbol{u}_{\mu,j}$
  $\boldsymbol{u}_{\mu,j} \leftarrow T_{\tau_\mu}[\boldsymbol{x}_{\mu,j}]$
$(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{c}) \leftarrow$
  $\mathcal{A}_1^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_T; r_{\mathcal{A}})$
$\mathsf{trace}(\mathcal{A}) \leftarrow (r_{\mathcal{A}}, \boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_T, \boldsymbol{h})$
$(\boldsymbol{w}_1, \boldsymbol{w}_2) \twoheadleftarrow \mathcal{E}^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(\mathsf{trace}(\mathcal{A}))$
for $\nu = 1$ to 2 do for $i = 1$ to $|\boldsymbol{Y}_\nu|$ do
  if $(\boldsymbol{v}_{\nu,i} \notin \mathrm{Rng}(T_{\tau_\nu}))$ then
    $v \leftarrow v + 1$; $T_{\tau_\nu}[\boldsymbol{R}_v] \leftarrow \boldsymbol{v}_{\nu,i}$
  $\boldsymbol{y}_{\nu,i} \leftarrow T_{\tau_\nu}^{-1}[\boldsymbol{v}_{\nu,i}]$
return $\mathsf{R}(Q, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_T, \boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{c}, \boldsymbol{w}_1, \boldsymbol{w}_2)$
```

```
Proc. $\mathsf{op}_\nu(h_1, h_2)$:
for $i = 1$ to 2 do
  if $(h_i \notin \mathrm{Rng}(T_{\tau_\nu}))$ then
    $v \leftarrow v + 1$; $T_{\tau_\nu}[\boldsymbol{R}_v] \leftarrow h_i$
  $x_i \leftarrow T_{\tau_\nu}^{-1}[h_i]$
$x \leftarrow x_1 + x_2$
if $(x \notin \mathrm{Dom}(T_{\tau_\nu}))$ then
  $h \twoheadleftarrow \mathsf{G}_\nu \setminus \mathrm{Rng}(T_{\tau_\nu})$
  $T_{\tau_\nu}[x] \leftarrow h$
$o \leftarrow o + 1$; $\boldsymbol{h}_o \leftarrow T_{\tau_\nu}[x]$; return $\boldsymbol{h}_o$
```

```
Proc. $\mathsf{H}_\nu(m)$:
if $(m \notin \mathrm{Dom}(T_{H_\nu}))$ then
  $v \leftarrow v + 1$
  $T_{\tau_\nu}[m] \leftarrow \boldsymbol{R}_v$
$r \leftarrow T_{H_\nu}[m]$
if $(r \notin \mathrm{Dom}(T_{\tau_\nu}))$ then
  $h \twoheadleftarrow \mathsf{G}_\nu \setminus \mathrm{Rng}(T_{\tau_\nu})$
  $T_{\tau_\nu}[r] \leftarrow h$
$o \leftarrow o + 1$; $\boldsymbol{h}_o \leftarrow T_{\tau_\nu}[r]$
return $\boldsymbol{h}_o$
```

```
Proc. $\mathsf{e}(h_1, h_2)$:
for $\nu = 1$ to 2 do
  if $(h_\nu \notin \mathrm{Rng}(T_{\tau_\nu}))$ then
    $v \leftarrow v + 1$; $T_{\tau_\nu}[\boldsymbol{R}_v] \leftarrow h_\nu$
  $x_\nu \leftarrow T_{\tau_\nu}^{-1}[h_\nu]$
$x \leftarrow x_1 x_2$
if $(x \notin \mathrm{Dom}(T_{\tau_T}))$ then
  $h \twoheadleftarrow \mathsf{G}_T \setminus \mathrm{Rng}(T_{\tau_T})$
  $T_{\tau_T}[x] \leftarrow h$
$o \leftarrow o + 1$; $\boldsymbol{h}_o \leftarrow T_{\tau_T}[x]$; return $\boldsymbol{h}_o$
```

Figure 20: Code of the intermediate games in the proof of Inequality (4). Here, ν is an index ranging over $\{1, 2\}$. Oracles op_T and H_T are as in the bottom part of Figure 19.

$\mathrm{G}_1 \rightsquigarrow \mathrm{G}_2$. Let Bad_μ be the event in G_2 that there are two different polynomials in $\mathrm{Dom}(T_{\tau_\mu})$ which result in the same value when evaluating $\boldsymbol{S}$ and $\boldsymbol{R}$ at random $\boldsymbol{s}$ and $\boldsymbol{r}$. Notice that G_1 and G_2 are identical until Bad_1 or Bad_2 or Bad_T, and by the fundamental lemma of game playing we therefore have that $|\Pr[\mathrm{G}_2] - \Pr[\mathrm{G}_1]| \leq \Pr[\mathsf{Bad}_1] + \Pr[\mathsf{Bad}_2] + \Pr[\mathsf{Bad}_T]$.

We bound the latter probabilities via Lemma 1. Consider the adversary $\mathcal{B}_1$ in the Schwartz–Zippel game defined in Figure 21. Here, $\mathcal{B}_1$ simulates G_2 to $\mathcal{A}$ and then returns all entries in $\mathrm{Dom}(T_{\tau_1})$. Notice that if Bad_1 occurs, then $\mathcal{B}_1$ wins the SZ-game, and that T_{τ_1} contains at most $m+n+3q_{\mathsf{op}}+q_{\mathsf{H}}+q_{\mathsf{e}}+1$ polynomials of degree at most d_P. By Lemma 1, $\Pr[\mathsf{Bad}_1] \leq (m+n+3q_{\mathsf{op}}+q_{\mathsf{H}}+q_{\mathsf{e}}+1)^2 \cdot d_P/2p$. We similarly bound $\Pr[\mathsf{Bad}_2]$ and $\Pr[\mathsf{Bad}_T]$ using adversaries $\mathcal{B}_2$ and $\mathcal{B}_T$ in the Schwartz–Zippel game defined in Figure 21, noting that T_{τ_2} contains at most $m + n + 3q_{\mathsf{op}} + q_{\mathsf{H}} + q_{\mathsf{e}} + 1$ polynomials of degree at most d_P, and at most $m + 3q_{\mathsf{op}} + q_{\mathsf{H}} + q_{\mathsf{e}}$ polynomials of degree at most $2d_P$. Therefore,

$$|\Pr[\mathrm{G}_2]-\Pr[\mathrm{G}_1]| \leq \Pr[\mathsf{Bad}_1]+\Pr[\mathsf{Bad}_2]+\Pr[\mathsf{Bad}_T] \leq \frac{2(m+n+3q_{\mathsf{op}}+q_{\mathsf{H}}+q_{\mathsf{e}}+1)^2 \cdot d_P}{p}.$$

$\mathrm{G}_2 \rightsquigarrow \mathrm{G}_3$. Let Bad' be the event in G_3 that either $Q(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_T, \boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{c}) \neq 0$, or $\boldsymbol{y}_{\nu,i} \neq \sum_{j=0}^{|\boldsymbol{X}_\nu|-1} \boldsymbol{w}_{\nu,ij} \boldsymbol{x}_{\nu,j}$ for some $\nu \in \{1, 2\}$ and $1 \leq i \leq |\boldsymbol{Y}_\nu|$, but the corresponding equality holds when evaluating $\boldsymbol{S}$ and $\boldsymbol{R}$ at random $\boldsymbol{s}$ and $\boldsymbol{r}$. Then G_2 and G_3 are identical until Bad', and by the fundamental lemma of game playing we have $|\Pr[\mathrm{G}_3] - \Pr[\mathrm{G}_2]| \leq \Pr[\mathsf{Bad}']$.

```
Adversaries $\mathcal{B}_\mu/\mathcal{B}'/\mathcal{B}'_{\nu,i}$:
$T_{\tau_1}, T_{\tau_2}, T_{\tau_T}, T_{H_1}, T_{H_2}, T_{H_T} \leftarrow [\,]$; $o, v \leftarrow 0$; $r_\mathcal{A} \twoheadleftarrow \mathcal{R}_\mathcal{A}$
$\boldsymbol{u}_{1,0} \twoheadleftarrow \mathsf{G}_1$; $\boldsymbol{u}_{2,0} \twoheadleftarrow \mathsf{G}_2$; $T_{\tau_1}[1] \leftarrow \boldsymbol{u}_{1,0}$; $T_{\tau_2}[1] \leftarrow \boldsymbol{u}_{2,0}$
$(Q, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_T) \leftarrow \mathcal{A}_0^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(\boldsymbol{u}_{1,0}, \boldsymbol{u}_{2,0}; r_\mathcal{A})$
$\boldsymbol{x}_1 \leftarrow \boldsymbol{P}_1(\boldsymbol{S})$; $\boldsymbol{x}_2 \leftarrow \boldsymbol{P}_2(\boldsymbol{S})$; $\boldsymbol{x}_T \leftarrow \boldsymbol{P}_T(\boldsymbol{S})$; $\boldsymbol{x}_{1,0}, \boldsymbol{x}_{2,0} \leftarrow 1$
for $\mu \in \{1, 2, T\}$ do for $j = 1$ to $|\boldsymbol{P}_\mu|$ do
    if $(\boldsymbol{x}_{\mu,j} \notin \mathrm{Dom}(T_{\tau_\mu}))$ then $\boldsymbol{u}_{\mu,j} \twoheadleftarrow \mathsf{G}_\mu \setminus \mathrm{Rng}(T_{\tau_\mu})$; $T_{\tau_\mu}[\boldsymbol{x}_{\mu,j}] \leftarrow \boldsymbol{u}_{\mu,j}$
    $\boldsymbol{u}_{\mu,j} \leftarrow T_{\tau_\mu}[\boldsymbol{x}_{\mu,j}]$
$(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{c}) \leftarrow \mathcal{A}_1^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_T; r_\mathcal{A})$; $\mathsf{trace}(\mathcal{A}) \leftarrow (r_\mathcal{A}, \boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_T, \boldsymbol{h})$
$(\boldsymbol{w}_1, \boldsymbol{w}_2) \twoheadleftarrow \mathcal{E}^{\mathsf{op}_1,\mathsf{op}_2,\mathsf{op}_T,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T,\mathsf{e}}(\mathsf{trace}(\mathcal{A}))$
for $\nu = 1$ to $2$ do for $i = 1$ to $|\boldsymbol{Y}_\nu|$ do
    if $(\boldsymbol{v}_{\nu,i} \notin \mathrm{Rng}(T_{\tau_\nu}))$ then $v \leftarrow v + 1$; $T_{\tau_\nu}[\boldsymbol{R}_v] \leftarrow \boldsymbol{v}_{\nu,i}$
    $\boldsymbol{y}_{\nu,i} \leftarrow T_{\tau_\nu}^{-1}[\boldsymbol{v}_{\nu,i}]$
$\mathcal{B}_\mu$: return $\mathrm{Dom}(T_{\tau_\mu})$                    $\mathcal{B}'_{\nu,i}$: return $\left(\boldsymbol{y}_{\nu,i} - \sum_{j=0}^{|\boldsymbol{X}_\nu|-1} \boldsymbol{w}_{\nu,ij}\boldsymbol{x}_{\nu,j}, 0\right)$
$\mathcal{B}'$: return $(Q(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_T, \boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{c}), 0)$
```

Figure 21: Definition of the adversaries $\mathcal{B}_\mu$, $\mathcal{B}'$ and $\mathcal{B}'_{\nu,i}$ from the proof of Theorem 3. In all cases, oracles op_μ, H_μ and e are defined as in Figure 20, and μ and ν are indices ranging over $\{1, 2, T\}$ and $\{1, 2\}$, respectively.

We bound the latter probability via Lemma 1. Consider the adversaries $\mathcal{B}'$ and $\mathcal{B}'_{\nu,i}$ in the Schwartz–Zippel game defined in Figure 21. Here, $\mathcal{B}'$ and $\mathcal{B}'_{\nu,i}$ simulate G_3 to $\mathcal{A}$ and return $(Q(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_T, \boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{c}), 0)$ and $\left(\boldsymbol{y}_{\nu,i} - \sum_{j=0}^{|\boldsymbol{X}_\nu|-1} \boldsymbol{w}_{\nu,ij}\boldsymbol{x}_{\nu,j}, 0\right)$, respectively. Notice that if Bad' occurs, then $\mathcal{B}'$ or $\mathcal{B}'_{\nu,i}$ win the SZ-game for some $\nu \in \{1, 2\}$ and $1 \leq i \leq |\boldsymbol{Y}_\nu|$, and that the polynomials returned by $\mathcal{B}'$ and $\mathcal{B}'_{\nu,i}$ have total degree at most $d_Q d_P$ and d_P, respectively. By Lemma 1, $\Pr[\mathsf{Bad}'] \leq d_Q d_P/p + 2nd_P/p$.

We conclude the proof by showing that the winning probability of $\mathcal{A}$ in G_3 is zero. Notice that if the output of $\mathcal{A}$ is such that the relation polynomial Q is not satisfied, then $\mathcal{A}$ has trivially lost the game. If on the other hand Q is satisfied, we obtain

$$\begin{aligned}\overline{Q_{i_1 i_2}}(\boldsymbol{S}) \left(\sum_{j=0}^{|\boldsymbol{X}_1|-1} \boldsymbol{w}_{1,i_1 j} \boldsymbol{P}_{1,j}(\boldsymbol{S}) + \sum_l \boldsymbol{b}_{1,i_1 l} \boldsymbol{R}_l \right) \left(\sum_{j=0}^{|\boldsymbol{X}_2|-1} \boldsymbol{w}_{2,i_2 j} \boldsymbol{P}_{2,j}(\boldsymbol{S}) + \sum_l \boldsymbol{b}_{2,i_2 l} \boldsymbol{R}_l \right) \\ + \sum_{\nu=1}^{2} \sum_{i=1}^{|\boldsymbol{Y}_\nu|} \overline{Q_{\nu,i}}(\boldsymbol{S}) \left(\sum_{j=0}^{|\boldsymbol{X}_\nu|-1} \boldsymbol{w}_{\nu,ij} \boldsymbol{P}_{\nu,j}(\boldsymbol{S}) + \sum_l \boldsymbol{b}_{\nu,il} \boldsymbol{R}_l \right) + \overline{Q_0}(\boldsymbol{S}) = 0\end{aligned} \tag{5}$$

as a polynomial in $\boldsymbol{S}$ and $\boldsymbol{R}$. We want to show that this implies $\boldsymbol{b}_{\nu,il} = 0$ for all $\nu \in \{1, 2\}$, all $1 \leq i \leq |\boldsymbol{Y}_\nu|$ and all l, since the representation returned by $\mathcal{E}$ will then be correct.

Assume for the moment that $\overline{Q_{i_1 i_2}}(\boldsymbol{S}) \neq 0$. We begin by proving that $\boldsymbol{b}_{\nu,i_\nu l} = 0$ for all $\nu \in \{1, 2\}$ and all l. Indeed, suppose this was not the case, and let $\bar{l}$ be an index such that $\boldsymbol{b}_{1,i_1\bar{l}} \neq 0$. From the term of degree two in $\boldsymbol{R}$ we obtain (1) $\overline{Q_{i_1 i_2}}(\boldsymbol{S}) \boldsymbol{b}_{1,i_1 l} \boldsymbol{b}_{2,i_2 l} = 0$ for all l, and (2) $\overline{Q_{i_1 i_2}}(\boldsymbol{S})(\boldsymbol{b}_{1,i_1 l} \boldsymbol{b}_{2,i_2 l'} + \boldsymbol{b}_{1,i_1 l'} \boldsymbol{b}_{2,i_2 l}) = 0$ for all $l \neq l'$. Then (1) gives $\boldsymbol{b}_{2,i_2\bar{l}} = 0$, and (2) then yields $\boldsymbol{b}_{2,i_2 l} = 0$ for all $l \neq \bar{l}$, i.e., $\boldsymbol{b}_{2,i_2 l} = 0$ for every l. The linear term in $\boldsymbol{R}_{\bar{l}}$ now becomes

$$\overline{Q_{i_1 i_2}}(\boldsymbol{S}) \boldsymbol{b}_{1,i_1\bar{l}} \sum_{j=0}^{|\boldsymbol{X}_2|-1} \boldsymbol{w}_{2,i_2 j} \boldsymbol{P}_{2,j}(\boldsymbol{S}) + \sum_{\nu=1}^{2} \sum_{i=1}^{|\boldsymbol{Y}_\nu|} \overline{Q_{\nu,i}}(\boldsymbol{S}) \boldsymbol{b}_{\nu,i\bar{l}} = 0\,,$$

which gives

$$\overline{Q_{i_1 i_2}}(\boldsymbol{S}) \sum_{j=0}^{|\boldsymbol{X}_2|-1} \boldsymbol{w}_{2,i_2 j} \boldsymbol{P}_{2,j}(\boldsymbol{S}) = -\sum_{\nu=1}^{2} \sum_{i=1}^{|\boldsymbol{Y}_\nu|} \frac{\boldsymbol{b}_{\nu,i\bar{l}}}{\boldsymbol{b}_{1,i_1\bar{l}}} \overline{Q_{\nu,i}}(\boldsymbol{S}) \,.$$

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

$$\begin{aligned}\left(\sum_{j=0}^{|\boldsymbol{X}_1|-1} \boldsymbol{w}_{1,i_1 j} \boldsymbol{P}_{1,j}(\boldsymbol{S})\right)\left(-\sum_{\nu=1}^{2} \sum_{i=1}^{|\boldsymbol{Y}_\nu|} \frac{\boldsymbol{b}_{\nu,i\bar{l}}}{\boldsymbol{b}_{1,i_1\bar{l}}} \overline{Q_{\nu,i}}(\boldsymbol{S})\right)& \\ + \sum_{\nu=1}^{2} \sum_{i=1}^{|\boldsymbol{Y}_\nu|} \overline{Q_{\nu,i}}(\boldsymbol{S}) \left(\sum_{j=0}^{|\boldsymbol{X}_\nu|-1} \boldsymbol{w}_{\nu,ij} \boldsymbol{P}_{\nu,j}(\boldsymbol{S})\right) + \overline{Q_0}(\boldsymbol{S}) = 0 \,,&\end{aligned}$$

which means that

$$\overline{Q_0}(\boldsymbol{S}) = \sum_{\nu=1}^{2} \sum_{i=1}^{|\boldsymbol{Y}_\nu|} \sum_{j=0}^{|\boldsymbol{X}_1|-1} \boldsymbol{w}_{1,i_1 j} \frac{\boldsymbol{b}_{\nu,i\bar{l}}}{\boldsymbol{b}_{1,i_1\bar{l}}} \overline{Q_{\nu,i}}(\boldsymbol{S}) \boldsymbol{P}_{1,j}(\boldsymbol{S}) - \sum_{\nu=1}^{2} \sum_{i=1}^{|\boldsymbol{Y}_\nu|} \sum_{j=0}^{|\boldsymbol{X}_\nu|-1} \boldsymbol{w}_{\nu,ij} \overline{Q_{\nu,i}}(\boldsymbol{S}) \boldsymbol{P}_{\nu,j}(\boldsymbol{S}) \,.$$

This, however, contradicts our assumption of $\overline{Q_0}$ not being in the linear span of $\overline{Q_{\nu,i}}\boldsymbol{P}_{\nu',j}$, from which we conclude that $\boldsymbol{b}_{1,i_1 l} = 0$ for every l. As a consequence, Equation (5) now becomes

$$\begin{aligned}\overline{Q_{i_1 i_2}}(\boldsymbol{S}) \left(\sum_{j=0}^{|\boldsymbol{X}_1|-1} \boldsymbol{w}_{1,i_1 j} \boldsymbol{P}_{1,j}(\boldsymbol{S})\right)\left(\sum_{j=0}^{|\boldsymbol{X}_2|-1} \boldsymbol{w}_{2,i_2 j} \boldsymbol{P}_{2,j}(\boldsymbol{S}) + \sum_{l} \boldsymbol{b}_{2,i_2 l} \boldsymbol{R}_l\right)& \\ + \sum_{\nu=1}^{2} \sum_{i=1}^{|\boldsymbol{Y}_\nu|} \overline{Q_{\nu,i}}(\boldsymbol{S}) \left(\sum_{j=0}^{|\boldsymbol{X}_\nu|-1} \boldsymbol{w}_{\nu,ij} \boldsymbol{P}_{\nu,j}(\boldsymbol{S}) + \sum_{l} \boldsymbol{b}_{\nu,il} \boldsymbol{R}_l\right) + \overline{Q_0}(\boldsymbol{S}) = 0 \,.&\end{aligned}$$

We can similarly show that $\boldsymbol{b}_{2,i_2 l} = 0$ for every l. Indeed, assume for the sake of contradiction that $\boldsymbol{b}_{2,i_2\bar{l}} \neq 0$ for some $\bar{l}$. The linear term in $\boldsymbol{R}_{\bar{l}}$ then is

$$\overline{Q_{i_1 i_2}}(\boldsymbol{S}) \boldsymbol{b}_{2,i_2\bar{l}} \sum_{j=0}^{|\boldsymbol{X}_1|-1} \boldsymbol{w}_{1,i_1 j} \boldsymbol{P}_{1,j}(\boldsymbol{S}) + \sum_{\nu=1}^{2} \sum_{i=1}^{|\boldsymbol{Y}_\nu|} \overline{Q_{\nu,i}}(\boldsymbol{S}) \boldsymbol{b}_{\nu,i\bar{l}} = 0 \,,$$

that is,

$$\overline{Q_{i_1 i_2}}(\boldsymbol{S}) \sum_{j=0}^{|\boldsymbol{X}_1|-1} \boldsymbol{w}_{1,i_1 j} \boldsymbol{P}_{1,j}(\boldsymbol{S}) = -\sum_{\nu=1}^{2} \sum_{i=1}^{|\boldsymbol{Y}_\nu|} \frac{\boldsymbol{b}_{\nu,i\bar{l}}}{\boldsymbol{b}_{2,i_2\bar{l}}} \overline{Q_{\nu,i}}(\boldsymbol{S}) \,.$$

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

$$\begin{aligned}\left(-\sum_{\nu=1}^{2} \sum_{i=1}^{|\boldsymbol{Y}_\nu|} \frac{\boldsymbol{b}_{\nu,i\bar{l}}}{\boldsymbol{b}_{2,i_2\bar{l}}} \overline{Q_{\nu,i}}(\boldsymbol{S})\right)\left(\sum_{j=0}^{|\boldsymbol{X}_2|-1} \boldsymbol{w}_{2,i_2 j} \boldsymbol{P}_{2,j}(\boldsymbol{S})\right) +& \\ \sum_{\nu=1}^{2} \sum_{i=1}^{|\boldsymbol{Y}_\nu|} \overline{Q_{\nu,i}}(\boldsymbol{S}) \left(\sum_{j=0}^{|\boldsymbol{X}_\nu|-1} \boldsymbol{w}_{\nu,ij} \boldsymbol{P}_{\nu,j}(\boldsymbol{S})\right) + \overline{Q_0}(\boldsymbol{S}) = 0 \,,&\end{aligned}$$

which means that

$$\overline{Q_0}(\boldsymbol{S}) = \sum_{\nu=1}^{2} \sum_{i=1}^{|\boldsymbol{Y}_\nu|} \sum_{j=0}^{|\boldsymbol{X}_2|-1} \boldsymbol{w}_{2,i_2 j} \frac{\boldsymbol{b}_{\nu,i\bar{l}}}{\boldsymbol{b}_{2,i_2\bar{l}}} \overline{Q_{\nu,i}}(\boldsymbol{S}) \boldsymbol{P}_{2,j}(\boldsymbol{S}) - \sum_{\nu=1}^{2} \sum_{i=1}^{|\boldsymbol{Y}_\nu|} \sum_{j=0}^{|\boldsymbol{X}_\nu|-1} \boldsymbol{w}_{\nu,ij} \overline{Q_{\nu,i}}(\boldsymbol{S}) \boldsymbol{P}_{\nu,j}(\boldsymbol{S}) \,.$$

This again contradicts our assumption of $\overline{Q_0}$ not being in the linear span of $\overline{Q_{\nu,i}}\boldsymbol{P}_{\nu',j}$, and thus $\boldsymbol{b}_{2,i_2 l} = 0$ for every l. Equation (5) then simplifies to

$$\overline{Q_{i_1 i_2}}(\boldsymbol{S})\left(\sum_{j=0}^{|\boldsymbol{X}_1|-1} \boldsymbol{w}_{1,i_1 j}\boldsymbol{P}_{1,j}(\boldsymbol{S})\right)\left(\sum_{j=0}^{|\boldsymbol{X}_2|-1} \boldsymbol{w}_{2,i_2 j}\boldsymbol{P}_{2,j}(\boldsymbol{S})\right)$$
$$+\sum_{\nu=1}^{2}\sum_{i=1}^{|\boldsymbol{Y}_\nu|}\overline{Q_{\nu,i}}(\boldsymbol{S})\left(\sum_{j=0}^{|\boldsymbol{X}_\nu|-1} \boldsymbol{w}_{\nu,ij}\boldsymbol{P}_{\nu,j}(\boldsymbol{S}) + \sum_l \boldsymbol{b}_{\nu,il}\boldsymbol{R}_l\right) + \overline{Q_0}(\boldsymbol{S}) = 0\,.$$

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

$$\sum_{\nu=1}^{2}\sum_{i=1}^{|\boldsymbol{Y}_\nu|}\overline{Q_{\nu,i}}(\boldsymbol{S})\boldsymbol{b}_{\nu,il} = 0\,. \tag{6}$$

Recall that, by assumption, polynomials $\overline{Q_{\nu,i}}$ are linearly independent, which means that $\boldsymbol{b}_{\nu,il} = 0$ for all $1 \le \nu \le 2$, all $1 \le i \le |\boldsymbol{Y}_\nu|$ and all l.

If on the other hand $\overline{Q_{i_1 i_2}}(\boldsymbol{S}) = 0$, then there are no terms of degree two in $\boldsymbol{R}$ in Equation (5). This means that we can jump directly to Equation (6) and conclude that $\boldsymbol{b}_{\nu,il} = 0$ for all $1 \le \nu \le 2$, all $1 \le i \le |\boldsymbol{Y}_\nu|$ and all l, since $\overline{Q_{\nu,i}}$ are linearly independent.

This proves that if $\mathcal{A}$ returns a valid output, then $\mathcal{E}$ returns an accurate representation of $\boldsymbol{v}$ in terms of $\boldsymbol{x}$, which means that $\Pr[\mathrm{G}_3] = 0$. Collecting all the terms above, we obtain

$$\begin{aligned}\mathrm{Adv}^{\mathrm{uk}}_{p,\mathbf{G},\mathcal{S},\mathcal{A},\mathcal{E}} &\le \frac{2(m+n+3q_{\mathsf{op}}+q_{\mathsf{H}}+2q_{\mathsf{e}}+1)^2\cdot d_P}{p} + \frac{d_Q d_P}{p} + \frac{2nd_P}{p}\\ &\le \mathcal{O}\left(\frac{(m+n+q_{\mathsf{op}}+q_{\mathsf{H}}+q_{\mathsf{e}}+d_Q)^2\cdot d_P}{p}\right).\end{aligned}$$

This concludes the proof. □

We now show that the specific knowledge assumptions considered in Section 4 all satisfy the condition stated in the theorem above (and the analogous theorem for simple groups proved as Theorem 7).

Corollary 1. *Let $d, p \in \mathbb{N}$ with p prime, and fix $\mathsf{G}, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_T \subseteq \{0,1\}^*$ with $|\mathsf{G}| = |\mathsf{G}_1| = |\mathsf{G}_2| = |\mathsf{G}_T| = p$. (1)* KEA1, KEA3*, and d-*PKE *hold in* GGM-H *with parameters (p, G). (2) d-*KZG*, d-*PKE*, and d-*GROTH16 *hold in* GBM3-H *with parameters $(p, \mathbf{G})$ (the latter for any class of first-stage algorithms $\mathfrak{A}$).*

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 (see Figure 11) is of the form considered in Theorem 3, and the polynomials $\overline{Q_{1,1}}(S) = 1$ and $\overline{Q_{1,2}}(S) = -S + c$ are linearly independent for every $c \in \mathbb{Z}_p$. There is no need to check the third condition of the theorem because Q has no degree-two term in $\boldsymbol{Y}$, and the requirement on the source is satisfied by definition.

For d-GROTH16, recall that the relation polynomial is given in Figure 12. Again, polynomial Q is of the form covered by Theorem 3, and $\overline{Q_{1,2}}(\boldsymbol{S}) = -\boldsymbol{S}_3\boldsymbol{S}_4^2$ is non-zero and thus linearly independent. To verify the third condition, recall that $\overline{Q}_0(\boldsymbol{S}) = -\boldsymbol{S}_1\boldsymbol{S}_2\boldsymbol{S}_3^2\boldsymbol{S}_4^2 - \sum_{i=0}^{\ell}\boldsymbol{c}_i\big(\boldsymbol{S}_2 U_i(\boldsymbol{S}_5)+\boldsymbol{S}_1 V_i(\boldsymbol{S}_5)+W_i(\boldsymbol{S}_5)\big)\boldsymbol{S}_3^2\boldsymbol{S}_4^2$. It is straightforward to see that $\boldsymbol{S}_1\boldsymbol{S}_2\boldsymbol{S}_3^2\boldsymbol{S}_4^2$ does not lie in the linear span of $\overline{Q_{\nu,i}}\boldsymbol{P}_{\nu',j}$, and neither does $\sum_{i=0}^{\ell}\boldsymbol{c}_i\big(\boldsymbol{S}_2 U_i(\boldsymbol{S}_5)+\boldsymbol{S}_1 V_i(\boldsymbol{S}_5)+W_i(\boldsymbol{S}_5)\big)\boldsymbol{S}_3^2\boldsymbol{S}_4^2$ contain such a term, so that the condition is verified. Again, the requirement on the source is satisfied by definition. □

7 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 GBM3-H hardness of UK, as the two models are formally incomparable for knowledge assumptions.

If we consider the classical definition of algebraic adversaries [FKL18], we can trivially build an extractor for every such adversary $\mathcal{A}$: output the scalar representation returned by $\mathcal{A}$ in the AGM as the linear relation between the outputs and the inputs. As mentioned, this justification does not take hashing into account, and thus we consider algebraic adversaries in the ABM3-H. In this model, the extractor above is no longer valid as it may output nonzero coefficients for hash values.

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 B, we include a proof of the hardness of linear UK in the AGM-H (i.e., for simple groups). As for relation polynomials Q of degree two in the output variables, in Theorem 5 we prove hardness of d-GROTH16 in the ABM3-H.

Theorem 4 (Linear UK holds in ABM3-H)**.** *Let* B *be a type-3 bilinear group scheme and* $d_P, d_Q \colon \mathbb{N} \to \mathbb{N}$ *be polynomials. Consider the class of PPT algorithms* $\mathfrak{A}$ *and the source* $\mathcal{S}$ *defined as follows:*

1. *For every* $\mathcal{A}_0 \in \mathfrak{A}$*, the relation polynomial* Q *returned by* $\mathcal{A}_0$ *is of the form*

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

2. *For every* $\mathcal{A}_0 \in \mathfrak{A}$*, every* $(Q, \boldsymbol{P}_\mu)$ *returned by* $\mathcal{A}_0$*, every* $\boldsymbol{c} \in \mathbb{Z}_p^{|\boldsymbol{C}|}$*, and every* $\nu \in \{1, 2\}$*, the polynomials* $\overline{Q_{\nu,i}}$*,* $1 \leq i \leq |\boldsymbol{Y}_\nu|$*, are linearly independent;*
3. *For every* $\mathcal{A}_0 \in \mathfrak{A}$ *and every* $(Q, \boldsymbol{P}_\mu)$ *returned by* $\mathcal{A}_0$*, every polynomial* P *in either* $\boldsymbol{P}_\mu$ *has total degree at most* d_P*, and* Q *has total degree at most* d_Q*;*
4. *For every* $\mathcal{A}_0 \in \mathfrak{A}$ *and every* $(Q, \boldsymbol{P}_\mu)$ *returned by* $\mathcal{A}_0$*,* $\mathcal{S}$ *samples* $\boldsymbol{s} \in \mathbb{Z}_p^k$ *at random and returns* $(\boldsymbol{P}_\mu(\boldsymbol{s}))$*.*

If (d_P, d_P)-DL *holds for* B*, then* UK *holds for* $(\mathrm{B}, \mathcal{S}, \mathfrak{A})$ *in the* ABM3-H*. More precisely, for every low-degree PPT adversary* $\mathcal{A}$ *with* $\mathcal{A}_0 \in \mathfrak{A}$*, there exist an extractor* $\mathcal{E}$ *and an adversary* $\mathcal{B}$ *against* (d_P, d_P)-DL*, both with approximately the same running time as* $\mathcal{A}$*, such that*

$$\mathrm{Adv}^{\mathrm{uk}}_{\mathrm{B},\mathcal{S},\mathcal{A},\mathcal{E}}(\lambda) \leq \left(1 - \frac{d_P(\lambda) d_Q(\lambda)}{2^{\lambda-1} - 1}\right)^{-1} \cdot \mathrm{Adv}^{(d_P,d_P)\text{-dl}}_{\mathrm{B},\mathcal{B}}(\lambda)\,. \tag{7}$$

Here, μ *in an index ranging over* $\{1, 2, T\}$*,* k *is an upper bound on the number of variables of every polynomial* P *in* $\boldsymbol{P}_\mu$*, and we let* $\overline{Q_{\nu,i}}(\boldsymbol{S}) \coloneqq Q_{\nu,i}(\boldsymbol{P}_\mu(\boldsymbol{S}), \boldsymbol{c})$ *for* $\nu \in \{1, 2\}$*, where we set* $\boldsymbol{P}_{\nu,0}(\boldsymbol{S}) \coloneqq 1$*.*

Proof overview. Fix an adversary $\mathcal{A}$ in the UK game as in the statement of the theorem, and define an extractor $\mathcal{E}$ as in Figure 22 (top). This extractor essentially re-runs $\mathcal{A}$ on its view to obtain $\mathcal{A}$'s output $(\boldsymbol{w}_\nu, \boldsymbol{v}_\nu, \boldsymbol{c})$. Recall that this means that $\mathcal{A}$ encodes group elements $[\boldsymbol{y}_{\nu,i}]_\nu = \prod_{j=0}^{|\boldsymbol{X}_\nu|-1} [\boldsymbol{w}_{\nu,ij} \boldsymbol{x}_{\nu,j}]_\nu \cdot \prod_l [\boldsymbol{v}_{\nu,il} \boldsymbol{h}_{\nu,l}]_\nu$ for $\nu \in \{1, 2\}$, where $[\boldsymbol{x}_\nu]_\nu$ and $[\boldsymbol{h}_\nu]_\nu$ are the vectors 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 $(\boldsymbol{w}_1, \boldsymbol{w}_2)$. Clearly, extractor $\mathcal{E}$ will be correct if $\boldsymbol{v}_1 = \boldsymbol{v}_2 = 0$ in the representation returned by $\mathcal{A}$.

We then show that if $\mathcal{A}$ returns a valid output and (d_P, d_P)-DL holds for B, this will likely be the case. To that end, consider the adversary $\mathcal{B}$ playing the (d_P, d_P)-DL game for B defined in Figure 22 (bottom). In essence, $\mathcal{B}$ runs $\mathcal{A}$ and simulates the UK

```
Extractor $\mathcal{E}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\mathsf{trace}(\mathcal{A}))$:
parse $\mathsf{trace}(\mathcal{A}) = (r_{\mathcal{A}}, \gamma, [\boldsymbol{x}_\mu]_\mu, [\boldsymbol{h}_\mu]_\mu)$; $o_1, o_2, o_T \leftarrow 0$
$(Q, \boldsymbol{P}_\mu) \leftarrow \mathcal{A}_0^{\overline{\mathsf{H}}_1,\overline{\mathsf{H}}_2,\overline{\mathsf{H}}_T}(\gamma; r_{\mathcal{A}})$
$(\boldsymbol{w}_\nu, \boldsymbol{v}_\nu, \boldsymbol{c}) \leftarrow \mathcal{A}_1^{\overline{\mathsf{H}}_1,\overline{\mathsf{H}}_2,\overline{\mathsf{H}}_T}(\gamma, [\boldsymbol{x}_\mu]_\mu; r_{\mathcal{A}})$
  // $\mathcal{A}$ encodes group elements
  // $[\boldsymbol{y}_{\nu,i}]_\nu = \prod_{j=0}^{|\boldsymbol{X}_\nu|-1}[\boldsymbol{w}_{\nu,ij}\boldsymbol{x}_{\nu,j}]_\nu \cdot \prod_l [\boldsymbol{v}_{\nu,il}\boldsymbol{h}_{\nu,l}]_\nu$
return $(\boldsymbol{w}_1, \boldsymbol{w}_2)$

Oracle $\overline{\mathsf{H}}_\mu(m)$:
$o_\mu \leftarrow o_\mu + 1$
return $[\boldsymbol{h}_{\mu,o_\mu}]_\mu$
```

```
Adversary $\mathcal{B}(\gamma, [t]_1, \ldots, [t^{d_P(\lambda)}]_1, [t]_2, \ldots, [t^{d_P(\lambda)}]_2)$:
$o_1, o_2 \leftarrow 0$; $U_1, U_2, U_T \leftarrow [\,]$; $\mathsf{S} \leftarrow \emptyset$; $r_{\mathcal{A}} \twoheadleftarrow \mathcal{R}_{\mathcal{A}}(\lambda)$
for $i = 0$ to $d_P(\lambda)$ do $[t^i]_T \leftarrow e([t^i]_1, [1]_2)$
$(Q, \boldsymbol{P}_\mu) \leftarrow \mathcal{A}_0^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\gamma; r_{\mathcal{A}})$; $\boldsymbol{\rho} \twoheadleftarrow \mathbb{Z}_p^k$; $\boldsymbol{\sigma} \twoheadleftarrow \mathbb{Z}_p^{*k}$
for $\mu \in \{1, 2, T\}$ do
  $\boldsymbol{X}_\mu(T) \leftarrow \boldsymbol{P}_\mu(\boldsymbol{\rho} + \boldsymbol{\sigma}T)$; $[\boldsymbol{x}_\mu]_\mu \leftarrow [\boldsymbol{X}_\mu(t)]_\mu$
$\boldsymbol{X}_{1,0}(T), \boldsymbol{X}_{2,0}(T) \leftarrow 1$
$(\boldsymbol{w}_\nu, \boldsymbol{v}_\nu, \boldsymbol{c}) \leftarrow \mathcal{A}_1^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\gamma, [\boldsymbol{x}_\mu]_\mu; r_{\mathcal{A}})$
  // $\mathcal{A}$ encodes group elements
  // $[\boldsymbol{y}_{\nu,i}]_\nu = \prod_{j=0}^{|\boldsymbol{X}_\nu|-1}[\boldsymbol{w}_{\nu,ij}\boldsymbol{x}_j]_\nu \cdot \prod_l [\boldsymbol{v}_{\nu,il}\boldsymbol{h}_{\nu,l}]_\nu$
for $\nu \in \{1, 2\}$ do
  for $i = 1$ to $|\boldsymbol{Y}_\nu|$ do
    $\boldsymbol{Y}_{\nu,i}(T) \leftarrow \sum_{j=0}^{|\boldsymbol{X}_\nu|-1} \boldsymbol{w}_{\nu,ij}\boldsymbol{X}_{\nu,j}(T) + \sum_l \boldsymbol{v}_{\nu,il}H_{\nu,l}(T)$
$Q'(T) \leftarrow Q(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_T, \boldsymbol{Y}_1, \boldsymbol{Y}_2, \boldsymbol{c})$
if $(Q'(T) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{Berlekamp}(Q', p)$
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
  $o_\nu \leftarrow o_\nu + 1$
  $\alpha_{\nu,o_\nu} \twoheadleftarrow \mathbb{Z}_p$
  $\beta_{\nu,o_\nu} \twoheadleftarrow \mathbb{Z}_p^*$
  $H_{\nu,o_\nu}(T)$
    $\leftarrow \alpha_{\nu,o_\nu} + \beta_{\nu,o_\nu}T$
  $U_\nu[m] \leftarrow [H_{\nu,o_\nu}(t)]_\nu$
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 $U_T[m]$
```

Figure 22: *Top:* Extractor $\mathcal{E}$ for the algebraic adversary $\mathcal{A}$ in the UK game. *Bottom:* Adversary $\mathcal{B}$ against (d_P, d_P)-DL. In all figures, μ and ν range over $\{1, 2, T\}$ and $\{1, 2\}$, respectively, and k is an upper bound on the number of variables appearing in any polynomial P in $\boldsymbol{P}_\mu$.

game. When preparing the group element inputs and answering hash queries, $\mathcal{B}$ embeds the (d_P, d_P)-DL instance it is tasked with solving. Note that this is possible because $\mathcal{B}$ is given the power-DL challenge up to power $d_P(\lambda)$ in both groups. By construction, if $\mathcal{A}$ returns an output that satisfies Q, then t is a root of the polynomial $Q'(T)$ defined by $\mathcal{B}$. This means that $\mathcal{B}$ will be able to find t by inspecting the roots of Q' whenever $Q'(T) \neq 0$. We show that the latter happens with high probability if $\boldsymbol{v}_\nu \neq 0$ for some $\nu \in \{1, 2\}$, which means $\boldsymbol{v}_\nu$ must vanish if (d_P, d_P)-DL holds for B.

Proof. We now formally implement the intuition presented in the proof overview above. Fix an adversary $\mathcal{A}$ as above, and define an extractor $\mathcal{E}$ and an adversary $\mathcal{B}$ as shown in Figure 22. We now show how to use adversary $\mathcal{B}$ to prove Inequality (7) for $\mathcal{A}$ and $\mathcal{E}$. To that end, consider the following sequence of games (the formal description of which can be found in Figure 23):

G$_0$: This is the original (d_P, d_P)-DL game for B run with adversary $\mathcal{B}$.

G$_1$: This game proceeds as G$_0$, but performs variable substitutions $(\boldsymbol{\rho}', \boldsymbol{\sigma}') = (\boldsymbol{\rho} + \boldsymbol{\sigma}t, \boldsymbol{\sigma})$ and $(\alpha'_{\nu,l}, \beta'_{\nu,l}) = (\alpha_{\nu,l} + \beta_{\nu,l}t, \beta_{\nu,l})$ in polynomials $\boldsymbol{X}_\mu$ and $H_{\nu,l}$. More precisely, polynomials $\boldsymbol{X}_\mu(T)$ are now defined as $\boldsymbol{X}_\mu(T) \leftarrow \boldsymbol{P}_\mu(\boldsymbol{\rho}' + \boldsymbol{\sigma}'(T - t))$ for random $\boldsymbol{\rho}'$ and invertible $\boldsymbol{\sigma}'$. Similarly, upon a query m to H_ν, game G$_2$ samples random $\alpha'_{\nu,l}$ and invertible $\beta'_{\nu,l}$, and sets $H_{\nu,l}(T) \leftarrow \alpha'_{\nu,l} + \beta'_{\nu,l}(T - t)$. Inputs $[\boldsymbol{x}_\mu]_\mu$ and hash replies $U_\nu[m]$ are still computed as $[\boldsymbol{X}_\mu(t)]_\mu = [\boldsymbol{P}_\mu(\boldsymbol{\rho}')]_\mu$ and $[H_{\nu,l}(t)]_\nu = [\alpha'_{\nu,l}]_\nu$, respectively.

```
Game $\mathrm{G}_0(\lambda)$:
$\gamma \twoheadleftarrow \mathrm{B}(1^\lambda)$, $t \twoheadleftarrow \mathbb{Z}_p$; $o_1, o_2 \leftarrow 0$; $U_1, U_2, U_T \leftarrow [\,]$
$\mathsf{S} \leftarrow \emptyset$; $r_\mathcal{A} \twoheadleftarrow \mathcal{R}_\mathcal{A}(\lambda)$
for $i = 0$ to $d_P(\lambda)$ do $[t^i]_T \leftarrow e([t^i]_1, [1]_2)$
$(Q, \boldsymbol{P}_\mu) \leftarrow \mathcal{A}_0^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\gamma; r_\mathcal{A})$; $\boldsymbol{\rho} \twoheadleftarrow \mathbb{Z}_p^k$; $\boldsymbol{\sigma} \twoheadleftarrow \mathbb{Z}_p^{*k}$
for $\mu \in \{1, 2, T\}$ do $\boldsymbol{X}_\mu(T) \leftarrow \boldsymbol{P}_\mu(\boldsymbol{\rho} + \boldsymbol{\sigma}T)$; $[\boldsymbol{x}_\mu]_\mu \leftarrow [\boldsymbol{X}_\mu(t)]_\mu$
$\boldsymbol{X}_{1,0}(T), \boldsymbol{X}_{2,0}(T) \leftarrow 1$; $(\boldsymbol{w}_\nu, \boldsymbol{v}_\nu, \boldsymbol{c}) \leftarrow \mathcal{A}_1^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\gamma, [\boldsymbol{x}_\mu]_\mu; r_\mathcal{A})$
for $\nu \in \{1, 2\}$ do
  for $i = 1$ to $|\boldsymbol{Y}_\nu|$ do
    $\boldsymbol{Y}_{\nu,i}(T) \leftarrow \sum_{j=0}^{|\boldsymbol{X}_\nu|-1} \boldsymbol{w}_{\nu,ij} \boldsymbol{X}_{\nu,j}(T) + \sum_l \boldsymbol{v}_{\nu,il} H_{\nu,l}(T)$
$Q'(T) \leftarrow Q(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_T, \boldsymbol{Y}_1, \boldsymbol{Y}_2, \boldsymbol{c})$
if $(Q'(T) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{Berlekamp}(Q', p)$
$t' \leftarrow 0$; for $z \in \mathsf{S}$ do if $([z]_1 = [t]_1)$ then return $t' \leftarrow z$; break
return $(t = t')$
```

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

```
Game $\mathrm{G}_1(\lambda)$:
$\gamma \twoheadleftarrow \mathrm{B}(1^\lambda)$, $t \twoheadleftarrow \mathbb{Z}_p$; $o_1, o_2 \leftarrow 0$; $U_1, U_2, U_T \leftarrow [\,]$
$\mathsf{S} \leftarrow \emptyset$; $r_\mathcal{A} \twoheadleftarrow \mathcal{R}_\mathcal{A}(\lambda)$
for $i = 0$ to $d_P(\lambda)$ do $[t^i]_T \leftarrow e([t^i]_1, [1]_2)$
$(Q, \boldsymbol{P}_\mu) \leftarrow \mathcal{A}_0^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\gamma; r_\mathcal{A})$; $\boldsymbol{\rho}' \twoheadleftarrow \mathbb{Z}_p^k$; $\boldsymbol{\sigma}' \twoheadleftarrow \mathbb{Z}_p^{*k}$
for $\mu \in \{1, 2, T\}$ do $\boldsymbol{X}_\mu(T) \leftarrow \boldsymbol{P}_\mu(\boldsymbol{\rho}' + \boldsymbol{\sigma}'(T - t))$; $[\boldsymbol{x}_\mu]_\mu \leftarrow [\boldsymbol{X}_\mu(t)]_\mu$
$\boldsymbol{X}_{1,0}(T), \boldsymbol{X}_{2,0}(T) \leftarrow 1$; $(\boldsymbol{w}_\nu, \boldsymbol{v}_\nu, \boldsymbol{c}) \leftarrow \mathcal{A}_1^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\gamma, [\boldsymbol{x}_\mu]_\mu; r_\mathcal{A})$
for $\nu \in \{1, 2\}$ do
  for $i = 1$ to $|\boldsymbol{Y}_\nu|$ do
    $\boldsymbol{Y}_{\nu,i}(T) \leftarrow \sum_{j=0}^{|\boldsymbol{X}_\nu|-1} \boldsymbol{w}_{\nu,ij} \boldsymbol{X}_{\nu,j}(T) + \sum_l \boldsymbol{v}_{\nu,il} H_{\nu,l}(T)$
$Q'(T) \leftarrow Q(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_T, \boldsymbol{Y}_1, \boldsymbol{Y}_2, \boldsymbol{c})$
if $(Q'(T) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{Berlekamp}(Q', p)$
$t' \leftarrow 0$; for $z \in \mathsf{S}$ do if $([z]_1 = [t]_1)$ then return $t' \leftarrow z$; break
return $(t = t')$
```

```
Oracle $\mathsf{H}_\nu(m)$:
if $(m \notin \mathrm{Dom}(U_\nu))$ then
  $o_\nu \leftarrow o_\nu + 1$
  $\alpha'_{\nu,o_\nu} \twoheadleftarrow \mathbb{Z}_p$; $\beta'_{\nu,o_\nu} \twoheadleftarrow \mathbb{Z}_p^*$
  $H_{\nu,o_\nu}(T)$
    $\leftarrow \alpha'_{\nu,o_\nu} + \beta'_{\nu,o_\nu}(T - t)$
  $U_\nu[m] \leftarrow [H_{\nu,o_\nu}(t)]_\nu$
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 $U_T[m]$
```

```
Game $\mathrm{G}_2(\lambda)$:
$\gamma \twoheadleftarrow \mathrm{B}(1^\lambda)$, $t \twoheadleftarrow \mathbb{Z}_p$; $o_1, o_2 \leftarrow 0$; $U_1, U_2, U_T \leftarrow [\,]$
$\mathsf{S} \leftarrow \emptyset$; $r_\mathcal{A} \twoheadleftarrow \mathcal{R}_\mathcal{A}(\lambda)$
for $i = 0$ to $d_P(\lambda)$ do $[t^i]_T \leftarrow e([t^i]_1, [1]_2)$
$(Q, \boldsymbol{P}_\mu) \leftarrow \mathcal{A}_0^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\gamma; r_\mathcal{A})$; $\boldsymbol{\rho}' \twoheadleftarrow \mathbb{Z}_p^k$
for $\mu \in \{1, 2, T\}$ do
  $\boldsymbol{X}_\mu(T, \boldsymbol{\Sigma}') \leftarrow \boldsymbol{P}_\mu(\boldsymbol{\rho}' + \boldsymbol{\Sigma}'(T - t))$; $[\boldsymbol{x}_\mu]_\mu \leftarrow [\boldsymbol{X}_\mu(t, \boldsymbol{\Sigma}')]_\mu$
$\boldsymbol{X}_{1,0}(T), \boldsymbol{X}_{2,0}(T) \leftarrow 1$; $(\boldsymbol{w}_\nu, \boldsymbol{v}_\nu, \boldsymbol{c}) \leftarrow \mathcal{A}_1^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\gamma, [\boldsymbol{x}_\mu]_\mu; r_\mathcal{A})$
for $\nu \in \{1, 2\}$ do
  for $i = 1$ to $|\boldsymbol{Y}_\nu|$ do
    $\boldsymbol{Y}_{\nu,i}(T, \boldsymbol{\Sigma}', \boldsymbol{B}'_\nu) \leftarrow$
      $\sum_{j=0}^{|\boldsymbol{X}_\nu|-1} \boldsymbol{w}_{\nu,ij} \boldsymbol{X}_{\nu,j}(T, \boldsymbol{\Sigma}') + \sum_l \boldsymbol{v}_{\nu,il} H_{\nu,l}(T, \boldsymbol{B}'_{\nu,l})$
$Q''(T, \boldsymbol{\Sigma}', \boldsymbol{B}'_1, \boldsymbol{B}'_2) \leftarrow Q(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_T, \boldsymbol{Y}_1, \boldsymbol{Y}_2, \boldsymbol{c})$
$\boldsymbol{\sigma}' \twoheadleftarrow \mathbb{Z}_p^{*k}$; $\boldsymbol{\beta}'_1 \twoheadleftarrow \mathbb{Z}_p^{*o_1}$; $\boldsymbol{\beta}'_2 \twoheadleftarrow \mathbb{Z}_p^{*o_2}$; $Q'(T) \leftarrow Q''(T, \boldsymbol{\sigma}', \boldsymbol{\beta}'_1, \boldsymbol{\beta}'_2)$
if $(Q'(T) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{Berlekamp}(Q', p)$
$t' \leftarrow 0$; for $z \in \mathsf{S}$ do if $([z]_1 = [t]_1)$ then return $t' \leftarrow z$; break
return $(t = t')$
```

```
Oracle $\mathsf{H}_\nu(m)$:
if $(m \notin \mathrm{Dom}(U_\nu))$ then
  $o_\nu \leftarrow o_\nu + 1$; $\alpha'_{\nu,o_\nu} \twoheadleftarrow \mathbb{Z}_p$
  $H_{\nu,o_\nu}(T, \boldsymbol{B}'_\nu)$
    $\leftarrow \alpha'_{\nu,o_\nu} + \boldsymbol{B}'_{\nu,o_\nu}(T - t)$
  $U_\nu[m] \leftarrow [H_{\nu,o_\nu}(t, \boldsymbol{B}'_\nu)]_\nu$
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 $U_T[m]$
```

Figure 23: Code of the intermediate games in the proof of Inequality (7). In all figures, μ and ν are indices ranging over $\{1, 2, T\}$ and $\{1, 2\}$, respectively, and k is an upper bound on the number of variables appearing in any polynomial P in $\boldsymbol{P}_\mu$.

G_2: This game proceeds as G_1, but the polynomials $\boldsymbol{X}_\mu$ and $H_{\nu,l}$ are now defined as $\boldsymbol{X}_\mu(T, \boldsymbol{\Sigma}') \leftarrow \boldsymbol{P}_\mu(\boldsymbol{\rho}' + \boldsymbol{\Sigma}'(T - t))$ and $H_{\nu,l}(T, \boldsymbol{B}'_\nu) \leftarrow \alpha'_{\nu,l} + \boldsymbol{B}'_{\nu,l}(T - t)$, where $\boldsymbol{\Sigma}'$ is a new vector of variables and $\boldsymbol{B}'_{\nu,l}$ 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 samples random $\boldsymbol{\sigma}'$ and invertible $\boldsymbol{\beta}'_\nu$, sets $Q'(T) \leftarrow Q''(T, \boldsymbol{\sigma}', \boldsymbol{\beta}'_1, \boldsymbol{\beta}'_2)$, and checks if $Q'(T) = 0$. From here on, game G_2 proceeds as G_1.

We now argue that subsequent games have identical success probabilities.

$\mathrm{G}_0 \rightsquigarrow \mathrm{G}_1$. Observe that for every fixed $\lambda \in \mathbb{N}$, γ returned by $\mathrm{B}(1^\lambda)$, $t \in \mathbb{Z}_p$, and randomness $r_\mathcal{A}$ returned by $\mathcal{R}_\mathcal{A}(\lambda)$, the random variates $\boldsymbol{\rho}'$, $\boldsymbol{\sigma}'$, $\alpha'_{\nu,l}$ and $\beta'_{\nu,l}$ in G_1 are related to the random variates $\boldsymbol{\rho}$, $\boldsymbol{\sigma}$, $\alpha_{\nu,l}$ and $\beta_{\nu,l}$ in G_0 via the transformation $\mathrm{diag}(\begin{smallmatrix} 1 & t \\ 0 & 1 \end{smallmatrix})$, which is invertible. Consequently, $\Pr[\mathrm{G}_0] = \Pr[\mathrm{G}_1]$, since there is a one-to-one correspondence between the random variables in the two games.

$\mathrm{G}_1 \rightsquigarrow \mathrm{G}_2$. Notice that $\mathcal{A}$ is oblivious to the changes to polynomials $\boldsymbol{X}_\mu$ and $H_{\nu,l}$, so the simulation of $\mathcal{A}$ is identical in both games. Indeed, in both games inputs to $\mathcal{A}$ and hash replies are computed in the same way. After running $\mathcal{A}$, G_2 derives the same polynomial Q' computed in G_1 by substituting random $\boldsymbol{\sigma}'$, $\boldsymbol{\beta}'_1$ and $\boldsymbol{\beta}'_2$ into Q'', so the winning condition is again the same in both games. Therefore, $\Pr[\mathrm{G}_1] = \Pr[\mathrm{G}_2]$.

We conclude the proof by studying the winning probability in G_2. First, notice that in this game adversary $\mathcal{A}$ plays the UK game, since the inputs of $\mathcal{A}$ are obtained by evaluating $\boldsymbol{P}_\mu$ at random points and hash replies are random group elements. Now for any $\lambda \in \mathbb{N}$, γ returned by $\mathrm{B}(1^\lambda)$, $t \in \mathbb{Z}_p$, randomness $r_\mathcal{A}$ returned by $\mathcal{R}_\mathcal{A}(\lambda)$, and vectors $\boldsymbol{\rho}'$, $\boldsymbol{\alpha}'_\nu$ and $\boldsymbol{\alpha}$ in $\mathbb{Z}_p$, denote by $\mathrm{G}' := \mathrm{G}'(\lambda, \gamma, t, r_\mathcal{A}, \boldsymbol{\rho}', \boldsymbol{\alpha}'_\nu, \boldsymbol{\alpha})$ the game $\mathrm{G}_2(\lambda)$ with these random choices fixed. Then $\Pr[\mathrm{G}_2(\lambda)] = \sum_{(\gamma, t, r_\mathcal{A}, \boldsymbol{\rho}', \boldsymbol{\alpha}'_\nu, \boldsymbol{\alpha})} \Pr[\mathrm{G}'] \Pr[\gamma, t, r_\mathcal{A}, \boldsymbol{\rho}', \boldsymbol{\alpha}'_\nu, \boldsymbol{\alpha}]$, where $\Pr[\gamma, t, r_\mathcal{A}, \boldsymbol{\rho}', \boldsymbol{\alpha}'_\nu, \boldsymbol{\alpha}]$ denotes the probability that such a tuple is drawn in $\mathrm{G}_2(\lambda)$, and the sum extends over all $(\gamma, t, r_\mathcal{A}, \boldsymbol{\rho}', \boldsymbol{\alpha}'_\nu, \boldsymbol{\alpha})$ such that $\Pr[\gamma, t, r_\mathcal{A}, \boldsymbol{\rho}', \boldsymbol{\alpha}'_\nu, \boldsymbol{\alpha}] \neq 0$.

Now consider the set X of all $(\gamma, t, r_\mathcal{A}, \boldsymbol{\rho}', \boldsymbol{\alpha}'_\nu, \boldsymbol{\alpha})$ in the sum above such that $\mathcal{A}$ returns $(Q, \boldsymbol{P}_\mu)$ and $(\boldsymbol{w}_\nu, \boldsymbol{v}_\nu, \boldsymbol{c})$ for which the relation polynomial in UK is satisfied and extractor $\mathcal{E}$ fails to compute a correct representation of the outputs. Notice that

$$\sum_{(\gamma, t, r_\mathcal{A}, \boldsymbol{\rho}', \boldsymbol{\alpha}'_\nu, \boldsymbol{\alpha}) \in \mathsf{X}} \Pr[\gamma, t, r_\mathcal{A}, \boldsymbol{\rho}', \boldsymbol{\alpha}'_\nu, \boldsymbol{\alpha}] = \mathrm{Adv}^{\mathrm{uk}}_{\mathrm{B}, \mathcal{S}, \mathcal{A}, \mathcal{E}}(\lambda)\,.$$

We claim that for any $(\gamma, t, r_\mathcal{A}, \boldsymbol{\rho}', \boldsymbol{\alpha}'_\nu, \boldsymbol{\alpha}) \in \mathsf{X}$, $\Pr[\mathrm{G}'] \geq 1 - d_P(\lambda) d_Q(\lambda) / (2^{\lambda - 1} - 1)$. Indeed, fix any $(\gamma, t, r_\mathcal{A}, \boldsymbol{\rho}', \boldsymbol{\alpha}'_\nu, \boldsymbol{\alpha}) \in \mathsf{X}$. Since $\mathcal{E}$ fails to return a correct representation of the output of $\mathcal{A}$, it must be $\boldsymbol{v}_1 \neq 0$ or $\boldsymbol{v}_2 \neq 0$, i.e., there exist $\nu^* \in \{1, 2\}$, $1 \leq i^* \leq |\boldsymbol{Y}_{\nu^*}|$ and l^* such that $\boldsymbol{v}_{\nu^*, i^* l^*} \neq 0$. We now claim that the polynomial $Q''(T, \boldsymbol{\Sigma}', \boldsymbol{B}'_1, \boldsymbol{B}'_2)$ constructed in G_2 after running $\mathcal{A}$ is not identically zero with overwhelming probability. Indeed, consider the polynomial

$$\begin{aligned} R(\boldsymbol{S}, \boldsymbol{H}_1, \boldsymbol{H}_2) &:= Q\Big(\boldsymbol{P}_\mu(\boldsymbol{S}), \textstyle\sum_{j=0}^{|\boldsymbol{X}_\nu|-1} \boldsymbol{w}_{\nu,ij} \boldsymbol{P}_{\nu,j}(\boldsymbol{S}) + \sum_l \boldsymbol{v}_{\nu,il} \boldsymbol{H}_{\nu,l}, \boldsymbol{c}\Big) \\ &= \sum_{\nu=1}^{2} \sum_{i=1}^{|\boldsymbol{Y}_\nu|} Q_{\nu,i}(\boldsymbol{P}_\mu(\boldsymbol{S}), \boldsymbol{c}) \Big(\textstyle\sum_{j=0}^{|\boldsymbol{X}_\nu|-1} \boldsymbol{w}_{\nu,ij} \boldsymbol{P}_{\nu,j}(\boldsymbol{S}) + \sum_l \boldsymbol{v}_{\nu,il} \boldsymbol{H}_{\nu,l}\Big) + Q_0(\boldsymbol{P}_\mu(\boldsymbol{S}), \boldsymbol{c})\,. \end{aligned}$$

Polynomial R is of total degree at most $d_P(\lambda) d_Q(\lambda)$ and not identically zero, because the coefficient of $\boldsymbol{H}_{\nu^*, l^*}$ is $\sum_{i=1}^{|\boldsymbol{Y}_{\nu^*}|} \overline{Q_{\nu^*,i}} \boldsymbol{v}_{\nu^*, il^*}$, which is non-zero since the polynomials $\overline{Q_{\nu^*,i}}$ are assumed to be linearly independent and $\boldsymbol{v}_{\nu^*, i^* l^*} \neq 0$. Now notice that

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

which again is non-zero by Lemma 2 and of degree in T at most $d_P(\lambda) d_Q(\lambda)$. Moreover, by Lemma 2, the leading coefficient in T of $Q''(T, \boldsymbol{\Sigma}', \boldsymbol{B}'_1, \boldsymbol{B}'_2)$ is a polynomial in $\boldsymbol{\Sigma}', \boldsymbol{B}'_1, \boldsymbol{B}'_2$ of total degree at most $d_P(\lambda) d_Q(\lambda)$, which for random invertible $\boldsymbol{\sigma}'$ and $\boldsymbol{\beta}'_\nu$ will be zero with probability at most $d_P(\lambda) d_Q(\lambda) / (2^{\lambda - 1} - 1)$ by Lemma 1. Thus, with probability at least $1 - d_P(\lambda) d_Q(\lambda) / (2^{\lambda - 1} - 1)$, $Q'(T) \neq 0$ in G'. We conclude by observing that whenever this happens, game G' will return 1, because t is a root of $Q'(T)$ by construction, and will therefore be found by inspecting its roots. This means

$$\begin{aligned}\mathrm{Adv}^{(d_P,d_P)\text{-dl}}_{\mathrm{B},\mathcal{B}}(\lambda) = \Pr[\mathrm{G}_0(\lambda)] = \Pr[\mathrm{G}_2(\lambda)] = \sum_{(\gamma,t,r_{\mathcal{A}},\boldsymbol{\rho}',\boldsymbol{\alpha}'_\nu,\boldsymbol{\alpha})} \Pr[\mathrm{G}'] \Pr[\gamma,t,r_{\mathcal{A}},\boldsymbol{\rho}',\boldsymbol{\alpha}'_\nu,\boldsymbol{\alpha}] \\ \geq \sum_{(\gamma,t,r_{\mathcal{A}},\boldsymbol{\rho}',\boldsymbol{\alpha}'_\nu,\boldsymbol{\alpha})\in \mathsf{X}} \Pr[\mathrm{G}'] \Pr[\gamma,t,r_{\mathcal{A}},\boldsymbol{\rho}',\boldsymbol{\alpha}'_\nu,\boldsymbol{\alpha}] \geq \left(1 - \frac{d_P(\lambda) d_Q(\lambda)}{2^{\lambda-1}-1}\right) \cdot \mathrm{Adv}^{\mathrm{uk}}_{\mathrm{B},\mathcal{A},\mathcal{E}}(\lambda)\,,\end{aligned}$$

which concludes the proof. □

Our requirements from the polynomials in the theorem above are identical to those needed for the linear case of Theorem 3 (and those needed in simple groups in Theorem 7). 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* DL *holds in* Γ*, then* KEA1 *holds in* Γ *in the* AGM-H*. (1b) If* 2-DL *holds in* Γ*, then* KEA3 *holds in* Γ *in the* AGM-H*. (1c) If* $(d+1)$-DL *holds in* Γ*, then* d-PKE *holds in* Γ *in the* AGM-H*.*

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

We conclude this section by proving the following theorem, which establishes the hardness of d-GROTH16 in the ABM3-H.

Theorem 5 (d-GROTH16 holds in ABM3-H)**.** *Let* B *be a type-3 bilinear group scheme,* $d\colon \mathbb{N}\to\mathbb{N}$ *a polynomial, and* $q(\lambda) := \max(3, d(\lambda)+1, 2d(\lambda)-1)$ *for every* $\lambda \in \mathbb{N}$*. If* (q,q)-DL *holds for* B*, then* d-GROTH16 *holds for* $(\mathrm{B},\mathfrak{A})$ *in the* ABM3-H *for any class of first-stage algorithms* $\mathfrak{A}$*. More precisely, for every PPT algebraic adversary* $\mathcal{A}$ *against* d-GROTH16*, there exist an extractor* $\mathcal{E}$ *and an adversary* $\mathcal{B}$ *against* (q,q)-DL*, both with approximately the same running time as* $\mathcal{A}$*, such that*

$$\mathrm{Adv}^{d\text{-groth16}}_{\mathrm{B},\mathcal{A},\mathcal{E}}(\lambda) \leq \left(1 - \frac{2q(\lambda)}{2^{\lambda-1}-1}\right)^{-1} \cdot \mathrm{Adv}^{(q,q)\text{-dl}}_{\mathrm{B},\mathcal{B}}(\lambda)\,. \tag{8}$$

Proof overview. Fix an adversary $\mathcal{A}$ in the d-GROTH16 game as in the statement of the theorem, and define an extractor $\mathcal{E}$ as in Figure 24 (top). This extractor essentially reruns $\mathcal{A}$ on its view to obtain $\mathcal{A}$'s output. The extractor then simply ignores the coefficients pertaining to the hash values and returns those associated with the input group elements. Clearly, extractor $\mathcal{E}$ will be correct if the coefficients of the hash values were zero in the representation returned by $\mathcal{A}$.

We then show that if $\mathcal{A}$ returns a valid output and (q,q)-DL holds for B, this will likely be the case. To that end, consider the adversary $\mathcal{B}$ playing the (q,q)-DL game for B defined in Figure 24 (bottom) (the polynomials $\boldsymbol{P}_1$, $\boldsymbol{P}_2$ and Q are as in Figure 12). In essence, $\mathcal{B}$ runs $\mathcal{A}$ and simulates the d-GROTH16 game. When preparing the group element inputs and answering hash queries, $\mathcal{B}$ embeds the (q,q)-DL instance it is tasked with solving. Note that this is possible because $\mathcal{B}$ is given the power-DL challenge up to power $q(\lambda)$ in both groups. By construction, if $\mathcal{A}$ returns an output that satisfies the relation polynomial of d-GROTH16, then t is a root of the polynomial $Q'(R)$ defined by $\mathcal{B}$. This means that $\mathcal{B}$ will be able to find t by inspecting the roots of Q' whenever $Q'(R) \neq 0$. We show that the latter happens with high probability if some hash coefficient is non-zero, which means that these must vanish if (q,q)-DL holds for B.

```
Extractor $\mathcal{E}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\mathsf{trace}(\mathcal{A}))$:
parse $\mathsf{trace}(\mathcal{A}) = (r_{\mathcal{A}}, \varpi, [\boldsymbol{x}_1]_1, [\boldsymbol{x}_2]_2, [\boldsymbol{h}_1]_1, [\boldsymbol{h}_2]_2, [\boldsymbol{h}_T]_T)$
$o_1, o_2, o_T \leftarrow 0$
$(\ell, (U_i, V_i, W_i)_{i=0}^m, T) \leftarrow \mathcal{A}_0^{\overline{\mathsf{H}}_1,\overline{\mathsf{H}}_2,\overline{\mathsf{H}}_T}(\varpi; r_{\mathcal{A}})$
$((f_i)_{i=1}^{\ell}, \boldsymbol{w}_{1,1}, \boldsymbol{v}_{1,1}, \boldsymbol{w}_{1,2}, \boldsymbol{v}_{1,2}, \boldsymbol{w}_2, \boldsymbol{v}_2) \leftarrow$
    $\mathcal{A}_1^{\overline{\mathsf{H}}_1,\overline{\mathsf{H}}_2,\overline{\mathsf{H}}_T}(\varpi, [\boldsymbol{x}_1]_1, [\boldsymbol{x}_2]_2; r_{\mathcal{A}})$
return $(\boldsymbol{w}_{1,1}, \boldsymbol{w}_{1,2}, \boldsymbol{w}_2)$
```

```
Oracle $\overline{\mathsf{H}}_\mu(m)$:
$o_\mu \leftarrow o_\mu + 1$
return $[\boldsymbol{h}_{\mu,o_\mu}]_\mu$
```

```
Adversary $\mathcal{B}(\varpi, [t]_1, \ldots, [t^{q(\lambda)}]_1, [t]_2, \ldots, [t^{q(\lambda)}]_2)$:
$o_1, o_2 \leftarrow 0$; $U_1, U_2, U_T \leftarrow [\,]$; $\mathsf{S} \leftarrow \emptyset$; $r_{\mathcal{A}} \twoheadleftarrow \mathcal{R}_{\mathcal{A}}(\lambda)$
$(\ell, (U_i, V_i, W_i)_{i=0}^m, T) \leftarrow \mathcal{A}_0^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\varpi; r_{\mathcal{A}})$
$\boldsymbol{\varphi} \twoheadleftarrow \mathbb{Z}_p^5$; $\boldsymbol{\psi} \twoheadleftarrow \mathbb{Z}_p^{*5}$
for $\nu \in \{1, 2\}$ do
    $\boldsymbol{X}_\nu(R) \leftarrow \boldsymbol{P}_\nu(\boldsymbol{\varphi} + \boldsymbol{\psi} R)$
    $[\boldsymbol{x}_\nu]_\nu \leftarrow [\boldsymbol{X}_\nu(t)]_\nu$
    $\boldsymbol{X}_{\nu,0}(R) \leftarrow 1$
$((f_i)_{i=1}^{\ell}, \boldsymbol{w}_{1,1}, \boldsymbol{v}_{1,1}, \boldsymbol{w}_{1,2}, \boldsymbol{v}_{1,2}, \boldsymbol{w}_2, \boldsymbol{v}_2) \leftarrow$
    $\mathcal{A}_1^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\varpi, [\boldsymbol{x}_\nu]_\nu; r_{\mathcal{A}})$
$\boldsymbol{Y}_{1,1}(R) \leftarrow \sum_{j=0}^{2d(\lambda)+m+3} \boldsymbol{w}_{1,1,j} \boldsymbol{X}_{1,j}(R) + \sum_l \boldsymbol{v}_{1,1,l} H_{1,l}(R)$
$\boldsymbol{Y}_{1,2}(R) \leftarrow \sum_{j=0}^{2d(\lambda)+m+3} \boldsymbol{w}_{1,2,j} \boldsymbol{X}_{1,j}(R) + \sum_l \boldsymbol{v}_{1,2,l} H_{1,l}(R)$
$Y_2(R) \leftarrow \sum_{j=0}^{d(\lambda)+1} \boldsymbol{w}_{2,j} \boldsymbol{X}_{2,j}(R) + \sum_l \boldsymbol{v}_{2,l} H_{2,l}(R)$
$f_0 \leftarrow 1$; $Q'(T) \leftarrow Q(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{Y}_1, Y_2, (f_i)_{i=0}^{\ell})$
if $(Q'(T) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{Berlekamp}(Q', p)$
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
    $o_\nu \leftarrow o_\nu + 1$
    $\rho_{\nu,o_\nu} \twoheadleftarrow \mathbb{Z}_p$
    $\sigma_{\nu,o_\nu} \twoheadleftarrow \mathbb{Z}_p^*$
    $H_{\nu,o_\nu}(T) \leftarrow$
        $\rho_{\nu,o_\nu} + \sigma_{\nu,o_\nu} R$
    $U_\nu[m] \leftarrow [H_{\nu,o_\nu}(t)]_\nu$
return $U_\nu[m]$
```

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

Figure 24: *Top:* Extractor $\mathcal{E}$ for the algebraic adversary $\mathcal{A}$ in the d-GROTH16 game. *Bottom:* Adversary $\mathcal{B}$ against (q, q)-DL. In all figures, μ and ν range over $\{1, 2, T\}$ and $\{1, 2\}$, respectively, and (vectors of) polynomials $\boldsymbol{P}_1$, $\boldsymbol{P}_2$ and Q are as in Figure 12.

Proof. We now formally implement the intuition presented in the proof overview above. Fix an adversary $\mathcal{A}$ as above, and define an extractor $\mathcal{E}$ and an adversary $\mathcal{B}$ as shown in Figure 24. We now show how to use adversary $\mathcal{B}$ to prove Inequality (8) for $\mathcal{A}$ and $\mathcal{E}$. To that end, consider the following sequence of games (the formal description of which can be found in Figure 25):

G_0: This is the original (q, q)-DL game for B run with adversary $\mathcal{B}$.

G_1: This game proceeds as G_0, but performs variable substitutions $(\boldsymbol{\varphi}', \boldsymbol{\psi}') = (\boldsymbol{\varphi} + \boldsymbol{\psi} t, \boldsymbol{\psi})$ and $(\rho'_{\nu,l}, \sigma'_{\nu,l}) = (\rho_{\nu,l} + \sigma_{\nu,l} t, \sigma_{\nu,l})$ in polynomials $\boldsymbol{X}_\nu$ and $H_{\nu,l}$. More precisely, polynomials $\boldsymbol{X}_\nu(R)$ are now defined as $\boldsymbol{X}_\nu(R) \leftarrow \boldsymbol{P}_\nu(\boldsymbol{\varphi}' + \boldsymbol{\psi}'(R - t))$ for random $\boldsymbol{\varphi}'$ and invertible $\boldsymbol{\psi}'$. Similarly, upon a query m to H_ν, game G_2 samples random $\rho'_{\nu,l}$ and invertible $\sigma'_{\nu,l}$, and sets $H_{\nu,l}(R) \leftarrow \rho'_{\nu,l} + \sigma'_{\nu,l}(R - t)$. Inputs $[\boldsymbol{x}_\nu]_\nu$ and hash replies $U_\nu[m]$ are still computed as $[\boldsymbol{X}_\nu(t)]_\nu = [\boldsymbol{P}_\nu(\boldsymbol{\varphi}')]_\nu$ and $[H_{\nu,l}(t)]_\nu = [\rho'_{\nu,l}]_\nu$, respectively.

G_2: This game proceeds as G_1, but $\boldsymbol{X}_\nu$ and $H_{\nu,l}$ are now defined as $\boldsymbol{X}_\nu(R, \boldsymbol{\Psi}') \leftarrow \boldsymbol{P}_\nu(\boldsymbol{\varphi}' + \boldsymbol{\Psi}'(R - t))$ and $H_{\nu,l}(R, \boldsymbol{\Sigma}'_\nu) \leftarrow \rho'_{\nu,l} + \boldsymbol{\Sigma}'_{\nu,l}(R - t)$, where $\boldsymbol{\Psi}'$ is a new vector of variables and $\boldsymbol{\Sigma}'_{\nu,l}$ is a fresh variable for every oracle call. Accordingly, the polynomial Q'' constructed after running $\mathcal{A}$ is now in variables R, $\boldsymbol{\Psi}'$, $\boldsymbol{\Sigma}'_1$ and $\boldsymbol{\Sigma}'_2$. After defining Q'', game G_2 samples random $\boldsymbol{\psi}'$ and invertible $\boldsymbol{\sigma}'_\nu$, sets $Q'(R) \leftarrow Q''(R, \boldsymbol{\psi}', \boldsymbol{\sigma}'_1, \boldsymbol{\sigma}'_2)$, and checks if $Q'(R) = 0$. From here on, game G_2 proceeds as G_1.

We now argue that subsequent games have identical success probabilities.

```
Game $\mathrm{G}_0(\lambda)$:
$\varpi \twoheadleftarrow \mathrm{B}(1^\lambda)$; $t \twoheadleftarrow \mathbb{Z}_p$; $o_1, o_2 \leftarrow 0$; $U_1, U_2, U_T \leftarrow [\,]$
$\mathsf{S} \leftarrow \emptyset$; $r_{\mathcal{A}} \twoheadleftarrow \mathcal{R}_{\mathcal{A}}(\lambda)$
$(\ell, (U_i, V_i, W_i)_{i=0}^m, T) \leftarrow \mathcal{A}_0^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\varpi; r_{\mathcal{A}})$; $\boldsymbol{\varphi} \twoheadleftarrow \mathbb{Z}_p^5$; $\boldsymbol{\psi} \twoheadleftarrow \mathbb{Z}_p^{*5}$
for $\nu \in \{1,2\}$ do
    $\boldsymbol{X}_\nu(R) \leftarrow \boldsymbol{P}_\nu(\boldsymbol{\varphi} + \boldsymbol{\psi}R)$; $[\boldsymbol{x}_\nu]_\nu \leftarrow [\boldsymbol{X}_\nu(t)]_\nu$; $\boldsymbol{X}_{\nu,0}(R) \leftarrow 1$
$((f_i)_{i=1}^\ell, \boldsymbol{w}_{1,1}, \boldsymbol{v}_{1,1}, \boldsymbol{w}_{1,2}, \boldsymbol{v}_{1,2}, \boldsymbol{w}_2, \boldsymbol{v}_2) \leftarrow \mathcal{A}_1^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\varpi, [\boldsymbol{x}_\nu]_\nu; r_{\mathcal{A}})$
$\boldsymbol{Y}_{1,1}(R) \leftarrow \sum_{j=0}^{2d(\lambda)+m+3} \boldsymbol{w}_{1,1,j}\boldsymbol{X}_{1,j}(R) + \sum_l \boldsymbol{v}_{1,1,l}H_{1,l}(R)$
$\boldsymbol{Y}_{1,2}(R) \leftarrow \sum_{j=0}^{2d(\lambda)+m+3} \boldsymbol{w}_{1,2,j}\boldsymbol{X}_{1,j}(R) + \sum_l \boldsymbol{v}_{1,2,l}H_{1,l}(R)$
$Y_2(R) \leftarrow \sum_{j=0}^{d(\lambda)+1} \boldsymbol{w}_{2,j}\boldsymbol{X}_{2,j}(R) + \sum_l \boldsymbol{v}_{2,l}H_{2,l}(R)$
$f_0 \leftarrow 1$; $Q'(R) \leftarrow Q(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{Y}_1, Y_2, (f_i)_{i=0}^\ell)$
if $(Q'(R) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{Berlekamp}(Q', p)$
$t' \leftarrow 0$; for $z \in \mathsf{S}$ do if $([z]_1 = [t]_1)$ then return $t' \leftarrow z$; break
return $(t = t')$
```

```
Oracle $\mathsf{H}_\nu(m)$:
if $(m \notin \mathrm{Dom}(U_\nu))$ then
    $o_\nu \leftarrow o_\nu + 1$
    $\rho_{\nu,o_\nu} \twoheadleftarrow \mathbb{Z}_p$
    $\sigma_{\nu,o_\nu} \twoheadleftarrow \mathbb{Z}_p^*$
    $H_{\nu,o_\nu}(R) \leftarrow$
        $\rho_{\nu,o_\nu} + \sigma_{\nu,o_\nu}R$
    $U_\nu[m] \leftarrow [H_{\nu,o_\nu}(t)]_\nu$
return $U_\nu[m]$

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

```
Game $\mathrm{G}_1(\lambda)$:
$\varpi \twoheadleftarrow \mathrm{B}(1^\lambda)$; $t \twoheadleftarrow \mathbb{Z}_p$; $o_1, o_2 \leftarrow 0$; $U_1, U_2, U_T \leftarrow [\,]$
$\mathsf{S} \leftarrow \emptyset$; $r_{\mathcal{A}} \twoheadleftarrow \mathcal{R}_{\mathcal{A}}(\lambda)$
$(\ell, (U_i, V_i, W_i)_{i=0}^m, T) \leftarrow \mathcal{A}_0^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\varpi; r_{\mathcal{A}})$; $\boldsymbol{\varphi}' \twoheadleftarrow \mathbb{Z}_p^5$; $\boldsymbol{\psi}' \twoheadleftarrow \mathbb{Z}_p^{*5}$
for $\nu \in \{1,2\}$ do
    $\boldsymbol{X}_\nu(R) \leftarrow \boldsymbol{P}_\nu(\boldsymbol{\varphi}' + \boldsymbol{\psi}'(R - t))$; $[\boldsymbol{x}_\nu]_\nu \leftarrow [\boldsymbol{X}_\nu(t)]_\nu$; $\boldsymbol{X}_{\nu,0}(R) \leftarrow 1$
$((f_i)_{i=1}^\ell, \boldsymbol{w}_{1,1}, \boldsymbol{v}_{1,1}, \boldsymbol{w}_{1,2}, \boldsymbol{v}_{1,2}, \boldsymbol{w}_2, \boldsymbol{v}_2) \leftarrow \mathcal{A}_1^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\varpi, [\boldsymbol{x}_\nu]_\nu; r_{\mathcal{A}})$
$\boldsymbol{Y}_{1,1}(R) \leftarrow \sum_{j=0}^{2d(\lambda)+m+3} \boldsymbol{w}_{1,1,j}\boldsymbol{X}_{1,j}(R) + \sum_l \boldsymbol{v}_{1,1,l}H_{1,l}(R)$
$\boldsymbol{Y}_{1,2}(R) \leftarrow \sum_{j=0}^{2d(\lambda)+m+3} \boldsymbol{w}_{1,2,j}\boldsymbol{X}_{1,j}(R) + \sum_l \boldsymbol{v}_{1,2,l}H_{1,l}(R)$
$Y_2(R) \leftarrow \sum_{j=0}^{d(\lambda)+1} \boldsymbol{w}_{2,j}\boldsymbol{X}_{2,j}(R) + \sum_l \boldsymbol{v}_{2,l}H_{2,l}(R)$
$f_0 \leftarrow 1$; $Q'(R) \leftarrow Q(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{Y}_1, Y_2, (f_i)_{i=0}^\ell)$
if $(Q'(R) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{Berlekamp}(Q', p)$
$t' \leftarrow 0$; for $z \in \mathsf{S}$ do if $([z]_1 = [t]_1)$ then return $t' \leftarrow z$; break
return $(t = t')$
```

```
Oracle $\mathsf{H}_\nu(m)$:
if $(m \notin \mathrm{Dom}(U_\nu))$ then
    $o_\nu \leftarrow o_\nu + 1$
    $\rho'_{\nu,o_\nu} \twoheadleftarrow \mathbb{Z}_p$
    $\sigma'_{\nu,o_\nu} \twoheadleftarrow \mathbb{Z}_p^*$
    $H_{\nu,o_\nu}(R) \leftarrow$
        $\rho'_{\nu,o_\nu} + \sigma'_{\nu,o_\nu}(R - t)$
    $U_\nu[m] \leftarrow [H_{\nu,o_\nu}(t)]_\nu$
return $U_\nu[m]$

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

```
Game $\mathrm{G}_2(\lambda)$:
$\varpi \twoheadleftarrow \mathrm{B}(1^\lambda)$; $t \twoheadleftarrow \mathbb{Z}_p$; $o_1, o_2 \leftarrow 0$; $U_1, U_2, U_T \leftarrow [\,]$
$\mathsf{S} \leftarrow \emptyset$; $r_{\mathcal{A}} \twoheadleftarrow \mathcal{R}_{\mathcal{A}}(\lambda)$
$(\ell, (U_i, V_i, W_i)_{i=0}^m, T) \leftarrow \mathcal{A}_0^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\varpi; r_{\mathcal{A}})$; $\boldsymbol{\varphi}' \twoheadleftarrow \mathbb{Z}_p^5$
for $\nu \in \{1,2\}$ do
    $\boldsymbol{X}_\nu(R, \boldsymbol{\Psi}') \leftarrow \boldsymbol{P}_\nu(\boldsymbol{\varphi}' + \boldsymbol{\Psi}'(R - t))$
    $[\boldsymbol{x}_\nu]_\nu \leftarrow [\boldsymbol{X}_\nu(t, \boldsymbol{\Psi}')]_\nu$; $\boldsymbol{X}_{\nu,0}(R) \leftarrow 1$
$((f_i)_{i=1}^\ell, \boldsymbol{w}_{1,1}, \boldsymbol{v}_{1,1}, \boldsymbol{w}_{1,2}, \boldsymbol{v}_{1,2}, \boldsymbol{w}_2, \boldsymbol{v}_2) \leftarrow \mathcal{A}_1^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_T}(\varpi, [\boldsymbol{x}_\nu]_\nu; r_{\mathcal{A}})$
$\boldsymbol{Y}_{1,1}(R, \boldsymbol{\Psi}', \boldsymbol{\Sigma}'_1) \leftarrow$
    $\sum_{j=0}^{2d(\lambda)+m+3} \boldsymbol{w}_{1,1,j}\boldsymbol{X}_{1,j}(R, \boldsymbol{\Psi}') + \sum_l \boldsymbol{v}_{1,1,l}H_{1,l}(R, \boldsymbol{\Sigma}'_1)$
$\boldsymbol{Y}_{1,2}(R, \boldsymbol{\Psi}', \boldsymbol{\Sigma}'_1) \leftarrow$
    $\sum_{j=0}^{2d(\lambda)+m+3} \boldsymbol{w}_{1,2,j}\boldsymbol{X}_{1,j}(R, \boldsymbol{\Psi}') + \sum_l \boldsymbol{v}_{1,2,l}H_{1,l}(R, \boldsymbol{\Sigma}'_1)$
$Y_2(R, \boldsymbol{\Psi}', \boldsymbol{\Sigma}'_2) \leftarrow \sum_{j=0}^{d(\lambda)+1} \boldsymbol{w}_{2,j}\boldsymbol{X}_{2,j}(R, \boldsymbol{\Psi}') + \sum_l \boldsymbol{v}_{2,l}H_{2,l}(R, \boldsymbol{\Sigma}'_2)$
$f_0 \leftarrow 1$; $Q''(R, \boldsymbol{\Psi}', \boldsymbol{\Sigma}'_1, \boldsymbol{\Sigma}'_2) \leftarrow Q(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{Y}_1, Y_2, (f_i)_{i=0}^\ell)$
$\boldsymbol{\psi}' \twoheadleftarrow \mathbb{Z}_p^{*5}$; $\boldsymbol{\sigma}'_1 \twoheadleftarrow \mathbb{Z}_p^{*o_1}$; $\boldsymbol{\sigma}'_2 \twoheadleftarrow \mathbb{Z}_p^{*o_2}$; $Q'(R) \leftarrow Q''(R, \boldsymbol{\psi}', \boldsymbol{\sigma}'_1, \boldsymbol{\sigma}'_2)$
if $(Q'(R) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{Berlekamp}(Q', p)$
$t' \leftarrow 0$; for $z \in \mathsf{S}$ do if $([z]_1 = [t]_1)$ then return $t' \leftarrow z$; break
return $(t = t')$
```

```
Oracle $\mathsf{H}_\nu(m)$:
if $(m \notin \mathrm{Dom}(U_\nu))$ then
    $o_\nu \leftarrow o_\nu + 1$
    $\rho'_{\nu,o_\nu} \twoheadleftarrow \mathbb{Z}_p$
    $H_{\nu,o_\nu}(R, \boldsymbol{\Sigma}'_\nu) \leftarrow$
        $\rho'_{\nu,o_\nu} + \boldsymbol{\Sigma}'_{\nu,o_\nu}(R - t)$
    $U_\nu[m] \leftarrow$
        $[H_{\nu,o_\nu}(t, \boldsymbol{\Sigma}'_\nu)]_\nu$
return $U_\nu[m]$

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

Figure 25: Code of the intermediate games in the proof of Inequality (8). In all figures, ν is an index ranging over $\{1, 2\}$, and (vectors of) polynomials $\boldsymbol{P}_1$, $\boldsymbol{P}_2$ and Q are as in Figure 12.

$\mathrm{G}_0 \rightsquigarrow \mathrm{G}_1$. Observe that for every fixed $\lambda \in \mathbb{N}$, ϖ returned by $\mathrm{B}(1^\lambda)$, $t \in \mathbb{Z}_p$, and randomness $r_{\mathcal{A}}$ returned by $\mathcal{R}_{\mathcal{A}}(\lambda)$, the random variates $\boldsymbol{\varphi}'$, $\boldsymbol{\psi}'$, $\rho'_{\nu,l}$ and $\sigma'_{\nu,l}$ in G_1 are related to the random variates $\boldsymbol{\varphi}$, $\boldsymbol{\psi}$, $\rho_{\nu,l}$ and $\sigma_{\nu,l}$ in G_0 via the transformation $\mathrm{diag}(\left(\begin{smallmatrix}1 & t\\ 0 & 1\end{smallmatrix}\right))$, which is invertible. Consequently, $\Pr[\mathrm{G}_0] = \Pr[\mathrm{G}_1]$, since there is a one-to-one correspondence between the random variables in the two games.

$\mathrm{G}_1 \rightsquigarrow \mathrm{G}_2$. Notice that $\mathcal{A}$ is oblivious to the changes to polynomials $\boldsymbol{X}_\nu$ and $H_{\nu,l}$, so the simulation of $\mathcal{A}$ is identical in both games. Indeed, in both games inputs to $\mathcal{A}$ and hash replies are computed in the same way. After running $\mathcal{A}$, G_2 derives the same polynomial Q' computed in G_1 by substituting random $\boldsymbol{\psi}'$, $\boldsymbol{\sigma}'_1$ and $\boldsymbol{\sigma}'_2$ into Q'', so the winning condition is again the same in both games. Therefore, $\Pr[\mathrm{G}_1] = \Pr[\mathrm{G}_2]$.

We conclude the proof by studying the winning probability in G_2. First, notice that in this game adversary $\mathcal{A}$ plays the d-GROTH16 game, since the inputs of $\mathcal{A}$ are obtained by evaluating $\boldsymbol{P}_\nu$ at random points and hash replies are random group elements. Now for any $\lambda \in \mathbb{N}$, ϖ returned by $\mathrm{B}(1^\lambda)$, $t \in \mathbb{Z}_p$, randomness $r_{\mathcal{A}}$ returned by $\mathcal{R}_{\mathcal{A}}(\lambda)$, and vectors $\boldsymbol{\varphi}'$, $\boldsymbol{\rho}'_\nu$ and $\boldsymbol{\rho}$ in $\mathbb{Z}_p$, denote by $\mathrm{G}' := \mathrm{G}'(\lambda, \varpi, t, r_{\mathcal{A}}, \boldsymbol{\varphi}', \boldsymbol{\rho}'_\nu, \boldsymbol{\rho})$ the game $\mathrm{G}_2(\lambda)$ with these random choices fixed. Then $\Pr[\mathrm{G}_2(\lambda)] = \sum_{(\varpi, t, r_{\mathcal{A}}, \boldsymbol{\varphi}', \boldsymbol{\rho}'_\nu, \boldsymbol{\rho})} \Pr[\mathrm{G}'] \Pr[\varpi, t, r_{\mathcal{A}}, \boldsymbol{\varphi}', \boldsymbol{\rho}'_\nu, \boldsymbol{\rho}]$, where $\Pr[\varpi, t, r_{\mathcal{A}}, \boldsymbol{\varphi}', \boldsymbol{\rho}'_\nu, \boldsymbol{\rho}]$ denotes the probability that such a tuple is drawn in $\mathrm{G}_2(\lambda)$, and the sum extends over all $(\varpi, t, r_{\mathcal{A}}, \boldsymbol{\varphi}', \boldsymbol{\rho}'_\nu, \boldsymbol{\rho})$ such that $\Pr[\varpi, t, r_{\mathcal{A}}, \boldsymbol{\varphi}', \boldsymbol{\rho}'_\nu, \boldsymbol{\rho}] \neq 0$.

Now consider the set X of all $(\varpi, t, r_{\mathcal{A}}, \boldsymbol{\varphi}', \boldsymbol{\rho}'_\nu, \boldsymbol{\rho})$ in the sum above such that $\mathcal{A}$ returns $(\ell, (U_i, V_i, W_i)_{i=0}^m, T)$ and $((f_i)_{i=1}^\ell, \boldsymbol{w}_{1,1}, \boldsymbol{v}_{1,1}, \boldsymbol{w}_{1,2}, \boldsymbol{v}_{1,2}, \boldsymbol{w}_2, \boldsymbol{v}_2)$ for which the relation polynomial in d-GROTH16 is satisfied and extractor $\mathcal{E}$ fails to compute a correct representation of the outputs. Notice that

$$\sum_{(\varpi, t, r_{\mathcal{A}}, \boldsymbol{\varphi}', \boldsymbol{\rho}'_\nu, \boldsymbol{\rho}) \in \mathsf{X}} \Pr[\varpi, t, r_{\mathcal{A}}, \boldsymbol{\varphi}', \boldsymbol{\rho}'_\nu, \boldsymbol{\rho}] = \mathrm{Adv}^{d\text{-groth16}}_{\mathrm{B},\mathcal{A},\mathcal{E}}(\lambda)\,.$$

We claim that for any $(\varpi, t, r_{\mathcal{A}}, \boldsymbol{\varphi}', \boldsymbol{\rho}'_\nu, \boldsymbol{\rho}) \in \mathsf{X}$, $\Pr[\mathrm{G}'] \geq 1 - 2q(\lambda)/(2^{\lambda-1} - 1)$. Indeed, fix any $(\varpi, t, r_{\mathcal{A}}, \boldsymbol{\varphi}', \boldsymbol{\rho}'_\nu, \boldsymbol{\rho}) \in \mathsf{X}$. Since $\mathcal{E}$ fails to return a correct representation of the output of $\mathcal{A}$, it must be either $\boldsymbol{v}_{1,1} \neq 0$, or $\boldsymbol{v}_{1,2} \neq 0$, or $\boldsymbol{v}_2 \neq 0$. We now show that either way, the polynomial $Q''(R, \boldsymbol{\Psi}', \boldsymbol{\Sigma}'_1, \boldsymbol{\Sigma}'_2)$ constructed in G_2 after running $\mathcal{A}$ is not identically zero with overwhelming probability. To that end, consider the polynomial $V(\boldsymbol{S}, \boldsymbol{H}_1, \boldsymbol{H}_2)$ given by

$$\begin{aligned} V := &\left(\sum_{j=0}^{2d+m+3} \boldsymbol{w}_{1,1,j} \boldsymbol{P}_{1,j}(\boldsymbol{S}) + \sum_l \boldsymbol{v}_{1,1,l} \boldsymbol{H}_{1,l}\right)\left(\sum_{j=0}^{d+1} \boldsymbol{w}_{2,j} \boldsymbol{P}_{2,j}(\boldsymbol{S}) + \sum_l \boldsymbol{v}_{2,l} \boldsymbol{H}_{2,l}\right) \\ &- \left(\sum_{j=0}^{2d+m+3} \boldsymbol{w}_{1,2,j} \boldsymbol{P}_{1,j}(\boldsymbol{S}) + \sum_l \boldsymbol{v}_{1,2,l} \boldsymbol{H}_{1,l}\right) \boldsymbol{P}_{2,3}(\boldsymbol{S}) \\ &- \boldsymbol{P}_{1,1}(\boldsymbol{S}) \boldsymbol{P}_{2,1}(\boldsymbol{S}) - \sum_{i=0}^{\ell} f_i \boldsymbol{P}_{1,2d+3+i}(\boldsymbol{S}) \boldsymbol{P}_{2,2}(\boldsymbol{S})\,. \end{aligned} \tag{9}$$

We will prove further down that polynomial $V(\boldsymbol{S}, \boldsymbol{H}_1, \boldsymbol{H}_2)$ is not identically zero and of total degree at most $2q(\lambda)$. Assuming for the moment that that is the case, notice that

$$Q''(R, \boldsymbol{\Psi}', \boldsymbol{\Sigma}'_1, \boldsymbol{\Sigma}'_2) = V\big(\boldsymbol{\varphi}' + \boldsymbol{\Psi}'(R - t), \boldsymbol{\rho}'_1 + \boldsymbol{\Sigma}'_1(R - t), \boldsymbol{\rho}'_2 + \boldsymbol{\Sigma}'_2(R - t)\big)\,,$$

which again is non-zero by Lemma 2 and of degree in R at most $2q(\lambda)$. Moreover, by Lemma 2, the leading coefficient in R of $Q''(R, \boldsymbol{\Psi}', \boldsymbol{\Sigma}'_1, \boldsymbol{\Sigma}'_2)$ is a polynomial in $\boldsymbol{\Psi}', \boldsymbol{\Sigma}'_1, \boldsymbol{\Sigma}'_2$ of total degree at most $2q(\lambda)$, which for random invertible $\boldsymbol{\psi}'$ and $\boldsymbol{\sigma}'_\nu$ will be zero with probability at most $2q(\lambda)/(2^{\lambda-1} - 1)$ by Lemma 1. Thus, we have $Q'(R) \neq 0$ in G' with probability at least $1 - 2q(\lambda)/(2^{\lambda-1} - 1)$. We can now derive Inequality (8) by observing that whenever this happens, game G' will return 1, because t is a root of $Q'(R)$ by construction, and will therefore be found by inspecting its roots. This means

$$\mathrm{Adv}^{(q,q)\text{-dl}}_{\mathrm{B},\mathcal{B}}(\lambda) = \Pr[\mathrm{G}_0(\lambda)] = \Pr[\mathrm{G}_2(\lambda)] = \sum_{(\varpi,t,r_{\mathcal{A}},\boldsymbol{\varphi}',\boldsymbol{\rho}'_{\nu},\boldsymbol{\rho})} \Pr[\mathrm{G}'] \Pr[\varpi, t, r_{\mathcal{A}}, \boldsymbol{\varphi}', \boldsymbol{\rho}'_{\nu}, \boldsymbol{\rho}]$$

$$\geq \sum_{(\varpi,t,r_{\mathcal{A}},\boldsymbol{\varphi}',\boldsymbol{\rho}'_{\nu},\boldsymbol{\rho})\in\mathsf{X}} \Pr[\mathrm{G}'] \Pr[\varpi, t, r_{\mathcal{A}}, \boldsymbol{\varphi}', \boldsymbol{\rho}'_{\nu}, \boldsymbol{\rho}] \geq \left(1 - \frac{2q(\lambda)}{2^{\lambda-1}-1}\right) \cdot \mathrm{Adv}^{d\text{-groth16}}_{\mathrm{B},\mathcal{A},\mathcal{E}}(\lambda)\,.$$

To conclude our proof, it remains to be shown that the polynomial V defined in (9) is not identically zero and of total degree at most $2q(\lambda)$. To prove that $V \neq 0$, assume by contradiction that $V = 0$. We proceed by case distinction, according to whether $\boldsymbol{v}_{1,1} \neq 0$, or $\boldsymbol{v}_{1,2} \neq 0$, or $\boldsymbol{v}_2 \neq 0$ in the output of $\mathcal{A}$.

First case: $\boldsymbol{v}_{1,1} \neq 0$. Let l^* be such that $\boldsymbol{v}_{1,1,l^*} \neq 0$. Then the term in $\boldsymbol{H}_{1,l^*}$ has coefficient

$$\boldsymbol{v}_{1,1,l^*}\left(\sum_{j=0}^{d+1} \boldsymbol{w}_{2,j}\boldsymbol{P}_{2,j}(\boldsymbol{S}) + \sum_{l} \boldsymbol{v}_{2,l}\boldsymbol{H}_{2,l}\right) - \boldsymbol{v}_{1,2,l^*}\boldsymbol{P}_{2,3}(\boldsymbol{S})\,, \tag{10}$$

which must be zero as a polynomial in $\boldsymbol{S}$ and $\boldsymbol{H}_2$ since we are assuming $V = 0$. Since $\boldsymbol{v}_{1,1,l^*} \neq 0$ and the polynomials in $\boldsymbol{P}_2$ are linearly independent, this means $\boldsymbol{v}_{2,l} = 0$ for all l and $\boldsymbol{w}_{2,j} = 0$ for all $j \neq 3$. Simplifying the equation $V = 0$ accordingly, we obtain

$$\left(\sum_{j=0}^{2d+m+3}\Big(\boldsymbol{w}_{2,4}\boldsymbol{w}_{1,1,j} - \boldsymbol{w}_{1,2,j}\Big)\boldsymbol{P}_{1,j}(\boldsymbol{S}) + \sum_{l}\Big(\boldsymbol{w}_{2,4}\boldsymbol{v}_{1,1,l} - \boldsymbol{v}_{1,2,l}\Big)\boldsymbol{H}_{1,l}\right)\boldsymbol{P}_{2,3}(\boldsymbol{S})$$
$$- \boldsymbol{P}_{1,1}(\boldsymbol{S})\boldsymbol{P}_{2,1}(\boldsymbol{S}) - \sum_{i=0}^{\ell} f_i \boldsymbol{P}_{1,2d+3+i}(\boldsymbol{S})\boldsymbol{P}_{2,2}(\boldsymbol{S}) = 0\,.$$

This, however, is a contradiction, because the monomial $\boldsymbol{P}_{1,1}(\boldsymbol{S})\boldsymbol{P}_{2,1}(\boldsymbol{S}) = \boldsymbol{S}_1\boldsymbol{S}_2\boldsymbol{S}_3^2\boldsymbol{S}_4^2$ is the only one with tuple of degrees $(1, 1, 2, 2, 0)$, and since it has coefficient $-1 \neq 0$, the equality above cannot hold.

Second case: $\boldsymbol{v}_{1,2} \neq 0$. Let l^* be such that $\boldsymbol{v}_{1,2,l^*} \neq 0$. We again look at the term in $\boldsymbol{H}_{1,l^*}$, whose coefficient given in (10) must be zero as a polynomial in $\boldsymbol{S}$ and $\boldsymbol{H}_2$ since we are assuming $V = 0$. Since $\boldsymbol{v}_{1,2,l^*} \neq 0$, it must be $\boldsymbol{v}_{1,1,l^*} \neq 0$ as well, because otherwise we would have $\boldsymbol{P}_{2,3}(\boldsymbol{S}) = 0$, a contradiction. From here on, the argument proceeds as in the first case above.

Third case: $\boldsymbol{v}_2 \neq 0$. Let l^* be such that $\boldsymbol{v}_{2,l^*} \neq 0$. Then the term in $\boldsymbol{H}_{2,l^*}$ has coefficient

$$\boldsymbol{v}_{2,l^*}\left(\sum_{j=0}^{2d+m+3} \boldsymbol{w}_{1,1,j}\boldsymbol{P}_{1,j}(\boldsymbol{S}) + \sum_{l} \boldsymbol{v}_{1,1,l}\boldsymbol{H}_{1,l}\right),$$

which must be zero as a polynomial in $\boldsymbol{S}$ and $\boldsymbol{H}_1$ since we are assuming $V = 0$. Taking into account that $\boldsymbol{v}_{2,l^*} \neq 0$ and simplifying the equation $V = 0$ accordingly, we obtain

$$-\left(\sum_{j=0}^{2d+m+3} \boldsymbol{w}_{1,2,j}\boldsymbol{P}_{1,j}(\boldsymbol{S}) + \sum_{l} \boldsymbol{v}_{1,2,l}\boldsymbol{H}_{1,l}\right)\boldsymbol{P}_{2,3}(\boldsymbol{S}) - \boldsymbol{P}_{1,1}(\boldsymbol{S})\boldsymbol{P}_{2,1}(\boldsymbol{S})$$
$$- \sum_{i=0}^{\ell} f_i \boldsymbol{P}_{1,2d+3+i}(\boldsymbol{S})\boldsymbol{P}_{2,2}(\boldsymbol{S}) = 0\,.$$

This is again a contradiction, because the monomial $\boldsymbol{P}_{1,1}(\boldsymbol{S})\boldsymbol{P}_{2,1}(\boldsymbol{S}) = \boldsymbol{S}_1\boldsymbol{S}_2\boldsymbol{S}_3^2\boldsymbol{S}_4^2$ is the only one with tuple of degrees $(1, 1, 2, 2, 0)$, and since it has coefficient $-1 \neq 0$, the equality above cannot hold.

This shows that $V(\boldsymbol{S}, \boldsymbol{H}_1, \boldsymbol{H}_2) \neq 0$. To prove the bound on the degree, notice that

$$\begin{aligned}
\deg(V) \leq \max\Bigg[&\deg(\boldsymbol{P}_{1,1}\boldsymbol{P}_{2,1}), \deg\left(\left(\sum_{j=0}^{2d+m+3}\boldsymbol{w}_{1,2,j}\boldsymbol{P}_{1,j} + \sum_{l}\boldsymbol{v}_{1,2,l}\boldsymbol{H}_{1,l}\right)\boldsymbol{P}_{2,3}\right), \\
&\deg\left(\left(\sum_{j=0}^{2d+m+3}\boldsymbol{w}_{1,1,j}\boldsymbol{P}_{1,j} + \sum_{l}\boldsymbol{v}_{1,1,l}\boldsymbol{H}_{1,l}\right)\left(\sum_{j=0}^{d+1}\boldsymbol{w}_{2,j}\boldsymbol{P}_{2,j} + \sum_{l}\boldsymbol{v}_{2,l}\boldsymbol{H}_{2,l}\right)\right), \\
&\deg\left(\sum_{i=0}^{\ell} f_i\boldsymbol{P}_{1,2d+3+i}\boldsymbol{P}_{2,2}\right)\Bigg] \\
= \max\Bigg[&\deg\left(\sum_{j=0}^{2d+m+3}\boldsymbol{w}_{1,1,j}\boldsymbol{P}_{1,j} + \sum_{l}\boldsymbol{v}_{1,1,l}\boldsymbol{H}_{1,l}\right) + \deg\left(\sum_{j=0}^{d+1}\boldsymbol{w}_{2,j}\boldsymbol{P}_{2,j} + \sum_{l}\boldsymbol{v}_{2,l}\boldsymbol{H}_{2,l}\right), 6, \\
&\deg\left(\sum_{j=0}^{2d+m+3}\boldsymbol{w}_{1,2,j}\boldsymbol{P}_{1,j} + \sum_{l}\boldsymbol{v}_{1,2,l}\boldsymbol{H}_{1,l}\right) + \deg(\boldsymbol{P}_{2,3}), \\
&\deg\left(\sum_{i=0}^{\ell} f_i\boldsymbol{P}_{1,2d+3+i}\right) + \deg(\boldsymbol{P}_{2,2})\Bigg] \\
\leq \max\Big[&\max_{j=0}^{2d+m+3}\big(\deg(\boldsymbol{P}_{1,j}), 1\big) + \max_{j=0}^{d+1}\big(\deg(\boldsymbol{P}_{2,j}), 1\big), 6, \max_{j=0}^{2d+m+3}\big(\deg(\boldsymbol{P}_{1,j}), 1\big) + 3, \\
&\max_{i=0}^{\ell}\deg(\boldsymbol{P}_{1,2d+3+i}) + 3\Big] \\
\leq \max\big[&2\max(d(\lambda), 1), 6, \max(d(\lambda), 1) + 3, d(\lambda) + 3\big] \leq 2q(\lambda)\,.
\end{aligned}$$

This concludes the proof. □

8 Conclusion and Relevance to Applications

We established in Theorem 3 that the UK assumption holds in bilinear generic groups for adversaries $\mathcal{A}_0$ that return flexible polynomials Q and $\boldsymbol{P}$. The d-PKE, d-KZG, and d-GROTH16 assumptions are instances of the UK assumption, where $\mathcal{A}_0$ returns specific polynomials Q and $\boldsymbol{P}$. 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 adversaries. We proved separately that d-GROTH16 holds in ABM3-H.

We may now base the *knowledge soundness* of the modified Groth16 SNARK [Gro16] 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-DL in the standard model: Run the existing AGM reduction [FKL18, Theorem 7.2], utilizing the coefficient representation output by the extractor as the coefficient representation needed by the AGM reduction. We obtain the following result.

Corollary 3. *Let* B *be a type-3 bilinear group scheme, and* $d\colon \mathbb{N} \to \mathbb{N}$ *a polynomial. Then* Groth16 *for degree-d QAPs is knowledge sound in the standard model, based on the* d-GROTH16 *and* $(2d-1)$-DL *assumptions.*[10]

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.[11]

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

Acknowledgments

We thank the anonymous reviewers who helped improve the presentation of our results. 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

[AGHO11] Masayuki Abe, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 649–666. Springer, Berlin, Heidelberg, August 2011. doi:10.1007/978-3-642-22792-9_37.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, Berlin, Heidelberg, May 2004. doi:10.1007/978-3-540-24676-3_4.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer, Berlin, Heidelberg, May 2005. doi:10.1007/11426639_26.

[BCC+17] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the SNARK. *Journal of Cryptology*, 30(4):989–1066, October 2017. doi:10.1007/s00145-016-9241-9.

[10] 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 we require $(2d-1)$-DL.

[11] We informed the authors of [FFK+23] that the same issue might arise in their simulation-extractability result for KZG polynomial commitments, at least when considering it in the standard model.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In Shafi Goldwasser, editor, *ITCS 2012: 3rd Innovations in Theoretical Computer Science*, pages 326–349. Association for Computing Machinery, January 2012. doi:10.1145/2090236.2090263.

[BD14] James Birkett and Alexander W. Dent. Security models and proof strategies for plaintext-aware encryption. *Journal of Cryptology*, 27(1):139–180, January 2014. doi:10.1007/s00145-012-9141-6.

[BDPV08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer, Berlin, Heidelberg, April 2008. doi:10.1007/978-3-540-78967-3_11.

[Ber67] Elwyn R. Berlekamp. Factoring polynomials over finite fields. *The Bell System Technical Journal*, 46(8):1853–1859, October 1967. doi:10.1002/j.1538-7305.1967.tb03174.x.

[BFHO22] Balthazar Bauer, Pooya Farshim, Patrick Harasser, and Adam O'Neill. Beyond uber: Instantiating generic groups via PGGs. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022: 20th Theory of Cryptography Conference, Part III*, volume 13749 of *Lecture Notes in Computer Science*, pages 212–242. Springer, Cham, November 2022. doi:10.1007/978-3-031-22368-6_8.

[BFL20] Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. A classification of computational assumptions in the algebraic group model. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 121–151. Springer, Cham, August 2020. doi:10.1007/978-3-030-56880-1_5.

[BFS16] Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 777–804. Springer, Berlin, Heidelberg, December 2016. doi:10.1007/978-3-662-53890-6_26.

[BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 677–706. Springer, Cham, May 2020. doi:10.1007/978-3-030-45721-1_24.

[BHK14] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Cryptography from compression functions: The UCE bridge to the ROM. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 169–187. Springer, Berlin, Heidelberg, August 2014. doi:10.1007/978-3-662-44371-2_10.

[Bla06] John Black. The ideal-cipher model, revisited: An uninstantiable blockcipher-based hash function. In Matthew J. B. Robshaw, editor, *Fast Software Encryption – FSE 2006*, volume 4047 of *Lecture Notes in Computer Science*, pages 328–340. Springer, Berlin, Heidelberg, March 2006. doi:10.1007/11799313_21.

[BP04a] Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 273–289. Springer, Berlin, Heidelberg, August 2004. doi:10.1007/978-3-540-28628-8_17.

[BP04b] Mihir Bellare and Adriana Palacio. Towards plaintext-aware public-key encryption without random oracles. In Pil Joong Lee, editor, *Advances in Cryptology – ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 48–62. Springer, Berlin, Heidelberg, December 2004. doi:10.1007/978-3-540-30539-2_4.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, November 1993. doi:10.1145/168588.168596.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, Berlin, Heidelberg, May / June 2006. doi:10.1007/11761679_25.

[Bro01] Daniel R. L. Brown. The exact security of ECDSA. Contributions to IEEE P1363a, January 2001. http://grouper.ieee.org/groups/1363/.

[BRS02] John Black, Phillip Rogaway, and Thomas Shrimpton. Black-box analysis of the block-cipher-based hash-function constructions from PGV. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 320–335. Springer, Berlin, Heidelberg, August 2002. doi:10.1007/3-540-45708-9_21.

[BV98] Dan Boneh and Ramarathnam Venkatesan. Breaking RSA may not be equivalent to factoring. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 59–71. Springer, Berlin, Heidelberg, May / June 1998. doi:10.1007/BFb0054117.

[CCS07] Liqun Chen, Zhaohui Cheng, and Nigel P. Smart. Identity-based key agreement protocols from pairings. *International Journal of Information Security*, 6(4):213–241, January 2007. doi:10.1007/s10207-006-0011-9.

[CDMP05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård revisited: How to construct a hash function. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448. Springer, Berlin, Heidelberg, August 2005. doi:10.1007/11535218_26.

[CFF$^+$21] Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. Lunar: A toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part III*, volume 13092 of *Lecture Notes in Computer Science*, pages 3–33. Springer, Cham, December 2021. doi:10.1007/978-3-030-92078-4_1.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th Annual ACM Symposium on Theory of Computing*, pages 209–218. ACM Press, May 1998. doi:10.1145/276698.276741.

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

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, Berlin, Heidelberg, August 1998. doi:10.1007/BFb0055717.

[Dam92] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 445–456. Springer, Berlin, Heidelberg, August 1992. doi:10.1007/3-540-46766-1_36.

[Den02] Alexander W. Dent. Adapting the weaknesses of the random oracle model to the generic group model. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 100–109. Springer, Berlin, Heidelberg, December 2002. doi:10.1007/3-540-36178-2_6.

[Den06a] Alexander W. Dent. The Cramer-Shoup encryption scheme is plaintext aware in the standard model. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 289–307. Springer, Berlin, Heidelberg, May / June 2006. doi:10.1007/11761679_18.

[Den06b] Alexander W. Dent. The hardness of the DHK problem in the generic group model. Cryptology ePrint Archive, Report 2006/156, 2006. URL: https://eprint.iacr.org/2006/156.

[DL78] Richard A. Demillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195, June 1978. doi:10.1016/0020-0190(78)90067-4.

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

[FFK$^+$23] Antonio Faonio, Dario Fiore, Markulf Kohlweiss, Luigi Russo, and Michal Zajac. From polynomial IOP and commitments to non-malleable zkSNARKs. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023: 21st Theory of Cryptography Conference, Part III*, volume 14371 of *Lecture Notes in Computer Science*, pages 455–485. Springer, Cham, November / December 2023. doi:10.1007/978-3-031-48621-0_16.

[FGJ18] Nils Fleischhacker, Vipul Goyal, and Abhishek Jain. On the existence of three round zero-knowledge proofs. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 3–33. Springer, Cham, April / May 2018. doi:10.1007/978-3-319-78372-7_1.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 33–62. Springer, Cham, August 2018. doi:10.1007/978-3-319-96881-0_2.

[FPS20] Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind Schnorr signatures and signed ElGamal encryption in the algebraic group model. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 63–95. Springer, Cham, May 2020. doi:10.1007/978-3-030-45724-2_3.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, Berlin, Heidelberg, August 1987. doi:10.1007/3-540-47721-7_12.

[GK16] Shafi Goldwasser and Yael Tauman Kalai. Cryptographic assumptions: A position paper. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 505–522. Springer, Berlin, Heidelberg, January 2016. doi:10.1007/978-3-662-49096-9_21.

[GM17] Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 581–612. Springer, Cham, August 2017. doi:10.1007/978-3-319-63715-0_20.

[GPS06] Stephen D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165, 2006. URL: https://eprint.iacr.org/2006/165.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 321–340. Springer, Berlin, Heidelberg, December 2010. doi:10.1007/978-3-642-17373-8_19.

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

[GS22] Jens Groth and Victor Shoup. On the security of ECDSA with additive key derivation and presignatures. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part I*, volume 13275 of *Lecture Notes in Computer Science*, pages 365–396. Springer, Cham, May / June 2022. doi:10.1007/978-3-031-06944-4_13.

[GT21] Ashrujit Ghoshal and Stefano Tessaro. Tight state-restoration soundness in the algebraic group model. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 64–93, Virtual Event, August 2021. Springer, Cham. doi:10.1007/978-3-030-84252-9_3.

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

[HKT11] Thomas Holenstein, Robin Künzler, and Stefano Tessaro. The equivalence of the random oracle model and the ideal cipher model, revisited. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 89–98. ACM Press, June 2011. doi:10.1145/1993636.1993650.

[HT98] Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 408–423. Springer, Berlin, Heidelberg, August 1998. doi:10.1007/BFb0055744.

[HT99] Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. Cryptology ePrint Archive, Report 1999/009, 1999. URL: https://eprint.iacr.org/1999/009.

[Jou04] Antoine Joux. A one round protocol for tripartite Diffie–Hellman. *Journal of Cryptology*, 17(4):263–276, September 2004. doi:10.1007/s00145-004-0312-y.

[KLT16] Aggelos Kiayias, Feng-Hao Liu, and Yiannis Tselekounis. Practical non-malleable codes from ℓ-more extractable hash functions. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 1317–1328. ACM Press, October 2016. doi:10.1145/2976749.2978352.

[KLX22] Julia Kastner, Julian Loss, and Jiayu Xu. On pairing-free blind signature schemes in the algebraic group model. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022: 25th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 13178 of *Lecture Notes in Computer Science*, pages 468–497. Springer, Cham, March 2022. doi:10.1007/978-3-030-97131-1_16.

[KM07] Neal Koblitz and Alfred J. Menezes. Another look at "provable security". *Journal of Cryptology*, 20(1):3–37, January 2007. doi:10.1007/s00145-005-0432-z.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 177–194. Springer, Berlin, Heidelberg, December 2010. doi:10.1007/978-3-642-17373-8_11.

[Lep02] Matthew Lepinski. On the existence of 3-round zero-knowledge proofs. Master's thesis, Massachusetts Institute of Technology, June 2002. URL: http://hdl.handle.net/1721.1/87273.

[Lip22] Helger Lipmaa. A unified framework for non-universal SNARKs. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022: 25th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 13177 of *Lecture Notes in Computer Science*, pages 553–583. Springer, Cham, March 2022. doi:10.1007/978-3-030-97121-2_20.

[LPS23] Helger Lipmaa, Roberto Parisella, and Janno Siim. Algebraic group model with oblivious sampling. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023: 21st Theory of Cryptography Conference, Part IV*, volume 14372 of *Lecture Notes in Computer Science*, pages 363–392. Springer, Cham, November / December 2023. doi:10.1007/978-3-031-48624-1_14.

[Mau05] Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12. Springer, Berlin, Heidelberg, December 2005. doi:10.1007/11586821_1.

[MBKM19] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 2111–2128. ACM Press, November 2019. doi:10.1145/3319535.3339817.

[Nao03] Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109. Springer, Berlin, Heidelberg, August 2003. doi:10.1007/978-3-540-45146-4_6.

[Nec94] Vassiliy Ilyich Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, February 1994. doi:10.1007/BF02113297.

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

[PV05] Pascal Paillier and Damien Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In Bimal K. Roy, editor, *Advances in Cryptology – ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 1–20. Springer, Berlin, Heidelberg, December 2005. doi:10.1007/11593447_1.

[RLB$^+$08] Andy Rupp, Gregor Leander, Endre Bangerter, Alexander W. Dent, and Ahmad-Reza Sadeghi. Sufficient conditions for intractability over black-box groups: Generic lower bounds for generalized DL and DH problems. In Josef Pieprzyk, editor, *Advances in Cryptology – ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 489–505. Springer, Berlin, Heidelberg, December 2008. doi:10.1007/978-3-540-89255-7_30.

[RS20] Lior Rotem and Gil Segev. Algebraic distinguishers: From discrete logarithms to decisional uber assumptions. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part III*, volume 12552 of *Lecture Notes in Computer Science*, pages 366–389. Springer, Cham, November 2020. doi:10.1007/978-3-030-64381-2_13.

[RZ21] Carla Ràfols and Arantxa Zapico. An algebraic framework for universal and updatable SNARKs. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 774–804, Virtual Event, August 2021. Springer, Cham. doi:10.1007/978-3-030-84242-0_27.

[Sch80] Jack T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, October 1980. doi:10.1145/322217.322225.

[Sha05] Hovav Shacham. *New Paradigms in Signature Schemes.* PhD thesis, Stanford University, December 2005. URL: https://hovav.net/ucsd/dist/thesis.pdf.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, Berlin, Heidelberg, May 1997. doi:10.1007/3-540-69053-0_18.

[Zha22] Mark Zhandry. To label, or not to label (in generic groups). In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part III*, volume 13509 of *Lecture Notes in Computer Science*, pages 66–96. Springer, Cham, August 2022. doi:10.1007/978-3-031-15982-4_3.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In Edward W. Ng, editor, *Symbolic and Algebraic Computation – EUROSAM 1979*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, Berlin, Heidelberg, June 1979. doi:10.1007/3-540-09519-5_73.

[ZZ23] Cong Zhang and Mark Zhandry. The relationship between idealized models under computationally bounded adversaries. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part VI*, volume 14443 of *Lecture Notes in Computer Science*, pages 390–419. Springer, Singapore, December 2023. doi:10.1007/978-981-99-8736-8_13.

[ZZK22] Cong Zhang, Hong-Sheng Zhou, and Jonathan Katz. An analysis of the algebraic group model. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022, Part IV*, volume 13794 of *Lecture Notes in Computer Science*, pages 310–322. Springer, Cham, December 2022. doi:10.1007/978-3-031-22972-5_11.

```
Extractor $\mathcal{E}^{\mathsf{op},\mathsf{H}}(\mathsf{trace}(\mathcal{A}))$:
parse $\mathsf{trace}(\mathcal{A}) = (r_{\mathcal{A}}, \boldsymbol{u}, \boldsymbol{h})$
$o, v \leftarrow 0$; $U_\tau, U_H \leftarrow [\,]$; $U_\tau[1] \leftarrow \boldsymbol{u}_0$
$(Q, \boldsymbol{P}) \leftarrow \mathcal{A}_0^{\overline{\mathsf{op}},\overline{\mathsf{H}}}(\boldsymbol{u}_0; r_{\mathcal{A}})$
for $j = 1$ to $|\boldsymbol{X}| - 1$ do
    $U_\tau[\boldsymbol{P}_j(\boldsymbol{S})] \leftarrow \boldsymbol{u}_j$
$(\boldsymbol{v}, \boldsymbol{c}) \leftarrow \mathcal{A}_1^{\overline{\mathsf{op}},\overline{\mathsf{H}}}(\boldsymbol{u}; r_{\mathcal{A}})$; $\boldsymbol{P}_0(\boldsymbol{S}) \leftarrow 1$
for $i = 1$ to $|\boldsymbol{Y}|$ do
    if $(\boldsymbol{v}_i \notin \mathrm{Rng}(U_\tau))$ then
        $v \leftarrow v + 1$; $U_\tau[\boldsymbol{R}_v] \leftarrow \boldsymbol{v}_i$
    parse $U_\tau^{-1}[\boldsymbol{v}_i] =$
        $\sum_{j=0}^{|\boldsymbol{X}|-1} \boldsymbol{w}_{ij} \boldsymbol{P}_j(\boldsymbol{S}) + \sum_l \boldsymbol{b}_{il} \boldsymbol{R}_l$
return $\boldsymbol{w}$
```

```
Proc. $\overline{\mathsf{op}}(h_1, h_2)$:
for $i = 1$ to 2 do
    if $(h_i \notin \mathrm{Rng}(U_\tau))$ then
        $v \leftarrow v + 1$; $U_\tau[\boldsymbol{R}_v] \leftarrow h_i$
    $x_i \leftarrow U_\tau^{-1}[h_i]$
$x \leftarrow x_1 + x_2$; $o \leftarrow o + 1$
if $(x \notin \mathrm{Dom}(U_\tau))$ then $U_\tau[x] \leftarrow \boldsymbol{h}_o$
return $U_\tau[x]$

Proc. $\overline{\mathsf{H}}(m)$:
if $(m \notin \mathrm{Dom}(U_H))$ then
    $v \leftarrow v + 1$; $U_H[m] \leftarrow \boldsymbol{R}_v$
$r \leftarrow U_H[m]$; $o \leftarrow o + 1$
if $(r \notin \mathrm{Dom}(U_\tau))$ then $U_\tau[r] \leftarrow \boldsymbol{h}_o$
return $U_\tau[r]$
```

Figure 26: Definition of the extractor $\mathcal{E}$ from the proof of Theorem 6.

Appendix A: Soundness of Linear UK in GGM-H

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

Theorem 6 (Linear UK holds in GGM-H)**.** *Let $p \in \mathbb{N}$ be prime, and fix $\mathsf{G} \subseteq \{0,1\}^*$ with $|\mathsf{G}| = p$. Consider the class of algorithms $\mathfrak{A}$ and the source $\mathcal{S}$ defined as follows:*

1. *For every $\mathcal{A}_0 \in \mathfrak{A}$, the relation polynomial Q returned by $\mathcal{A}_0$ is of the form*

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

2. *For every $\mathcal{A}_0 \in \mathfrak{A}$, every $(Q, \boldsymbol{P})$ returned by $\mathcal{A}_0$, and every $\boldsymbol{c} \in \mathbb{Z}_p^{|\boldsymbol{C}|}$, the polynomials $\overline{Q_i}$, $1 \le i \le |\boldsymbol{Y}|$, are linearly independent;*
3. *For every $\mathcal{A}_0 \in \mathfrak{A}$ and every $(Q, \boldsymbol{P})$ 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 first-stage adversaries $\mathfrak{A}$ and source $\mathcal{S}$ above. More precisely, for every low-degree adversary $\mathcal{A}$ with $\mathcal{A}_0 \in \mathfrak{A}$, there exists an extractor $\mathcal{E}$ such that*

$$\mathrm{Adv}^{\mathrm{uk}}_{p,\mathsf{G},\mathcal{S},\mathcal{A},\mathcal{E}} \le \mathcal{O}\left(\frac{(m + n + q_{\mathsf{op}} + q_{\mathsf{H}} + d_Q)^2 \cdot d_P}{p}\right). \tag{11}$$

Here, d_Q is an upper bound on the total degree of Q, d_P and k are upper bounds on the total degree and the number of variables of every polynomial P in $\boldsymbol{P}$, m and n are upper bounds on $|\boldsymbol{X}| - 1$ and $|\boldsymbol{Y}|$, q_{op} and q_{H} are upper bounds on the number of queries made by $\mathcal{A}$ to the respective oracles, and we let $\boldsymbol{P}_0(\boldsymbol{S}) \coloneqq 1$ in $\overline{Q_i}(\boldsymbol{S}) \coloneqq Q_i(\boldsymbol{P}(\boldsymbol{S}), \boldsymbol{c})$.

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 Figure 26. This extractor essentially re-runs $\mathcal{A}$ on its view and observes its oracle queries, keeping track of the discrete logarithms of the elements queried by $\mathcal{A}$ via a table U_τ. Whenever $\mathcal{E}$ is unable to "explain" an element in G, it instead stores a fresh variable $\boldsymbol{R}_v$ in U_τ.

We claim that this extractor allows proving Inequality (11). To that end, consider the following sequence of games (the formal description of which can be found in Figure 27):

G_0: This is the original UK game in the GGM-H with parameters (p, G) and source $\mathcal{S}$, run with adversary $\mathcal{A}$ and extractor $\mathcal{E}$. We omit repeated invocations of op to create the inputs of $\mathcal{A}_1$, and instead compute $\tau(\boldsymbol{x})$ directly. We also reformulate the winning condition by not applying τ in the last two clauses, which results in an equivalent game since τ is injective. The operation, hashing and pairing oracles are augmented to construct the view of $\mathcal{A}$ along the way.

G_1: This game proceeds as G_0, but the encoding τ is implemented via lazy sampling. More precisely, instead of sampling τ, G_1 initializes a table $T_\tau \leftarrow [\,]$. Oracles op and H are then implemented via lazy sampling from G using table T_τ.

G_2: This game proceeds as G_1, but it 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_τ, it instead saves a fresh variable $\boldsymbol{R}_v$. Only after $\mathcal{A}$ and $\mathcal{E}$ are run, G_2 samples random $\boldsymbol{s}$ and $\boldsymbol{r}$ and evaluates the inputs and outputs of $\mathcal{A}$ at these points, and checks the winning condition as in G_1. Notice that in this game, table T_τ is populated exactly as table U_τ compiled by $\mathcal{E}$.

G_3: This game proceeds as G_2, but we omit the sampling of $\boldsymbol{s}$ and $\boldsymbol{r}$, and instead regard the winning condition as a set of (in)equalities between polynomials in $\boldsymbol{S}$ and $\boldsymbol{R}$.

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 be the event in G_2 that there are two different polynomials in $\mathrm{Dom}(T_\tau)$ 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 the adversary $\mathcal{B}$ in the Schwartz–Zippel game defined in Figure 28. Here, $\mathcal{B}$ simulates G_2 to $\mathcal{A}$ and then returns all entries in $\mathrm{Dom}(T_\tau)$. Notice that if Bad occurs, then $\mathcal{B}$ wins the SZ-game, and that T_τ contains at most $m + n + 3q_{\mathsf{op}} + q_{\mathsf{H}} + 1$ polynomials of degree at most d_P. By Lemma 1, $\Pr[\mathsf{Bad}] \leq (m + n + 3q_{\mathsf{op}} + q_{\mathsf{H}} + 1)^2 \cdot d_P/2p$.

$\mathrm{G}_2 \rightsquigarrow \mathrm{G}_3$. Let Bad' be the event in G_3 that $Q(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c}) \neq 0$ or $\boldsymbol{y}_i \neq \sum_{j=0}^{|\boldsymbol{X}|-1} \boldsymbol{w}_{ij}\boldsymbol{x}_j$ for some $1 \leq i \leq |\boldsymbol{Y}|$, but the corresponding equality holds when evaluating $\boldsymbol{S}$ and $\boldsymbol{R}$ at random $\boldsymbol{s}$ and $\boldsymbol{r}$. Then G_2 and G_3 are identical until Bad', and by the fundamental lemma of game playing we have $|\Pr[\mathrm{G}_3] - \Pr[\mathrm{G}_2]| \leq \Pr[\mathsf{Bad}']$.

We again bound the latter probability via Lemma 1. Consider the adversaries $\mathcal{B}'$ and $\mathcal{B}'_i$ in the Schwartz–Zippel game defined in Figure 28. Here, $\mathcal{B}'$ and $\mathcal{B}'_i$ simulate G_3 to $\mathcal{A}$ and then return $(Q(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c}), 0)$ and $(\boldsymbol{y}_i - \sum_{j=0}^{|\boldsymbol{X}|-1} \boldsymbol{w}_{ij}\boldsymbol{x}_j, 0)$, respectively. Notice that if Bad' occurs, then $\mathcal{B}'$ or $\mathcal{B}'_i$ win the SZ-game for some $1 \leq i \leq |\boldsymbol{Y}|$, and that the polynomials returned by $\mathcal{B}'$ and $\mathcal{B}'_i$ have total degree at most $d_Q d_P$ and d_P, respectively. By Lemma 1, $\Pr[\mathsf{Bad}'] \leq d_Q d_P/p + n \cdot d_P/p$.

We conclude the proof by showing that the winning probability of $\mathcal{A}$ in G_3 is zero. 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. If on the other hand Q is satisfied, we obtain

$$\sum_{i=1}^{|\boldsymbol{Y}|} \overline{Q_i}(\boldsymbol{S}) \left(\sum_{j=0}^{|\boldsymbol{X}|-1} \boldsymbol{w}_{ij}\boldsymbol{P}_j(\boldsymbol{S}) + \sum_l \boldsymbol{b}_{il}\boldsymbol{R}_l \right) + \overline{Q_0}(\boldsymbol{S}) = 0$$

as a polynomial in $\boldsymbol{S}$ and $\boldsymbol{R}$. We want to show that this implies $\boldsymbol{b}_{il} = 0$ for all $1 \leq i \leq |\boldsymbol{Y}|$ and all l, since the representation returned by $\mathcal{E}$ will be correct if that is the case. Looking

```
Game $\mathrm{G}_0$:
$\tau \twoheadleftarrow \mathrm{Inj}(\mathbb{Z}_p, \mathsf{G})$; $T_H \leftarrow [\,]$; $\boldsymbol{u}_0 \leftarrow \tau(1)$; $o \leftarrow 0$
$r_{\mathcal{A}} \twoheadleftarrow \mathcal{R}_{\mathcal{A}}$; $(Q, \boldsymbol{P}) \leftarrow \mathcal{A}_0^{\mathsf{op},\mathsf{H}}(\boldsymbol{u}_0; r_{\mathcal{A}})$
$\boldsymbol{s} \twoheadleftarrow \mathbb{Z}_p^k$; $\boldsymbol{x} \leftarrow \boldsymbol{P}(\boldsymbol{s})$; $\boldsymbol{x}_0 \leftarrow 1$
$\boldsymbol{u} \leftarrow \tau(\boldsymbol{x})$; $(\boldsymbol{v}, \boldsymbol{c}) \leftarrow \mathcal{A}_1^{\mathsf{op},\mathsf{H}}(\boldsymbol{u}; r_{\mathcal{A}})$
$\mathsf{trace}(\mathcal{A}) \leftarrow (r_{\mathcal{A}}, \boldsymbol{u}, \boldsymbol{h})$; $\boldsymbol{w} \twoheadleftarrow \mathcal{E}^{\mathsf{op},\mathsf{H}}(\mathsf{trace}(\mathcal{A}))$
$\boldsymbol{y} \leftarrow \tau^{-1}(\boldsymbol{v})$
return $(Q(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{c}) \neq 0) \wedge (Q(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c}) = 0)$
    $\wedge \big((\exists i)(\boldsymbol{y}_i \neq \sum_{j=0}^{|\boldsymbol{X}|-1} \boldsymbol{w}_{ij}\boldsymbol{x}_j)\big)$
```

```
Proc. $\mathsf{op}(h_1, h_2)$:
$x_1 \leftarrow \tau^{-1}(h_1)$; $x_2 \leftarrow \tau^{-1}(h_2)$
$o \leftarrow o + 1$; $\boldsymbol{h}_o \leftarrow \tau(x_1 + x_2)$; return $\boldsymbol{h}_o$

Proc. $\mathsf{H}(m)$:
if $m \notin \mathrm{Dom}(T_H)$ then $r \twoheadleftarrow \mathbb{Z}_p$; $T_H[m] \leftarrow r$
$r \leftarrow T_H[m]$; $o \leftarrow o + 1$; $\boldsymbol{h}_o \leftarrow \tau(r)$
return $\boldsymbol{h}_o$
```

```
Game $\mathrm{G}_1$:
$T_\tau, T_H \leftarrow [\,]$; $\boldsymbol{u}_0 \twoheadleftarrow \mathsf{G}$; $T_\tau[1] \leftarrow \boldsymbol{u}_0$; $o \leftarrow 0$
$r_{\mathcal{A}} \twoheadleftarrow \mathcal{R}_{\mathcal{A}}$; $(Q, \boldsymbol{P}) \leftarrow \mathcal{A}_0^{\mathsf{op},\mathsf{H}}(\boldsymbol{u}_0; r_{\mathcal{A}})$
$\boldsymbol{s} \twoheadleftarrow \mathbb{Z}_p^k$; $\boldsymbol{x} \leftarrow \boldsymbol{P}(\boldsymbol{s})$; $\boldsymbol{x}_0 \leftarrow 1$
for $j = 1$ to $|\boldsymbol{X}| - 1$ do
    if $(\boldsymbol{x}_j \notin \mathrm{Dom}(T_\tau))$ then
        $\boldsymbol{u}_j \twoheadleftarrow \mathsf{G} \setminus \mathrm{Rng}(T_\tau)$; $T_\tau[\boldsymbol{x}_j] \leftarrow \boldsymbol{u}_j$
    $\boldsymbol{u}_j \leftarrow T_\tau[\boldsymbol{x}_j]$
$(\boldsymbol{v}, \boldsymbol{c}) \leftarrow \mathcal{A}_1^{\mathsf{op},\mathsf{H}}(\boldsymbol{u}; r_{\mathcal{A}})$
$\mathsf{trace}(\mathcal{A}) \leftarrow (r_{\mathcal{A}}, \boldsymbol{u}, \boldsymbol{h})$; $\boldsymbol{w} \twoheadleftarrow \mathcal{E}^{\mathsf{op},\mathsf{H}}(\mathsf{trace}(\mathcal{A}))$
for $i = 1$ to $|\boldsymbol{Y}|$ do
    if $(\boldsymbol{v}_i \notin \mathrm{Rng}(T_\tau))$ then
        $\boldsymbol{y}_i \twoheadleftarrow \mathbb{Z}_p \setminus \mathrm{Dom}(T_\tau)$; $T_\tau[\boldsymbol{y}_i] \leftarrow \boldsymbol{v}_i$
    $\boldsymbol{y}_i \leftarrow T_\tau^{-1}[\boldsymbol{v}_i]$
return $(Q(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{c}) \neq 0) \wedge (Q(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c}) = 0)$
    $\wedge \big((\exists i)(\boldsymbol{y}_i \neq \sum_{j=0}^{|\boldsymbol{X}|-1} \boldsymbol{w}_{ij}\boldsymbol{x}_j)\big)$
```

```
Proc. $\mathsf{op}(h_1, h_2)$:
for $i = 1$ to 2 do
    if $(h_i \notin \mathrm{Rng}(T_\tau))$ then
        $x_i \leftarrow \mathbb{Z}_p \setminus \mathrm{Dom}(T_\tau)$; $T_\tau[x_i] \leftarrow h_i$
    $x_i \leftarrow T_\tau^{-1}[h_i]$
$x \leftarrow x_1 + x_2$
if $(x \notin \mathrm{Dom}(T_\tau))$ then
    $h \twoheadleftarrow \mathsf{G} \setminus \mathrm{Rng}(T_\tau)$; $T_\tau[x] \leftarrow h$
$o \leftarrow o + 1$; $\boldsymbol{h}_o \leftarrow T_\tau[x]$; return $\boldsymbol{h}_o$

Proc. $\mathsf{H}(m)$:
if $(m \notin \mathrm{Dom}(T_H))$ then $r \twoheadleftarrow \mathbb{Z}_p$; $T_H[m] \leftarrow r$
$r \leftarrow T_H[m]$
if $(r \notin \mathrm{Dom}(T_\tau))$ then
    $h \twoheadleftarrow \mathsf{G} \setminus \mathrm{Rng}(T_\tau)$; $T_\tau[r] \leftarrow h$
$o \leftarrow o + 1$; $\boldsymbol{h}_o \leftarrow T_\tau[r]$; return $\boldsymbol{h}_o$
```

```
Game $\mathrm{G}_2$:
$T_\tau, T_H \leftarrow [\,]$; $o, v \leftarrow 0$; $\boldsymbol{u}_0 \twoheadleftarrow \mathsf{G}$; $T_\tau[1] \leftarrow \boldsymbol{u}_0$
$r_{\mathcal{A}} \twoheadleftarrow \mathcal{R}_{\mathcal{A}}$; $(Q, \boldsymbol{P}) \leftarrow \mathcal{A}_0^{\mathsf{op},\mathsf{H}}(\boldsymbol{u}_0; r_{\mathcal{A}})$
$\boldsymbol{x} \leftarrow \boldsymbol{P}(\boldsymbol{S})$; $\boldsymbol{x}_0 \leftarrow 1$
for $j = 1$ to $|\boldsymbol{X}| - 1$ do
    if $(\boldsymbol{x}_j \notin \mathrm{Dom}(T_\tau))$ then
        $\boldsymbol{u}_j \twoheadleftarrow \mathsf{G} \setminus \mathrm{Rng}(T_\tau)$; $T_\tau[\boldsymbol{x}_j] \leftarrow \boldsymbol{u}_j$
    $\boldsymbol{u}_j \leftarrow T_\tau[\boldsymbol{x}_j]$
$(\boldsymbol{v}, \boldsymbol{c}) \leftarrow \mathcal{A}_1^{\mathsf{op},\mathsf{H}}(\boldsymbol{u}; r_{\mathcal{A}})$
$\mathsf{trace}(\mathcal{A}) \leftarrow (r_{\mathcal{A}}, \boldsymbol{u}, \boldsymbol{h})$; $\boldsymbol{w} \twoheadleftarrow \mathcal{E}^{\mathsf{op},\mathsf{H}}(\mathsf{trace}(\mathcal{A}))$
for $i = 1$ to $|\boldsymbol{Y}|$ do
    if $(\boldsymbol{v}_i \notin \mathrm{Rng}(T_\tau))$ then $v \leftarrow v + 1$; $T_\tau[\boldsymbol{R}_v] \leftarrow \boldsymbol{v}_i$
    $\boldsymbol{y}_i \leftarrow T_\tau^{-1}[\boldsymbol{v}_i]$
    parse $\boldsymbol{y}_i = \sum_{j=0}^{|\boldsymbol{X}|-1} \boldsymbol{w}_{ij}\boldsymbol{P}_j(\boldsymbol{S}) + \sum_l \boldsymbol{b}_{il}\boldsymbol{R}_l$
$\boldsymbol{s} \twoheadleftarrow \mathbb{Z}_p^k$; $\boldsymbol{r} \twoheadleftarrow \mathbb{Z}_p^{2q_{\mathsf{op}}+q_{\mathsf{H}}+|\boldsymbol{Y}|}$; $\boldsymbol{x} \leftarrow \boldsymbol{P}(\boldsymbol{s})$
for $i = 1$ to $|\boldsymbol{Y}|$ do $\boldsymbol{y}_i \leftarrow \sum_{j=0}^{|\boldsymbol{X}|-1} \boldsymbol{w}_{ij}\boldsymbol{x}_j + \sum_l \boldsymbol{b}_{il}\boldsymbol{r}_l$
return $(Q(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{c}) \neq 0) \wedge (Q(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c}) = 0)$
    $\wedge \big((\exists i)(\boldsymbol{y}_i \neq \sum_{j=0}^{|\boldsymbol{X}|-1} \boldsymbol{w}_{ij}\boldsymbol{x}_j)\big)$

Proc. $\mathsf{op}(h_1, h_2)$:
for $i = 1$ to 2 do
    if $(h_i \notin \mathrm{Rng}(T_\tau))$ then $v \leftarrow v + 1$; $T_\tau[\boldsymbol{R}_v] \leftarrow h_i$
    $x_i \leftarrow T_\tau^{-1}[h_i]$
$x \leftarrow x_1 + x_2$
if $(x \notin \mathrm{Dom}(T_\tau))$ then $h \twoheadleftarrow \mathsf{G}\setminus\mathrm{Rng}(T_\tau)$; $T_\tau[x] \leftarrow h$
$o \leftarrow o + 1$; $\boldsymbol{h}_o \leftarrow T_\tau[x]$; return $\boldsymbol{h}_o$
```

```
Game $\mathrm{G}_3$:
$T_\tau, T_H \leftarrow [\,]$; $o, v \leftarrow 0$; $\boldsymbol{u}_0 \twoheadleftarrow \mathsf{G}$; $T_\tau[1] \leftarrow \boldsymbol{u}_0$
$r_{\mathcal{A}} \twoheadleftarrow \mathcal{R}_{\mathcal{A}}$; $(Q, \boldsymbol{P}) \leftarrow \mathcal{A}_0^{\mathsf{op},\mathsf{H}}(\boldsymbol{u}_0; r_{\mathcal{A}})$
$\boldsymbol{x} \leftarrow \boldsymbol{P}(\boldsymbol{S})$; $\boldsymbol{x}_0 \leftarrow 1$
for $j = 1$ to $|\boldsymbol{X}| - 1$ do
    if $(\boldsymbol{x}_j \notin \mathrm{Dom}(T_\tau))$ then
        $\boldsymbol{u}_j \twoheadleftarrow \mathsf{G} \setminus \mathrm{Rng}(T_\tau)$; $T_\tau[\boldsymbol{x}_j] \leftarrow \boldsymbol{u}_j$
    $\boldsymbol{u}_j \leftarrow T_\tau[\boldsymbol{x}_j]$
$(\boldsymbol{v}, \boldsymbol{c}) \leftarrow \mathcal{A}_1^{\mathsf{op},\mathsf{H}}(\boldsymbol{u}; r_{\mathcal{A}})$
$\mathsf{trace}(\mathcal{A}) \leftarrow (r_{\mathcal{A}}, \boldsymbol{u}, \boldsymbol{h})$; $\boldsymbol{w} \twoheadleftarrow \mathcal{E}^{\mathsf{op},\mathsf{H}}(\mathsf{trace}(\mathcal{A}))$
for $i = 1$ to $|\boldsymbol{Y}|$ do
    if $(\boldsymbol{v}_i \notin \mathrm{Rng}(T_\tau))$ then
        $v \leftarrow v + 1$; $T_\tau[\boldsymbol{R}_v] \leftarrow \boldsymbol{v}_i$
    $\boldsymbol{y}_i \leftarrow T_\tau^{-1}[\boldsymbol{v}_i]$
return $(Q(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{c}) \neq 0) \wedge (Q(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c}) = 0)$
    $\wedge \big((\exists i)(\boldsymbol{y}_i \neq \sum_{j=0}^{|\boldsymbol{X}|-1} \boldsymbol{w}_{ij}\boldsymbol{x}_j)\big)$

Proc. $\mathsf{H}(m)$:
if $(m \notin \mathrm{Dom}(T_H))$ then
    $v \leftarrow v + 1$; $T_\tau[m] \leftarrow \boldsymbol{R}_v$
$r \leftarrow T_H[m]$
if $(r \notin \mathrm{Dom}(T_\tau))$ then
    $h \twoheadleftarrow \mathsf{G} \setminus \mathrm{Rng}(T_\tau)$; $T_\tau[r] \leftarrow h$
$o \leftarrow o + 1$; $\boldsymbol{h}_o \leftarrow T_\tau[r]$; return $\boldsymbol{h}_o$
```

Figure 27: Code of the intermediate games in the proof of Inequality (11).

```
Adversaries $\mathcal{B}/\mathcal{B}'/\mathcal{B}'_i$:
$T_\tau, T_H \leftarrow []$; $o, v \leftarrow 0$; $\boldsymbol{u}_0 \twoheadleftarrow \mathsf{G}$; $T_\tau[1] \leftarrow \boldsymbol{u}_0$; $r_{\mathcal{A}} \twoheadleftarrow \mathcal{R}_{\mathcal{A}}$
$(Q, \boldsymbol{P}) \leftarrow \mathcal{A}_0^{\mathsf{op},\mathsf{H}}(\boldsymbol{u}_0; r_{\mathcal{A}})$; $\boldsymbol{x} \leftarrow \boldsymbol{P}(\boldsymbol{S})$; $\boldsymbol{x}_0 \leftarrow 1$
for $j = 1$ to $|\boldsymbol{X}| - 1$ do
  if $(\boldsymbol{x}_j \notin \mathrm{Dom}(T_\tau))$ then $\boldsymbol{u}_j \twoheadleftarrow \mathsf{G} \setminus \mathrm{Rng}(T_\tau)$; $T_\tau[\boldsymbol{x}_j] \leftarrow \boldsymbol{u}_j$
  $\boldsymbol{u}_j \leftarrow T_\tau[\boldsymbol{x}_j]$
$(\boldsymbol{v}, \boldsymbol{c}) \leftarrow \mathcal{A}_1^{\mathsf{op},\mathsf{H}}(\boldsymbol{u}; r_{\mathcal{A}})$; $\mathsf{trace}(\mathcal{A}) \leftarrow (r_{\mathcal{A}}, \boldsymbol{u}, \boldsymbol{h})$; $\boldsymbol{w} \twoheadleftarrow \mathcal{E}^{\mathsf{op},\mathsf{H}}(\mathsf{trace}(\mathcal{A}))$
for $i = 1$ to $|\boldsymbol{Y}|$ do
  if $(\boldsymbol{v}_i \notin \mathrm{Rng}(T_\tau))$ then $v \leftarrow v + 1$; $T_\tau[\boldsymbol{R}_v] \leftarrow \boldsymbol{v}_i$
  $\boldsymbol{y}_i \leftarrow T_\tau^{-1}[\boldsymbol{v}_i]$
$\mathcal{B}$: return $\mathrm{Dom}(T_\tau)$      $\mathcal{B}'$: return $(Q(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c}), 0)$      $\mathcal{B}'_i$: return $\big(\boldsymbol{y}_i - \sum_{j=0}^{|\boldsymbol{X}|-1} \boldsymbol{w}_{ij} \boldsymbol{x}_j, 0\big)$
```

Figure 28: Definition of the adversaries $\mathcal{B}$, $\mathcal{B}'$ and $\mathcal{B}'_i$ from the proof of Theorem 6. In all cases, oracles op and H are defined as in Figure 27 (bottom).

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

$$\sum_{i=1}^{|\boldsymbol{Y}|} \overline{Q_i}(\boldsymbol{S}) \boldsymbol{b}_{il} = 0\,.$$

Recall that, by assumption, polynomials $\overline{Q_i}$ are linearly independent, which means that $\boldsymbol{b}_{il} = 0$ for all $1 \leq i \leq |\boldsymbol{Y}|$ and all l. This proves that if $\mathcal{A}$ returns a valid output, then $\mathcal{E}$ returns an accurate representation of $\boldsymbol{y}$ in terms of $\boldsymbol{x}$, which means that $\Pr[\mathrm{G}_2] = 0$.

Collecting all the terms above, we obtain

$$\begin{aligned}\mathrm{Adv}^{\mathrm{uk}}_{p,\mathsf{G},\mathcal{S},\mathcal{A},\mathcal{E}} \leq \frac{(m+n+3q_{\mathsf{op}}+q_{\mathsf{H}}+1)^2 \cdot d_P}{2p} + \frac{d_Q d_P}{p} + \frac{n d_P}{p} \\ \leq \mathcal{O}\left(\frac{(m+n+q_{\mathsf{op}}+q_{\mathsf{H}}+d_Q)^2 \cdot d_P}{p}\right),\end{aligned}$$

which concludes the proof. □

Appendix B: Soundness of Linear UK in AGM-H

In this appendix, we give a self-contained proof of the hardness of the UK assumption 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 and $d_P, d_Q \colon \mathbb{N} \to \mathbb{N}$ be polynomials. Consider the class of PPT algorithms $\mathfrak{A}$ and the source $\mathcal{S}$ defined as follows:*

1. *For every $\mathcal{A}_0 \in \mathfrak{A}$, the relation polynomial Q returned by $\mathcal{A}_0$ is of the form*

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

2. *For every $\mathcal{A}_0 \in \mathfrak{A}$, every $(Q, \boldsymbol{P})$ returned by $\mathcal{A}_0$, and every $\boldsymbol{c} \in \mathbb{Z}_p^{|\boldsymbol{C}|}$, the polynomials $\overline{Q_i}$, $1 \leq i \leq |\boldsymbol{Y}|$, are linearly independent;*
3. *For every $\mathcal{A}_0 \in \mathfrak{A}$ and every $(Q, \boldsymbol{P})$ returned by $\mathcal{A}_0$, every polynomial P in $\boldsymbol{P}$ has total degree at most d_P, and Q has total degree at most d_Q;*

```
Extractor $\mathcal{E}^{\mathsf{H}}(\mathsf{trace}(\mathcal{A}))$:
parse $\mathsf{trace}(\mathcal{A}) = (r_{\mathcal{A}}, \gamma, [\boldsymbol{x}], [\boldsymbol{h}])$; $o \leftarrow 0$
$(Q, \boldsymbol{P}) \leftarrow \mathcal{A}_0^{\overline{\mathsf{H}}}(\gamma; r_{\mathcal{A}})$; $(\boldsymbol{w}, \boldsymbol{v}, \boldsymbol{c}) \leftarrow \mathcal{A}_1^{\overline{\mathsf{H}}}(\gamma, [\boldsymbol{x}]; r_{\mathcal{A}})$
    // $\mathcal{A}$ encodes elements $[\boldsymbol{y}_i] = \prod_{j=0}^{|\boldsymbol{X}|-1} [\boldsymbol{w}_{ij}\boldsymbol{x}_j] \cdot \prod_l [\boldsymbol{v}_{il}\boldsymbol{h}_l]$
return $\boldsymbol{w}$

Oracle $\overline{\mathsf{H}}(m)$:
$o \leftarrow o + 1$
return $[\boldsymbol{h}_o]$
```

```
Adversary $\mathcal{B}(\gamma, [t], [t^2], \ldots, [t^{d_P(\lambda)}])$:
$o \leftarrow 0$; $U \leftarrow [\,]$; $\mathsf{S} \leftarrow \emptyset$; $r_{\mathcal{A}} \twoheadleftarrow \mathcal{R}_{\mathcal{A}}(\lambda)$; $(Q, \boldsymbol{P}) \leftarrow \mathcal{A}_0^{\mathsf{H}}(\gamma; r_{\mathcal{A}})$
$\boldsymbol{\rho} \twoheadleftarrow \mathbb{Z}_p^k$; $\boldsymbol{\sigma} \twoheadleftarrow \mathbb{Z}_p^{*k}$; $\boldsymbol{X}(T) \leftarrow \boldsymbol{P}(\boldsymbol{\rho} + \boldsymbol{\sigma}T)$; $[\boldsymbol{x}] \leftarrow [\boldsymbol{X}(t)]$
$(\boldsymbol{w}, \boldsymbol{v}, \boldsymbol{c}) \leftarrow \mathcal{A}_1^{\mathsf{H}}(\gamma, [\boldsymbol{x}]; r_{\mathcal{A}})$
    // $\mathcal{A}$ encodes elements $[\boldsymbol{y}_i] = \prod_{j=0}^{|\boldsymbol{X}|-1} [\boldsymbol{w}_{ij}\boldsymbol{x}_j] \cdot \prod_l [\boldsymbol{v}_{il}\boldsymbol{h}_l]$
for $i = 1$ to $|\boldsymbol{Y}|$ do $\boldsymbol{Y}_i(T) \leftarrow \sum_{j=0}^{|\boldsymbol{X}|-1} \boldsymbol{w}_{ij}\boldsymbol{X}_j(T) + \sum_l \boldsymbol{v}_{il}H_l(T)$
$Q'(T) \leftarrow Q(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{c})$
if $(Q'(T) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{Berlekamp}(Q', p)$
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
    $o \leftarrow o + 1$
    $\alpha_o \twoheadleftarrow \mathbb{Z}_p$
    $\beta_o \twoheadleftarrow \mathbb{Z}_p^*$
    $H_o(T) \leftarrow \alpha_o + \beta_o T$
    $U[m] \leftarrow [H_o(t)]$
return $U[m]$
```

Figure 29: *Top:* Extractor $\mathcal{E}$ for the algebraic adversary $\mathcal{A}$ in the UK game. *Bottom:* Adversary $\mathcal{B}$ against d_P-DL.

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

If d_P-DL holds for Γ, then UK *holds for $(\Gamma, \mathcal{S}, \mathfrak{A})$ in the* AGM-H. *More precisely, for every low-degree PPT adversary $\mathcal{A}$ with $\mathcal{A}_0 \in \mathfrak{A}$, there exist an extractor $\mathcal{E}$ and an adversary $\mathcal{B}$ against d_P-DL, both with approximately the same running time as $\mathcal{A}$, such that*

$$\mathrm{Adv}^{\mathrm{uk}}_{\Gamma,\mathcal{S},\mathcal{A},\mathcal{E}}(\lambda) \leq \left(1 - \frac{d_P(\lambda) d_Q(\lambda)}{2^{\lambda-1} - 1}\right)^{-1} \cdot \mathrm{Adv}^{d_P\text{-dl}}_{\Gamma,\mathcal{B}}(\lambda)\,. \tag{12}$$

Here, k is an upper bound on the number of variables of every polynomial P in $\boldsymbol{P}$, and we let $\overline{Q_i}(\boldsymbol{S}) \coloneqq Q_i(\boldsymbol{P}(\boldsymbol{S}), \boldsymbol{c})$, where we set $\boldsymbol{P}_0(\boldsymbol{S}) \coloneqq 1$.

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 Figure 29 (top). This extractor essentially re-runs $\mathcal{A}$ on its view to obtain $\mathcal{A}$'s output $(\boldsymbol{w}, \boldsymbol{v}, \boldsymbol{c})$. Recall that this means that $\mathcal{A}$ encodes group elements $[\boldsymbol{y}_i] = \prod_{j=0}^{|\boldsymbol{X}|-1} [\boldsymbol{w}_{ij}\boldsymbol{x}_j] \cdot \prod_l [\boldsymbol{v}_{il}\boldsymbol{h}_l]$, where $[\boldsymbol{x}]$ and $[\boldsymbol{h}]$ are the vectors of input group elements and of hash replies. The extractor then simply ignores the coefficients $\boldsymbol{v}$ pertaining to the hash values and returns $\boldsymbol{w}$. Clearly, extractor $\mathcal{E}$ will be correct if $\boldsymbol{v} = 0$ in the representation returned by $\mathcal{A}$.

We now show that if $\mathcal{A}$ returns a valid output and d_P-DL holds for Γ, this will likely be the case. To that end, consider the adversary $\mathcal{B}$ playing the d_P-DL game for Γ defined in Figure 29 (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 d_P-DL instance it is tasked with solving. Note that this is possible because $\mathcal{B}$ is given the power-DL challenge up to power $d_P(\lambda)$. By construction, if $\mathcal{A}$ returns an output that satisfies Q, then t is a root of the polynomial $Q'(T)$ defined by $\mathcal{B}$. This means that $\mathcal{B}$ will be able to find t by inspecting the roots of Q' whenever $Q'(T) \neq 0$. We show that the latter happens with overwhelming probability if $\boldsymbol{v} \neq 0$, which means that $\boldsymbol{v}$ must vanish if d_P-DL holds for Γ.

We now show how to use adversary $\mathcal{B}$ to prove Inequality (12) for $\mathcal{A}$ and $\mathcal{E}$. To that end, consider the following sequence of games (the formal description of which can be found in Figure 30):

```
Game $\mathrm{G}_0(\lambda)$:
$\gamma \twoheadleftarrow \Gamma(1^\lambda)$; $t \twoheadleftarrow \mathbb{Z}_p$; $o \leftarrow 0$; $U \leftarrow [\,]$; $\mathsf{S} \leftarrow \emptyset$
$r_\mathcal{A} \twoheadleftarrow \mathcal{R}_\mathcal{A}(\lambda)$; $(Q, \boldsymbol{P}) \leftarrow \mathcal{A}_0^{\mathsf{H}}(\gamma; r_\mathcal{A})$
$\boldsymbol{\rho} \twoheadleftarrow \mathbb{Z}_p^k$; $\boldsymbol{\sigma} \twoheadleftarrow \mathbb{Z}_p^{*k}$; $\boldsymbol{X}(T) \leftarrow \boldsymbol{P}(\boldsymbol{\rho} + \boldsymbol{\sigma}T)$; $[\boldsymbol{x}] \leftarrow [\boldsymbol{X}(t)]$
$(\boldsymbol{w}, \boldsymbol{v}, \boldsymbol{c}) \leftarrow \mathcal{A}_1^{\mathsf{H}}(\gamma, [\boldsymbol{x}]; r_\mathcal{A})$
for $i = 1$ to $|\boldsymbol{Y}|$ do $\boldsymbol{Y}_i(T) \leftarrow \sum_{j=0}^{|\boldsymbol{X}|-1} \boldsymbol{w}_{ij}\boldsymbol{X}_j(T) + \sum_l \boldsymbol{v}_{il} H_l(T)$
$Q'(T) \leftarrow Q(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{c})$; if $(Q'(T) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{Berlekamp}(Q', p)$
$t' \leftarrow 0$; for $z \in \mathsf{S}$ do if $([z] = [t])$ then return $t' \leftarrow z$; break
return $(t = t')$
```

```
Oracle $\mathsf{H}(m)$:
if $(m \notin \mathrm{Dom}(U))$ then
    $o \leftarrow o + 1$
    $\alpha_o \twoheadleftarrow \mathbb{Z}_p$
    $\beta_o \twoheadleftarrow \mathbb{Z}_p^*$
    $H_o(T) \leftarrow \alpha_o + \beta_o T$
    $U[m] \leftarrow [H_o(t)]$
return $U[m]$
```

```
Game $\mathrm{G}_1(\lambda)$:
$\gamma \twoheadleftarrow \Gamma(1^\lambda)$; $t \twoheadleftarrow \mathbb{Z}_p$; $o \leftarrow 0$; $U \leftarrow [\,]$; $\mathsf{S} \leftarrow \emptyset$
$r_\mathcal{A} \twoheadleftarrow \mathcal{R}_\mathcal{A}(\lambda)$; $(Q, \boldsymbol{P}) \leftarrow \mathcal{A}_0^{\mathsf{H}}(\gamma; r_\mathcal{A})$
$\boldsymbol{\rho}' \twoheadleftarrow \mathbb{Z}_p^k$; $\boldsymbol{\sigma}' \twoheadleftarrow \mathbb{Z}_p^{*k}$; $\boldsymbol{X}(T) \leftarrow \boldsymbol{P}(\boldsymbol{\rho}' + \boldsymbol{\sigma}'(T - t))$; $[\boldsymbol{x}] \leftarrow [\boldsymbol{X}(t)]$
$(\boldsymbol{w}, \boldsymbol{v}, \boldsymbol{c}) \leftarrow \mathcal{A}_1^{\mathsf{H}}(\gamma, [\boldsymbol{x}]; r_\mathcal{A})$
for $i = 1$ to $|\boldsymbol{Y}|$ do $\boldsymbol{Y}_i(T) \leftarrow \sum_{j=0}^{|\boldsymbol{X}|-1} \boldsymbol{w}_{ij}\boldsymbol{X}_j(T) + \sum_l \boldsymbol{v}_{il} H_l(T)$
$Q'(T) \leftarrow Q(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{c})$; if $(Q'(T) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{Berlekamp}(Q', p)$
$t' \leftarrow 0$; for $z \in \mathsf{S}$ do if $([z] = [t])$ then return $t' \leftarrow z$; break
return $(t = t')$
```

```
Oracle $\mathsf{H}(m)$:
if $(m \notin \mathrm{Dom}(U))$ then
    $o \leftarrow o + 1$
    $\alpha'_o \twoheadleftarrow \mathbb{Z}_p$
    $\beta'_o \twoheadleftarrow \mathbb{Z}_p^*$
    $H_o(T) \leftarrow \alpha'_o + \beta'_o(T - t)$
    $U[m] \leftarrow [H_o(t)]$
return $U[m]$
```

```
Game $\mathrm{G}_2(\lambda)$:
$\gamma \twoheadleftarrow \Gamma(1^\lambda)$; $t \twoheadleftarrow \mathbb{Z}_p$; $o \leftarrow 0$; $U \leftarrow [\,]$; $\mathsf{S} \leftarrow \emptyset$
$r_\mathcal{A} \twoheadleftarrow \mathcal{R}_\mathcal{A}(\lambda)$; $(Q, \boldsymbol{P}) \leftarrow \mathcal{A}_0^{\mathsf{H}}(\gamma; r_\mathcal{A})$
$\boldsymbol{\rho}' \twoheadleftarrow \mathbb{Z}_p^k$; $\boldsymbol{X}(T, \boldsymbol{\Sigma}') \leftarrow \boldsymbol{P}(\boldsymbol{\rho}' + \boldsymbol{\Sigma}'(T - t))$; $[\boldsymbol{x}] \leftarrow [\boldsymbol{X}(t, \boldsymbol{\Sigma}')]$
$(\boldsymbol{w}, \boldsymbol{v}, \boldsymbol{c}) \leftarrow \mathcal{A}_1^{\mathsf{H}}(\gamma, [\boldsymbol{x}]; r_\mathcal{A})$
for $i = 1$ to $|\boldsymbol{Y}|$ do
    $\boldsymbol{Y}_i(T, \boldsymbol{\Sigma}', \boldsymbol{B}') \leftarrow \sum_{j=0}^{|\boldsymbol{X}|-1} \boldsymbol{w}_{ij}\boldsymbol{X}_j(T, \boldsymbol{\Sigma}') + \sum_l \boldsymbol{v}_{il} H_l(T, \boldsymbol{B}'_l)$
$Q''(T, \boldsymbol{\Sigma}', \boldsymbol{B}') \leftarrow Q(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{c})$
$\boldsymbol{\sigma}' \twoheadleftarrow \mathbb{Z}_p^{*k}$; $\boldsymbol{\beta}' \twoheadleftarrow \mathbb{Z}_p^{*o}$; $Q'(T) \leftarrow Q''(T, \boldsymbol{\sigma}', \boldsymbol{\beta}')$
if $(Q'(T) \neq 0)$ then $\mathsf{S} \leftarrow \mathsf{Berlekamp}(Q', p)$
$t' \leftarrow 0$; for $z \in \mathsf{S}$ do if $([z] = [t])$ then return $t' \leftarrow z$; break
return $(t = t')$
```

```
Oracle $\mathsf{H}(m)$:
if $(m \notin \mathrm{Dom}(U))$ then
    $o \leftarrow o + 1$
    $\alpha'_o \twoheadleftarrow \mathbb{Z}_p$
    $H_o(T, \boldsymbol{B}') \leftarrow$
        $\alpha'_o + \boldsymbol{B}'_o(T - t)$
    $U[m] \leftarrow [H_o(t, \boldsymbol{B}')]$
return $U[m]$
```

Figure 30: Code of the intermediate games in the proof of Inequality (12). In all figures, k is an upper bound on the number of variables appearing in any polynomial P in $\boldsymbol{P}$.

G_0: This is the original d_P-DL game for Γ run with adversary $\mathcal{B}$.

G_1: This game proceeds as G_0, but performs variable substitutions $\boldsymbol{\rho}' = \boldsymbol{\rho} + \boldsymbol{\sigma}t$ and $\boldsymbol{\sigma}' = \boldsymbol{\sigma}$, and $\alpha'_l = \alpha_l + \beta_l t$ and $\beta'_l = \beta_l$, in polynomials $\boldsymbol{X}$ and H_l. More precisely, polynomials $\boldsymbol{X}(T)$ are now defined as $\boldsymbol{X}(T) \leftarrow \boldsymbol{P}(\boldsymbol{\rho}' + \boldsymbol{\sigma}'(T - t))$ for random $\boldsymbol{\rho}'$ and invertible $\boldsymbol{\sigma}'$. Similarly, upon a query m to H, game G_2 samples random α'_l and invertible β'_l, and sets $H_l(T) \leftarrow \alpha'_l + \beta'_l(T - t)$. Inputs $[\boldsymbol{x}]$ and hash replies $U[m]$ are still computed as $[\boldsymbol{X}(t)] = [\boldsymbol{P}(\boldsymbol{\rho}')]$ and $[H_l(t)] = [\alpha'_l]$, respectively.

G_2: This game proceeds as G_1, but polynomials $\boldsymbol{X}$ and H_l are now defined as $\boldsymbol{X}(T, \boldsymbol{\Sigma}') \leftarrow \boldsymbol{P}(\boldsymbol{\rho}' + \boldsymbol{\Sigma}'(T - t))$ and $H_l(T, \boldsymbol{B}') \leftarrow \alpha'_l + \boldsymbol{B}'_l(T - t)$, where $\boldsymbol{\Sigma}'$ is a new vector of variables and $\boldsymbol{B}'_l$ 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 samples random $\boldsymbol{\sigma}'$ and invertible $\boldsymbol{\beta}'$, sets $Q'(T) \leftarrow Q''(T, \boldsymbol{\sigma}', \boldsymbol{\beta}')$, and checks if $Q'(T) = 0$. From here on, game G_2 proceeds as G_1.

We now argue that subsequent games have identical success probabilities.

$\mathrm{G}_0 \rightsquigarrow \mathrm{G}_1$. Observe that for every fixed $\lambda \in \mathbb{N}$, γ returned by $\Gamma(1^\lambda)$, $t \in \mathbb{Z}_p$, and randomness $r_\mathcal{A}$ returned by $\mathcal{R}_\mathcal{A}(\lambda)$, the random variates $\boldsymbol{\rho}'$, $\boldsymbol{\sigma}'$, α'_l and β'_l in G_1 are related to the random variates $\boldsymbol{\rho}$, $\boldsymbol{\sigma}$, α_l and β_l in G_0 via the transformation $\mathrm{diag}(\begin{smallmatrix} 1 & t \\ 0 & 1 \end{smallmatrix})$, which is invertible. Consequently, $\Pr[\mathrm{G}_0] = \Pr[\mathrm{G}_1]$, since there is a one-to-one correspondence between the random variables in the two games.

$\mathrm{G}_1 \rightsquigarrow \mathrm{G}_2$. Notice that $\mathcal{A}$ is oblivious to the changes to polynomials $\boldsymbol{X}$ and H_l, so the simulation of $\mathcal{A}$ is identical in both games. Indeed, in both games inputs to $\mathcal{A}$ and hash replies are computed in the same way. After running $\mathcal{A}$, G_2 derives the same polynomial Q' computed in G_1 by substituting random $\boldsymbol{\sigma}'$ and $\boldsymbol{\beta}'$ into Q'', so the winning condition is again the same in both games. Therefore, $\Pr[\mathrm{G}_1] = \Pr[\mathrm{G}_2]$.

We conclude the proof by studying the winning probability in G_2. First, notice that in this game adversary $\mathcal{A}$ plays the UK game, since the inputs of $\mathcal{A}$ are obtained by evaluating $\boldsymbol{P}$ at random points and hash replies are random group elements. Now for any $\lambda \in \mathbb{N}$, γ returned by $\Gamma(1^\lambda)$, $t \in \mathbb{Z}_p$, randomness $r_\mathcal{A}$ returned by $\mathcal{R}_\mathcal{A}(\lambda)$, and vectors $\boldsymbol{\rho}'$ and $\boldsymbol{\alpha}'$ in $\mathbb{Z}_p$, denote by $\mathrm{G}' := \mathrm{G}'(\lambda, \gamma, t, r_\mathcal{A}, \boldsymbol{\rho}', \boldsymbol{\alpha}')$ the game $\mathrm{G}_2(\lambda)$ with these random choices fixed. Then $\Pr[\mathrm{G}_2(\lambda)] = \sum_{(\gamma,t,r_\mathcal{A},\boldsymbol{\rho}',\boldsymbol{\alpha}')} \Pr[\mathrm{G}'] \Pr[\gamma, t, r_\mathcal{A}, \boldsymbol{\rho}', \boldsymbol{\alpha}']$, where $\Pr[\gamma, t, r_\mathcal{A}, \boldsymbol{\rho}', \boldsymbol{\alpha}']$ denotes the probability that such a tuple is drawn in $\mathrm{G}_2(\lambda)$, and the sum extends over all $(\gamma, t, r_\mathcal{A}, \boldsymbol{\rho}', \boldsymbol{\alpha}')$ such that $\Pr[\gamma, t, r_\mathcal{A}, \boldsymbol{\rho}', \boldsymbol{\alpha}'] \neq 0$.

Now consider the set X of all $(\gamma, t, r_\mathcal{A}, \boldsymbol{\rho}', \boldsymbol{\alpha}')$ in the sum above such that $\mathcal{A}$ returns $(Q, \boldsymbol{P})$ and $(\boldsymbol{w}, \boldsymbol{v}, \boldsymbol{c})$ for which the relation polynomial in UK is satisfied and extractor $\mathcal{E}$ fails to compute a correct representation of the outputs. Notice that

$$\sum_{(\gamma,t,r_\mathcal{A},\boldsymbol{\rho}',\boldsymbol{\alpha}')\in\mathsf{X}} \Pr[\gamma, t, r_\mathcal{A}, \boldsymbol{\rho}', \boldsymbol{\alpha}'] = \mathrm{Adv}^{\mathrm{uk}}_{\Gamma,\mathcal{S},\mathcal{A},\mathcal{E}}(\lambda)\,.$$

We claim that for any $(\gamma, t, r_\mathcal{A}, \boldsymbol{\rho}', \boldsymbol{\alpha}') \in \mathsf{X}$, $\Pr[\mathrm{G}'] \geq 1 - d_P(\lambda) d_Q(\lambda)/(2^{\lambda-1}-1)$. Indeed, fix any $(\gamma, t, r_\mathcal{A}, \boldsymbol{\rho}', \boldsymbol{\alpha}') \in \mathsf{X}$. Since $\mathcal{E}$ fails to return a correct representation of the output of $\mathcal{A}$, it must be $\boldsymbol{v} \neq 0$, i.e., there exist $1 \leq i^* \leq |\boldsymbol{Y}|$ and l^* such that $\boldsymbol{v}_{i^*l^*} \neq 0$. We now claim that the polynomial $Q''(T, \boldsymbol{\Sigma}', \boldsymbol{B}')$ constructed in G_2 after running $\mathcal{A}$ is not identically zero with overwhelming probability. Indeed, consider the polynomial

$$\begin{aligned} R(\boldsymbol{S}, \boldsymbol{H}) &:= Q\left(\boldsymbol{P}(\boldsymbol{S}), \sum_{j=0}^{|\boldsymbol{X}|-1} \boldsymbol{w}_{ij} \boldsymbol{P}_j(\boldsymbol{S}) + \sum_l \boldsymbol{v}_{il} \boldsymbol{H}_l, \boldsymbol{c}\right) \\ &= \sum_{i=1}^{|\boldsymbol{Y}|} Q_i(\boldsymbol{P}(\boldsymbol{S}), \boldsymbol{c}) \left(\sum_{j=0}^{|\boldsymbol{X}|-1} \boldsymbol{w}_{ij} \boldsymbol{P}_j(\boldsymbol{S}) + \sum_l \boldsymbol{v}_{il} \boldsymbol{H}_l\right) + Q_0(\boldsymbol{P}(\boldsymbol{S}), \boldsymbol{c})\,. \end{aligned}$$

Polynomial R is of total degree at most $d_P(\lambda) d_Q(\lambda)$ and not identically zero, because the coefficient of $\boldsymbol{H}_{l^*}$ is $\sum_{i=1}^{|\boldsymbol{Y}|} \overline{Q_i} \boldsymbol{v}_{il^*}$, which is non-zero since the polynomials $\overline{Q_i}$ are assumed to be linearly independent and $\boldsymbol{v}_{i^*l^*} \neq 0$. Now notice that

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

which again is non-zero by Lemma 2 and of degree in T at most $d_P(\lambda) d_Q(\lambda)$. Moreover, by Lemma 2, the leading coefficient in T of $Q''(T, \boldsymbol{\Sigma}', \boldsymbol{B}')$ is a polynomial in $\boldsymbol{\Sigma}', \boldsymbol{B}'$ of total degree at most $d_P(\lambda) d_Q(\lambda)$, which for random invertible $\boldsymbol{\sigma}'$ and $\boldsymbol{\beta}'$ will be zero with probability at most $d_P(\lambda) d_Q(\lambda)/(2^{\lambda-1} - 1)$ by Lemma 1. Thus, with probability at least $1 - d_P(\lambda) d_Q(\lambda)/(2^{\lambda-1} - 1)$, $Q'(T) \neq 0$ in G'. We conclude by observing that whenever this happens, game G' will return 1, because t is a root of $Q'(T)$ by construction, and will therefore be found by inspecting its roots. This means

$$\begin{aligned} \mathrm{Adv}^{d_P\text{-dl}}_{\Gamma,\mathcal{B}}(\lambda) = \Pr[\mathrm{G}_0(\lambda)] = \Pr[\mathrm{G}_2(\lambda)] &= \sum_{(\gamma,t,r_\mathcal{A},\boldsymbol{\rho}',\boldsymbol{\alpha}')} \Pr[\mathrm{G}'] \Pr[\gamma, t, r_\mathcal{A}, \boldsymbol{\rho}', \boldsymbol{\alpha}'] \\ \geq \sum_{(\gamma,t,r_\mathcal{A},\boldsymbol{\rho}',\boldsymbol{\alpha}')\in\mathsf{X}} \Pr[\mathrm{G}'] \Pr[\gamma, t, r_\mathcal{A}, \boldsymbol{\rho}', \boldsymbol{\alpha}'] &\geq \left(1 - \frac{d_P(\lambda) d_Q(\lambda)}{2^{\lambda-1} - 1}\right) \cdot \mathrm{Adv}^{\mathrm{uk}}_{\Gamma,\mathcal{A},\mathcal{E}}(\lambda)\,, \end{aligned}$$

which concludes the proof. □

Appendix C: Relations Between Models

We start by recalling the compilation of games in the TSM to games in the GGM.

RR-compilation. Let p be a prime, $\mathsf{G} \subseteq \{0,1\}^*$ a finite set with $|\mathsf{G}| = p$, and let G be a game in the TSM with parameter p. Following Zhandry [Zha22],[12] we let the random-representation (RR) compilation of G with respect to G be the game $\mathrm{RR}(\mathrm{G}, \mathsf{G})$ in the GGM with parameters (p, G) defined as follows.

Game $\mathrm{RR}(\mathrm{G}, \mathsf{G})$ samples a random injection $\tau \in \mathrm{Inj}(\mathbb{Z}_p, \mathsf{G})$ and then operates as G, with the following modifications. All parties are run in input $\tau(1)$ and are offered the GGM operation oracle op defined by τ. Whenever G sends a handle to (resp., receives a handle from) any party, $\mathrm{RR}(\mathrm{G}, \mathsf{G})$ instead sends a string in G to (resp., receives a string in G from) the same party. The strings sent (resp., received) by $\mathrm{RR}(\mathrm{G}, \mathsf{G})$ are obtained (resp., operated on) by performing the same computations on strings as G does on handles. This is possible because, by type safety, game G acts on handles only through the TSM oracles op', eq' and cp'. Therefore, whenever G computes $\mathsf{op}'(\{x_1\}, \{x_2\})$, $\mathsf{eq}'(\{x_1\}, \{x_2\})$ or $\mathsf{cp}'(\{x\})$, $\mathrm{RR}(\mathrm{G}, \mathsf{G})$ can compute $\mathsf{op}(h_1, h_2)$, $(h_1 = h_2)$, and (h, h), respectively. Here, $h_i, h \in \mathsf{G}$ are the strings considered by the compiled game in place of the handles $\{x_i\}$ and $\{x\}$ considered by G. Any other communication between $\mathrm{RR}(\mathrm{G}, \mathsf{G})$ and the parties is processed as in G.

If G is a game in the TSM-H with parameter p, then $\mathrm{RR}(\mathrm{G}, \mathsf{G})$ is the game in the GGM-H with parameters (p, G) defined as above, except that oracle H is still offered to all algorithms.

Suppose that the advantage of an adversary $\mathcal{A}$ in winning G is defined as some function applied to the probability of $\mathcal{A}$ winning G. Then we define the advantage of an adversary $\mathcal{B}$ in winning $\mathrm{RR}(\mathrm{G}, \mathsf{G})$ as the same function applied to the probability of $\mathcal{B}$ winning $\mathrm{RR}(\mathrm{G}, \mathsf{G})$.

In the next two theorems, we show that the relation between GGM and TSM as initially established by Zhandry [Zha22] extends to the corresponding models with hashing for standard games.

Theorem 8 (GGM-H $\Longrightarrow$ TSM-H)**.** *Let p be a prime, and $\mathsf{G} \subseteq \{0,1\}^*$ a finite set with $|\mathsf{G}| = p$. Let* G *be a game in the* TSM-H *with parameter p, and $\mathrm{G}' := \mathrm{RR}(\mathrm{G}, \mathsf{G})$ the RR-compilation of* G *with respect to G. If G' is secure, then so is* G*. More precisely, for every adversary $\mathcal{A}$ in the* TSM-H *with parameter p against* G*, there exists an adversary $\mathcal{B}$ in the* GGM-H *with parameters (p, G) against G' such that*

$$\mathrm{Adv}^{\mathrm{G}}_{p,\mathcal{A}} = \mathrm{Adv}^{\mathrm{G}'}_{p,\mathsf{G},\mathcal{B}}\,, \tag{13}$$

and $q'_{\mathsf{op}} = q_{\mathsf{op}}$ and $q'_{\mathsf{H}} = q_{\mathsf{H}}$. Here, q_{op} and q_{H} (resp., q'_{op} and q'_{H}) are upper bounds on the number of queries made by $\mathcal{A}$ (resp., $\mathcal{B}$) to the respective oracles.

Proof. The proof follows that of Zhandry [Zha22, Theorem 3.4]. Given an adversary $\mathcal{A}$ against G, we construct an adversary $\mathcal{B}$ against G' by applying RR-compilation to $\mathcal{A}$.

In more detail, adversary $\mathcal{B}$ is run on input $\tau(1)$ and receives access to the GGM operation oracle op' and the hashing oracle H'. It then operates as $\mathcal{A}$, with the following modifications. Whenever $\mathcal{A}$ sends a handle to (resp., receives a handle from) the game, $\mathcal{B}$ instead sends a string in G to (resp., receives a string in G from) the game. Strings sent (resp., received) by $\mathcal{B}$ are obtained (resp., operated on) by performing the same computations on strings as $\mathcal{A}$ does on handles. This is possible because, by type safety, adversary $\mathcal{A}$ acts on handles only through the TSM oracles op, eq and cp. Therefore, whenever $\mathcal{A}$

[12] The analogous construction is called *canonical translation* in [Zha22].

```
Oracle $\mathsf{op}'(h_1, h_2)$:
for $i = 1$ to 2 do
  if ($h_i \in \mathrm{Rng}(T)$) then
    $\{x_i\} \leftarrow T^{-1}[h_i]$
  else
    $x_i \twoheadleftarrow \mathbb{Z}_p$; $T[\{x_i\}] \leftarrow h_i$
$\{x\} \leftarrow \mathsf{op}(\{x_1\}, \{x_2\})$
if $\{x\} \in \mathrm{Dom}(T)$ then
  $h \leftarrow T[\{x\}]$
else
  $h \twoheadleftarrow \mathsf{G} \setminus \mathrm{Rng}(T)$; $T[\{x\}] \leftarrow h$
return $h$
```

```
Oracle $\mathsf{H}'(m)$:
$\{x\} \leftarrow \mathsf{H}(m)$
if $\{x\} \in \mathrm{Dom}(T)$ then
  $h \leftarrow T[\{x\}]$
else
  $h \twoheadleftarrow \mathsf{G} \setminus \mathrm{Rng}(T)$
  $T[\{x\}] \leftarrow h$
return $h$
```

Figure 31: Oracles offered by $\mathcal{B}$ in the simulation of G′ to $\mathcal{A}$ in the proof of Theorem 9.

queries $\mathsf{op}(\{x_1\}, \{x_2\})$, $\mathsf{eq}(\{x_1\}, \{x_2\})$ or $\mathsf{cp}(\{x\})$, adversary $\mathcal{B}$ queries $\mathsf{op}'(h_1, h_2)$ or locally computes $(h_1 = h_2)$ and (h, h), respectively. Here, $h_i, h \in \mathsf{G}$ are the strings considered by $\mathcal{B}$ in place of the handles $\{x_i\}$ and $\{x\}$ considered by $\mathcal{A}$. Any other communication between $\mathcal{B}$ and the game is processed as done by $\mathcal{A}$.

We now claim that adversary $\mathcal{B}$ allows proving Equation (13). Indeed, notice that running G′ with adversary $\mathcal{B}$ is equivalent to running G with adversary $\mathcal{A}$, except that handles $\{x\}$ are replaced by the corresponding strings $\tau(x)$. Consequently, the operation is implemented via τ, and equality checks and copies are done on strings. This, however, does not change the winning probability of $\mathcal{A}$ in G, because type-safe algorithms are oblivious to what group element handles concretely are. Thus, $\mathcal{B}$ wins the RR-compilation G′ of G if and only if $\mathcal{A}$ wins G.

As for the query complexity, notice that $\mathcal{B}$ makes one oracle call for each query made by $\mathcal{A}$, which means that the upper bounds coincide. This concludes the proof. □

A similar implication also holds in the reverse direction, and we again follow the proof provided by Zhandry [Zha22, Theorem 3.5]. The main differences are that we use the Turing machine model of type-safe games and the set of group representations G is fixed.

Theorem 9 (TSM-H $\Longrightarrow$ GGM-H)**.** *Let p be a prime, and $\mathsf{G} \subseteq \{0,1\}^*$ a finite set with $|\mathsf{G}| = p$. Let G be a single-stage game in the* TSM-H *with parameter p, and $\mathrm{G}' := \mathrm{RR}(\mathrm{G}, \mathsf{G})$ the RR-compilation of* G *with respect to G. If* G *is secure, then so is* G′*. More precisely, for every adversary $\mathcal{A}$ in the* GGM-H *with parameter (p, G) against* G′*, there exists an adversary $\mathcal{B}$ in the* TSM-H *with parameters p against* G *such that*

$$\mathrm{Adv}^{\mathrm{G}'}_{p,\mathsf{G},\mathcal{A}} \leq \mathrm{Adv}^{\mathrm{G}}_{p,\mathcal{B}} + \mathcal{O}\left(\frac{(q'_{\mathsf{op}} + q'_{\mathsf{H}})^2}{p}\right), \tag{14}$$

and $q_{\mathsf{op}} = \tilde{\mathcal{O}}(q'_{\mathsf{op}})$ and $q_{\mathsf{H}} = q'_{\mathsf{H}}$. Here, q_{op} and q_{H} (resp., q'_{op} and q'_{H}) are upper bounds on the number of queries made by $\mathcal{B}$ (resp., $\mathcal{A}$) to the respective oracles.

Proof. Given an adversary $\mathcal{A}$ against the RR-compilation G′ of G, we construct an adversary $\mathcal{B}$ against G as follows. Adversary $\mathcal{B}$ is run on input handle $\{1\}$, and receives access to the TSM-H oracles op, eq, cp, and H. It then initializes a table T, sets $T[\{1\}] \leftarrow g$ for a randomly sampled $g \twoheadleftarrow \mathsf{G}$, and runs $\mathcal{A}$ on input g with oracles op' and H' as follows.

Whenever G sends a handle $\{x\}$ to $\mathcal{B}$, $\mathcal{B}$ uses its equality and copy oracles to check if $\{x\}$ is already stored in $\mathrm{Dom}(T)$; if so, it sends $T[\{x\}]$ to $\mathcal{A}$, and if not, it sets $T[\{1\}] \leftarrow h$ for a random $h \twoheadleftarrow \mathsf{G} \setminus \mathrm{Rng}(T)$, and then sends h to $\mathcal{A}$. Oracles op' and H' that $\mathcal{A}$ is run on are defined in Figure 31; note that creating a fresh handle $\{x_i\}$ involves repeated invocations of oracle op by $\mathcal{B}$. Finally, whenever $\mathcal{A}$ sends a string h to $\mathcal{B}$, adversary $\mathcal{B}$ looks h up in T and, if present, sends the corresponding handle back to G. If not, it creates handle $\{x\}$ for a random $x \in \mathbb{Z}_p$ and sends that. Any other communication from G or $\mathcal{A}$ is relayed by $\mathcal{B}$.

Note that this construction uses the fact that G is single-stage: For a multi-stage G, G′ and $\mathcal{A}$ would be multi-stage algorithms as well, and so would $\mathcal{B}$. But to ensure a consistent simulation, $\mathcal{B}$ would have to pass table T down its various stages, which is not allowed.

We now claim that adversary $\mathcal{B}$ allows proving Equation (14). To that end, consider the following sequence of games:

G_0: This is the original game G in the TSM-H with parameters p run with adversary $\mathcal{B}$. We omit repeated calls to op to create handles for randomly sampled integers, and instead issue these handles directly.

G_1: This game proceeds as G_0, but whenever a random x_i or x is sampled from $\mathbb{Z}_p$ (either during the simulation of op' or to process the output of $\mathcal{A}$), G_1 ensures that it is fresh.

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

$\mathrm{G}_0 \rightsquigarrow \mathrm{G}_1$. Let Bad be the event in G_1 that there is a collision between a sampled x_i or x and the content of any handle previously issued in the game. Notice that G_0 and G_1 are identical until Bad, and by the fundamental lemma of game playing we therefore have that $|\Pr[\mathrm{G}_1] - \Pr[\mathrm{G}_0]| \leq \Pr[\mathsf{Bad}]$.

To bound the latter probability, observe that T contains at most $3q'_{\mathsf{op}} + q'_{\mathsf{H}} + n + 1$ many entries at any time, where n is an upper bound on the number of elements sent by G to $\mathcal{B}$ at the outset and by $\mathcal{A}$ to $\mathcal{B}$ at the end. Since G_1 samples at most $2q'_{\mathsf{op}} + n$ many integers, we obtain that

$$|\Pr[\mathrm{G}_1] - \Pr[\mathrm{G}_0]| \leq \Pr[\mathsf{Bad}] \leq (2q'_{\mathsf{op}} + n)\frac{3q'_{\mathsf{op}} + q'_{\mathsf{H}} + n + 1}{p} = \mathcal{O}\left(\frac{(q'_{\mathsf{op}} + q'_{\mathsf{H}})^2}{p}\right).$$

We conclude the proof by observing that G_1 is equivalent to game G′ played by $\mathcal{A}$. Indeed, notice that oracles op' and H' are offered to $\mathcal{A}$ in G_1 are equivalent to the GGM-H oracles, with random injection τ lazily sampled via table T.

As for the query complexity, notice that for every call to op' made by $\mathcal{A}$, $\mathcal{B}$ calls op up to $4\lceil \log p \rceil + 1 = \tilde{\mathcal{O}}(1)$ many times to create handles hiding random integers, and makes one oracle call to H' for each hash query made by $\mathcal{A}$. This concludes the proof. □

Remark. A natural way to extend Theorem 8 (GGM-H $\Longrightarrow$ TSM-H) to extractor games is as follows. Recall that, given a TSM-H adversary $\mathcal{A}$, we need to define a TSM-H extractor $\mathcal{E}$ for $\mathcal{A}$. To do so, first (somehow) convert $\mathcal{A}$ into a GGM-H adversary $\mathcal{B}$, for which there exists an extractor $\mathcal{F}$. The natural choice now would be to define $\mathcal{E}$ in terms of $\mathcal{F}$. To that end, we would need to convert the TSM-H view of $\mathcal{A}$ into a GGM-H view, in order to run $\mathcal{F}$. This, however, does not seem to be possible: Group element handles need to be converted to the same random group representations which $\mathcal{B}$ was run on, but $\mathcal{E}$ does not have access to them. The intuitive reason for this failure is that GGM-H adversaries have a "richer view" (random group elements) compared to TSM-H adversaries (group element handles), and the latter cannot be converted to the former.

We face similar obstacles when trying to extend Theorem 9 (TSM-H $\Longrightarrow$ GGM-H) to extractor games: Given a GGM-H adversary $\mathcal{A}$, first convert it to a TSM-H adversary $\mathcal{B}$, for which we know that there exists an extractor $\mathcal{F}$. We are now given a view in GGM-H and extractor $\mathcal{F}$, and need to convert the GGM-H view into a TSM-H view in order to run $\mathcal{F}$. Again, this does not seem to be possible, because the GGM-H extractor $\mathcal{E}$ we are constructing has no way to create or manipulate group element handles (recall that it is given GGM-H oracles, and not oracles in TSM-H).