A Classification of Computational Assumptions in the Algebraic Group Model

Balthazar Bauer, Georg Fuchsbauer, Julian Loss

An extended abstract of this work appears in *CRYPTO'20*. This is the full version.

A Classification of Computational Assumptions in the Algebraic Group Model

Balthazar Bauer[1] Georg Fuchsbauer[2] Julian Loss[3]

[1] Inria, ENS, CNRS, PSL, France
[2] TU Wien, Austria
first.last@{ens.fr,tuwien.ac.at}
[3] University of Maryland, USA
jloss@umiacs.umd.edu

Abstract

a We give a taxonomy of computational assumptions in the algebraic group model (AGM). We first analyze Boyen's Uber assumption family for bilinear groups and then extend it in several ways to cover assumptions as diverse as Gap Diffie-Hellman and LRSW. We show that in the AGM every member of these families is implied by the q-discrete logarithm (DL) assumption, for some q that depends on the degrees of the polynomials defining the Uber assumption.

Using the meta-reduction technique, we then separate $(q+1)$-DL from q-DL, which yields a classification of all members of the extended Uber-assumption families. We finally show that there are strong assumptions, such as one-more DL, that provably fall outside our classification, by proving that they cannot be reduced from q-DL even in the AGM.

Keywords: Algebraic Group Model, Uber Assumption, Pairing-Based Cryptography

1 Introduction

A central paradigm for assessing the security of a cryptographic scheme or hardness assumption is to analyze it within an *idealized model of computation*. A line of work initiated by the seminal work of Nechaev [Nec94] introduced the *generic group model* (GGM) [Sho97, Mau05], in which all algorithms and adversaries are treated as generic algorithms, i.e., algorithms that do not exploit any particular structure of a group and hence can be run in *any* group. Because for many groups used in cryptography (in particular, groups defined over some elliptic curves), the best known algorithms are in fact generic, the GGM has for many years served as the canonical tool to establish confidence in new cryptographic hardness assumptions. Moreover, when cryptographic schemes have been too difficult to analyze in the standard model, they have also directly been proven secure in the GGM (for example LRSW signatures [LRSW99, CL04]).

Following the approach first used in [ABM15], a more recent work by Fuchsbauer, Kiltz, and Loss [FKL18] introduces the *algebraic group model*, in which all algorithms are assumed to be algebraic [PV05]. An *algebraic algorithm* generalizes the notion of a generic algorithm in that all of its output group elements must still be computed by generic operations; however, the algorithm can freely access the structure of the group and obtain more information than what would be possible by purely generic means. This places the AGM between the GGM and the standard model. In contrast to the GGM, one cannot give information-theoretic lower bounds

in the AGM; instead, one analyzes the security of a scheme by giving security reductions from computational hardness assumptions.

Because of its generality and because it provides a powerful framework that simplifies the security analyses of complex systems, the AGM has readily been adopted, in particular in the context of SNARK systems [FKL18, MBKM19, Lip19, GWC19]. It has also recently been used to analyze blind (Schnorr) signatures [FPS20], which are notoriously difficult to prove secure in the standard or random oracle model. Another recent work by Agrikola, Hofheinz and Kastner [AHK20] furthermore shows that the AGM constitutes a plausible model, which is instantiable under falsifiable assumptions in the standard model.

Since its inception, many proofs in the AGM have followed a similar structure, which often consists of a series of tedious case distinctions. A natural question is whether it is possible to unify a large body of relevant hardness assumptions under a general 'Uber' assumption. This would avoid having to prove a reduction to a more well-studied hardness assumption for each of them in the AGM separately. In this work, we present a very rich framework of such Uber assumptions, which contain, as special cases, reductions between hardness assumptions in the AGM from prior work [FKL18, Los19]. We also show that there exists a natural hierarchy among Uber assumptions of different strengths. Together, our results give an almost complete classification in the AGM of common hardness assumptions over (bilinear) groups of prime order.

1.1 Boyen's Uber Assumption Framework

The starting point of our generalizations is the Uber assumption framework by Boyen [Boy08], which serves as an umbrella assumption in the bilinear GGM. Consider a bilinear group $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p)$, where $\mathbb{G}_i$ is a group of prime order p and $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a (non-degenerate) bilinear map, and let g_1, g_2 and g_T be generators of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, respectively. Boyen's framework captures assumptions that are parametrized by polynomials $R_1, \ldots, R_r, S_1, \ldots, S_s, T_1, \ldots, T_t$ and F in a set of formal variables $X_1, \ldots, X_m$ as follows. The challenger picks a vector of randomly chosen points $\vec{x} = (x_1, \ldots, x_m) \in \mathbb{Z}_p^m$ and gives the adversary a list of group elements

$$\left(g_1^{R_1(\vec{x})}, \ldots, g_1^{R_r(\vec{x})}, g_2^{S_1(\vec{x})}, \ldots, g_2^{S_s(\vec{x})}, g_T^{T_1(\vec{x})}, \ldots, g_T^{T_t(\vec{x})}\right).$$

The adversary is considered successful it is able to compute $g_T^{F(\vec{x})}$. Note that for this *not* to be trivially computable, F must be independent from $\vec{R}, \vec{S}$ and $\vec{T}$. That is, it must not be a linear combination of elements from $\vec{T}$ and (pairwise products of) elements of $\vec{R}$ and $\vec{S}$; otherwise, $g_T^{F(x)}$ could be computed from the given group elements via group operations and the bilinear map.

Boyen gives lower bounds for this family of assumptions following the common proof paradigm within the GGM. He also extends the idea of this first Uber assumption [BBG05] with a fixed target polynomial F to an adaptive version called *flexible Uber Assumption*, in which the adversary can choose the target polynomial F itself (as long as it satisfies the same notion of independence from $\vec{R}$, $\vec{S}$ and $\vec{T}$ that makes computing $g_T^{F(\vec{x})}$ non-trivial). Finally, Boyen proposes an extension of his bounds to assumptions in which the elements of $(\vec{R}, \vec{S}, \vec{T})$ and F may be *rational fractions*, that is, fractions of polynomials. We start with considering a straightforward generalization of Boyen's framework where the solution the adversary must find can also be in one of the source groups, that is, of the form $g_1^{F_1(\vec{x})}$ or $g_2^{F_2(\vec{x})}$, as long as they satisfy some non-triviality conditions (Def. 3.3). We next discuss the details of our adaptation of (our generalization of) Boyen's framework to the AGM.

1.2 An Uber-Assumption Framework for the AGM

The main challenge in analyzing Boyen's framework in the AGM setting is that we can no longer prove lower bounds as in the GGM. The next best thing would be to reduce the Uber assumption to a well-established assumption such as the discrete logarithm (DLog) assumption. Due to the general nature of the Uber assumption, this turns out to be impossible; in particular, our negative result (see below) establishes that *algebraic reductions in the AGM* can only reduce DLog to Uber assumptions that are defined by *linear polynomials*.

Indeed, as for Boyen's [Boy08] proofs in the GGM, the degrees of the involved polynomials are expected to appear in our reductions. In our first theorem in Sect. 3 we show that in the AGM any Uber assumption is implied by a parametrized variant of the discrete logarithm problem: in the q-DLog problem the adversary, on top of the instance g^z, is also given $g^{z^2}, \ldots, g^{z^q}$ and must compute z. We prove that if the maximum total degree of the challenge polynomials in $(\vec{R}, \vec{S}, \vec{T})$ of an Uber assumption is at most q, then it is implied by the hardness of the q-DLog problem. This establishes that under q-DLog, anything that is not trivially computable from a given instance (represented by $(\vec{R}, \vec{S}, \vec{T})$) is infeasible to compute. We prove this by generalizing a technique first used by Fuchsbauer et al. [FKL18] to prove soundness of Groth's SNARK [Gro16] under the q-DLog assumption in the AGM.

Proof idea. To convey our main idea, consider a simple instance of the Uber assumption parametrized by polynomials $R_1, \ldots, R_r, F_1$ and let $\vec{S} = \vec{T} = \emptyset$. That is, the the adversary is given group elements $\mathbf{U}_1 = g_1^{R_1(\vec{x})}, \ldots, \mathbf{U}_r = g_1^{R_r(\vec{x})}$ for a random $\vec{x}$ and must compute $\mathbf{U}' = g_1^{F_1(\vec{x})}$. For this problem to be non-trivial, F_1 must be linearly independent of $R_1, \ldots, R_r$, that is, for all $\vec{a} \in \mathbb{Z}_p^r$ we have $R'(\vec{X}) \neq \sum_i a_i R_i(\vec{X})$.

Since the adversary is assumed to be algebraic (see Def. 2.4), it computes its output $\mathbf{U}'$ from its inputs $\mathbf{U}_1, \ldots, \mathbf{U}_r$ by generic group operations, that is, for some vector $\vec{\mu}$ we have $\mathbf{U}' = \prod_i \mathbf{U}_i^{\mu_i}$. In the AGM, the adversary is assumed to output this vector $\vec{\mu}$. Taking the logarithm of the previous equation yields

$$R'(\vec{x}) = \sum_{i=1}^{r} \mu_i R_i(\vec{x}). \tag{1}$$

Since R' is independent from $\vec{R}$, the polynomial $P(\vec{X}) := R'(\vec{X}) - \sum_i \mu_i R_i(\vec{X})$ is non-zero. On the other hand, (1) yields $P(\vec{x}) = 0$ for a successful adversary.

The adversary has thus (implicitly) found a non-zero polynomial P, which has the secret $\vec{x}$ among its roots. Now, in order to use this to solve a q-DLog instance $(g_1, g_1^z, \ldots, g_1^{z^q})$, we embed a randomized version of z into every coordinate of $\vec{x}$. In particular, for random vectors $\vec{y}$ and $\vec{v}$, we implicitly let $x_i := y_i z + v_i \bmod p$. By leveraging linearity, the reduction can compute the group elements $\mathbf{U}_i = g_1^{R_i(\vec{x})}$, etc, from its DLog instance.

If $P(\vec{X})$ is non-zero then $Q(Z) := P(y_1 Z + v_1, \ldots, y_m Z + v_m)$ is non-zero with overwhelming probability: the values v_i guarantee that the values y_i are perfectly hidden from the adversary and, as we show (Lemma 2.1), the leading coefficient of Q is a non-zero polynomial evaluated at $y_1, \ldots, y_m$, values that are independent of the adversary's behavior. Schwartz-Zippel thus bounds the probability that the leading coefficient of Q is zero, and thus, that $Q \equiv 0$. Since $Q(z) = P(\vec{x}) = 0$, we can factor the univariate polynomial Q and find the DLog solution z, which is among its roots.

Extensions. We next extend our approach to a flexible (i.e., adaptive) version of the static Uber assumption, where the adversary can adaptively choose the polynomials (Sect. 4) as well as a generalization from polynomials to rational fractions (Sect. 5). We combine the flexible

framework with the rational fraction framework in Sect. 6. After these generalizations, our framework covers assumptions such as strong Diffie-Hellman [BB08], where the adversary must compute a rational fraction of its own choice in the exponent.

In a next step (Sect. 7), we extend our framework to also cover gap-type assumptions such as Gap Diffie-Hellman (GDH) [OP01], which was recently proven equivalent to the DLog assumption in the AGM by Loss [Los19]. GDH states that the CDH assumption remains true even when the DDH assumption no longer holds. Informally, the idea of the proof given in [Los19] (first presented in [FKL18] for a restricted version of GDH) is to argue that the DDH oracle given to an algebraic adversary is useless, unless the adversary succeeds in breaking CDH during an oracle query. The reduction simulates the DDH oracle by always returning false. We generalize this to a broader class of assumptions, using a different simulation strategy, which avoids a security loss.

We also present (Sect. 8) an extension of our (adaptive) framework that allows to capture assumptions as strong as the LRSW assumption [LRSW99], which forms the basis of the Camenisch-Lysyanskaya signature scheme [CL04]. The LRSW assumption falls outside (even the adaptive version of) Boyen's Uber framework, since the adversary need not output the polynomial it is computing in the exponent.

The LRSW and GDH assumptions were previously studied in the AGM in the works of [FKL18, Los19], who gave very technical proofs spanning multiple pages of case distinctions. By comparison, our Uber Framework offers a more general and much simpler proof for both of these assumptions. Finally, we are able to prove all these results using *tight reductions*. This, in particular, improves upon the non-tight reduction of DLog to LRSW in [FKL18].

1.3 Classifying Assumptions in our Framework

Finally, we prove two separation results that show the following:

Separating $(q+1)$-DLog from q-DLog. This shows that with respect to currently known (i.e., algebraic) reduction techniques, the Uber assumption, for increasing degrees of the polynomials, defines a natural hierarchy of assumptions in the AGM. More concretely, the q-lowest class within the established hierarchy consists of all assumptions that are covered by a specific instantiation of the Uber assumption which can be reduced from the q-DLog problem. Our separation result (Theorem 9.1) shows that there is no algebraic reduction from the q-DLog problem to the $(q+1)$-DLog problem in the AGM. This implies that assumptions within different classes are separated with respect to algebraic reductions. Interestingly, we are even able to show our separation for reductions that can rewind and choose the random coins of the solver for the $(q+1)$-DLog problem freely.

Separating OMDL from q-DLog. Our second result (Theorem 10.1) shows a separation result between the one-more-DLog problem (OMDL) (where the adversary has to solve q DLog instances and is given an oracle that computes discrete logarithms, which it can access $q-1$ times) and the q-DLog problem (for any q) in the AGM. Our result strengthens a previous result by Bresson, Monnerat, and Vergnaud [BMV08], who showed a separation between the discrete logarithm problem (i.e, where $q=1$) and the 2-one-more-DLog problem with respect to *black-box reductions*. By comparison, our result holds even in the AGM, where reductions are inherently non-black-box, as the AGM implicitly assumes an efficient extractor algorithm that extracts algebraic coefficients from the algebraic adversary. As the extractor is non-black-box (since it depends on the algebraic adversary), neither is any reduction that non-trivially leverages the AGM.

Our result clearly establishes the limits of our framework, as it excludes the OMDL family of assumptions. Unlike our first separation, this one comes with the caveat that it only applies to reductions that are "black-box in the AGM", meaning that they simply obtain the algebraic coefficients via the extractor, but cannot rewind the adversary or choose its random coins.

1.4 Related Work

A long line of research has considered frameworks to capture general classes of assumptions. We give an overview of the most closely related works. The first Uber assumptions were introduced by Boyen et al. [BBG05, Boy08]. Others later gave alternative concepts to classify assumptions within cyclic groups. The works of Chase et al. [CM14, CMM16] study assumptions in bilinear groups of *composite* order, which are not considered in the original Uber framework. They show that several q-type assumptions are implied by (static) "subgroup-hiding" assumptions. This gives evidence that this type of assumption, which is specific to composite-order groups, is particularly strong.

More recently, Ghadafi and Groth [GG17] studied a broader class of assumptions in which the adversary must compute a group element from $\mathbb{G}_T$. Like our work, their framework applies to prime-order groups and extends to the case where the exponents can be described by rational fractions, and they also separate classes of assumptions from each other. However, their framework only deals with non-interactive assumptions, which do not cover the adaptive type of assumptions we study in our flexible variants (in fact, the authors mention extending their work to interactive assumptions as an open problem [GG17]). Their work does not cover assumptions such as GDH or LRSW, which we view as particularly interesting (and challenging) to classify. Indeed, our framework appears to be the first in this line of work that offers a classification comprising this type of assumptions.

A key difference is that Ghadafi and Groth's results are in the standard model whereas we work in the AGM. While this yields stronger results for reductions, their separations are weaker (in addition to separating less broad types of Uber assumptions), as they are with respect to generic reductions, whereas ours hold against algebraic reductions that can assume that the *adversary is algebraic*. Furthermore, their work considers *black-box reductions* that treat the underlying solver as an (imperfect) oracle, while we show the non-existence of reductions in the AGM, which are, by definition, non-black-box (see above). A final difference to the work of [GG17] lies in the tightness of all our reductions, whereas non of theirs are tight.

At CT-RSA'19 Mizuide, Takayasu, and Takagi [MTT19] studied static (i.e., non-flexible) variants and generalizations of the Diffie-Hellman problem in prime-order groups (also with extensions to the bilinear setting) by extending proofs from [FKL18] in the obvious manner. Most of their results are special cases of our Uber assumption framework. Concretely, when restricting the degrees of all input polynomials to 1 in our static Uber Assumption, our non-triviality condition implies all corresponding theorems in their paper (except the ones relating to Matrix assumptions, which are outside the scope of this work). By our separation of q-DLog for different q, our results for higher degrees do not follow from theirs by currently known techniques. Finally, they do not cover the flexible (adaptive) variants nor oracle-enhanced- and hidden-polynomial-type assumptions (such as GDH and LRSW).

A further distinction that sets our work apart from these prior works is our formulation of the aforementioned 'hidden-type' assumptions, where we allow the adversary to solve the problem with respect to a group generator *of its own choice* instead of the one provided by the game. A recent work [BMZ19] shows that even in the GGM, allowing randomly chosen generators results in unexpected complications when proving lower bounds. Similarly, giving the adversary this additional freedom makes proving (and formalizing) our results more challenging. We also give

this freedom to the reductions that we study (and prove impossible) in our separation results.

2 Algebraic Algorithms and Preliminaries

ALGORITHMS. We denote by $s \xleftarrow{\$} S$ the uniform sampling of the variable s from the (finite) set S. All our algorithms are probabilistic (unless stated otherwise) and written in uppercase letters A, B. To indicate that algorithm A runs on some inputs $(x_1, \ldots, x_n)$ and returns y, we write $y \xleftarrow{\$} \mathsf{A}(x_1, \ldots, x_n)$. If A has access to an algorithm B (via oracle access) during its execution, we write $y \xleftarrow{\$} \mathsf{A}^{\mathsf{B}}(x_1, \ldots, x_n)$.

POLYNOMIALS AND RATIONAL FRACTIONS. We denote polynomials by uppercase letters P, Q and specify them by a list of their coefficients. If m is an integer, we denote by $\mathbb{Z}_p[X_1, \ldots, X_m]$ the set of m-variate polynomials with coefficients in $\mathbb{Z}_p$ and by $\mathbb{Z}_p(X_1, \ldots, X_m)$ the set of rational fractions in m variables with coefficients in $\mathbb{Z}_p$. We define the *total degree* of a polynomial $P(X_1, \ldots, X_m) = \sum_{\vec{i} \in \mathbb{N}^m} \lambda_{i_1,\ldots,i_m} \prod_{j=1}^m X_j^{i_j} \in \mathbb{Z}_p[X_1, \ldots, X_m]$ as $\max_{\vec{i} \in \mathbb{N}^m \,:\, \lambda_{i_1,\ldots,i_m} \not\equiv_p 0} \{\sum_{j=1}^m i_j\}$.

For the degree of rational fractions we will use the "French" definition [AW98]: for $(P, Q) \in \mathbb{Z}_p[X_1, \ldots, X_m] \times (\mathbb{Z}_p[X_1, \ldots, X_m] \setminus \{0\})$ we define

$$\deg \frac{P}{Q} := \deg P - \deg Q.$$

This definition has the following properties: The degree does not depend on the choice of the representative; it generalizes the definition for polynomials; and the following holds: $\deg(F_1 \cdot F_2) = \deg F_1 + \deg F_2$, and $\deg(F_1 + F_2) \leq \max\{\deg F_1, \deg F_2\}$.

We state the following technical lemma, which we will use in our reductions.

Lemma 2.1 *Let P be a non-zero multivariate polynomial in $\mathbb{Z}_p[X_1, \ldots, X_m]$ of total degree d. Define $Q(Z) \in (\mathbb{Z}_p[Y_1, \ldots, Y_m, V_1, \ldots, V_m])[Z]$ as $Q(Z) := P(Y_1 Z + V_1, \ldots, Y_m Z + V_m)$. Then the coefficient of maximal degree of Q is a polynomial in $\mathbb{Z}_p[Y_1, \ldots, Y_m]$ of degree d.*

Proof. P is of the form $P(\vec{X}) = \sum_{\vec{i} \in \mathbb{N}^m \,:\, \sum_j i_j \leq d} \lambda_{i_1,\ldots,i_m} \prod_{j=1}^m X_j^{i_j}$ for coefficients $\lambda_{i_1,\ldots,i_m}$ and thus

$$\begin{aligned}
Q(Z) &= \sum_{\vec{i} \in \mathbb{N}^m \,:\, \sum_j i_j \leq d} \lambda_{i_1,\ldots,i_m} \prod_{j=1}^m (Y_j Z + V_j)^{i_j} \\
&= \sum_{\vec{i} \in \mathbb{N}^m \,:\, \sum_j i_j \leq d} \lambda_{\vec{i}} \prod_{j=1}^m \left(\sum_{k=0}^{i_j} \binom{i_j}{k} Y_j^k \, Z^k \, V_j^{i_j - k} \right) \\
&= \sum_{\vec{i} \in \mathbb{N}^m \,:\, \sum_j i_j \leq d} \lambda_{\vec{i}} \sum_{k_1=0}^{i_1} \cdots \sum_{k_m=0}^{i_m} \prod_{j=1}^m \binom{i_j}{k_j} Y_j^{k_j} \, Z^{k_j} \, V_j^{i_j - k_j} \\
&= \sum_{\ell=0}^{d} \sum_{\vec{i} \in \mathbb{N}^m \,:\, \sum_j i_j \leq d} \lambda_{\vec{i}} \sum_{\vec{k} \in \mathbb{N}^m \,:\, \sum_j k_j = \ell} \prod_{j=1}^m \binom{i_j}{k_j} Y_j^{k_j} \, Z^{k_j} \, V_j^{i_j - k_j} = \textstyle\sum_{\ell=0}^{d} \lambda'_\ell \, Z^\ell \\
&\qquad \text{with} \quad \lambda'_\ell := \sum_{\vec{i} \in \mathbb{N}^m \,:\, \sum_j i_j \leq d} \lambda_{\vec{i}} \sum_{\vec{k} \in \mathbb{N}^m \,:\, \sum_j k_j = \ell} \prod_{j=1}^m \binom{i_j}{k_j} Y_j^{k_j} \, V_j^{i_j - k_j}.
\end{aligned}$$

By assumption $P \not\equiv 0$. Thus for some $i_1, \ldots, i_m \geq 0$ with $\sum_j i_j = d$ we have $\lambda_{i_1,\ldots,i_m} \neq 0$, while $\lambda_{i_1,\ldots,i_m} = 0$ when $\sum_j i_j > d$. By the latter we have

$$\lambda'_d \;=\; \sum_{\vec{i}\in\mathbb{N}^m\,:\,\Sigma_j i_j \leq d} \lambda_{\vec{i}} \sum_{\vec{k}\,:\,\Sigma_j k_j = d} \prod_{j=1}^{m} \tbinom{i_j}{k_j}\, Y_j^{k_j}\, V_j^{i_j-k_j} \;=\; \sum_{\vec{i}\in\mathbb{N}^m\,:\,\Sigma_j i_j = d} \lambda_{\vec{i}} \prod_{j=1}^{m} Y_j^{i_j},$$

where the last step follows since $k_j \leq i_j$ for all j and $\sum_j i_j \leq d$, $\sum_j k_j = d$ implies that $k_j = i_j$ for all j. Since P is a polynomial of total degree d, for some $\vec{i} \in \mathbb{N}^m$ we have $\sum_j i_j = d$ and $\lambda_{\vec{i}} \neq 0$. We conclude that λ'_d is a polynomial in $(Y_1, \ldots, Y_m)$ of total degree d. ∎

We will use the following version of the Schwartz-Zippel lemma [DL77]:

Lemma 2.2 *Let $P \in \mathbb{Z}_p[X_1, \ldots, X_m]$ be a non-zero polynomial of total degree d. Let $r_1, \ldots, r_m$ be selected at random independently and uniformly from $\mathbb{Z}_p^*$. Then*

$$\Pr\left[P(r_1, \ldots, r_m) \equiv_p 0\right] \leq \tfrac{d}{p-1}.$$

Bilinear Groups. We next state the definition of a *bilinear group*.

Definition 2.3 (Bilinear group). A *bilinear group* is a tuple $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \phi, \psi, p)$ where

- $\mathbb{G}_i$ is a cyclic group of prime order p, for $i \in \{1, 2, T\}$;
- e is a non-degenerate bilinear map $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, that is, for all $a, b \in \mathbb{Z}_p$ and all generators g_1 of $\mathbb{G}_1$ and g_2 of $\mathbb{G}_2$ we have that $g_T := e(g_1, g_2)$ generates $\mathbb{G}_T$ and $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab} = g_T^{ab}$;
- ϕ is an isomorphism $\phi\colon \mathbb{G}_1 \to \mathbb{G}_2$, and ψ is an isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$.

All group operations and the bilinear map e must be efficiently computable. *$\mathcal{G}$ is of Type 1* if the maps ϕ and ψ are efficiently computable; *$\mathcal{G}$ is of Type 2* if there is no efficiently computable map ϕ; and *$\mathcal{G}$ is of Type 3* if there are no efficiently computable maps ϕ and ψ. We require that there exist an efficient algorithm $\mathsf{GenSamp}$ that returns generators g_1 of $\mathbb{G}_1$ and g_2 of $\mathbb{G}_2$, so that g_2 is uniformly random, and (for Types 1 and 2) $g_1 = \psi(g_2)$ or (Type 3) g_1 is also uniformly random. By $\mathsf{GenSamp}_i$ we denote a restricted version that only returns g_i.

In the following, we fix a bilinear group $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \phi, \psi, p)$.

(Algebraic) Security games. We use a variant of (code-based) *security games* [BR04]. In game $\mathbf{G}_\mathcal{G}$ (defined relative to $\mathcal{G}$), an adversary A interacts with a challenger that answers oracle queries issued by A. The game has a main procedure and (possibly zero) oracle procedures which describe how oracle queries are answered. We denote the output of a game $\mathbf{G}_\mathcal{G}$ between a challenger and an adversary A by $\mathbf{G}_\mathcal{G}^\mathsf{A}$. A is said to *win* if $\mathbf{G}_\mathcal{G}^\mathsf{A} = 1$. We define the *advantage* of A in $\mathbf{G}_\mathcal{G}$ as $\mathbf{Adv}_{\mathcal{G},\mathsf{A}}^\mathbf{G} := \Pr\left[\mathbf{G}_\mathcal{G}^\mathsf{A} = 1\right]$ and the running time of $\mathbf{G}_\mathcal{G}^\mathsf{A}$ as $\mathbf{Time}_{\mathcal{G},\mathsf{A}}^\mathbf{G}$. In this work, we are primarily concerned with *algebraic security games* $\mathbf{G}_\mathcal{G}$, in which we syntactically distinguish between elements of groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ (written in bold, uppercase letters, e.g., $\mathbf{Z}$) and all other elements, which must not depend on any group elements.

We next define algebraic algorithms. Intuitively, the only way for an algebraic algorithm to output a new group element $\mathbf{Z}$ is to derive it via group operations from known group elements.

Definition 2.4 (Algebraic algorithm for bilinear groups). An algorithm $\mathsf{A}_{\mathsf{alg}}$ executed in an algebraic game $\mathbf{G}_{\mathcal{G}}$ is called *algebraic* if for all group elements $\mathbf{Z} \in \mathbb{G}$ (where $\mathbb{G} \in \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T\}$) that $\mathsf{A}_{\mathsf{alg}}$ outputs, it additionally provides a representation in terms of received group elements in $\mathbb{G}$ and those from groups from which there is an efficient mapping to $\mathbb{G}$; in particular: if $\mathbf{U}_0, \ldots, \mathbf{U}_\ell \in \mathbb{G}_1$, $\mathbf{V}_0, \ldots, \mathbf{V}_m \in \mathbb{G}_2$ and $\mathbf{W}_0, \ldots, \mathbf{W}_t \in \mathbb{G}_T$ are the group elements received so far then $\mathsf{A}_{\mathsf{alg}}$ provides vectors $\vec{\mu}, \vec{\nu}, \vec{\zeta}, \vec{\eta}, \vec{\delta}$ and matrices $A = (\alpha_{i,j}), B = (\beta_{i,j}), \Gamma = (\gamma_{i,j})$ such that

- $\mathbf{Z} \in \mathbb{G}_1$ (Type 1 and 2): $\mathbf{Z} = \prod_i \mathbf{U}_i^{\mu_i} \cdot \prod_i \psi(\mathbf{V}_i)^{\nu_i}$
  (Type 3): $\mathbf{Z} = \prod_i \mathbf{U}_i^{\mu_i}$
- $\mathbf{Z} \in \mathbb{G}_2$ (Type 1): $\mathbf{Z} = \prod_i \phi(\mathbf{U}_i)^{\zeta_i} \cdot \prod_i \mathbf{V}_i^{\eta_i}$
  (Type 2 and 3): $\mathbf{Z} = \prod_i \mathbf{V}_i^{\eta_i}$
- $\mathbf{Z} \in \mathbb{G}_T$: $\mathbf{Z} = \prod_i \prod_j e(\mathbf{U}_i, \mathbf{V}_j)^{\alpha_{i,j}} \cdot \prod_i \prod_j e(\mathbf{U}_i, \phi(\mathbf{U}_j))^{\beta_{i,j}} \cdot \prod_i \prod_j e(\psi(\mathbf{V}_i), \mathbf{V}_j)^{\gamma_{i,j}} \cdot \prod_i \mathbf{W}_i^{\delta_i}$, where $\beta_{i,j} = 0$ for Type 2 and $\beta_{i,j} = \gamma_{i,j} = 0$ for Type 3.

We remark that oracle access to an algorithm B in the AGM includes any (usually non-black-box) access to B that is needed to extract the algebraic coefficients. Thus, our notion of black-box access in the AGM mainly rules out techniques such as rewinding B or running it on non-uniform random coins.

2.1 Generic Security Games and Algorithms

Generic algorithms $\mathsf{A}_{\mathsf{gen}}$ are only allowed to use generic properties of a group. Informally, an algorithm is generic if it works regardless of what group it is run in. This is usually modeled by giving an algorithm indirect access to group elements via abstract handles. It is straight-forward to translate all of our algebraic games into games that are syntactically compatible with generic algorithms accessing group elements only via abstract handles. We measure the running times of generic algorithms as queries to an oracle that implements the abstract group operation, i.e., every query accounts for one step of the algorithm. We highlight this difference by denoting the running time of a generic algorithm with the letter o rather than t. We say that winning algebraic game $\mathbf{G}_{\mathcal{G}}$ is (ε, o)-*hard in the generic group model* if for every generic algorithm $\mathsf{A}_{\mathsf{gen}}$ it holds that

$$\mathbf{Time}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}_{\mathsf{gen}}} \leq o \implies \mathbf{Adv}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}_{\mathsf{gen}}} \leq \varepsilon.$$

As all of our reductions run the adversary only once and without rewinding, the overhead in the running time of our reductions is *additive* only. We make the reasonable assumption that, compared to the running time of the adversary, this is typically small, and therefore ignore the losses in the running times for this work in order to keep notational overhead low.

We assume that a generic algorithm $\mathsf{A}_{\mathsf{gen}}$ provides the representation of $\mathbf{Z}$ relative to all previously received group elements, for all group elements $\mathbf{Z}$ that it outputs. This assumption is w.l.o.g. since a generic algorithm can only obtain new group elements by querying two known group elements to the generic group oracle; hence a reduction can always extract a valid representation of a group element output by a generic algorithm. This way, every generic algorithm is also an algebraic algorithm.

Furthermore, if $\mathsf{B}_{\mathsf{gen}}$ is a generic oracle algorithm and $\mathsf{A}_{\mathsf{alg}}$ is an algebraic algorithm, then $\mathsf{B}_{\mathsf{alg}} := \mathsf{B}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{gen}}$ is also an algebraic algorithm. We refer to [Mau05] for more on generic algorithms.

q-**dlog** $^{\mathsf{A}}_{\mathcal{G}_i}$	(q_1, q_2)-**dlog** $^{\mathsf{A}}_{\mathcal{G}}$
01 $g \xleftarrow{\$} \mathsf{GenSamp}_i$	01 $(g_1, g_2) \xleftarrow{\$} \mathsf{GenSamp}$
02 $z \xleftarrow{\$} \mathbb{Z}_p^*$	02 $z \xleftarrow{\$} \mathbb{Z}_p^*$
03 $z^* \xleftarrow{\$} \mathsf{A}(g, g^z, g^{z^2}, \ldots, g^{z^q})$	03 $z^* \xleftarrow{\$} \mathsf{A}(g_1, g_1^z, g_1^{z^2}, \ldots, g_1^{z^{q_1}}, g_2, g_2^z, \ldots, g_2^{z^{q_2}})$
04 Return $(z^* = z)$	04 Return $(z^* = z)$

Figure 1: q-discrete logarithm game q-**dlog** $_{\mathcal{G}_i}$ (left) and (q_1, q_2)-discrete logarithm game (q_1, q_2)-**dlog** $_{\mathcal{G}}$ (right) relative to group $\mathcal{G}_i, i \in \{1, 2\}$ and $\mathcal{G}$, resp., and adversary A.

Security Reductions. All our security reductions are (bilinear) generic algorithms, which allows us to compose all of our reductions with hardness bounds in the (bilinear) generic group model (see next paragraph). Let $\mathbf{G}_{\mathcal{G}}, \mathbf{H}_{\mathcal{G}}$ be security games. We say that algorithm $\mathsf{R}_{\mathsf{gen}}$ is a *generic* $(\Delta_\varepsilon^{(\cdot)}, \Delta_\varepsilon^{(+)}, \Delta_o^{(\cdot)}, \Delta_o^{(+)})$*-reduction from* $\mathbf{H}_{\mathcal{G}}$ *to* $\mathbf{G}_{\mathcal{G}}$ if $\mathsf{R}_{\mathsf{gen}}$ is generic and if for every algebraic algorithm $\mathsf{A}_{\mathsf{alg}}$, algorithm $\mathsf{B}_{\mathsf{alg}}$ defined as $\mathsf{B}_{\mathsf{alg}} := \mathsf{R}_{\mathsf{gen}}^{\mathsf{A}_{\mathsf{alg}}}$ satisfies

$$\mathbf{Adv}^{\mathbf{H}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}} \geq \frac{1}{\Delta_\varepsilon^{(\cdot)}} \cdot \left(\mathbf{Adv}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}} - \Delta_\varepsilon^{(+)}\right),$$
$$\mathbf{Time}^{\mathbf{H}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}} \leq \Delta_o^{(\cdot)} \cdot \left(\mathbf{Time}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}} + \Delta_o^{(+)}\right).$$

Furthermore, for simplicity of notation, we will make the convention of referring to $(1, \Delta_\varepsilon, 1, \Delta_o)$-reductions as $(\Delta_\varepsilon, \Delta_o)$-reductions.

Composing information-theoretic lower bounds with reductions in the AGM. The following lemma from [Los19] explains how statements in the AGM compose with bounds from the GGM.

Lemma 2.5 *Let* $\mathbf{G}_{\mathcal{G}}$ *and* $\mathbf{H}_{\mathcal{G}}$ *be algebraic security games and let* $\mathsf{R}_{\mathsf{gen}}$ *be a generic* $(\Delta_\varepsilon^{(\cdot)}, \Delta_\varepsilon^{(+)}, \Delta_o^{(\cdot)}, \Delta_o^{(+)})$*-reduction from* $\mathbf{H}_{\mathcal{G}}$ *to* $\mathbf{G}_{\mathcal{G}}$*. If* $\mathbf{H}_{\mathcal{G}}$ *is* (ε, o)*-secure in the GGM, then* $\mathbf{G}_{\mathcal{G}}$ *is* (ε', o')*-secure in the GGM where*

$$\varepsilon' = \varepsilon \cdot \Delta_\varepsilon^{(\cdot)} + \Delta_\varepsilon^{(+)}, \quad o' = o/\Delta_o^{(\cdot)} - \Delta_o^{(+)}.$$

The q-discrete logarithm assumption and variants. For this work, we consider two generalizations of the DLog assumption, which are parametrized (i.e., "q-type") variants of the DLog assumption. We describe them via the algebraic security games q-**dlog** $_{\mathcal{G}_i}$ and (q_1, q_2)-**dlog** $_{\mathcal{G}}$ in Fig. 1.

The following lemma, which follows similarly to the generic security of q-SDH [BB08], was proved (asymptotically) by Lipmaa [Lip12]. For completeness, we give a concrete proof in Appendix A.

Lemma 2.6 *Let* $o, q_1, q_2 \in \mathbb{N}$*, let* $q := \max\{q_1, q_2\}$*. Then* q*-DLog and* (q_1, q_2)*-DLog are* $(\frac{(o+q+1)^2 q}{p-1}, o)$*-secure in the bilinear generic group model.*

We remark that all though our composition results are stated in the bilinear GGM, it is straight forward to translate them to the standard GGM if the associated hardness assumption is stated over a pairing-free group. This is true, because in those cases, our reductions will also be pairing-free and hence are standard generic algorithms themselves.

3 The Uber-Assumption Family

Boyen [Boy08] extended the Uber-assumption framework he initially introduced with Boneh and Goh [BBG05]. We start with defining notions of independence for polynomials and rational fractions (of which polynomials are a special case):

Definition 3.1 Let $\vec{R} \in \mathbb{Z}_p(X_1, \ldots, X_m)^r$ and $W \in \mathbb{Z}_p(X_1, \ldots, X_m)$. We say that W is *linearly dependent* on $\vec{R}$ if there exist coefficients $(a_i)_{i=1}^r \in \mathbb{Z}_p^r$ such that

$$W = \sum_{i=1}^{r} a_i R_i.$$

We say that W is *(linearly) independent* from $\vec{R}$ if it is not linearly dependent on $\vec{R}$.

Definition 3.2 Let $\vec{R}$, $\vec{S}$, $\vec{F}$ and W be vectors of rational fractions from $\mathbb{Z}_p(X_1, \ldots, X_m)$ of length r, s, f and 1, respectively. We say that W is *("bilinearly") dependent* on $(\vec{R}, \vec{S}, \vec{F})$ if there exist coefficients $\{a_{i,j}\}$, $\{b_{i,j}\}$, $\{c_{i,j}\}$ and $\{d_k\}$ in $\mathbb{Z}_p$ such that

$$W = \sum_{i=1}^{r}\sum_{j=1}^{s} a_{i,j} R_i S_j + \sum_{i=1}^{r}\sum_{j=1}^{r} b_{i,j} R_i R_j + \sum_{i=1}^{s}\sum_{j=1}^{s} c_{i,j} S_i S_j + \sum_{k=1}^{f} d_k F_k.$$

We call the dependency of *Type 2* if $b_{i,j} = 0$ for all i, j and of *Type 3* if $b_{i,j} = c_{i,j} = 0$ for all i, j. Else, it is of *Type 1*. We say that W is *(Type-τ) independent* from $(\vec{R}, \vec{S}, \vec{F})$ if it is not (Type-τ) dependent on $(\vec{R}, \vec{S}, \vec{F})$. (Thus, W can be Type-3 independent but Type-2 dependent.)

Consider the Uber-assumption game in Fig. 2, which is parametrized by vectors of polynomials $\vec{R}$, $\vec{S}$ and $\vec{F}$ and polynomials R', S' and F'. For a random vector $\vec{x}$, the adversary receives the evaluation of the (vectors of) polynomials in the exponents of the generators g_1, g_2 and g_T; its goal is to find the evaluation of the polynomials R', S' and F' at $\vec{x}$ in the exponents. Note that we do not explicitly give the generators to the adversary. This is without loss of generality because we can always set $R_1 = S_1 = F_1 \equiv 1$.

The game can be efficiently solved if one of the following conditions hold (where we distinguish the different types of bilinear groups and interpret all polynomials over $\mathbb{Z}_p$):

(Type 1) If R' is dependent on $\vec{R}$ and $\vec{S}$, and S' is dependent on $\vec{R}$ and $\vec{S}$, and F' is Type-1 dependent (Def. 3.2) on $(\vec{R}, \vec{S}, \vec{F})$.

(Type 2) If R' is dependent on $\vec{R}$ and $\vec{S}$, S' is dependent on $\vec{S}$, and F' is Type-2 dependent (Def. 3.2) on $(\vec{R}, \vec{S}, \vec{F})$.

(Type 3) If R' is dependent on $\vec{R}$, S' is dependent on $\vec{S}$, and F' is Type-3 dependent (Def. 3.2) on $(\vec{R}, \vec{S}, \vec{F})$.

For example, in Type-2 groups, if $R' = \sum_i a'_i R_i + \sum_i b'_i S_i$ and $S' = 1$, and $F' = \sum_i \sum_j a_{i,j} R_i S_i + \sum_i \sum_j c_{i,j} S_i S_j$, then from a challenge $(\vec{\mathbf{U}}, \vec{\mathbf{V}}, \vec{\mathbf{W}})$, one can easily compute a solution $\mathbf{U}' := \prod_i \mathbf{U}_i^{a'_i} \cdot \psi(\prod_i \mathbf{V}_i^{b'_i})$, $\mathbf{V}' := g_2$, $\mathbf{W}' := \prod_i \prod_j e(\mathbf{U}_i, \mathbf{V}_j)^{a_{i,j}} \cdot \prod_i \prod_j e(\psi(\mathbf{V}_i), \mathbf{V}_j)^{b_{i,j}}$.

In our main theorem, we show that whenever the game in Fig. 2 cannot be trivially won, then for groups of Type $\tau \in \{1, 2\}$, it can be reduced from q-**dlog** $_{\mathcal{G}_2}$, and for Type-3 groups, it can be reduced from (q_1, q_2)-**dlog** $_{\mathcal{G}}$ (for appropriate values of q, q_1, q_2). To state the theorem for all types of groups, we first define the following non-triviality condition (which again we state for the more general case of rational fractions):

$(\vec{R}, \vec{S}, \vec{F}, R', S', F')$-**über**$_{\mathcal{G}}^{\mathsf{A_{alg}}}$

01 $(g_1, g_2) \xleftarrow{\$} \mathsf{GenSamp}$; $g_T \leftarrow e(g_1, g_2)$

02 $\vec{x} = (x_1, \ldots, x_m) \xleftarrow{\$} \mathbb{Z}_p^m$

03 $\vec{\mathbf{U}} := (g_1^{R_1(\vec{x})}, \ldots, g_1^{R_r(\vec{x})})$

04 $\vec{\mathbf{V}} := (g_2^{S_1(\vec{x})}, \ldots, g_2^{S_s(\vec{x})})$

05 $\vec{\mathbf{W}} := (g_T^{F_1(\vec{x})}, \ldots, g_T^{F_f(\vec{x})})$

06 $(\mathbf{U}', \mathbf{V}', \mathbf{W}') \xleftarrow{\$} \mathsf{A_{alg}}(\vec{\mathbf{U}}, \vec{\mathbf{V}}, \vec{\mathbf{W}})$

07 Return $((\mathbf{U}', \mathbf{V}', \mathbf{W}') = (g_1^{R'(\vec{x})}, g_2^{S'(\vec{x})}, g_T^{F'(\vec{x})}))$

Figure 2: Algebraic game for the Uber assumption relative to bilinear group $\mathcal{G}$ and adversary $\mathsf{A_{alg}}$, parametrized by (vectors of) or polynomials $\vec{R}, \vec{S}, \vec{F}, R', S'$ and F'

Definition 3.3 (Non-triviality). Let $\vec{R} \in \mathbb{Z}_p(X_1, \ldots, X_m)^r$, $\vec{S} \in \mathbb{Z}_p(X_1, \ldots, X_m)^s$, $\vec{F} \in \mathbb{Z}_p(X_1, \ldots, X_m)^f$, $R', S', F' \in \mathbb{Z}_p(X_1, \ldots, X_m)$. We say that the tuple $(\vec{R}, \vec{S}, \vec{F}, R', S', F')$ is *non-trivial for groups of type* τ, for $\tau \in \{1, 2, 3\}$, if the following holds:

- **either** R' is linearly independent from $\vec{R}$ and $\vec{S}$ in case $\tau \in \{1, 2\}$,
  R' is linearly independent from $\vec{R}$ in case $\tau = 3$; (τ.1)
- **or** S' is linearly independent from $\vec{R}$ and $\vec{S}$ in case $\tau = 1$,
  S' is linearly independent from $\vec{S}$ in case $\tau \in \{2, 3\}$; (τ.2)
- **or** F' is Type-τ "bilinearly" independent (Def. 3.2) from $(\vec{R}, \vec{S}, \vec{F})$. ($\tau$.T)

We have argued above that if the tuple $(\vec{R}, \vec{S}, \vec{F}, R', S', F')$ is trivial then the $(\vec{R}, \vec{S}, \vec{F}, R', S', F')$-**über** problem is trivial to solve, even with a generic algorithm. In Theorem 3.5 we now show that if the tuple is *non-trivial* then the corresponding Uber assumption holds for algebraic algorithms, as long as a type of q-DLog assumption holds (whose type depends on the type of bilinear group).

The (additive) security loss of the reduction depends on the degrees of the polynomials involved (as well as the group type and its order). E.g., in Type-3 groups, if R' is independent of $\vec{R}$ then the probability that the reduction fails is the maximum degree of R' and the components of $\vec{R}$, divided by the order of $\mathcal{G}$. In Type-1 and Type-2 groups, due to the homomorphism ψ, the loss depends on the maximum degree of R', $\vec{R}$ and $\vec{S}$. Similar bounds hold when S' is independent of $\vec{S}$ (and $\vec{R}$ for Type 1); and slightly more involved ones for the independence of F'. If several of R', S' and F' are independent then the reduction chooses the strategy that minimizes the security loss.

Definition 3.4 (Degree of non-trivial tuple of polynomials). Let $(\vec{R}, \vec{S}, \vec{F}, R', S', F')$ be a non-trivial tuple of polynomials in $\mathbb{Z}_p[X_1, \ldots, X_m]$. Define $d_{\vec{R}} := \max\{\deg R_i\}_{1 \le i \le r}$, $d_{\vec{S}} := \max\{\deg S_i\}_{1 \le i \le s}$, $d_{\vec{F}} := \max\{\deg F_i\}_{1 \le i \le f}$. We define the *type-$\tau$ degree* d_τ of $(\vec{R}, \vec{S}, \vec{F}, R', S', F')$ as follows:

- If (τ.1) holds, let $d_{\tau.1} := \max\{\deg R', d_{\vec{R}}, d_{\vec{S}}\}$ in case $\tau \in \{1, 2\}$ and
  $d_{\tau.1} := \max\{\deg R', d_{\vec{R}}\}$ in case $\tau = 3$.
- If (τ.2) holds, let $d_{\tau.2} := \max\{\deg S', d_{\vec{R}}, d_{\vec{S}}\}$ in case $\tau = 1$ and
  $d_{\tau.2} := \max\{\deg S', d_{\vec{S}}\}$ in case $\tau \in \{2, 3\}$.

- If $(\tau.\mathrm{T})$ holds, let $d_{\tau.T} := \max\{\deg F', 2\,d_{\vec{R}}, 2\,d_{\vec{S}}, d_{\vec{F}}\}$ when $\tau = 1$,
  $d_{\tau.T} := \max\{\deg F', d_{\vec{R}} + d_{\vec{S}}, 2\,d_{\vec{S}}, d_{\vec{F}}\}$ in case $\tau = 2$ and
  $d_{\tau.T} := \max\{\deg F', d_{\vec{R}} + d_{\vec{S}}, d_{\vec{F}}\}$ in case $\tau = 3$.

If (τ, i) does not hold, we set $d_{\tau,i} := \infty$ and define $d_\tau := \min\{d_{\tau.1}, d_{\tau.2}, d_{\tau.T}\}$. (By non-triviality, we have $d_\tau < \infty$.)

Theorem 3.5 (DLog implies Uber in the AGM). *Let $\mathcal{G}$ be of type $\tau \in \{1,2,3\}$ and $(\vec{R}, \vec{S}, \vec{F}, R', S', F') \in (\mathbb{Z}_p[X_1, \ldots, X_m])^{r+s+f+3}$ be a tuple of polynomials that is non-trivial for type τ and define $d_{\vec{R}} := \max\{\deg R_i\}$, $d_{\vec{S}} := \max\{\deg S_i\}$, $d_{\vec{F}} := \max\{\deg F_i\}$. Let q, q_1, q_2 be such that $q \geq \max\{d_{\vec{R}}, d_{\vec{S}}, d_{\vec{F}}/2\}$ as well as $q_1 \geq d_{\vec{R}}$, $q_2 \geq d_{\vec{S}}$ and $q_1 + q_2 \geq d_{\vec{F}}$. If*

(Type 1) *q-**dlog**$_{\mathcal{G}_1}$ or q-**dlog**$_{\mathcal{G}_2}$ is (ε, t)-secure in the AGM,*

(Type 2) *q-**dlog**$_{\mathcal{G}_2}$ is (ε, t)-secure in the AGM,*

(Type 3) *(q_1, q_2)-**dlog**$_{\mathcal{G}}$ is (ε, t)-secure in the AGM,*

*then $(\vec{R}, \vec{S}, \vec{F}, R', S', F')$-**über**$_{\mathcal{G}}$ is (ε', t')-secure in the AGM with*

$$\varepsilon' \leq \varepsilon + \tfrac{d_\tau}{p-1} \quad \text{and} \quad t' \leq t + o_1,$$

where d_τ is the maximal degree of $(\vec{R}, \vec{S}, \vec{F}, R', S', F')$, as defined in Def. 3.4,

$$o_1 := o_0 + 2 + (2\lfloor \log_2(p) \rfloor)((d_{\vec{R}} + 1)r + (d_{\vec{S}} + 1)s + (d_{\vec{F}} + 1)f + d_\tau) + rd_{\vec{R}} + sd_{\vec{S}} + fd_{\vec{F}}$$

with $o_0 := d_{\vec{R}} + d_{\vec{F}} + 2$ for Types 1 and 2, and $o_0 := d_{\vec{F}} + 1$ for in Type 3.

Proof. We give a detailed proof for Type-2 bilinear groups and then explain how to adapt it to Types 1 and 3. For $u \in \mathbb{Z}_p$ and $i \in \{1, 2, T\}$ we let $[u]_i$ denote g_i^u.

Let $\mathsf{A}_{\mathsf{alg}}$ be an algebraic algorithm against **über**$_{\mathcal{G}}$ that wins with advantage ε in time t. We construct a generic reduction with oracle access to $\mathsf{A}_{\mathsf{alg}}$, which yields an algebraic adversary $\mathsf{B}_{\mathsf{alg}}$ against q-**dlog**$_{\mathcal{G}_2}$. There are three (non-exclusive) reasons for $(\vec{R}, \vec{S}, \vec{F}, R', S', F')$ being non-trivial, which correspond to conditions (2.1), (2.2) and (2.T) in Def. 3.3. Each condition enables a different type of reduction, of which $\mathsf{B}_{\mathsf{alg}}$ runs the one that minimizes d_2 from Def. 3.4.

We start with Case (2.1), that is, R' is linearly independent from $\vec{R}$ and $\vec{S}$.

Adversary $\mathsf{B}_{\mathsf{alg}}(g_2, \mathbf{Z}_1, \ldots, \mathbf{Z}_q)$*:* On input a problem instance of game q-**dlog**$_{\mathcal{G}_2}$ with $\mathbf{Z}_i = [z^i]_2$, $\mathsf{B}_{\mathsf{alg}}$ simulates **über**$_{\mathcal{G}}$ for $\mathsf{A}_{\mathsf{alg}}$. It defines $g_1 \leftarrow \psi(g_2)$ and $g_T \leftarrow e(g_1, g_2)$. Then, it picks random values $\vec{y} \xleftarrow{\$} (\mathbb{Z}_p^*)^m$ and $\vec{v} \xleftarrow{\$} \mathbb{Z}_p^m$, implicitly sets $x_i := y_i z + v_i \bmod p$ and computes $\vec{\mathbf{U}} := [\vec{R}\,(x_1, \ldots, x_m)]_1$, $\vec{\mathbf{V}} := [\vec{S}\,(x_1, \ldots, x_m)]_2$, $\vec{\mathbf{W}} := [\vec{F}\,(x_1, \ldots, x_m)]_T$ from its q-DLog instance, the isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$ and the pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. It can do so efficiently since the total degrees of the polynomials in $\vec{R}$, $\vec{S}$ and $\vec{F}$ are bounded by q, q and $2q$ respectively.[1]

Next, $\mathsf{B}_{\mathsf{alg}}$ runs $(\mathbf{U}', \mathbf{V}', \mathbf{W}') \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}(\vec{\mathbf{U}}, \vec{\mathbf{V}}, \vec{\mathbf{W}})$. Since $\mathsf{A}_{\mathsf{alg}}$ is algebraic, it also returns vectors and matrices $\vec{\mu}, \vec{\nu}, \vec{\zeta}, \vec{\delta}, A = (\alpha_{i,j})_{i,j}, \Gamma = (\gamma_{i,j})_{i,j}$ such that

$$\mathbf{U}' = \textstyle\prod_i \mathbf{U}_i^{\mu_i} \cdot \prod_i \psi(\mathbf{V}_i)^{\nu_i} \tag{2a}$$
$$\mathbf{V}' = \textstyle\prod_i \mathbf{V}_i^{\eta_i} \tag{2b}$$

[1] E.g., $\mathsf{B}_{\mathsf{alg}}$ can compute $[x_1^q]_1 = [(y_1 z + v_1)^q]_1$ as $\prod_i \psi(\mathbf{Z}_i)^{\binom{q}{i} y_1^i v_1^{q-i}}$ and $[x_1^{2q}]_T$ as $e\big(\prod_i \psi(\mathbf{Z}_i)^{\binom{q}{i} y_1^i v_1^{q-i}}, \prod_i \mathbf{Z}_i^{\binom{q}{i} y_1^i v_1^{q-i}}\big)$ and similarly for terms in more variables.

$$\mathbf{W}' = \prod_i \prod_j e(\mathbf{U}_i, \mathbf{V}_j)^{\alpha_{i,j}} \cdot \prod_i \prod_j e(\psi(\mathbf{V}_i), \mathbf{V}_j)^{\gamma_{i,j}} \cdot \prod_i \mathbf{W}_i^{\delta_i}. \tag{2c}$$

$\mathsf{B}_{\mathsf{alg}}$ then computes the following multivariate polynomial, which corresponds to the exponents of (2a):

$$P_1(\vec{X}) = R'(\vec{X}) - \textstyle\sum_{i=1}^{r} \mu_i R_i(\vec{X}) - \sum_{i=1}^{s} \nu_i S_i(\vec{X}), \tag{3}$$

which is non-zero because in Case (2.1) R' is independent from $\vec{R}$ and $\vec{S}$. From P_1, it defines the univariate polynomial

$$Q_1(Z) := P_1(y_1 Z + v_1, \ldots, y_m Z + v_m). \tag{4}$$

If Q_1 is the zero polynomial then $\mathsf{B}_{\mathsf{alg}}$ aborts. $(*)$

Else, it factors Q_1 to obtain its roots $z_1, \ldots$ (of which there are at most $\max\{\deg R', d_{\vec{R}}, d_{\vec{S}}\}$; we analyse the degree of Q_1 below). If for one of them we have $g_2^{z_i} = \mathbf{Z}$, then $\mathsf{B}_{\mathsf{alg}}$ returns z_i.

We analyze $\mathsf{B}_{\mathsf{alg}}$'s success probability. First note that $\mathsf{B}_{\mathsf{alg}}$ perfectly simulates game $\mathbf{\ddot{u}ber}_{\mathcal{G}}$, as the values x_i are uniformly distributed in $\mathbb{Z}_p$ and $\vec{\mathbf{U}}$, $\vec{\mathbf{V}}$ and $\vec{\mathbf{W}}$ are correctly computed. Moreover, if $\mathsf{B}_{\mathsf{alg}}$ does not abort in $(*)$ and $\mathsf{A}_{\mathsf{alg}}$ wins game $\mathbf{\ddot{u}ber}_{\mathcal{G}}$, then $\mathbf{U}' = [R'(\vec{x})]_1$. On the other hand,

$$\mathbf{U}' = \prod_i \mathbf{U}_i^{\mu_i} \cdot \psi(\prod_i \mathbf{V}_i^{\nu_i}) = \left[\textstyle\sum_i \mu_i R_i(\vec{x}) + \sum_i \nu_i S_i(\vec{x})\right]_1.$$

Together, this means that $P_1(\vec{x}) \equiv_p 0$ and since $x_i \equiv_p y_i z + v_i$, moreover $Q_1(z) \equiv_p 0$. By factoring Q_1, reduction $\mathsf{B}_{\mathsf{alg}}$ finds thus the solution z.

It remains to bound the probability that $\mathsf{B}_{\mathsf{alg}}$ aborts in $(*)$, that is, the event that $0 \equiv Q_1(Z) = P_1(y_1 Z + v_1, \ldots, y_m Z + v_m)$. Interpreting Q_1 as an element from $(\mathbb{Z}_p[Y_1, \ldots, Y_m, V_1, \ldots, V_m])[Z]$, Lemma 2.1 yields that its maximal coefficient is a polynomial $Q_1^{\max}$ in $Y_1, \ldots, Y_m$ whose degree is the same as the maximal (total) degree d of P_1. From $P_1 \not\equiv 0$ and $P_1(\vec{x}) = 0$, we have $d > 0$.

We note that the values $y_1 z, \ldots, y_m z$ are completely hidden from $\mathsf{A}_{\mathsf{alg}}$ because they are "one-time-padded" with $v_1, \ldots, v_m$, respectively. This means that the values $(\vec{\mu}, \vec{\nu})$ returned by $\mathsf{A}_{\mathsf{alg}}$ are independent from $\vec{y}$. Since $\vec{y}$ is moreover independent from R', $\vec{R}$ and $\vec{S}$, it is also independent from P_1, Q_1 and $Q_1^{\max}$. The probability that $Q_1 \equiv 0$ is thus upper-bounded by the probability that its maximal coefficient $Q_1^{\max}(\vec{y}) \equiv_p 0$ when evaluated at the random point $\vec{y}$. By the Schwartz-Zippel lemma, the probability that $Q_1(Z) \equiv 0$ is thus upper-bounded by $\frac{d}{p-1}$. The degree d of Q_1 (and thus of $Q_1^{\max}$) is upper-bounded by the total degrees of P_1, which is $\max\{d'_R, d_{\vec{R}}, d_{\vec{S}}\} = d_{2.1}$ in Def. 3.4. $\mathsf{B}_{\mathsf{alg}}$ thus aborts in line (*) with probability at most $\frac{d_{2,1}}{p-1}$.

Case (2.2), that is, S' is linearly independent from $\vec{S}$, follows completely analogously, but with $d = d_{2.2} = \max\{d_{S'}, d_{\vec{S}}\}$.

Case (2.T), when F' is type-2-independent of $\vec{R}$, $\vec{S}$ and $\vec{F}$, is also analogous; we highlight the necessary changes: From $\mathsf{A}_{\mathsf{alg}}$'s representation $(A = (\alpha_{i,j}), \Gamma = (\gamma_{i,j}), \vec{\delta}) \in \mathbb{Z}_p^{r \times s} \times \mathbb{Z}_p^{s \times s} \times \mathbb{Z}_p^f$ for $\mathbf{W}'$ (see (2c)), that is,

$$\begin{aligned} \mathbf{W}' &= \prod_i \prod_j e(\mathbf{U}_i, \mathbf{V}_j)^{\alpha_{i,j}} \cdot \prod_i \prod_j e(\psi(\mathbf{V}_i), \mathbf{V}_j)^{\gamma_{i,j}} \cdot \prod_i \mathbf{W}_i^{\delta_i} \\ &= \left[\textstyle\sum_i \sum_j \alpha_{i,j} R_i(\vec{x}) S_j(\vec{x}) + \sum_i \sum_j \gamma_{i,j} S_i(\vec{x}) S_j(\vec{x}) + \sum_i \delta_i F_i(\vec{x})\right]_T. \end{aligned} \tag{5}$$

Analogously to (3) we define

$$\begin{aligned} P_T(\vec{X}) := F'(\vec{X}) - \textstyle\sum_{i=1}^{r} \sum_{j=1}^{s} \alpha_{i,j} R_i(\vec{X}) S_j(\vec{X}) \\ - \textstyle\sum_{i=1}^{s} \sum_{j=1}^{s} \gamma_{i,j} S_i(\vec{X}) S_j(\vec{X}) - \sum_{i=1}^{f} \delta_i F_i(\vec{X}), \end{aligned} \tag{6}$$

which is of degree at most $d_{2.T} := \max\{\deg F', d_{\vec{R}} + d_{\vec{S}}, 2 \cdot d_{\vec{S}}, d_{\vec{F}}\}$. Polynomial P_T is non-zero by Type-2-independence of F' (Def. 3.2). The reduction also computes $Q_T(Z) := P_T(y_1 Z + v_1, \ldots, y_m Z + v_m)$.

If $\mathsf{A}_{\mathsf{alg}}$ wins then $\mathbf{W}' = [F'(\vec{x})]_T$, which together with (5) implies that $P_T(\vec{x}) \equiv_p 0$ and thus $Q_T(z) \equiv_p 0$. Reduction $\mathsf{B}_{\mathsf{alg}}$ can find z by factoring Q_T; unless $Q_T(Z) \equiv 0$, which by an analysis analogous to the one for case (2.1) happens with probability $\frac{d_{2.T}}{p-1}$. (We detail the reduction for the case where $\mathbf{W}'$ is independent in the proof of Theorem 5.2, which proves a more general statement.)

Theorem 3.5 for Type-2 groups follows since $\mathbf{Adv}^{q\text{-}\mathbf{dlog}}_{\mathcal{G}_2,\mathsf{B}_{\mathsf{alg}}} \geq \mathbf{Adv}^{\ddot{\mathbf{u}}\mathbf{ber}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}} - \Pr[\mathsf{B}_{\mathsf{alg}} \text{ aborts}]$ and $\mathsf{B}_{\mathsf{alg}}$ follows the type of reduction that minimizes its abort probability to $\min\{\frac{d_{2.1}}{p-1}, \frac{d_{2.2}}{p-1}, \frac{d_{2.T}}{p-1}\} = \frac{d_2}{p-1}$.

Groups of Type 1 and 3. The reduction for bilinear groups of Type 1 to q-$\mathbf{dlog}_{\mathcal{G}_2}$ is almost the same proof. The only change is that for Case (1.T) the polynomial P_T in (6) has an extra term $-\sum_{i=1}^{r}\sum_{j=1}^{r}\beta_{i,j}R_i(\vec{X})R_j(\vec{X})$, because of the representation of $\mathbf{W}'$ in Type-1 groups (see Def. 2.4); the degree of P_T is then bounded by $\max\{\deg F', 2\,d_{\vec{R}}, 2\,d_{\vec{S}}, d_{\vec{F}}\}$. Analogously for Case (1.2), S' can now depend on $\vec{S}$ as well as $\vec{R}$. The reduction for Type-1 groups to q-$\mathbf{dlog}_{\mathcal{G}_1}$ is completely symmetric by swapping the roles of $\mathbb{G}_1$ and $\mathbb{G}_2$ and replacing ψ by ϕ.

The reduction for Type-3 groups relies on the (q_1, q_2)-$\mathbf{dlog}_{\mathcal{G}}$ assumption, as it requires $\{[z_i]_1\}_{i=1}^{q_1}$ and $\{[z_i]_2\}_{i=1}^{q_2}$ to simulate $\{[R_i(\vec{x})]_1\}_{i=1}^{r}$ and $\{[S_i(\vec{x})]_2\}_{i=1}^{s}$. without using any homomorphism ϕ or ψ. Apart from this, the proof is again analogous. (We treat the Type-3 case in detail in the proof of Theorem 5.2.) In Appendix B we detail the analysis of the running times of these reductions. ■

Using Lemmas 2.5 and 2.6 we obtain the following corollary to Theorem 3.5:

Corollary 3.6 *Let $\mathcal{G}$ be of type τ and $(\vec{R}, \vec{S}, \vec{F}, R', S', F')$ be non-trivial for τ of maximal degree d_τ. Then $(\vec{R}, \vec{S}, \vec{F}, R', S', F')$-$\ddot{\mathbf{u}}\mathbf{ber}_{\mathcal{G}}$ is $(\frac{(o+o_1+1+q)^2 q}{p-1} + \frac{d_\tau}{p-1}, o)$-secure in the generic bilinear group model.*

Comparison to previous GGM results. Boneh, Boyen and Goh [BBG05, Theorem A.2] claim that the decisional Uber assumption for the particular case $r = s$ and $f = 0$ is $(\frac{(o+2r+2)^2 q}{2p}, o)$-secure in the generic group model, and with the same reasoning, one can obtain the more general bound $(\frac{(o+r+s+f+2)^2 q}{2p}, o)$. Note that the loss in their bound is only linear in the maximum degree while ours *cubic*. Our looser bound is a result of our reduction, whereas Boneh, Boyen and Goh prove their bound directly in the GGM.[2]

4 The Flexible Uber Assumption

Boyen [Boy08] generalizes the Uber assumption framework to *flexible* assumptions, where the adversary can define the target polynomials (R', S' and F' in Fig. 2) itself, conditioned on the solution not being trivially computable from the instance, for *non-triviality* as in Def. 3.3. In Sect. 6 we consider this kind of flexible Uber assumption in our generalization to rational fractions and thereby cover assumptions like q-strong Diffie-Hellman [BB08].

For polynomials, we generalize this further by allowing the adversary to also (adaptively) choose the polynomials that constitute the challenge. The adversary is provided with an oracle that takes input a value $i \in \{1, 2, T\}$ and a polynomial $P(\vec{X})$ of the adversary's choice, and returns $g_i^{P(\vec{x})}$, where $\vec{x}$ is the secret value chosen during the game. The adversary then wins if it returns

[2] We did not consider the bound from [Boy08, Theorem 1], as it is an incorrect copy of the one in [BBG05].

m-**f-über**$^{\mathsf{A}_{\mathsf{alg}}}_{\mathcal{G}}$	$\underline{\mathsf{O}}(i, P(\vec{X}))$
01 $\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_T \leftarrow \emptyset$	06 $\mathcal{Q}_i = \mathcal{Q}_i \cup \{P\}$
02 $(g_1, g_2) \xleftarrow{\$} \mathsf{GenSamp}$; $g_T \leftarrow e(g_1, g_2)$	07 Return $g_i^{P(\vec{x})}$
03 $\vec{x} = (x_1, \ldots, x_m) \xleftarrow{\$} \mathbb{Z}_p^m$	
04 $((\mathbf{U}^*, \mathbf{V}^*, \mathbf{W}^*), (R^*, S^*, F^*)) \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}^{\mathsf{O}(\cdot,\cdot)}()$	
05 Return $\Big((\mathbf{U}^*, \mathbf{V}^*, \mathbf{W}^*) = (g_1^{R^*(\vec{x})}, g_2^{S^*(\vec{x})}, g_T^{F^*(\vec{x})})$ $\wedge$ $(\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_T, R^*, S^*, F^*)$ non-trivial for type of $\mathcal{G}\Big)$	

Figure 3: Algebraic game for the flexible Uber assumption

polynomials (R^*, S^*, F^*), which are independent from its queries, and $(g_1^{R^*(\vec{x})}, g_2^{S^*(\vec{x})}, g_T^{F^*(\vec{x})})$. The game for this flexible Uber assumption is specified in Fig. 3.

Theorem 4.1 *Let $m \geq 1$, let $\mathcal{G}$ be a bilinear-group of type $\tau \in \{1, 2, 3\}$ and consider an adversary $\mathsf{A}_{\mathsf{alg}}$ in game m-**f-über**$_{\mathcal{G}}$. Let $d_1', d_2', d_T', d_1^*, d_2^*, d_T^*$ be such that $\mathsf{A}_{\mathsf{alg}}$'s queries $(i, P(\vec{X}))$ satisfy $\deg P \leq d_i'$ and its output values R^*, S^*, F^* satisfy $\deg R^* \leq d_1^*$, $\deg S^* \leq d_2^*$, $\deg F^* \leq d_T^*$. Let q, q_1, q_2 be such that $q \geq \max\{d_1', d_2', d_T'/2\}$ as well as $q_1 \geq d_1'$, $q_2 \geq d_2'$ and $q_1 + q_2 \geq d_T'$. If*

(Type 1) q-**dlog**$_{\mathcal{G}_1}$ *or* q-**dlog**$_{\mathcal{G}_2}$ *is (ε, t)-secure in the AGM,*

(Type 2) q-**dlog**$_{\mathcal{G}_2}$ *is (ε, t)-secure in the AGM,*

(Type 3) (q_1, q_2)-**dlog**$_{\mathcal{G}}$ *is (ε, t)-secure in the AGM,*

*then m-**f-über**$_{\mathcal{G}}$ is (ε', t')-secure in the AGM with*

$$\varepsilon' \leq \varepsilon + \tfrac{d_\tau}{p-1} \quad \text{and} \quad t' \approx t,$$

where d_τ is as in Def. 3.4 after the following replacements: $d_{\vec{R}} \leftarrow d_1'$, $d_{\vec{S}} \leftarrow d_2'$, $d_{\vec{F}} \leftarrow d_F'$, $\deg R' \leftarrow d_1^*$, $\deg S' \leftarrow d_2^*$ and $\deg F' \leftarrow d_T^*$.

Proof sketch. Inspecting the proof of Theorem 3.5, note that the values $[R_i(\vec{x})]_1$, $[S_i(\vec{x})]_2$ and $[F_i(\vec{x})]_T$ need not be known in advance and can be computed by the reduction at any point, as long as the degrees of R_i and S_i are bounded by q and those of F_i by $2q$. The adversary could thus specify the polynomials via oracle calls and the reduction can compute $\mathbf{U}_i$, $\mathbf{V}_i$ and $\mathbf{F}_i$ on the fly.

Likewise, the polynomials P_1, P_2 and P_T (and their univariate counterparts which the reduction factors) are only defined after $\mathsf{A}_{\mathsf{alg}}$ stops; therefore, R', S' and F', from which they are defined, need only be known then. The proof of Theorem 3.5 is thus adapted to prove Theorem 4.1 in a very straightforward way. ∎

5 The Uber Assumption for Rational Fractions

Reconsider the Uber assumption in Fig. 2, but now let $\vec{R}, \vec{S}, \vec{F}, R', S'$ and F' be *rational fractions* over $\mathbb{Z}_p$ rather than polynomials. We will show that even this generalization of the Uber assumption is implied by q-DLog assumptions. We start with introducing some notation. We view a rational fraction as defined by two polynomials, its numerator and its denominator, and assume that the fraction is reduced. For a rational fraction $R \in \mathbb{Z}_p(X_1, \ldots, X_m)$, we denote its numerator by $\hat{R}$ and its denominator by $\check{R}$. That is $\hat{R}, \check{R} \in \mathbb{Z}_p[X_1, \ldots, X_m]$ are such that

$(\vec{R}, \vec{S}, \vec{F}, R', S', F')$-**rüber**$^{\mathsf{A}_{\mathsf{alg}}}_{\mathcal{G}}$ // $R_i = \hat{R}_i/\check{R}_i$, for $\hat{R}_i, \check{R}_i \in \mathbb{Z}_p[\vec{X}]$, etc

01 $(g_1, g_2) \xleftarrow{\$} \mathsf{GenSamp}$; $g_T \leftarrow e(g_1, g_2)$

02 $\vec{x} = (x_1, \ldots, x_m) \xleftarrow{\$} \mathbb{Z}_p^m$

03 If for some i: $\check{R}_i(\vec{x}) \equiv_p 0$ or $\check{S}_i(\vec{x}) \equiv_p 0$ or $\check{F}_i(\vec{x}) \equiv_p 0$ then return 1

04 If $\check{R}'(\vec{x}) \equiv_p 0$ or $\check{S}'(\vec{x}) \equiv_p 0$ or $\check{F}'(\vec{x}) \equiv_p 0$ then return 1

05 $\vec{\mathbf{U}} := (g_1^{R_1(\vec{x})}, \ldots, g_1^{R_r(\vec{x})})$; $\vec{\mathbf{V}} := (g_2^{S_1(\vec{x})}, \ldots, g_2^{S_s(\vec{x})})$; $\vec{\mathbf{W}} := (g_T^{F_1(\vec{x})}, \ldots, g_T^{F_f(\vec{x})})$

06 $(\mathbf{U}', \mathbf{V}', \mathbf{W}') \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}(\vec{\mathbf{U}}, \vec{\mathbf{V}}, \vec{\mathbf{W}})$

07 Return $((\mathbf{U}', \mathbf{V}', \mathbf{W}') = (g_1^{R'(\vec{x})}, g_2^{S'(\vec{x})}, g_T^{F'(\vec{x})}))$

Figure 4: Algebraic game for the Uber assumption relative to bilinear group $\mathcal{G}$ and adversary $\mathsf{A}_{\mathsf{alg}}$, parametrized by (vectors of) rational fractions $\vec{R}, \vec{S}, \vec{F}, R', S'$ and F'

$R = \hat{R}/\check{R}$. As rational fractions are not defined everywhere, we modify the game from Fig. 2 so the adversary wins should the experiment choose an input $\vec{x}$ for which one of the rational fractions is not defined. The rational-fraction uber game is given in Fig. 4.

For a vector of rational fractions $\vec{R} \in \mathbb{Z}_p(X_1, \ldots, X_m)^r$, we define its *common denominator* $\mathrm{Den}(\vec{R})$ as a least common multiple of the denominators of the components of $\vec{R}$. In particular, fix an algorithm LCM that given a set of polynomials returns a least common multiple of them. Then we define:

$$\mathrm{Den}(\vec{R}) = \mathrm{Den}(\hat{R}_1/\check{R}_1, \ldots, \hat{R}_r/\check{R}_r) := \mathrm{LCM}\{\check{R}_1, \ldots, \check{R}_r\}.$$

We let $\check{d}_{\vec{R}}$ denote the degree of $\mathrm{Den}(\vec{R})$ and $d_{\vec{R}}$ denote the maximal degree of the elements of $\vec{R}$, that is $d_{\vec{R}} := \max\{\deg(R_1), \ldots, \deg(R_r)\}$. Note that this integer could be negative and is lower bounded by $-\check{d}_{\vec{R}}$. The security loss in Theorem 5.2 depends on the type of the bilinear group, the reason for the tuple $(\vec{R}, \vec{S}, \vec{F}, R', S', F')$ being non-trivial, as well as the degrees of the numerators and denominators of the involved rational fractions. We summarize this in the following technical definition.

Definition 5.1 (Degree of non-trivial tuple of rational fractions). Let $(\vec{R}, \vec{S}, \vec{F}, R', S', F')$ be a non-trivial tuple whose elements are rational fractions in $\mathbb{Z}_p(X_1, \ldots, X_m)$. Let $d_{\mathrm{den}} := \check{d}_{\vec{R}\|\vec{S}\|\vec{F}\|R'\|S'\|F'}$. We define the *type-$\tau$ degree* d_τ of $(\vec{R}, \vec{S}, \vec{F}, R', S', F')$ as follows, distinguishing the kinds of non-triviality defined in Def. 3.3.

(Type 1)
- If (1.1) holds, let $d_{1.1} := d_{\mathrm{den}} + \check{d}_{R'} + \check{d}_{\vec{R}\|\vec{S}} + \max\{d_{R'}, d_{\vec{R}}, d_{\vec{S}}\}$
- if (1.2) holds, let $d_{1.2} := d_{\mathrm{den}} + \check{d}_{S'} + \check{d}_{\vec{R}\|\vec{S}} + \max\{d_{S'}, d_{\vec{R}}, d_{\vec{S}}\}$
- if (1.T) holds, $d_{1.T} := d_{\mathrm{den}} + \check{d}_{F'} + \check{d}_{\vec{R}\|\vec{S}\|\vec{F}} + \check{d}_{\vec{R}\|\vec{S}} + \max\{d_{F'}, 2d_{\vec{S}}, 2d_{\vec{R}}, d_{\vec{F}}\}$

(Type 2)
- If (2.1) holds, let $d_{2.1} := d_{\mathrm{den}} + \check{d}_{R'} + \check{d}_{\vec{R}\|\vec{S}} + \max\{d_{R'}, d_{\vec{R}}, d_{\vec{S}}\}$
- if (2.2) holds, let $d_{2.2} := d_{\mathrm{den}} + \check{d}_{S'} + \check{d}_{\vec{S}} + \max\{d_{S'}, d_{\vec{S}}\}$
- if (2.T) holds, $d_{2.T} := d_{\mathrm{den}} + \check{d}_{F'} + \check{d}_{\vec{R}\|\vec{S}\|\vec{F}} + \check{d}_{\vec{S}} + \max\{d_{F'}, 2d_{\vec{S}}, d_{\vec{R}} + d_{\vec{S}}, d_{\vec{F}}\}$

(Type 3)
- If (3.1) holds, let $d_{3.1} := d_{\mathrm{den}} + \check{d}_{R'} + \check{d}_{\vec{R}} + \max\{d_{R'}, d_{\vec{R}}\}$
- if (3.2) holds, let $d_{3.2} := d_{\mathrm{den}} + \check{d}_{S'} + \check{d}_{\vec{S}} + \max\{d_{S'}, d_{\vec{S}}\}$

- if (3.T) holds, $d_{3.T} := d_{\text{den}} + \check{d}_{F'} + \check{d}_{\vec{R}\|\vec{F}} + \check{d}_{\vec{S}} + \max\{d_{F'}, d_{\vec{R}} + d_{\vec{S}}, d_{\vec{F}}\}$

If (τ, i) does not hold, set $d_{\tau,i} := \infty$. Define $d_\tau := \min\{d_{\tau.1}, d_{\tau.2}, d_{\tau.T}\}$.

Theorem 5.2 (DLog implies Uber for rational fractions in the AGM). *Let $\mathcal{G}$ be a bilinear group of type $\tau \in \{1,2,3\}$ and let $(\vec{R}, \vec{S}, \vec{F}, R', S', F') \in (\mathbb{Z}_p(X_1, \ldots, X_m))^{r+s+f+3}$ be a tuple of rational fractions that is non-trivial for type τ (Def. 3.3). Let q, q_1 and q_2 be such that $q \geq \check{d}_{\vec{R}\|\vec{S}\|\vec{F}} + \max\{d_{\vec{R}}, d_{\vec{S}}, d_{\vec{F}}/2\}$ and $q_1 \geq \check{d}_{\vec{R}\|\vec{F}} + d_{\vec{R}}$ and $q_2 \geq \check{d}_{\vec{S}} + d_{\vec{S}}$ as well as $q_1 + q_2 \geq \check{d}_{\vec{R}\|\vec{F}} + \check{d}_{\vec{S}} + d_{\vec{F}}$. If*

(Type 1) *q-**dlog**$_{\mathcal{G}_1}$ or q-**dlog**$_{\mathcal{G}_2}$ is (ε, t)-secure in the AGM,*

(Type 2) *q-**dlog**$_{\mathcal{G}_2}$ is (ε, t)-secure in the AGM,*

(Type 3) *(q_1, q_2)-**dlog**$_{\mathcal{G}}$ is (ε, t)-secure in the AGM,*

*then $(\vec{R}, \vec{S}, \vec{F}, R', S', F')$-**rüber**$_{\mathcal{G}}$, as defined in Fig. 4, is (ε', t')-secure in the AGM with*

$$\varepsilon' \leq \varepsilon + \tfrac{d_\tau}{p-1} \quad \text{and} \quad t' \approx t,$$

where d_τ is the maximal degree of $(\vec{R}, \vec{S}, \vec{F}, R', S', F')$, as defined in Def. 5.1.

The proof extends the ideas used to prove Theorem 3.5 by employing a technique from [BB08]. Consider a group of Type 1 or 2. The reduction computes $D := \text{Den}(\vec{R}\|\vec{S}\|\vec{F})$, a least common multiple of the denominators of the instance. Given a q-DLog instance $g_2, g_2^z, g_2^{z^2}, \ldots$, it first implicitly sets $x_i := y_i z + v_i \bmod p$, then it checks whether any denominator evaluates to zero at $\vec{x}$ (this entails the additive loss d_{den}). Then it computes a new random generator $h_2 := g_2^{D(y_1 z + v_1, \ldots, y_m z + v_m)}$ and $h_1 := \psi(h_2)$. For rational fractions $S_i = \hat{S}_i / \check{S}_i$, it then uses h_1, h_2 to compute the Uber challenge elements $h_2^{S_i(\vec{x})}$ as $g_2^{\overline{S}(\vec{x})}$ for the *polynomial* $\overline{S}(\vec{X}) := (\hat{S}_i \cdot D / \check{S}_i)(\vec{X})$, and likewise for R_i and F_i. This explains the lower bound on q in the theorem statement. When the adversary returns a group element $h_i^{R'(\vec{x})}$ so that R' is non-trivial, then from the algebraic representations of this element we can define a polynomial (which with overwhelming probability is non-zero) that vanishes at z. The difference here is that we expand by the denominator of R' in order to obtain a polynomial. The degree of this polynomial is bounded by the values in Def. 5.1, which also bound the failure probability of the reduction. In Type-3 groups, the reduction can set $h_1 := g_1^{\text{Den}(\vec{R}\|\vec{F})(\vec{x})}$ and $h_1 := g_2^{\text{Den}(\vec{S})(\vec{x})}$, which leads to better bounds. We detail this case in our proof of Theorem 5.2, which can be found in Appendix C.

6 The Uber Assumption for Rational Fractions and Flexible Targets

For rational fractions, we can also define a flexible generalization, where the adversary can choose the target polynomials R', S' and F' in Fig. 2 itself, conditioned on the tuple $(\vec{R}, \vec{S}, \vec{F}, R', S', F')$ being non-trivial. The game is specified in Fig. 5. This extension covers assumptions such as the q-strong DH assumption by Boneh and Boyen [BB08], which they proved secure in the generic group model. A q-SDH adversary is given $(g_i, g_i^z, g_i^{z^2}, \ldots, g_i^{z^q})$ for $i = 1, 2$ and must compute $(g_1^{(z+c)^{-1}}, c)$ for any $c \in \mathbb{Z}_p \setminus \{-z\}$ of its choice. This is an instance of the flexible game in Fig. 5 when setting $m = 1$, $r = s = q + 1$, $f = 0$ and $R_i(X) = S_i(X) = X^{i-1}$, and the adversary returns $R^*(X) = 1/(X + c)$, $S^*(X) = F^*(X) = 0$.

$(\vec{R},\vec{S},\vec{F})$-**f-rüber**$^{\mathsf{A}_{\mathsf{alg}}}_{\mathcal{G}}$

$(g_1,g_2) \xleftarrow{\$} \mathsf{GenSamp}$; $g_T \leftarrow e(g_1,g_2)$

$\vec{x} = (x_1,\ldots,x_m) \xleftarrow{\$} \mathbb{Z}_p^m$

If for some i: $\check{R}_i(\vec{x}) \equiv_p 0$ or $\check{S}_i(\vec{x}) \equiv_p 0$ or $\check{F}_i(\vec{x}) \equiv_p 0$ then return 1

$\vec{\mathbf{U}} := (g_1^{R_1(\vec{x})},\ldots,g_1^{R_r(\vec{x})})$; $\vec{\mathbf{V}} := (g_2^{S_1(\vec{x})},\ldots,g_2^{S_s(\vec{x})})$; $\vec{\mathbf{W}} := (g_T^{F_1(\vec{x})},\ldots,g_T^{F_f(\vec{x})})$

$((\mathbf{U}^*,\mathbf{V}^*,\mathbf{W}^*),(R^*,S^*,F^*)) \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}(\vec{\mathbf{U}},\vec{\mathbf{V}},\vec{\mathbf{W}})$

Return $\Big((\mathbf{U}^*,\mathbf{V}^*,\mathbf{W}^*) = (g_1^{R^*(\vec{x})},g_2^{S^*(\vec{x})},g_T^{F^*(\vec{x})})$

$\wedge\ (\vec{R},\vec{S},\vec{F},R^*,S^*,F^*)$ non-trivial for type of $\mathcal{G}\Big)$

Figure 5: Algebraic game for the flexible-targets Uber assumption

Theorem 6.1 (DLog implies flexible-target Uber for rational fractions in the AGM). *Let $\mathcal{G}$ be a bilinear group of type $\tau \in \{1,2,3\}$ and let $(\vec{R},\vec{S},\vec{F}) \in (\mathbb{Z}_p(X_1,\ldots,X_m))^{r+s+f}$ be a tuple of rational fractions.*

*Consider an adversary $\mathsf{A}_{\mathsf{alg}}$ in game $(\vec{R},\vec{S},\vec{F})$-**f-rüber** (Fig. 5) and let $d_1^*,d_2^*,d_T^*,\check{d}_1^*,\check{d}_2^*,\check{d}_T^*$ be such that $\mathsf{A}_{\mathsf{alg}}$'s outputs R^*,S^*,F^* satisfy $\deg R^* \le d_1^*$, $\deg S^* \le d_2^*$, $\deg F^* \le d_T^*$, $\deg \check{R}^* \le \check{d}_1^*$, $\deg \check{S}^* \le \check{d}_2^*$ and $\deg \check{F}^* \le \check{d}_T^*$.*

Let q,q_1 and q_2 be such that $q \ge \check{d}_{\vec{R}\|\vec{S}\|\vec{F}} + \max\{d_{\vec{R}},d_{\vec{S}},d_{\vec{F}}/2\}$ and let $q_1 \ge \check{d}_{\vec{R}\|\vec{F}} + d_{\vec{R}}$ and $q_2 \ge \check{d}_{\vec{S}} + d_{\vec{S}}$ and $q_1+q_2 \ge \check{d}_{\vec{R}\|\vec{F}} + \check{d}_{\vec{S}} + d_{\vec{F}}$, where $\check{d}_{\vec{R}} = \check{d}_{(\hat{R}_1/\check{R}_1,\ldots,\hat{R}_r/\check{R}_r)} = \deg \mathrm{LCM}\{\check{R}_1,\ldots,\check{R}_r\}$. If

(Type 1) *q-**dlog**$_{\mathcal{G}_1}$ or q-**dlog**$_{\mathcal{G}_2}$ is (ε,t)-secure in the AGM,*

(Type 2) *q-**dlog**$_{\mathcal{G}_2}$ is (ε,t)-secure in the AGM,*

(Type 3) *(q_1,q_2)-**dlog**$_{\mathcal{G}}$ is (ε,t)-secure in the AGM,*

*then $(\vec{R},\vec{S},\vec{F})$-**f-rüber**$_{\mathcal{G}}$ is (ε',t')-secure in the AGM with*

$$\varepsilon' \le \varepsilon + \tfrac{d_\tau}{p-1} \quad \text{and} \quad t' \approx t,$$

where d_τ is defined as in Def. 5.1, except for defining $d_{\mathrm{den}} := \check{d}_{\vec{R}\|\vec{S}\|\vec{F}}$ and replacing $\check{d}_{R'},\check{d}_{S'},\check{d}_{F'}$, $d_{R'},d_{S'}$, and $d_{F'}$ by $\check{d}_1^*,\check{d}_2^*,\check{d}_T^*,d_1^*,d_2^*$, and d_T^*), respectively.

Proof sketch. Much in the way the proof of Theorem 3.5 is adapted to Theorem 4.1, Theorem 6.1 is proved similarly to Theorem 5.2. Since P_1,P_2 and P_T are only defined once the adversary returns its rational fractions R^*,S^*,F^*, they need not be known in advance. (Note that, unlike for polynomials (Theorem 4.1), the instance $(\vec{R},\vec{S},\vec{F})$ does have to be fixed, as the reduction uses it to set up the generators h_1 and h_2.) A difference to Theorem 6.1 is the value d_{den} in the security loss, which is now smaller since the experiment need not check the denominators of the target fractions. ∎

7 Uber Assumptions with Decisional Oracles

In this section we show that we can provide the adversary, essentially *for free*, with an oracle that checks whether the logarithms of given group elements satisfy any polynomial relation. In more detail, the adversary is given access to an oracle that takes as input a polynomial $P \in \mathbb{Z}_p[X_1,\ldots,X_n]$ and group elements $\mathbf{Y}_1,\ldots,\mathbf{Y}_n$ (from any group $\mathbb{G}_1,\mathbb{G}_2$ or $\mathbb{G}_T$) and checks

$(\vec{R},\vec{S},\vec{F})$-**f-drüber**$^{\mathsf{A}_{\mathsf{alg}}}_{\mathcal{G}}$

01 $(g_1,g_2) \xleftarrow{\$} \mathsf{GenSamp}$; $g_T \leftarrow e(g_1,g_2)$
02 $\vec{x}=(x_1,\ldots,x_m) \xleftarrow{\$} \mathbb{Z}_p^m$
03 If for some i: $\check{R}_i(\vec{x}) \equiv_p 0$ or $\check{S}_i(\vec{x}) \equiv_p 0$ or $\check{F}_i(\vec{x}) \equiv_p 0$
04 then return 1
05 $\vec{\mathbf{U}} := (g_1^{R_1(\vec{x})},\ldots,g_1^{R_r(\vec{x})})$
06 $\vec{\mathbf{V}} := (g_2^{S_1(\vec{x})},\ldots,g_2^{S_s(\vec{x})})$
07 $\vec{\mathbf{W}} := (g_T^{F_1(\vec{x})},\ldots,g_T^{F_f(\vec{x})})$
08 $((\mathbf{U}^*,\mathbf{V}^*,\mathbf{W}^*),(R^*,S^*,F^*)) \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}^{\mathsf{O}(\cdot,\cdot)}(\vec{\mathbf{U}},\vec{\mathbf{V}},\vec{\mathbf{W}})$
09 Return $\Big((\mathbf{U}^*,\mathbf{V}^*,\mathbf{W}^*) = (g_1^{R^*(\vec{x})},g_2^{S^*(\vec{x})},g_T^{F^*(\vec{x})})$
$\wedge\ (\vec{R},\vec{S},\vec{F},R^*,S^*,F^*)$ non-trivial for type of $\mathcal{G}\Big)$

$\underline{\mathsf{O}}(P(\vec{X}),(\mathbf{Y}_1,\ldots,\mathbf{Y}_n))$
10 For $i=1,\ldots,n$ do
11 let $\iota_i \in \{1,2,T\}$ s.t. $\mathbf{Y}_i \in \mathbb{G}_{\iota_i}$
12 $y_i \leftarrow \log_{g_{\iota_i}} \mathbf{Y}_i$
13 Return $(P(\vec{y}) \equiv_p 0)$

Figure 6: Algebraic game for the flexible-targets Uber assumption with decisional oracles

whether $P(\log \mathbf{Y}_1,\ldots,\log \mathbf{Y}_n) \equiv_p 0$. Decisional oracles can be added to any type of Uber assumption; for concreteness, we extend the most general variant from the previous section. The game **f-drüber** ("d" for decisional oracles) is defined in Fig. 6. This extension covers assumptions such as Gap Diffie-Hellman (DH) [OP01], where the adversary must solve a DH instance while being given an oracle that checks whether a triple $(\mathbf{Y}_1,\mathbf{Y}_2,\mathbf{Y}_3)$ is a DH tuple, i.e., $\mathbf{Y}_1^{\log \mathbf{Y}_2} = \mathbf{Y}_3$. This oracle is a special case of the one in Fig. 6, when called with $P(X_1,X_2,X_3) := X_1X_2 - X_3$.

Theorem 7.1 (DLog implies flexible-target Uber for rational fractions with decisional oracles in the AGM). *The statement of Theorem 6.1 holds when* **f-rüber** *is replaced by* **f-drüber**.

Proof sketch. The reduction $\mathsf{B}_{\mathsf{alg}}$ from $(\vec{R},\vec{S},\vec{F})$-**f-drüber** to q-DLog (or (q_1,q_2)-DLog) works as for Theorem 6.1 (as detailed in the proof of Theorem 5.2), except that $\mathsf{B}_{\mathsf{alg}}$ must also answer $\mathsf{A}_{\mathsf{alg}}$'s oracle queries, which we describe in the following for Type-3 groups.

As for Theorem 6.1, $\mathsf{B}_{\mathsf{alg}}$, on input $(\vec{\mathbf{Y}},\vec{\mathbf{Z}})$ with $\mathbf{Y}_i = [z^i]_1$ and $\mathbf{Z}_j = [z^j]_2$, for $0 \le i \le q_1$ and $0 \le j \le q_2$, computes LCMs of denominators $D := \mathrm{Den}(\vec{R}\|\vec{S}\|\vec{F})$, $D_1 := \mathrm{Den}(\vec{R}\|\vec{F})$ and $D_2 := \mathrm{Den}(\vec{S})$. It picks $\vec{y} \xleftarrow{\$} (\mathbb{Z}_p^*)^m$ and $\vec{v} \xleftarrow{\$} \mathbb{Z}_p^m$, implicitly sets $x_i := y_i z + v_i \bmod p$ and checks if $D(\vec{x}) \equiv_p 0$. If so, the reduction derives the corresponding univariate polynomial and finds z. Otherwise it computes $h_1 := [D_1(\vec{x})]_1$, $h_2 := [D_2(\vec{x})]_2$ (note that $D_1(\vec{x})$ and $D_2(\vec{x})$ are non-zero), $\mathbf{U}_i = [(D_1 \cdot R_i)(\vec{x})]_1$, $\mathbf{V}_i = [(D_2 \cdot S_i(\vec{x})]_2$ and $\mathbf{W}_i = [(D_1 \cdot D_2 \cdot F_i)(\vec{x})]_T$.

Consider a query $\mathsf{O}(P,(\mathbf{Y}_1,\ldots,\mathbf{Y}_n))$ for some n and $P \in \mathbb{Z}_p[X_1,\ldots,X_n]$, and $\mathbf{Y}_i \in \mathbb{G}_{\iota_i}$ for $\iota_i \in \{1,2,T\}$. Since $\mathsf{A}_{\mathsf{alg}}$ is algebraic, it provides representations of the group elements $\mathbf{Y}_i$ with respect to its input $(\vec{\mathbf{U}},\vec{\mathbf{V}},\vec{\mathbf{W}})$; in particular, for each $\mathbf{Y}_i$, depending on the group, it provides $\vec{\mu}_i$ or $\vec{\eta}_i$ or $(A_i,\vec{\delta}_i)$ such that:

$$(\mathbf{Y}_i \in \mathbb{G}_1)\quad \mathbf{Y}_i = \textstyle\prod_{j=1}^r \mathbf{U}_j^{\mu_{i,j}} = \big[\sum_{j=1}^r \mu_{i,j}(D_1 \cdot R_j)(\vec{x})\big]_1 =: \big[Q_i(z)\big]_1$$

$$(\mathbf{Y}_i \in \mathbb{G}_2)\quad \mathbf{Y}_i = \textstyle\prod_{j=1}^s \mathbf{V}_j^{\eta_{i,j}} = \big[\sum_{j=1}^s \eta_{i,j}(D_2 \cdot S_j)(\vec{x})\big]_2 =: \big[Q_i(z)\big]_2$$

$$(\mathbf{Y}_i \in \mathbb{G}_T)\quad \mathbf{Y}_i = \textstyle\prod_{j=1}^r \prod_{k=1}^s e(\mathbf{U}_j,\mathbf{V}_k)^{\alpha_{i,j,k}} \cdot \prod_{j=1}^f \mathbf{W}_j^{\delta_{i,j}}$$
$$= \big[\textstyle\sum_{j=1}^r \sum_{k=1}^s \alpha_{i,j,k}(D_1 \cdot R_j)(\vec{x})\,(D_2 \cdot S_j)(\vec{x}) + \sum_{j=1}^f \delta_{i,j}(D_1 \cdot D_2 \cdot F_j)(\vec{x})\big]_T =: \big[Q_i(z)\big]_T,$$

where $D_1 \cdot R_j$ as well as $D_2 \cdot S_j$ and $D_1 \cdot D_2 \cdot F_j$ are multivariate polynomials (not rational fractions) and Q_i is the polynomial defined by replacing X_i by $y_i Z + v_i$. Let $D_i'(Z)$ be defined from $D_i(\vec{X})$

analogously. Then we have $\log_{g_{\iota_i}} \mathbf{Y}_i = Q_i(z)$ and furthermore $\log_{h_{\iota_i}} \mathbf{Y}_i = Q_i(z)/D_{\iota_i}(z)$, where $D_T := D_1 \cdot D_2$.

To answer the oracle query, $\mathsf{B}_{\mathsf{alg}}$ must therefore determine whether the function $P(Q_1/D_{i_1}, \ldots, Q_n/D_{i_n})$ vanishes at z. Since $D_1(z), D_2(z) \not\equiv_p 0$, this is the case precisely when $\overline{P} := D_1^d \cdot D_2^d \cdot P(Q_1/D_{i_1}, \ldots, Q_n/D_{i_n})$ vanishes at z, where d is the maximal degree of P. Note that $\overline{P}$ is a polynomial. The reduction distinguishes 3 cases:

1. $\overline{P} \equiv 0$: in this case, the oracle replies 1.
2. $\overline{P} \not\equiv 0$: in this case, $\mathsf{B}_{\mathsf{alg}}$ factorizes $\overline{P}$ to find its roots $z_1, \ldots$, checks whether $\mathbf{Z}_1 = g^{z_i}$ for some i. If this is the case, it stops and returns the solution z_i to its (q_1, q_2)-DLog instance.
3. Else, the oracle replies 0.

Correctness of the simulation is immediate, since the correct oracle reply is 1 if and only if $\overline{P}(z) \equiv_p 0$. ■

8 The Flexible Gegenuber Assumption

In this section, we show how to extend the Uber framework even further, by letting the adversary generate its own generators (for the outputs), yielding the *GeGenUber* assumption. Consider the LRSW assumption [LRSW99] in Type-1 bilinear groups: given $(X = g^x, Y = g^y)$ (which can be viewed as a signature verification key [CL04]) and an oracle, which on input (a message) $m \in \mathbb{Z}_p$ returns (a signature) $(g^a, g^{ay}, g^{a(x+mxy)})$ for a random $a \xleftarrow{\$} \mathbb{Z}_p$, it is infeasible to return (a signature on a fresh message) $((g^{a^*}, g^{a^*y}, g^{a^*(x+m^*xy)}, m^*))$ for any a^* and m^* different from the queried values. Since the adversary need not return the value a^*, this cannot be cast into the Uber framework. Associating the values $a_1, \ldots, a_\ell$ chosen by the signing oracle to formal variables $A_1, \ldots, A_\ell$, in the Uber framework $\vec{X}$ would correspond to $(X, Y, A_1, \ldots, A_\ell)$ and signing queries to the polynomials A_i, A_iY and $A_iX + m_iA_iXY$. Now the adversary can choose a fresh generator $g^* := g^{a^*}$ and must return $((g^*)^{R_i^*(\vec{X})})_{i=1}^3$ for $R_1^* \equiv 1, R_2^* = Y$ and $R_3^* = X + m^*XY$ for some $m^* \in \mathbb{Z}_p^*$ of its choice.

Another example is the simultaneous Diffie-Hellman assumption [PW17], stating that given (g, g^{x_1}, g^{x_2}) it is hard to compute $(Y, Y^{1/x_1}, Y^{1/x_2})$ for some generator Y of the adversary's choice.[3]

Our last generalization now extends the flexible Uber assumption from Sect. 4 by letting the adversary generate its own generators $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{W}$ of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, resp., and return polynomials R^*, S^* and F^*, as well as $(\mathbf{U}^{R^*(\vec{x})}, \mathbf{V}^{S^*(\vec{x})}, \mathbf{W}^{F^*(\vec{x})})$. The game m-**gegenüber** is defined in Fig. 7. This additional freedom for the adversary induces a necessary change in the definition of non-triviality, as illustrated by the following simple (univariate) example: after the challenger chooses $x \xleftarrow{\$} \mathbb{Z}_p$, the adversary makes queries $R_1 := X$ and $R_2 := X^3$ and receives $\mathbf{U}_1$ and $\mathbf{U}_2$. For all Uber assumptions so far, the polynomial $R^* := X^2$ would be considered non-trivial. However, in game **gegenüber** the adversary could return $\mathbf{U} := \mathbf{U}_1 = g^x$ and $\mathbf{U}^* := \mathbf{U}_2 = g^{x^3} = \mathbf{U}^{R^*(x)}$.

Whereas until now the target polynomial R^* was not allowed to be a linear combination of the queried polynomials $P_1, \ldots, P_\ell$ (or of products of such polynomials, depending on the group types), for the Gegenuber assumption we also need to exclude fractions of such linear combinations (such as X^3/X in the example above) to thwart trivial attacks.

[3] As originally stated [PW17], the adversary is given $(g^a, g^{ax_1}, g^{ax_2})$ for a random a. This is equivalent to our statement since we consider a random generator g.

m-**gegenüber**$^{\mathsf{A}_{\mathsf{alg}}}_{\mathcal{G}}$	$\mathsf{O}(i, P(\vec{X}))$
01 $\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_T \leftarrow \emptyset$	06 $\mathcal{Q}_i = \mathcal{Q}_i \cup \{P\}$
02 $(g_1, g_2) \xleftarrow{\$} \mathsf{GenSamp}$; $g_T \leftarrow e(g_1, g_2)$	07 Return $g_i^{P(\vec{x})}$
03 $\vec{x} = (x_1, \ldots, x_m) \xleftarrow{\$} \mathbb{Z}_p^m$	
04 $((\mathbf{U}, \mathbf{V}, \mathbf{W}, \mathbf{U}^*, \mathbf{V}^*, \mathbf{W}^*), (R^*, S^*, F^*)) \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}^{\mathsf{O}(\cdot,\cdot)}()$	
05 Return $\Big(\mathbf{U} \neq 1 \wedge \mathbf{V} \neq 1 \wedge \mathbf{W} \neq 1$	
$\wedge\ (\mathbf{U}^*, \mathbf{V}^*, \mathbf{W}^*) = (\mathbf{U}^{R^*(\vec{x})}, \mathbf{V}^{S^*(\vec{x})}, \mathbf{W}^{F^*(\vec{x})})$	
$\wedge\ (\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_T, R^*, S^*, F^*)$ gegenuber-non-trivial for type of $\mathcal{G}\Big)$	

Figure 7: Algebraic game for the flexible Gegenuber assumption

For a family E of polynomials, we denote by $\mathrm{Span}(E)$ all linear combinations of elements of E, which we extend to fractions as

$$\mathrm{FrSp}(E) := \{\hat{P}/\check{P} \mid (\hat{P}, \check{P}) \in \mathrm{Span}(E) \times (\mathrm{Span}(E) \setminus \{0\})\}.$$

Moreover, by $E_1 * E_2$ we denote the set $\{P_1 \cdot P_2 \mid (P_1, P_2) \in E_1 \times E_2\}$.

Definition 8.1 (Non-triviality for Gegenuber assumption). Let $\mathcal{Q}_1, \mathcal{Q}_2$ and $\mathcal{Q}_T$ be sets of polynomials and let R^*, S^* and F^* be polynomials. We say that $(\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_T, R^*, S^*, F^*)$ is *gegenuber-non-trivial* for groups of type τ, if the following holds:

- **Either** $R^* \notin \mathrm{FrSp}(\mathcal{Q}_1 \cup \mathcal{Q}_2)$ for $\tau \in \{1, 2\}$ and $R^* \notin \mathrm{FrSp}(\mathcal{Q}_1)$ for $\tau = 3$, $(\tau.1)$
- **or** $S^* \notin \mathrm{FrSp}(\mathcal{Q}_1 \cup \mathcal{Q}_2)$ for $\tau = 1$ and $S^* \notin \mathrm{FrSp}(\mathcal{Q}_2)$ for $\tau \in \{2, 3\}$, $(\tau.2)$
- **or** $F^* \notin \mathrm{FrSp}(\mathcal{Q}_T \cup (\mathcal{Q}_1 \cup \mathcal{Q}_2) * (\mathcal{Q}_1 \cup \mathcal{Q}_2))$ for $\tau = 1$ $(1.T)$
  $F^* \notin \mathrm{FrSp}(\mathcal{Q}_T \cup (\mathcal{Q}_1 \cup \mathcal{Q}_2) * \mathcal{Q}_2)$ for $\tau = 2$ $(2.T)$
  $F^* \notin \mathrm{FrSp}(\mathcal{Q}_T \cup (\mathcal{Q}_1 * \mathcal{Q}_2))$ for $\tau = 3$. $(3.T)$

Definition 8.2 (Degree of non-triviality for Gegenuber assumption). Let $(\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_T, R^*, S^*, F^*)$ be a non-trivial tuple of polynomials in $\mathbb{Z}_p[X_1, \ldots, X_m]$. For $i \in \{1, 2, T\}$ define $d'_i := \max\{\deg P\}_{P \in \mathcal{Q}_i}$. We define the *type-$\tau$ degree* d_τ of $(\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_T, R^*, S^*, F^*)$ as follows:

- If $(\tau.1)$ holds, let $d_{\tau.1} := \max\{1, \deg R^*\} \cdot \max\{d'_1, d'_2\}$ in case $\tau \in \{1, 2\}$ and
  $d_{\tau.1} := \max\{1, \deg R^*\} \cdot d'_1$ in case $\tau = 3$.
- If $(\tau.2)$ holds, let $d_{\tau.2} := \max\{1, \deg S^*\} \cdot \max\{d'_1, d'_2\}$ in case $\tau = 1$ and
  $d_{\tau.2} := \max\{1, \deg S^*\} \cdot d'_2$ in case $\tau \in \{2, 3\}$.
- If $(\tau.T)$ holds, let $d_{\tau.T} := \max\{1, \deg F^*\} \cdot \max\{2\,d'_1, 2\,d'_2, d'_T\}$ for $\tau = 1$
  $d_{\tau.T} := \max\{1, \deg F^*\} \cdot \max\{d'_1 + d'_2, 2\,d'_2, d'_T\}$ for $\tau = 2$,
  $d_{\tau.T} := \max\{1, \deg F^*\} \cdot \max\{d'_1 + d'_2, d'_T\}$ for $\tau = 3$.

If (τ, i) does not hold, set $d_{\tau,i} := \infty$. Define $d_\tau := \min\{d_{\tau.1}, d_{\tau.2}, d_{\tau.T}\}$.

Note that for all Uber variants, the adversary only outputs one element per group $\mathbb{G}_i$, which is without loss of generality, as a vector of group elements would be non-trivial if at least one component is non-trivial. We defined the Gegenuber assumption analogously, so one might wonder how this covers LRSW, where the adversary must output group elements corresponding to two polynomials Y and $X + m^*XY$. The reason is that LRSW holds even if the adversary

only has to output the latter polynomial (the former is required to verify the validity of a solution using the pairing): In the GGM this follows from LRSW being an instance of Gegenuber, Theorem 8.3 (see below for both) and Lemmas 2.5 and 2.6.

To show that LRSW is gegenuber-non-trivial, consider the set of queries an adversary can make, namely:

$$\mathcal{Q} := \{1, X, Y, \{A_i, A_iY, A_iX + m_iA_iXY\}_{i=1}^{\ell}\}.$$

To prove that the stronger variant of LRSW satisfies non-triviality as defined in Def. 8.1, we show that for $0 \neq m^* \notin \{m_1, \ldots, m_\ell\}$: $R^* := X + m^*XY \notin \mathrm{FrSp}(\mathcal{Q})$. For the sake of contradiction, assume that for some $\check{P}, \hat{P} \in \mathrm{Span}(\mathcal{Q})$, $\check{P} \not\equiv 0$:

$$(X + m^*XY)\check{P} = \hat{P}. \tag{7}$$

Since $\hat{P} \in \mathrm{Span}(\mathcal{Q})$, its total degree in X and Y is at most 2, which implies that $\check{P}$ must be of degree 0 in X and Y. We can thus write $\check{P}$ as $\eta + \sum_{j=1}^{\ell} \alpha_j A_j$ for some η and $\vec{\alpha}$. Since X is a factor of the left-hand side of (7), $\hat{P}$ cannot have terms without X and must therefore be of the form $\hat{P} = \xi X + \sum_{j=1}^{\ell} \mu_j A_j \, (X + m_j XY)$ for some ξ and $\vec{\mu}$. Equation (7) becomes thus

$$(X + m^*XY)\left(\eta + \textstyle\sum_{j=1}^{\ell} \alpha_j A_j\right) = \xi X + \sum_{j=1}^{\ell} \mu_j A_j \, (X + m_j XY)\,.$$

By equating coefficients, we get: $\eta = \xi$ (from coeff. X) and $m^*\eta \equiv_p 0$ (from XY) and for all $j \in [1, \ell]$: $\alpha_j = \mu_j$ (from A_jX) and $\alpha_j m^* \equiv_p \mu_j m_j$ (from A_jXY). Since $m^* \neq 0$, we have $\eta = \xi = 0$. Furthermore, if $\alpha_j \neq 0$ for some j, then $m_j = m^*$, meaning it was not a valid solution. If $\alpha_j = 0$ for all j then $\check{P} \equiv 0$, which shows such $\hat{P}$ and $\check{P}$ do not exist and thus $X + m^*XY \notin \mathrm{FrSp}(\mathcal{Q})$.

Theorem 8.3 *Let $m \geq 1$, let $\mathcal{G}$ be a bilinear group of type $\tau \in \{1, 2, 3\}$ and let $\mathsf{A}_{\mathsf{alg}}$ be an adversary in game m-**gegenüber**$_{\mathcal{G}}$. Let $d_1', d_2', d_T', d_1^*, d_2^*, d_T^*$ be such that $\mathsf{A}_{\mathsf{alg}}$'s queries $(i, P(\vec{X}))$ satisfy $\deg P \leq d_i'$ and its output satisfies $\deg R^* \leq d_1^*$, $\deg S^* \leq d_2^*$, $\deg F^* \leq d_T^*$. Let q, q_1, q_2 be such that $q \geq \max\{d_1', d_2', d_T'/2\}$, as well as $q_1 \geq d_1'$, $q_2 \geq d_2'$ and $q_1 + q_2 \geq d_T'$. If*

(Type 1) *q-**dlog**$_{\mathcal{G}_1}$ or q-**dlog**$_{\mathcal{G}_2}$ is (ε, t)-secure in the AGM,*

(Type 2) *q-**dlog**$_{\mathcal{G}_2}$ is (ε, t)-secure in the AGM,*

(Type 3) *(q_1, q_2)-**dlog**$_{\mathcal{G}}$ is (ε, t)-secure in the AGM,*

*then **gegenüber**$_{\mathcal{G}}$ is (ε', t')-secure in the AGM with*

$$\varepsilon' \leq \varepsilon + \tfrac{d_\tau}{p-1} \quad \text{and} \quad t' \approx t,$$

with d_τ from Def. 8.2 after replacing $\deg R^*$, $\deg S^*$ and $\deg F^*$ by d_1^*, d_2^* and d_T^*, respectively.

The proof can be found in Appendix D. It is an adaptation of the one for the flexible uberassumption, which adapts the proof of Theorem 3.5. We highlight the main difference for non-triviality of type (2.1). Recall that in the proof Theorem 3.5, the adversary returns a solution $\mathbf{U}'$ and its algebraic representation so that (2a) holds. From this and the fact that $\mathbf{U}' = [R'(\vec{x})]_1$, we derived the polynomial P_1 in (3), which is non-zero by non-triviality.

In the proof of Theorem 8.3, the adversary also outputs a new generator $\mathbf{U}$, whose algebraic representation yields a polynomial $Q \in \mathrm{Span}(\mathcal{Q}_1 \cup \mathcal{Q}_2)$ so that $\mathbf{U} = [Q(\vec{x})]_1$. For a valid solution $\mathbf{U}^*$, we thus have $\mathbf{U}^* = [R^*(\vec{x}) \cdot Q(\vec{x})]_1$. On the other hand, the representation of $\mathbf{U}^*$ yields $\mathbf{U}^* = [Q^*(\vec{x})]_1$ for some $Q^* \in \mathrm{Span}(\mathcal{Q}_1 \cup \mathcal{Q}_2)$. We thus have that $P_1(\vec{X}) := R^*(\vec{X}) \cdot Q(\vec{X}) - Q^*(\vec{X})$ vanishes at $\vec{x}$. By non-triviality, $R^* \notin \mathrm{FrSp}(\mathcal{Q}_1 \cup \mathcal{Q}_2)$, which implies $P_1 \not\equiv 0$, so the reduction can find the roots of $P_1(y_1 Z + x_1, \ldots, y_n Z + x_n)$ and solve q-DLog.

9 Separation of $(q+1)$-DL from q-DL

Now that we have shown that every Uber assumption falls into a (minimal) class of assumptions that are equivalent to q-DLog, we show that these classes can be separated according to their parameter q. We prove that, assuming that q-DLog is hard, there does not exist an algebraic reduction from q-DLog to $(q+1)$-DLog. In particular, we show that if there exists a reduction $\mathsf{R}_{\mathsf{alg}}$ that has access to a $(q+1)$-DLog (algebraic) adversary $\mathsf{A}_{\mathsf{alg}}$ and can solve q-DLog, then there exists a meta-reduction that uses $\mathsf{R}_{\mathsf{alg}}$ to break q-DLog. In the following, we use the notation $\mathsf{R}_{\mathsf{alg}}(\mathsf{A}_{\mathsf{alg}})$ to denote that $\mathsf{R}_{\mathsf{alg}}$ has complete access to $\mathsf{A}_{\mathsf{alg}}$'s internal state. In particular, $\mathsf{R}_{\mathsf{alg}}$ is allowed to rewind $\mathsf{A}_{\mathsf{alg}}$ to any point of an execution and run $\mathsf{A}_{\mathsf{alg}}$ on any choice of random coins as many times as it wants.

Theorem 9.1 *Let $\mathcal{G}_i$ be a group of prime order p. There exists an algorithm M such that the following holds. Let $\mathsf{R}_{\mathsf{alg}}$ be an algebraic algorithm s.t. for every algorithm $\mathsf{A}_{\mathsf{alg}}$ that (t,ϵ)-breaks $(q+1)$-**dlog**$_{\mathcal{G}_i}$, $\mathsf{B} = \mathsf{R}_{\mathsf{alg}}(\mathsf{A}_{\mathsf{alg}})$ is an algorithm that (t',ϵ')-breaks q-**dlog**$_{\mathcal{G}_i}$. If $t \geq 2\,(2q+1)\lceil \log_2 p \rceil$ then $\mathsf{M}^{\mathsf{R}_{\mathsf{alg}}}$ (t',ϵ')-breaks q-**dlog**$_{\mathcal{G}_i}$.*

We start with a proof overview. Consider a reduction $\mathsf{R}_{\mathsf{alg}}$, which on input a q-DLog instance $(g, g^x, \ldots, g^{x^q})$ can run an algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ multiple times on $(q+1)$-DLog instances $(\mathbf{Z}, \mathbf{Z}^y, \ldots, \mathbf{Z}^{y^{q+1}})$; that is, $\mathsf{R}_{\mathsf{alg}}$ can choose a new generator $\mathbf{Z}$ and a new problem solution y. Since $\mathsf{R}_{\mathsf{alg}}$ is algebraic, it outputs a representation of the group elements composing its $(q+1)$-DLog instance in terms of the received q-DLog instance. We distinguish two cases: if (a) y is independent from x then the representation reveals y, which means that a meta-reduction M can simulate a successful $\mathsf{A}_{\mathsf{alg}}$ to $\mathsf{R}_{\mathsf{alg}}$, and the latter must thus find x. On the other hand, if (b) y depends on x, then this yields a non-trivial equation in x, which the meta-reduction can solve and thereby (without needing to simulate $\mathsf{A}_{\mathsf{alg}}$) solve the q-DLog instance.

To simplify the probability analysis, we let M simulate $\mathsf{A}_{\mathsf{alg}}$ even when $\mathsf{R}_{\mathsf{alg}}$ behaves as in the second case (as it can compute y from x). To correctly argue about the probability distributions, we ensure that any malformed instance provided by the algebraic reduction $\mathsf{R}_{\mathsf{alg}}$ is detected. We will use the following lemma in the proof of Theorem 9.1:

Lemma 9.2 *Let $q \geq 1$, let $F \in \mathbb{Z}_p(X)$ and let $0 \not\equiv P \in \mathbb{Z}_p[X]$ be of degree at most q. If $F^{q+1} \cdot P$ is a polynomial and of degree at most q, then F is constant.*

Proof. Let $\hat{F}, \check{F} \in \mathbb{Z}_p[X]$ be coprime such that $F = \hat{F}/\check{F}$. Then $\hat{F}^{q+1}$ and $\check{F}^{q+1}$ are coprime as well. From this and the premise that $\hat{F}^{q+1} \cdot P/\check{F}^{q+1}$ is a polynomial, we get that $\check{F}^{q+1}$ divides P, and thus $(q+1) \cdot \deg \check{F} \leq \deg P$. Since the latter is at most q, we have $\deg \check{F} = 0$.

Furthermore, we assumed that $q \geq \deg(F^{q+1} \cdot P) = (q+1) \cdot \deg \hat{F} + \deg P$, and thus $\deg \hat{F} = 0$. Together, this means F is constant. ∎

Proof of Theorem 9.1. Let $\mathsf{R}_{\mathsf{alg}}$ be an algebraic algorithm s.t. for every algorithm $\mathsf{A}_{\mathsf{alg}}$ that (t,ϵ)-breaks $(q+1)$-**dlog**$_{\mathcal{G}_i}$, $\mathsf{B} = \mathsf{R}_{\mathsf{alg}}(\mathsf{A}_{\mathsf{alg}})$ is an algorithm that (t',ϵ')-breaks q-**dlog**$_{\mathcal{G}_i}$. In the following, we describe a meta-reduction M s.t. $\mathsf{M}^{\mathsf{R}_{\mathsf{alg}}}$ (t',ϵ')-breaks q-**dlog**$_{\mathcal{G}_i}$.

$\mathsf{M}(g, \mathbf{X}_1, \ldots, \mathbf{X}_q)$**:** Run $\mathsf{R}_{\mathsf{alg}}$ on the received q-DLog instance $(g, g^x, \ldots, g^{x^q})$. Whenever $\mathsf{R}_{\mathsf{alg}}$ runs adversary $\mathsf{A}_{\mathsf{alg}}$ on $(q+1)$-DLog input $(\mathbf{Z}_0, \mathbf{Z}_1, \ldots, \mathbf{Z}_{q+1})$, do the following. Let $\vec{z}_i = (z_{i,0}, \ldots, z_{i,q})$ for $0 \leq i \leq q+1$ be the representation vectors for $\mathbf{Z}_0, \ldots, \mathbf{Z}_{q+1}$ provided by $\mathsf{R}_{\mathsf{alg}}$; that is, $\mathbf{Z}_i = \prod_{j=0}^{q} (g^{x^j})^{z_{i,j}}$.

If $\mathbf{Z}_0 = 1$ then return $\perp$. $(*)$

Else define

$$P_i(X) := \textstyle\sum_{j=0}^{q} z_{i,j} X^j \quad \text{for } 0 \leq i \leq q+1 \quad \text{and} \tag{8}$$

$$Q_i := P_{i+1}P_0 - P_iP_1 \quad \text{for } 0 \le i \le q. \tag{9}$$

M now distinguishes two cases:

(a) $Q_i \equiv 0$ for all $i \in [0,q]$: Then (as we argue below) $P_1/P_0 \equiv c$, that is, a constant polynomial. M returns c as $\mathsf{A}_{\mathsf{alg}}$'s output.

(b) For some $k \in [0,q]$: $Q_k \not\equiv 0$: Compute the roots $x_1, \ldots$ of Q_k and check if for some j: $g^{x_j} = \mathbf{X}_1$. If not, then return $\perp$ as $\mathsf{A}_{\mathsf{alg}}$'s output. $(**)$
Else let $y := P_1(x_j)/P_0(x_j) \bmod p$. $(1 \neq \mathbf{Z}_0 = g^{P_0(x_j)}$, thus $P_0(x_j) \not\equiv_p 0.)$ If for some $i \in [1, q+1] : \mathbf{Z}_i \neq \mathbf{Z}_0^{y^i}$, return $\perp$ as $\mathsf{A}_{\mathsf{alg}}$'s output. $(***)$
Else return y.

Correctness of simulation. We now argue that M always correctly simulates an adversary $\mathsf{A}_{\mathsf{alg}}$ that solves $(q+1)$-DLog if it received a correct instance, and returns $\perp$ otherwise. Consider the case where $\mathsf{R}_{\mathsf{alg}}$ provides a valid $(q+1)$-DLog instance, that is $\mathbf{Z}_0 \neq 1$ and

$$\exists\, y \in \mathbb{Z}_p^* \; \forall\, i \in [1, q+1] : \mathbf{Z}_i = \mathbf{Z}_0^{y^i}. \tag{10}$$

By (8), we have $\mathbf{Z}_i = g^{P_i(x)}$ for all i. Since $\mathbf{Z}_0 \neq 1$, M does not stop in line $(*)$ and $P_0(x) \not\equiv_p 0$. Moreover, from (10) we have $y \equiv_p P_1(x)/P_0(x)$ and

$$P_{i+1}(x) \equiv_p P_1(x)/P_0(x) \cdot P_i(x), \tag{11}$$

in other words, $Q_i(x) \equiv_p 0$ for all $i \in [0,q]$.

In case (a), $Q_i \equiv 0$, and thus, letting $F := P_1/P_0$, we have (by definition (9)) $P_{i+1} = F \cdot P_i$ for all i, and by induction:

$$\forall\, i \in [0, q+1] : P_i = F^i \cdot P_0,$$

and in particular $P_{q+1} = F^{q+1} \cdot P_0$. Since P_{q+1} and P_0 are polynomials of degree at most q, by Lemma 9.2 we get $F \equiv c$ for $c \in \mathbb{Z}_p$. The meta-reduction M thus returns $c \equiv_p F(x) \equiv_p P_1(x)/P_0(x) \equiv_p y$.

In case (b), since $Q_k \not\equiv 0$ but, by (11), $Q_k(x) \equiv_p 0$, the meta-reduction finds x and M does not stop in line $(**)$ and neither in line $(***)$, since $y \equiv_p P_1(x)/P_0(x)$.

Now consider the case that $\mathsf{R}_{\mathsf{alg}}$ sends a malformed instance: if $\mathbf{Z}_0$ is not a generator then $\mathsf{A}_{\mathsf{alg}}$ returns $\perp$ in line $(*)$. Assume $\mathbf{Z}_0$ is a generator (and thus $P_0(x) \not\equiv_p 0$), but (10) is not satisfied. Using the algebraic representations of $\mathbf{Z}_0, \ldots, \mathbf{Z}_{q+1}$, this is equivalent to

$$\forall\, y \in \mathbb{Z}_p^* \; \exists\, k \in [1, q+1] : P_k(x) \not\equiv_p y^k P_0(x). \tag{12}$$

We first show that the meta-reduction M goes to case (b): Indeed, if $Q_i \equiv 0$ for all $i \in [0,q]$, then $P_{i+1}(x) \equiv_p P_1(x)/P_0(x) \cdot P_i(x)$ for all i and, by induction, $P_i(x) \equiv_p (P_1(x)/P_0(x))^i \cdot P_0(x)$, which, setting $y := P_1(x)/P_0(x) \bmod p$, contradicts (12).

Let $k \in [0,q]$ be such that $Q_k \not\equiv 0$. If x is not among the roots of Q_k, then M returns $\perp$ in line $(**)$. Otherwise, it sets $y := P_1(x)/P_0(x) \bmod p$. If for some $i \in [1, q+1]$: $P_i(x) \not\equiv_p y^i P_0(x)$ then M returns $\perp$ in line $(***)$. If not then this contradicts (12). Therefore, the simulation will return $\perp$ on invalid inputs.

For the simulation of $\mathsf{A}_{\mathsf{alg}}$ the meta-reduction needs to compute at most $2q+1$ exponentiations, each of which require at most $2\lceil \log_2 p \rceil$ group operations using square and multiply. The simulation of $\mathsf{A}_{\mathsf{alg}}$ is thus perfect and takes at most t steps. The meta-reduction M succeeds in winning q-$\mathbf{dlog}_{\mathcal{G}_i}$ whenever $\mathsf{B} = \mathsf{R}_{\mathsf{alg}}(\mathsf{A}_{\mathsf{alg}})$ wins q-$\mathbf{dlog}_{\mathcal{G}_i}$. Therefore, we obtain

$$\Pr\left[q\text{-}\mathbf{dlog}_{\mathcal{G}_i}^{\mathsf{M}} = 1\right] = \Pr\left[q\text{-}\mathbf{dlog}_{\mathcal{G}_i}^{\mathsf{B}} = 1\right] \ge \varepsilon'.$$

Moreover, the running time of M is that of B, i.e., t'. This completes the proof. ∎

A similar result can be shown for (q_1, q_2)-$\mathbf{dlog}_{\mathcal{G}}$, that is, (q_1, q_2)-$\mathbf{dlog}_{\mathcal{G}}$ is not implied by (q_1', q_2')-$\mathbf{dlog}_{\mathcal{G}}$ if $q_1' > q_1$ or $q_2' > q_2$.

10 Separation of 2-One-More DL from q-DL

We conclude with showing that "one-more"-discrete logarithm (OMDL) assumptions fall outside of our q-DLog taxonomy. While it is known that there is no black-box reduction from DLog to OMDL [BMV08], we show that there is no algebraic reduction either, even one for algebraic adversaries.

To obtain the strongest possible impossibility result, we show that for no $q \in \mathbb{N}$, there exists an algebraic reduction from q-DLog (a stronger assumption than DLog) to 2-OMDL (the weakest variant of OMDL assumptions), unless q-DLog is easy. The game for 2-OMDL is depicted in Fig. 8. The proof uses the same high-level idea as for Theorem 10.1. If the representation of the group elements the reduction gives to the adversary is independent of its own q-DLog challenge, then the meta-reduction can directly simulate the adversary. Else, they depend on the q-DLog challenge in a way that allows the meta-reduction to derive a q-DLog solution. Compared to the previous section, we now restrict the algebraic reduction to only have black-box access (according to our notion of black-box) to the adversary. This is because a reduction that can choose the random coins of the adversary in a non-uniform (adaptive) way can make the simulation of the adversary by our meta-reduction fail.

We need to define the adversary's behavior, that is, its oracle call, beforehand; in particular, it must not depend on the type of representations obtained from the reduction, which makes the proof more complicated and restricts the simulation to adversaries that can fail with negligible probability. The adversary, after receiving a 2-OMDL challenge $(\mathbf{Y}_0, \mathbf{Y}_1, \mathbf{Y}_2)$, makes a query $(\mathbf{Y}_1^{r_1}\mathbf{Y}_2^{r_2})$ for random r_1, r_2 to its DLog oracle and then returns the 2-OMDL solution $(\log_{\mathbf{Y}_0}\mathbf{Y}_1, \log_{\mathbf{Y}_0}\mathbf{Y}_2)$. We now show how the meta-reduction simulates this adversary.

Since the reduction is algebraic, it provides with its 2-OMDL instance representations $\vec{z}_i = (z_{i,0}, \ldots, z_{i,q})$ in terms of its q-DLog challenge $(g, g^x, \ldots, g^{x^q})$, such that $\log_g \mathbf{Y}_i \equiv_p \sum_{j=0}^{q} z_{i,j}x^j$. From the reply y to the adversary's single oracle query, we get the following equation: $0 = \log_g(\mathbf{Y}_1^{r_1}\mathbf{Y}_2^{r_2}) - \log_g \mathbf{Y}_0^{y} \equiv_p \sum_{j=0}^{q} (r_1 z_{1,j} + r_2 z_{2,j} - y z_{0,j})x^j$.

The q-DLog challenge x is thus the root of the polynomial with coefficients $a_j := (r_1 z_{1,j} + r_2 z_{2,j} - y z_{0,j}) \bmod p$, and the meta-reduction can find x if one of these coefficients is non-zero. Using x, it can then compute $\log_g \mathbf{Y}_i$ for all i from the representations and from that the OMDL solution $(\log_{\mathbf{Y}_0}\mathbf{Y}_1, \log_{\mathbf{Y}_0}\mathbf{Y}_2)$.

If, on the other hand, $a_j = 0$ for some j then by plugging in the definition of y, we get another polynomial which vanishes at x. We then show that, due to the randomizers r_1 and r_2, with overwhelming probability, the coefficients of this polynomial are non-zero – unless for some c_1, c_2 we have $\vec{z}_1 = c_1\vec{z}_0$ and $\vec{z}_2 = c_2\vec{z}_0$. But in this case (c_1, c_2) is the solution to the 2-OMDL instance and the meta-reduction can therefore finish the simulation of the adversary.

Theorem 10.1 *Let $\mathcal{G}_i$ be a group of prime order p. There exists an algorithm M such that the following holds: Let $\mathsf{R}_{\mathsf{alg}}$ be an algebraic reduction s.t. for every algorithm $\mathsf{A}_{\mathsf{alg}}$ that (t, ϵ)-breaks 2-$\mathbf{omdl}_{\mathcal{G}_i}$, $\mathsf{B} = \mathsf{R}_{\mathsf{alg}}^{\mathsf{A}_{\mathsf{alg}}}$ is an algorithm that (t', ϵ')-breaks q-$\mathbf{dlog}_{\mathcal{G}_i}$. If $t \geq (6 + 2q)\lceil \log_2 p \rceil + 1$ and $\epsilon \leq 1 - 1/p$ then $\mathsf{M}^{\mathsf{R}_{\mathsf{alg}}}$ (t', ϵ')-breaks q-$\mathbf{dlog}_{\mathcal{G}_i}$.*

Proof. Let $\mathsf{R}_{\mathsf{alg}}$ be an algebraic reduction s.t. for every algorithm $\mathsf{A}_{\mathsf{alg}}$ that (t, ϵ)-breaks 2-$\mathbf{omdl}_{\mathcal{G}_i}$, $\mathsf{B} = \mathsf{R}_{\mathsf{alg}}^{\mathsf{A}_{\mathsf{alg}}}$ is an algorithm that (t', ϵ')-breaks q-$\mathbf{dlog}_{\mathcal{G}_i}$.

We first specify a hypothetical algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ which ($t = (6+2q)\cdot\lceil \log_2 p \rceil + 1$, $\epsilon = 1 - 1/p$)-breaks 2-OMDL: on input $(\mathbf{Y}_0, \mathbf{Y}_1, \mathbf{Y}_2)$, if $\mathbf{Y}_0$ is not a generator, it returns $\perp$. Else, it chooses

$\underline{\text{2-omdl}}^{\mathsf{A}_{\mathsf{alg}}}_{\mathcal{G}_i}$	$\underline{\mathsf{O}}(\mathbf{Z})$
00 $Q \leftarrow 0$	05 if $Q = 0$ then
01 $g \xleftarrow{\$} \mathsf{GenSamp}_i$	06 $\quad Q \leftarrow 1$
02 $y_1, y_2 \xleftarrow{\$} \mathbb{Z}_p^*$	07 $\quad$ Return $\log_g \mathbf{Z}$
03 $(y_1^*, y_2^*) \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}^{\mathsf{O}(\cdot)}(g, g^{y_1}, g^{y_2})$	08 Return $\perp$
04 Return $((y_1^*, y_2^*) = (y_1, y_2))$	

Figure 8: Game for 2-one-more discrete logarithm 2-$\mathbf{omdl}_{\mathcal{G}_i}$ relative to bilinear group $\mathcal{G}_i, i \in \{1, 2\}$ and adversary $\mathsf{A}_{\mathsf{alg}}$

two uniform values $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$ and queries its oracle $\mathsf{O}(\cdot)$ on $\mathbf{Y}_1^{r_1}\mathbf{Y}_2^{r_2}$, providing representation $(0, r_1, r_2)$ in basis $(\mathbf{Y}_0, \mathbf{Y}_1, \mathbf{Y}_2)$. If it does not obtain the correct answer $\log_{\mathbf{Y}_0}\mathbf{Y}_1^{r_1}\mathbf{Y}_2^{r_2}$, it returns $\perp$. Else it returns the 2-OMDL solution $\log_{\mathbf{Y}_0}\mathbf{Y}_1$ and $\log_{\mathbf{Y}_0}\mathbf{Y}_2$ with probability $1 - 1/p$.

We now construct a meta-reduction M, s.t. $\mathsf{M}^{\mathsf{R}_{\mathsf{alg}}}$ (t', ϵ')-breaks q-$\mathbf{dlog}_{\mathcal{G}_i}$, as follows. On input a q-$\mathbf{dlog}_{\mathcal{G}_i}$ instance $(g, g^x, \ldots, g^{x^q})$, the meta-reduction runs $\mathsf{R}_{\mathsf{alg}}$ on $(g, g^x, \ldots, g^{x^q})$. Every time $\mathsf{R}_{\mathsf{alg}}$ invokes 2-OMDL adversary $\mathsf{A}_{\mathsf{alg}}$ on a challenge $(\mathbf{Y}_0, \mathbf{Y}_1, \mathbf{Y}_2)$, M simulates $\mathsf{A}_{\mathsf{alg}}$ as follows.

If $\mathbf{Y}_0$ is not a generator, it returns $\perp$. Since $\mathsf{R}_{\mathsf{alg}}$ is algebraic, along with $(\mathbf{Y}_0, \mathbf{Y}_1, \mathbf{Y}_2)$, it provides corresponding representation vectors $\vec{z}_i = (z_{i,0}, \ldots, z_{i,q}) \in \mathbb{Z}_p^{q+1}$ for $i \in [0, 2]$ such that

$$\mathbf{Y}_i = \textstyle\prod_{j=0}^{q}(g^{x^j})^{z_{i,j}} = g^{\sum_{j=0}^{q} z_{i,j}x^j} \quad \text{for } i \in [0, 2] \ . \tag{13}$$

Let k be such that $z_{0,k} \neq 0$ (which exists, since $\mathbf{Y}_0 \neq 1$). The meta-reduction chooses $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$ and queries $\mathsf{O}(\mathbf{Y}_1^{r_1}\mathbf{Y}_2^{r_2})$ to obtain y. If $\mathbf{Y}_0^y \neq \mathbf{Y}_1^{r_1}\mathbf{Y}_2^{r_2}$ then M returns $\perp$. Otherwise, let

$$\begin{aligned} d_{i,j} &:= z_{i,k}z_{0,j} - z_{i,j}z_{0,k} \bmod p \\ a_j &:= yz_{0,j} - r_1z_{1,j} - r_2z_{2,j} \bmod p \quad \text{for } 1 \leq i \leq 2 \text{ and } 1 \leq j \leq q \ . \end{aligned}$$

M distinguishes three cases and simulates $\mathsf{A}_{\mathsf{alg}}$ accordingly:

(a) $d_{i,j} = 0$ for all i, j: in this case, M computes (modulo p) $z_{1,k}z_{0,k}^{-1}$ and $z_{2,k}z_{0,k}^{-1}$, which $\mathsf{A}_{\mathsf{alg}}$ returns as the OMDL solution (we argue correctness below).

(b) $a_j \neq 0$ for some j: in this case, M factors the polynomial with coefficients $a_0, \ldots, a_q$. If x is among its roots then M computes the following modulo p and lets $\mathsf{A}_{\mathsf{alg}}$ return it:

$$\left(\textstyle\sum_{j=0}^{q} z_{1,j}x^j\right)\left(\sum_{j=0}^{q} z_{0,j}x^j\right)^{-1} \quad \text{and} \quad \left(\textstyle\sum_{j=0}^{q} z_{2,j}x^j\right)\left(\sum_{j=0}^{q} z_{0,j}x^j\right)^{-1} \tag{14}$$

(since $((\log_g \mathbf{Y}_1)(\log_g \mathbf{Y}_0)^{-1}, (\log_g \mathbf{Y}_2)(\log_g \mathbf{Y}_0)^{-1}) \equiv_p (\log_{\mathbf{Y}_0}\mathbf{Y}_1, \log_{\mathbf{Y}_0}\mathbf{Y}_2)$, by (13), this is thus the OMDL solution).

(c) For the remaining case, let i^*, j^* be such that $d_{i^*,j^*} \neq 0$. In this case, M factors the polynomial with coefficients $d_{i^*,0}, \ldots, d_{i^*,q}$. If x is among its roots then $\mathsf{A}_{\mathsf{alg}}$ returns the values from (14). Otherwise M returns $\perp$.

When $\mathsf{R}_{\mathsf{alg}}$ stops and returns x then M also stops and returns x.

We now show that the meta-reduction M correctly simulates every run of $\mathsf{A}_{\mathsf{alg}}$ that $\mathsf{R}_{\mathsf{alg}}$ invokes, in particular, that $\mathsf{A}_{\mathsf{alg}}$ solves the OMDL instance with probability at least $1 - 1/p$. First note that $\mathbf{Y}_0^y = \mathbf{Y}_1^{r_1}\mathbf{Y}_2^{r_2}$, which together with (13) yields:

$$0 \equiv_p \textstyle\sum_{j=0}^{q}(yz_{0,j} - r_1z_{1,j} - r_2z_{2,j})x^j \equiv_p \sum_{j=0}^{q} a_jx^j =: P_{\vec{a}}(x) \ . \tag{15}$$

(where $P_{\vec{a}}$ is the polynomial with coefficients $(a_0, \ldots, a_q)$).
In Case (a), we have that $(z_{1,k}z_{0,k}^{-1}, z_{2,k}z_{0,k}^{-1})$ is the OMDL solution since for $i = 1, 2$:

$$\mathbf{Y}_0^{z_{i,k}z_{0,k}^{-1}} \overset{(13)}{=} g^{\sum_{j=0}^{q} z_{i,k}z_{0,k}^{-1}z_{0,j}x^j} \overset{d_{i,j}=0}{=} g^{\sum_{j=0}^{q} z_{i,j}x^j} \overset{(13)}{=} \mathbf{Y}_i \ .$$

In Case (b), the polynomial $P_{\vec{a}}$ defined in (15) is non-zero. Since by (15), x is one of its roots, M can find it and use it to compute $\log_{\mathbf{Y}_0} \mathbf{Y}_i$ via (14).
In Case (c), we have $a_k = 0$ and thus (by definition of a_k):

$$yz_{0,k} \equiv_p r_1 z_{1,k} + r_2 z_{2,k} \ . \tag{16}$$

Multiplying (15) by $z_{0,k}$ we get

$$\textstyle\sum_{j=0}^{q}(yz_{0,k}z_{0,j} - r_1 z_{1,j}z_{0,k} - r_2 z_{2,j}z_{0,k})x^j \equiv_p 0 \ ,$$

which, when substituting $yz_{0,k}$ by the right-hand side of (16), yields

$$r_1 \textstyle\sum_{j=0}^{q}(z_{1,k}z_{0,j} - z_{1,j}z_{0,k})x^j + r_2 \sum_{j=0}^{q}(z_{2,k}z_{0,j} - z_{2,j}z_{0,k})x^j \equiv_p 0 \ .$$

Using $d_{i,j} \equiv_p z_{i,k}z_{0,j} - z_{i,j}z_{0,k}$, this can be written as

$$r_1 \textstyle\sum_{j=0}^{q} d_{1,j}x^j + r_2 \sum_{j=0}^{q} d_{2,j}x^j \equiv_p 0 \ . \tag{17}$$

Recall that in Case (c), for some i^*, j^*: $d_{i^*,j^*} \neq 0$. If $D := \sum_{j=0}^{q} d_{i^*,j}x^j \equiv_p 0$, then M finds x by factoring $P_{\vec{d}_{i^*}}$ and thus finishes the simulation. In the remaining case we have $D \not\equiv_p 0$; moreover, the values x and $\vec{d}_{i^*}$ are independent of r_{i^*}, which was chosen after the reduction (implicitly) defined $\vec{d}_{i^*}$. Equation (17), that is, $r_{i^*}D \equiv_p -r_{3-i^*} \sum_{j=0}^{q} d_{3-i^*,j}x^j$, thus only holds with probability $1/p$. The meta-reduction therefore makes $\mathsf{A}_{\mathsf{alg}}$ return $\perp$ on correct inputs with probability at most $1/p$.

Finally, we show that the simulation of $\mathsf{A}_{\mathsf{alg}}$ takes at most t steps. Computing $\mathbf{Y}_0^y$, $\mathbf{Y}_1^{r_1}$, and $\mathbf{Y}_2^{r_2}$ via square and multiply requires $3 \cdot 2\lceil \log_2(p) \rceil$ group operations, and computing $\mathbf{Y}_1^{r_1} \cdot \mathbf{Y}_2^{r_2}$ takes one. Checking if the roots of the polynomial $\sum_{i=0}^{q} a_i X^i$ (in Case (b)) or $\sum_{j=0}^{q} d_{i^*,j}X^j$ (in Case (c)) are equal to x requires $q \cdot 2\lceil \log_2(p) \rceil$ group operations.

M succeeds in winning q-$\mathbf{dlog}_{\mathcal{G}_i}$ whenever $\mathsf{B}_{\mathsf{alg}} = \mathsf{R}_{\mathsf{alg}}^{\mathsf{A}_{\mathsf{alg}}}$ wins q-$\mathbf{dlog}_{\mathcal{G}_i}$. Therefore, we obtain

$$\Pr\left[q\text{-}\mathbf{dlog}_{\mathcal{G}_i}^{\mathsf{M}} = 1\right] = \Pr\left[q\text{-}\mathbf{dlog}_{\mathcal{G}_i}^{\mathsf{B}} = 1\right] \geq \varepsilon' \ .$$

Moreover, the running time of M is that of B, i.e., t'. This completes the proof. ■

Acknowledgements. This work is funded in part by the MSR–Inria Joint Centre. The second author is supported by the Vienna Science and Technology Fund (WWTF) through project VRG18-002. Parts of this work were done while he was visiting the Simons Institute for the Theory of Computing.

References

[ABM15] Michel Abdalla, Fabrice Benhamouda, and Philip MacKenzie. Security of the J-PAKE password-authenticated key exchange protocol. In *2015 IEEE Symposium on Security and Privacy*, pages 571–587. IEEE Computer Society Press, 2015.

[AFG+10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, Heidelberg, August 2010.

[AHK20] Thomas Agrikola, Dennis Hofheinz, and Julia Kastner. On instantiating the algebraic group model from falsifiable assumptions. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 96–126. Springer, 2020.

[AW98] Claude Deschamps André Warusfel, François Moulin. *Mathématiques 1ère année : Cours et exercices corrigés*. Editions Dunod, 1998.

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, 2008.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, 2005.

[BMV08] Emmanuel Bresson, Jean Monnerat, and Damien Vergnaud. Separation results on the "one-more" computational problems. In Tal Malkin, editor, *CT-RSA 2008*, volume 4964 of *LNCS*, pages 71–87. Springer, 2008.

[BMZ19] James Bartusek, Fermi Ma, and Mark Zhandry. The distinction between fixed and random generators in group-based assumptions. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 801–830. Springer, 2019.

[Boy08] Xavier Boyen. The uber-assumption family (invited talk). In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 39–56. Springer, 2008.

[BR04] Mihir Bellare and Phillip Rogaway. Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Report 2004/331, 2004. http://eprint.iacr.org/2004/331.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, 2004.

[CM14] Melissa Chase and Sarah Meiklejohn. Déjà Q: Using dual systems to revisit q-type assumptions. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 622–639. Springer, 2014.

[CMM16] Melissa Chase, Mary Maller, and Sarah Meiklejohn. Déjà Q all over again: Tighter and broader reductions of q-type assumptions. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 655–681. Springer, 2016.

[DL77] Richard A DeMillo and Richard J Lipton. A probabilistic remark on algebraic program testing. Technical report, Georgia Inst of Tech Atlanta School of Information and Computer Science, 1977.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, 2018.

[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, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 63–95. Springer, 2020.

[GG17] Essam Ghadafi and Jens Groth. Towards a classification of non-interactive computational assumptions in cyclic groups. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 66–96. Springer, 2017.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, 2016.

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

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, 2012.

[Lip19] Helger Lipmaa. Simulation-extractable SNARKs revisited. *ePrint Cryptogoly Archive, Report 2019/612*, 2019.

[Los19] Julian Loss. *New techniques for the modular analysis of digital signature schemes*. PhD thesis, Ruhr University Bochum, Germany, 2019.

[LRSW99] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard M. Heys and Carlisle M. Adams, editors, *SAC 1999*, volume 1758 of *LNCS*, pages 184–199. Springer, 1999.

[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 *LNCS*, pages 1–12. Springer, 2005.

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

[MTT19] Taiga Mizuide, Atsushi Takayasu, and Tsuyoshi Takagi. Tight reductions for Diffie-Hellman variants in the algebraic group model. In Mitsuru Matsui, editor, *CT-RSA 2019*, volume 11405 of *LNCS*, pages 169–188. Springer, 2019.

[Nec94] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994.

[OP01] Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 104–118. Springer, 2001.

[PV05] Pascal Paillier and Damien Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In Bimal K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 1–20. Springer, 2005.

[PW17] David Pointcheval and Guilin Wang. VTBPEKE: Verifier-based two-basis password exponential key exchange. In Ramesh Karri, Ozgur Sinanoglu, Ahmad-Reza Sadeghi, and Xun Yi, editors, *ASIACCS 17*, pages 301–312. ACM Press, 2017.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, 1997.

A Proof of Lemma 2.6

We prove the statement for (q_1, q_2)-DLog; the proof for q-DLog follows analogously. We give a proof in Maurer's version of the GGM [Mau05], in which an adversary $\mathsf{A}_{\mathsf{gen}}$ can access elements

from the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ only via abstract handles. These are maintained by the challenger in lists L_1, L_2, and L_T, which correspond to the groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, respectively.

For (q_1, q_2)-DLog the lists L_1 and L_2 initially contain the handles to the elements $1, Z, Z^2, \ldots, Z^{q_1}$ and $1, Z, Z^2, \ldots, Z^{q_2} \in \mathbb{Z}_p[Z]$, respectively, that correspond to the (q_1, q_2)-DLog challenge given to $\mathsf{A}_{\mathsf{gen}}$. The challenger also samples $z \xleftarrow{\$} \mathbb{Z}_p^*$, the solution, and we will argue that this value remains information-theoretically hidden from $\mathsf{A}_{\mathsf{gen}}$.

The adversary is granted access to oracles of two types. Oracles $\mathsf{O}_i(\cdot)$, for $i \in \{1, 2, T\}$, take as input two handles $h_1, h_2 \in L_i$ for polynomials $P_1(Z), P_2(Z) \in \mathbb{Z}_p[Z]$, respectively, and output a handle h for the polynomial $P_1(Z) + P_2(Z)$; L_i is accordingly updated with the handle h. Oracle $\mathsf{O}_e(\cdot)$, on input two handles $h_1 \in L_1$ and $h_2 \in L_2$ for $P_1(Z)$ and $P_2(Z)$, outputs the handle h_T to the element $P_1(Z) \cdot P_2(Z)$ and updates L_T accordingly.

Following the standard argument in the generic group model, we consider a so-called *collision event* $\mathcal{E}$ which occurs if there exist two distinct handles $h_1, h_2 \in L_1 \cup L_2 \cup L_T$ that point to polynomials $P_1(Z)$ and $P_2(Z)$, respectively, such that $P_1(Z) \neq P_2(Z)$, yet $P_1(z) = P_2(z)$.

One can show that non-adaptively generating group elements until a collision occurs is the best possible strategy for a generic algorithm. Thus, to lower-bound the number of oracle interactions that are needed until $\mathsf{A}_{\mathsf{gen}}$ finds z, it suffices to lower-bound the time until $\mathcal{E}$ happens.

Before $\mathcal{E}$ occurs, z is a uniformly random value and thus the probability that two computed elements are equal after t steps of computation (i.e., oracle calls) can be upper-bounded by the Schwartz-Zippel Lemma (Lemma 2.2). In particular, let $P_1(Z)$ and $P_2(Z)$ be distinct polynomials of degree at most $2q$. Then Schwartz-Zippel upper-bounds the probability that $P_1(z) \equiv_p P_2(z)$ for a uniform element $z \in \mathbb{Z}_p^*$ by $2q/(p-1)$. As initially the set $L_1 \cup L_2 \cup L_T$ is of size $q+1$ (where $q := \max\{q_1, q_2\}$) and an oracle call by $\mathsf{A}_{\mathsf{gen}}$ adds at most one polynomial of degree at most $q_1 + q_2 < 2q$ to one of the lists, there are at most $\binom{t+q+1}{2}$ such equations after t steps of computation. Thus, the probability of a collision occurring is at most $(t+q+1)^2 \cdot 2q/(2(p-1)) = (t+q+1)^2 \cdot q/(p-1)$.

B Running Time of the Generic Reduction of Theorem 3.5

Fact B.1 Let $\mathbb{G}$ a group of order p. The square-and-multiply algorithm in $\mathbb{G}$ takes as input a group element $\mathbf{X} \in \mathbb{G}$ and a scalar $a \in \mathbb{Z}_p$ and returns $\mathbf{X}^a$ after computing at most $2\lceil \log(p) \rceil$ group operations.

Analysis. We give an upper bound on the running time of computation, in terms of bilinear group operations, of the reduction B in Theorem 3.5. Let $o_{\exp}$ denote the number of group operations required to compute an exponentiation.

First, B computes the "core" elements, that is, $g_T^{z^k}$ for $0 \leq k \leq d_{\vec{T}}$ and $g_1^{z^k}$ for $0 \leq k \leq d_{\vec{R}}$ in Types 1 and 2, which requires $d_{\vec{T}} + 1$ computations of e and (for Types 1 and 2) $d_{\vec{R}} + 1$ computations of the morphism ψ.

Let $i \in \{1, \ldots, r\}$, $j \in \{1, \ldots, s\}$, and $k \in \{1, \ldots, f\}$. Then we denote

$$R_i(y_1 Z + x_1, \ldots, y_m Z + x_m) = \sum_{\ell=0}^{d_{\vec{R}}} \lambda_{R_i,\ell} Z^\ell,$$

$$S_j(y_1 Z + x_1, \ldots, y_m Z + x_m) = \sum_{\ell=0}^{d_{\vec{S}}} \lambda_{S_j,\ell} Z^\ell, \text{ and}$$

$$F_k(y_1 Z + x_1, \ldots, y_m Z + x_m) = \sum_{\ell=0}^{d_{\vec{F}}} \lambda_{F_k,\ell} Z^\ell.$$

Next, B computes the monomials $(g_1^{z^\ell})^{\lambda_{R_i,\ell}} = g_1^{\lambda_{R_i,\ell} z^i}$, for all $\ell \in \{0,\dots,d_{\vec{R}}\}$, as well as $(g_2^{z^\ell})^{\lambda_{S_j,\ell}} = g_2^{\lambda_{S_j,\ell} z^\ell}$ for $\ell \in \{0,\dots,d_{\vec{S}}\}$, and $(g_T^{z^\ell})^{\lambda_{F_k,\ell}} = g_T^{\lambda_{F_k,\ell} z^\ell}$ for $\ell \in \{0,\dots,d_{\vec{F}}\}$. This requires at most $(rd_{\vec{R}} + sd_{\vec{S}} + fd_{\vec{F}} + 3)o_{\exp}$ group operations. Finally, B computes, for all $i \in \{0,\dots,r\}$, all $j \in \{1,\dots,s\}$, and $k \in \{1,\dots,f\}$:

$$\prod_{\ell=0}^{d_{\vec{R}}} g_T^{\lambda_{R_i,\ell} z^\ell} = g_1^{R_i(y_1 z + x_1,\dots,y_m z + x_m)},$$
$$\prod_{\ell=0}^{d_{\vec{S}}} g_2^{\lambda_{S_j,\ell} z^\ell} = g_2^{S_j(y_1 z + x_1,\dots,y_m z + x_m)}, \text{ and}$$
$$\prod_{\ell=0}^{d_{\vec{F}}} g_T^{\lambda_{F_k,\ell} z^\ell} = g_T^{F_k(y_1 z + x_1,\dots,y_m z + x_m)},$$

which requires $rd_{\vec{R}} + sd_{\vec{S}} + fd_{\vec{F}}$ group operations. This concludes the computation of $\mathsf{A}_{\mathsf{alg}}$'s input.

After computing Q_1 (or Q_2 or Q_T depending on the case) from $\mathsf{A}_{\mathsf{alg}}$'s output, and its roots $z_1,\dots$, reduction B verifies $g^{z_i} \stackrel{?}{=} g^z$. Since the polynomial has at most d_τ roots, this last step requires at most $d_\tau \cdot o_{\exp}$ group operations.

We can therefore upper-bound the number of group operations performed by B^{A} by $t + o_1$ with $o_1 := o_0 + 2 + ((d_{\vec{R}}+1)r + (d_{\vec{S}}+1)s + (d_{\vec{F}}+1)f + d_\tau)o_{\exp} + rd_{\vec{R}} + sd_{\vec{S}} + fd_{\vec{F}}$ with $o_0 := d_{\vec{R}} + d_{\vec{F}} + 2$ for Types 1 and 2, and $o_0 := d_{\vec{F}} + 1$ for Type 3. By applying Fact B.1, we obtain the time bound claimed in Theorem 3.5.

C Proof of Theorem 5.2

We will use the following lemma in the proof of Theorem 5.2.

Lemma C.1 *Let n be an integer, $(P_i)_{i=1}^n \in (\mathbb{Z}_p[X_1,\dots,X_m])^n$ and $\vec{x} \in \mathbb{Z}_p^m$. Then $P_i(\vec{x}) \equiv_p 0$ for some $i \in [1,n]$ if and only if $\mathrm{LCM}(P_1,\dots,P_n)(\vec{x}) \equiv_p 0$.*

Proof. Let i be such that $P_i(\vec{x}) \equiv_p 0$. Since P_i divides $\mathrm{LCM}(P_1,\dots,P_n)$, we have

$$\mathrm{LCM}(P_1,\dots,P_n)(\vec{x}) \equiv_p 0.$$

Now suppose $\mathrm{LCM}(P_1,\dots,P_n)(\vec{x}) \equiv_p 0$. Since $\mathrm{LCM}(P_1,\dots,P_n)$ divides $\prod_{i=1}^n P_i$, we have $\prod_{i=1}^n P_i(\vec{x}) \equiv_p 0$. Because $\mathbb{Z}_p$ is an integer domain, this implies that $P_i(\vec{x}) \equiv_p 0$ for some i. ∎

Whereas in the proof of Theorem 3.5 we focused on Type-2 groups, for the sake of variation, we give the details for Type-3 groups, and then explain what changes for Types 1 and 2. Again, for $i \in \{1,2,T\}$, we denote g_i^u by $[u]_i$ for $u \in \mathbb{Z}_p$.

Let $\mathsf{A}_{\mathsf{alg}}$ be an algebraic algorithm against $\mathbf{r\ddot{u}ber}_{\mathcal{G}}$, defined by a tuple $(\vec{R},\vec{S},\vec{F},R',S',F')$ of (vectors of) rational fractions, that wins with advantage ε in time t. We construct an algebraic adversary $\mathsf{B}_{\mathsf{alg}}$ against (q_1,q_2)-$\mathbf{dlog}_{\mathcal{G}}$. Depending on why the tuple is non-trivial (conditions (3.1), (3.2) and/or (3.T) in Def. 3.3), $\mathsf{B}_{\mathsf{alg}}$ uses a different strategy, minimizing d_3 in Def. 5.1.

We detail the most complex case, which is (3.T); that is, F' is Type-3 independent from $(\vec{R},\vec{S},\vec{F})$.

Adversary $\mathsf{B}_{\mathsf{alg}}(g_1,\mathbf{Y}_1,\dots,\mathbf{Y}_{q_1},g_2,\mathbf{Z}_1,\dots,\mathbf{Z}_{q_2})$*:* On input a (q_1,q_2)-DLog instance with $\mathbf{Y}_i = [z^i]_1$ and $\mathbf{Z}_i = [z^i]_2$, reduction $\mathsf{B}_{\mathsf{alg}}$ simulates $\mathbf{r\ddot{u}ber}_{\mathcal{G}}$ for $\mathsf{A}_{\mathsf{alg}}$.

It first computes a least common multiple $D := \mathrm{Den}(\vec{R}\|\vec{S}\|\vec{F}\|R'\|S'\|F')$ of the denominators of $\vec{R}, \vec{S}, \vec{F}, R', S', F'$. It then picks random values $\vec{y} \xleftarrow{\$} (\mathbb{Z}_p^*)^m$ and $\vec{v} \xleftarrow{\$} \mathbb{Z}_p^m$ and uses them to implicitly define $x_i := y_i z + v_i \bmod p$.

If $D'(Z) := D(y_1 Z + v_1, \ldots, y_m Z + v_m)$ is the zero polynomial, it aborts. $(*)$

Else, for each root z_i of D', the reduction checks whether $g_1^{z_i} = \mathbf{Y}_1$; if such z_i exists, it stops the simulation and outputs z_i. $(\sharp)$

If $\mathsf{B}_{\mathsf{alg}}$ has not stopped then $D(\vec{x}) \not\equiv_p 0$. It now computes a least common multiple $D_1 := \mathrm{Den}(\vec{R}\|\vec{F})$ of the denominators of $\vec{R}$ and $\vec{F}$ and $D_2 := \mathrm{Den}(\vec{S})$.

From its (q_1, q_2)-DLog instance, it then computes $h_1 := g_1^{D_1(\vec{x})}$, $h_2 := g_2^{D_2(\vec{x})}$. Note that for all $i \in \{1, 2\}, h_i \neq 1$ (since D_i is a divisor of D, and $D(\vec{x}) \not\equiv_p 0$).

For $1 \leq i \leq r$, $\mathsf{B}_{\mathsf{alg}}$ computes $\mathbf{U}_i := [(\hat{R}_i \cdot D_1/\check{R}_i)(\vec{x})]_1 = h_1^{R_i(\vec{x})}$ (where $D_1/\check{R}_i$ is a polynomial by construction of D_1). This can be computed from $\mathbf{Y}_1, \ldots, \mathbf{Y}_{q_1}$, since $q_1 \geq \deg(D_1) + d_{\vec{R}}$. Likewise, for $1 \leq i \leq s$, $\mathsf{B}_{\mathsf{alg}}$ computes $\mathbf{V}_i := [(\hat{S}_i \cdot D_2/\check{S}_i)(\vec{x})]_2 = h_2^{S_i(\vec{x})}$ from $\mathbf{Z}_1, \ldots, \mathbf{Z}_{q_2}$. Finally, $\mathsf{B}_{\mathsf{alg}}$ computes $e(g_1, g_2)^{z^i} = [z^i]_T$ for $1 \leq i \leq q_1 + q_2$ from its instance, from which it then computes $\mathbf{W}_i := [D_2 \cdot \hat{F}_i \cdot D_1/\check{F}_i(\vec{x})]_T = e(h_1, h_2)^{F_i(\vec{x})}$ for $1 \leq i \leq f$.

With these values, $\mathsf{B}_{\mathsf{alg}}$ runs $(\mathbf{U}', \mathbf{V}', \mathbf{W}') \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}(\vec{\mathbf{U}}, \vec{\mathbf{V}}, \vec{\mathbf{W}})$, which also returns representations $\vec{\mu}$ for $\mathbf{U}'$, $\vec{\eta}$ for $\mathbf{V}'$ and $(A, \vec{\delta})$ for $\mathbf{W}'$ (so that (2) holds with $\nu_i = \gamma_{i,j} = 0$ for all i, j).

$\mathsf{B}_{\mathsf{alg}}$ defines the following polynomial:

$$P_T := \check{F}' \cdot D_1 \cdot D_2 \cdot (F' - \textstyle\sum_{i=1}^{r} \sum_{j=1}^{s} \alpha_{i,j} R_i S_j - \sum_{i=k}^{f} \delta_k F_k). \tag{18}$$

Note that P_T is indeed a polynomial, because $\check{F}' \cdot F'$, as well as $D_1 \cdot R_i$, $D_2 \cdot S_j$ and $D_1 \cdot F_k$ for all i, j, k are all polynomials.

From P_T, the reduction defines $P_T'(Z) := P_T(y_1 Z + v_1, \ldots, y_m Z + v_m)$. If P_T' is the zero polynomial then $\mathsf{B}_{\mathsf{alg}}$ aborts. $(**)$

Else, it factors P_T' to obtain its roots $z_1, \ldots$ If for one of them we have $g_1^{z_i} = \mathbf{Y}_1$, then $\mathsf{B}_{\mathsf{alg}}$ returns z_i. $(\sharp\sharp)$

We analyze $\mathsf{B}_{\mathsf{alg}}$'s success probability. First note that $\mathsf{B}_{\mathsf{alg}}$ solves the DLog challenge if it stops in line $(\sharp)$ and it fails if it aborts in line $(*)$. Otherwise, $\mathsf{B}_{\mathsf{alg}}$ perfectly simulates game **rüber** (Fig. 4), as the values x_i are uniformly distributed in $\mathbb{Z}_p$, and the tests in lines 03 and 04 in **rüber** are all done: For some i: $\check{R}_i(\vec{x}) \equiv_p 0$ or $\check{S}_i(\vec{x}) \equiv_p 0$ or $\check{F}_i(\vec{x}) \equiv_p 0$, or $\check{R}'(\vec{x}) \equiv_p 0$ or $\check{S}'(\vec{x}) \equiv_p 0$ or $\check{F}'(\vec{x}) \equiv_p 0$ if and only if a least common denominator of all these polynomials vanishes at $\vec{x}$, i.e. $D(\vec{x}) \equiv_p 0$ (by Lemma C.1), and $\mathsf{B}_{\mathsf{alg}}$ only proceeds if the latter is not the case. In this case $\mathsf{A}_{\mathsf{alg}}$'s inputs, $\vec{\mathbf{U}}$, $\vec{\mathbf{V}}$ and $\vec{\mathbf{W}}$ are correctly computed.

If $\mathsf{B}_{\mathsf{alg}}$ does not stop in line $(**)$ either and $\mathsf{A}_{\mathsf{alg}}$ is successful then

$$\mathbf{W}' = e(h_1, h_2)^{F'(\vec{x})} = \left[D_1(\vec{x}) D_2(\vec{x}) F'(\vec{x})\right]_T . \tag{19}$$

On the other hand, from $\mathsf{A}_{\mathsf{alg}}$'s representation $(A, \vec{\delta})$ of $\mathbf{W}'$ and from the definitions of $\mathbf{U}_i$, $\mathbf{V}_j$ and $\mathbf{W}_k$ we have:

$$\begin{aligned} \mathbf{W}' &= \textstyle\prod_i \prod_j e(\mathbf{U}_i, \mathbf{V}_j)^{\alpha_{i,j}} \cdot \prod_k \mathbf{W}_k^{\delta_k} \\ &= \left[\textstyle\sum_i \sum_j \alpha_{i,j} (\hat{R}_i \cdot D_1/\check{R}_i)(\vec{x}) \, (\hat{S}_j \cdot D_2/\check{S}_j)(\vec{x}) + \sum_k \delta_k (D_2 \cdot \hat{F}_k \cdot D_1/\check{F}_k)(\vec{x})\right]_T . \end{aligned} \tag{20}$$

Equating the representations of $\mathbf{W}'$ in (19) and (20) in base $e(g_1, g_2)$ yields

$$(D_1 \cdot D_2 \cdot F' - \textstyle\sum_i \sum_j \alpha_{i,j}\, \hat{R}_i \cdot D_1/\check{R}_i \cdot \hat{S}_j \cdot D_2/\check{S}_j + \sum_k \delta_k\, D_2 \cdot \hat{F}_k \cdot D_1/\check{F}_k)(\vec{x}) \equiv_p 0\ . \quad (21)$$

Multiplying the above by $\check{F}'(\vec{x})$ yields $P_T(\vec{x}) \equiv_p 0$ and since $x_i \equiv_p y_i z + v_i$, we have $P'_T(z) \equiv_p 0$ as well. By factoring P'_T, reduction $\mathsf{B}_{\mathsf{alg}}$ finds thus the solution z.

We have shown that unless $\mathsf{B}_{\mathsf{alg}}$ aborts in lines $(*)$ or $(**)$, it finds the (q_1, q_2)-DLog solution whenever $\mathsf{A}_{\mathsf{alg}}$ wins **rüber**. In the remainder of the proof we bound the probability that $\mathsf{B}_{\mathsf{alg}}$ aborts. It aborts if and only if the following polynomial is zero: $Q(Z) := (D' \cdot P'_T)(Z) = (D \cdot P_T)\,(y_1 Z + v_1, \ldots, y_m Z + v_m)$. It thus suffices to upper-bound the probability that the coefficient of maximal degree of this polynomial is zero. By Lemma 2.1, this coefficient can be represented as a polynomial $Q^{\max}$ in variables $(Y_1, \ldots, Y_m)$ that is of the same degree as $D \cdot P_T$, which we bound as follows (recall that $d_{\mathrm{den}} = \deg D$; cf. Def. 5.1):

$$\begin{aligned}
\deg(D \cdot P_T) &\le d_{\mathrm{den}} + \deg \check{F}' + \deg D_1 + \deg D_2 + \\
&\qquad \max\{\deg F',\ \deg R_i + \deg S_j,\ \deg F_k\}_{1\le i\le r, 1\le j\le s, 1\le k\le f} \\
&= d_{\mathrm{den}} + \check{d}_{F'} + \check{d}_{\vec{R}\|\vec{F}} + \check{d}_{\vec{S}} + \max\{d_{F'}, d_{\vec{R}} + d_{\vec{S}}, d_{\vec{F}}\} \\
&= d_{3.T}
\end{aligned}$$

As the values $y_1 z, \ldots, y_m z$ are completely hidden from $\mathsf{A}_{\mathsf{alg}}$ (they are "one-time-padded" with $v_1, \ldots, v_m$, resp.), the values $(A, \vec{\delta})$ returned by $\mathsf{A}_{\mathsf{alg}}$ are independent from $\vec{y}$. Since $\vec{y}$ is moreover independent from F', $\vec{R}$, $\vec{S}$ and $\vec{F}$, it is also independent from P_T, D, Q and $Q^{\max}$. The probability that $Q \equiv 0$ is thus upper-bounded by the probability that $Q^{\max}(\vec{y}) \equiv_p 0$ when evaluated at the random point $\vec{y}$. By Lemma 2.2 (Schwartz-Zippel), the probability that $Q^{\max}(\vec{y}) \equiv_p 0$ is thus upper-bounded by $\frac{d_{3.T}}{p-1}$.

We turn to the remaining cases (3.1) and (3.2), which follow similarly, except that P_T in (18) is replaced by different polynomials P. Moreover, we can optimize the loss of the security reduction, by reducing the degree of P. Note that the two properties which are used in the proof are:

- P is a non-zero polynomial, and
- if $\mathsf{A}_{\mathsf{alg}}$ wins the game then $P(\vec{x}) \equiv_p 0$.

In Case (3.1), that is, when R' is linearly independent from $\vec{R}$, from $\mathsf{A}_{\mathsf{alg}}$'s output $\mathbf{U}' = h_1^{R'(\vec{x})} = [D_1(\vec{x})\, R'(\vec{x})]_1$ and its representation $\vec{\mu}$ of $\mathbf{U}' = \prod_i \mathbf{U}_i^{\mu_i} = [\sum_i (\hat{R}_i \cdot D_1/\check{R}_i)(\vec{x})]_1$, we get that the following function vanishes at $\vec{x}$ (which corresponds to (21) above):

$$D_1 \cdot R' - \textstyle\sum_i \mu_i\, \hat{R}_i \cdot D_1/\check{R}\ .$$

Multiplying this by $\check{R}'$ yields a polynomial. In contrast to case (3.T), we can moreover divide by $D_1/\mathrm{Den}(\vec{R})$ and still obtain a polynomial. We thus define:

$$P_1 := \check{R}' \cdot \mathrm{Den}(\vec{R}) \cdot (R' - \textstyle\sum_{i=1}^{r} \mu_i R_i)\ .$$

Since P_1 is of degree at most $\check{d}_{R'} + \check{d}_{\vec{R}} + \max\{d_{R'}, d_{\vec{R}}\}$, the probability of $\mathsf{B}_{\mathsf{alg}}$ aborting is $\frac{d_{3.1}}{p-1}$.

For Case (3.2), from $\mathsf{A}_{\mathsf{alg}}$'s representation $\vec{\eta}$ of $\mathbf{V}'$, we analogously define

$$P_2 := \check{S}' \cdot \mathrm{Den}(\vec{S}) \cdot (S' - \textstyle\sum_{i=1}^{s} \eta_i S_i)\ ,$$

which is of degree at most $\check{d}_{S'} + \check{d}_{\vec{S}} + \max\{d_{S'}, d_{\vec{S}}\} = d_{3.2}$.

The theorem for Type-3 groups now follows because

$$\mathbf{Adv}^{(q_1,q_2)\text{-}\mathbf{dlog}}_{\mathcal{G},\mathsf{B_{alg}}} \geq \mathbf{Adv}^{\mathbf{rüber}}_{\mathcal{G},\mathsf{A_{alg}}} - \Pr[\mathsf{B_{alg}} \text{ aborts}]$$

and $\mathsf{B_{alg}}$ follows the strategy that minimizes its abort probability to $\min\{\frac{d_{3.1}}{p-1}, \frac{d_{3.2}}{p-1}, \frac{d_{3.T}}{p-1}\} = \frac{d_3}{p-1}$.

Groups of Type 1 and 2. The reductions for bilinear groups of Types 1 and 2 to q-$\mathbf{dlog}_{\mathcal{G}_2}$ work analogously; the main difference is that we can only redefine g_2 as h_2, since h_2 defines $h_1 = \psi(h_2)$. This means that instead of D_1 and D_2 as in the proof above, we define $D_{1,2} := \mathrm{Den}(\vec{R}\|\vec{S}\|\vec{F})$ and set $h_2 := g_2^{D_{1,2}(\vec{x})}$. For Type-2 groups, let $(\vec{\mu},\vec{\nu}),\vec{\eta}$ and $(A,\Gamma,\vec{\delta})$ be $\mathsf{A_{alg}}$'s representations of $\mathbf{U}'$, $\mathbf{V}'$ and $\mathbf{W}'$, respectively. Then $\mathsf{B_{alg}}$ defines the polynomials P_1, P_2 and P_T as follows:

$$\begin{aligned}
P_1 &:= \check{R}' \cdot \mathrm{Den}(\vec{R}\|\vec{S}) \cdot (R' - \textstyle\sum_{i=1}^{r} \mu_i R_i - \sum_{i=1}^{s} \nu_i S_i) \\
P_2 &:= \check{S}' \cdot \mathrm{Den}(\vec{S}) \cdot (S' - \textstyle\sum_{i=1}^{s} \eta_i S_i) \\
P_T &:= \check{F}' \cdot \mathrm{Den}(\vec{R}\|\vec{S}\|\vec{F}) \cdot \mathrm{Den}(\vec{S}) \cdot \\
&\qquad (F' - \textstyle\sum_{i=1}^{r}\sum_{j=1}^{s} \alpha_{i,j}\, R_i \cdot S_j - \sum_{i=1}^{s}\sum_{j=1}^{s} \gamma_{i,j}\, S_i \cdot S_j - \sum_{i=1}^{f} \delta_i F_i) \ .
\end{aligned}$$

As for the case $(3.T)$ above, by applying Lemmas 2.1 and 2.2 to $D \cdot P_1, D \cdot P_2$, and $D \cdot P_T$, we deduce that the probability of aborting is bounded by $\frac{d_2}{p-1}$.

For Type-1 groups, let $(\vec{\mu},\vec{\nu}),(\vec{\eta},\vec{\zeta})$ and $(A,B,\Gamma,\vec{\delta})$ be $\mathsf{A_{alg}}$'s representations of $\mathbf{U}'$, $\mathbf{V}'$ and $\mathbf{W}'$, respectively. Then $\mathsf{B_{alg}}$ defines the polynomials P_1, P_2 and P_T as follows:

$$\begin{aligned}
P_1 &:= \check{R}' \cdot \mathrm{Den}(\vec{R}\|\vec{S}) \cdot (R' - \textstyle\sum_{i=1}^{r} \mu_i R_i - \sum_{i=1}^{s} \nu_i S_i) \\
P_2 &:= \check{S}' \cdot \mathrm{Den}(\vec{R}\|\vec{S}) \cdot (S' - \textstyle\sum_{i=1}^{r} \zeta_i R_i - \sum_{i=1}^{s} \eta_i S_i) \\
P_T &:= \check{F}' \cdot \mathrm{Den}(\vec{R}\|\vec{S}\|\vec{F}) \cdot \mathrm{Den}(\vec{R}\|\vec{S}) \cdot (F' - \textstyle\sum_{i=1}^{r}\sum_{j=1}^{s} \alpha_{i,j}\, R_i \cdot S_j \\
&\qquad - \textstyle\sum_{i=1}^{r}\sum_{j=1}^{r} \beta_{i,j}\, R_i \cdot R_j - \sum_{i=1}^{s}\sum_{j=1}^{s} \gamma_{i,j}\, S_i \cdot S_j - \sum_{i=1}^{f} \delta_i F_i) \ .
\end{aligned}$$

By an analysis analogous to the above, the abort probability is bounded by $\frac{d_1}{p-1}$.

D Proof of Theorem 8.3

We give a detailed proof for Type-2 bilinear groups. Let $\mathsf{A_{alg}}$ be an algebraic algorithm against $\mathbf{gegenüber}_{\mathcal{G}}$ that wins with advantage ϵ in time t. We construct a generic reduction with oracle access to $\mathsf{A_{alg}}$, which yields an algebraic adversary $\mathsf{B_{alg}}$ against q-$\mathbf{dlog}_{\mathcal{G}_2}$. There are three (non-exclusive) types of reasons that $(\mathcal{Q}_1,\mathcal{Q}_2,\mathcal{Q}_T,R^*,S^*,F^*)$ is non-trivial; that is $(2.i)$ from Def. 8.1 holds for some $i \in \{1,2,T\}$. Each condition enables a different type of reduction, of which $\mathsf{B_{alg}}$ runs the one that minimizes d_τ. We start with Case (2.1), that is, $R^* \notin \mathrm{FrSp}(\mathcal{Q}_1 \cup \mathcal{Q}_2)$.

Adversary $\mathsf{B_{alg}}(g_2,\mathbf{Z}_1,\ldots,\mathbf{Z}_q)$*:* On input a problem instance of game q-$\mathbf{dlog}_{\mathcal{G}_2}$ with $\mathbf{Z}_i = [z^i]_2$, $\mathsf{B_{alg}}$ defines $g_1 \leftarrow \psi(g_2)$ and $g_T \leftarrow e(g_1,g_2)$. Then, it picks random values $\vec{y} \xleftarrow{\$} (\mathbb{Z}_p^*)^m$ and $\vec{v} \xleftarrow{\$} \mathbb{Z}_p^m$, sets $x_i := y_i z + v_i \bmod p$ (implicitly), and runs $((\mathbf{U}_1,\mathbf{V}_1,\mathbf{W}_1,\mathbf{U}_2,\mathbf{V}_2,\mathbf{W}_2),(R^*,S^*,F^*)) \xleftarrow{\$} \mathsf{A_{alg}}^{\mathsf{O}(\cdot,\cdot)}()$. Oracle calls $\mathsf{O}(i,P(\vec{X}))$ are answered by computing and returning $\mathbf{Y}_{P,i} := [P(x_1,\ldots,x_m)]_i$ from the q-DLog instance, the morphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$ and the pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. $\mathsf{B_{alg}}$ can do so efficiently since the total degree of the polynomials in $\mathcal{Q}_1$, $\mathcal{Q}_2$ and $\mathcal{Q}_T$ are bounded by q, q and $2q$, respectively.

Since $\mathsf{A}_{\mathsf{alg}}$ is algebraic, for all $k_1, k_2, k_3 \in \{1,2\}$ it also returns vectors and matrices $\vec{\mu}^{(k_1)}, \vec{\nu}^{(k_1)}, \vec{\zeta}^{(k_2)}, \vec{\delta}^{(k_3)}, A^{(k_3)} = (\alpha_{j,k}^{(k_3)})_{j,k}$, $\Gamma^{(k_3)} = (\gamma_{j,k}^{(k_3)})_{j,k}$ respectively indexed by $\mathcal{Q}_1$, $\mathcal{Q}_2$, $\mathcal{Q}_1$, $\mathcal{Q}_T$, $\mathcal{Q}_1 \times \mathcal{Q}_2$ and $\mathcal{Q}_2 \times \mathcal{Q}_2$ such that

$$\mathbf{U}_{k_1} = \prod_{R\in\mathcal{Q}_1} \mathbf{Y}_{R,1}^{\mu_R^{(k_1)}} \cdot \prod_{R\in\mathcal{Q}_2} \psi(\mathbf{Y}_{R,2})^{\nu_R^{(k_1)}} \tag{22a}$$

$$\mathbf{V}_{k_2} = \prod_{R\in\mathcal{Q}_2} \mathbf{Y}_{R,2}^{\zeta_R^{(k_2)}} \tag{22b}$$

$$\mathbf{W}_{k_3} = \prod_{R\in\mathcal{Q}_1}\prod_{S\in\mathcal{Q}_2} e(\mathbf{Y}_{R,1}, \mathbf{Y}_{S,2})^{\alpha_{R,S}^{(k_3)}} \tag{22c}$$

$$\cdot \prod_{R\in\mathcal{Q}_2}\prod_{S\in\mathcal{Q}_2} e(\psi(\mathbf{Y}_{R,2}), \mathbf{Y}_{S,2})^{\gamma_{R,S}^{(k_3)}} \cdot \prod_{R\in\mathcal{Q}_T} \mathbf{Y}_{R,T}^{\delta_R^{(k_3)}} . \tag{22d}$$

$\mathsf{B}_{\mathsf{alg}}$ then computes the following multivariate polynomial, which corresponds to the logarithm of $\mathbf{U}_1^{R^*(\vec{x})} \cdot \mathbf{U}_2^{-1}$ in base g_1:

$$P_1(\vec{X}) := R^*(\vec{X})\big(\textstyle\sum_{R\in\mathcal{Q}_1} \mu_R^{(1)} R(\vec{X}) + \sum_{R\in\mathcal{Q}_2} \nu_R^{(1)} R(\vec{X})\big) \\ - \textstyle\sum_{R\in\mathcal{Q}_1} \mu_R^{(2)} R(\vec{X}) - \sum_{R\in\mathcal{Q}_2} \nu_R^{(2)} R(\vec{X}).$$

Since in Case (2.1) we have $R^* \notin \mathrm{FrSp}\,(\mathcal{Q}_1 \cup \mathcal{Q}_2)$, the polynomial P_1 is non-zero. From P_1, the reduction defines $Q_1(Z) := P_1(y_1 Z + v_1, \ldots, y_m Z + v_m)$. If Q_1 is the zero polynomial then $\mathsf{B}_{\mathsf{alg}}$ aborts. $(*)$

Else, it factors Q_1 to obtain its roots $z_1, \ldots$ (of which there are at most $\max\{\deg R^*, 1\} \cdot \max\{d_1', d_2'\}$). If for one of them we have $g_2^{z_i} = \mathbf{Z}$, then $\mathsf{B}_{\mathsf{alg}}$ returns z_i.

We analyze $\mathsf{B}_{\mathsf{alg}}$'s success probability. First note that $\mathsf{B}_{\mathsf{alg}}$ perfectly simulates game $\mathbf{gegenüber}_{\mathcal{G}}$, as the values x_i are uniformly distributed in $\mathbb{Z}_p$ and oracle calls are correctly computed.

Moreover, if $\mathsf{B}_{\mathsf{alg}}$ does not abort in $(*)$ and $\mathsf{A}_{\mathsf{alg}}$ wins game $\mathbf{gegenüber}_{\mathcal{G}}$, then $\mathbf{U}_1^{R^*(\vec{x})} \cdot \mathbf{U}_2^{-1} = 1$. Substituting the right-hand side of (22a) for $\mathbf{U}_1$ and $\mathbf{U}_2$ in this equation and considering the discrete logarithm in base g_1, this yields $P_1(\vec{x}) \equiv_p 0$. Since $x_i = y_i z + v_i$, we moreover have $Q_1(z) = 0$. By factoring Q_1, reduction $\mathsf{B}_{\mathsf{alg}}$ finds thus the solution z.

It remains to bound the probability that $\mathsf{B}_{\mathsf{alg}}$ aborts in $(*)$, that is, the probability that $0 \equiv Q_1(Z) = P_1\,(y_1 Z + v_1, \ldots, y_m Z + v_m)$. The analysis is analogous to all previous theorems:

We upper-bound the probability that the coefficient of maximal degree, say d, is zero. By Lemma 2.1, this coefficient can be represented as a polynomial $Q_1^{\max}$ in variables $(Y_1, \ldots, Y_m)$ of the same degree d. The values $y_1 z, \ldots, y_m z$ are completely hidden from $\mathsf{A}_{\mathsf{alg}}$ because they are "one-time-padded" with $v_1, \ldots, v_m$, respectively. This means that the values $\vec{\mu}^{(1)}, \vec{\mu}^{(2)}, \vec{\nu}^{(1)}$ nad $\vec{\nu}^{(2)}$ returned by $\mathsf{A}_{\mathsf{alg}}$ are independent from $\vec{y}$. Since $\vec{y}$ is moreover independent from $\vec{R^*}$, $\mathcal{Q}_1$ and $\mathcal{Q}_2$, $\mathcal{Q}_T$, it is also independent from P_1, Q_1 and $Q_1^{\max}$. The probability that $Q_1 \equiv 0$ is thus upper-bounded by the probability that $Q_1^{\max}(\vec{y}) \equiv_p 0$ when evaluated at the random point $\vec{y}$. By the Schwartz-Zippel lemma, the probability that $Q_1(Z) \equiv 0$ is thus upper-bounded by $\frac{d}{p-1}$. The degree d of Q_1 (and thus of $Q_1^{\max}$) is upper-bounded by the total degrees of P_1, which is $\max\{1, \deg R^*\} \cdot \max\{d_1', d_2'\} = d_{2.1}$ in Def. 8.2. $\mathsf{B}_{\mathsf{alg}}$ thus aborts in line (*) with probability at most $\frac{d_{2,1}}{p-1}$.

Cases (2.2) (where we have $S^* \notin \mathrm{FrSp}(\mathcal{Q}_2)$) and $(2.T)$ are treated analogously. Theorem 8.3 for Type-2 groups now follows because

$$\mathbf{Adv}^{q\text{-}\mathbf{dlog}}_{\mathcal{G}_2,\mathsf{B}_{\mathsf{alg}}} \geq \mathbf{Adv}^{\mathbf{gegenüber}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}} - \Pr[\mathsf{B}_{\mathsf{alg}} \text{ aborts}]$$

and $\mathsf{B}_{\mathsf{alg}}$ follows the strategy that minimizes its abort probability to $\min\{\frac{d_{2.1}}{p-1}, \frac{d_{2.2}}{p-1}, \frac{d_{2.T}}{p-1}\} = \frac{d_2}{p-1}$.

The proofs for groups of Type 1 and Type 3 are done by analogous adaptations of Theorem 3.5 as just shown for Type 2.