Revisiting the Uber Assumption in the Algebraic Group Model: Fine-Grained Bounds in Hidden-Order Groups and Improved Reductions in Bilinear Groups

Lior Rotem*

Abstract

We prove strong security guarantees for a wide array of computational and decisional problems, both in hidden-order groups and in bilinear groups, within the algebraic group model (AGM) of Fuchsbauer, Kiltz and Loss (CRYPTO '18). As our first contribution, we put forth a new fine-grained variant of the Uber family of assumptions in hidden-order groups. This family includes in particular the repeated squaring function of Rivest, Shamir and Wagner, which underlies their time-lock puzzle as well as the main known candidates for verifiable delay functions; and a computational variant of the generalized BBS problem, which underlies the timed commitments of Boneh and Naor (CRYPTO '00). We then provide two results within a variant of the AGM, which show that the hardness of solving problems in this family in a less-than-trivial number of steps is implied by well-studied assumptions. The first reduction may be applied in any group (and in particular, class groups), and is to the RSA assumption; and our second reduction is in RSA groups with a modulus which is the product of two safe primes, and is to the factoring assumption.

Additionally, we prove that the hardness of any computational problem in the Uber family of problems in bilinear groups is implied by the hardness of the q-discrete logarithm problem. The parameter q in our reduction is the maximal degree in which a variable appears in the polynomials which define the specific problem within the Uber family. This improves upon a recent result of Bauer, Fuchsbauer and Loss (CRYPTO '20), who obtained a similar implication but for a parameter q which is lower bounded by the maximal *total* degree of one of the above polynomials. We discuss the implications of this improvement to prominent group key-exchange protocols.

*School of Computer Science and Engineering, Hebrew University of Jerusalem, Jerusalem 91904, Israel. Email: lior.rotem@cs.huji.ac.il. Supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities and by the European Union's Horizon 2020 Framework Program (H2020) via an ERC Grant (Grant No. 714253).

Contents

1 Introduction

The algebraic group model (the AGM) was introduced by Fuchsbauer, Kiltz and Loss[1] [FKL18] with the aim of striking a middle ground between the generic group model (the GGM) and the standard model. In contrast to the GGM, algorithms within the AGM (also known as algebraic algorithms) do receive the representation of group elements and may use it in any way they see fit. The restriction, however, is that whenever an algebraic algorithm outputs a group element, it must provide alongside it a representation of it in the basis of its input group elements. This representation serves as an explanation as to how the output element was computed from the input elements in an algebraic manner. Given the less restrictive nature of the AGM when compared to the GGM,[2] a central line of research over the last couple of years has focused on establishing the security of cryptographic schemes and assumptions within the AGM; see for example [FKL18, MTT19, MBK+19, ABB+20, AGK20, BDF+20, BFL20, CHM+20, CH20, FPS20, GRW+20, KLX20, RS20a, ABK+21, GT21, KLX22].

The sequentiality of repeated squaring. The "repeated squaring" function in hidden-order groups, first suggested by Rivest, Shamir and Wagner [RSW96], serves as the basis for the main candidate constructions of both time-lock puzzles and of verifiable delay functions [BBB+18, Pie19, Wes19]. For years, however, the sequentiality of this function remained purely as an assumption, and there was no known reduction (in idealized models or otherwise) relating it to the hardness of a better-established assumptions. Recently, Katz, Loss and Xu [KLX20] presented a strengthened version of the AGM (the strong AGM) and provided evidence for the sequentiality of repeated squaring within this model.[3] Concretely, they showed that any strongly-algebraic algorithm which manages to speed-up the repeated squaring function in the group QR_N of quadratic residues modulo N (where N is the product of two safe primes), can be used in order to factor the modulus N. Their result provides a novel and important corroboration for the sequentiality of repeated squaring in the group QR_N. However, it is limited in two respects: Firstly, it inherently relies on the algebraic structure of QR_N and does not apply to other hidden-order groups of interest, such as RSA groups or the class group of imaginary quadratic number fields [BH01, BBH+02, BBF18, Wes19]. Secondly, it addresses only the repeated squaring function, and leaves out other possible fine-grained problems in hidden-order groups.

The Uber problem in bilinear groups. The Uber family of problems in bilinear groups was introduced by Boneh, Boyen and Goh [BBG05, Boy08] as a unified framework for reasoning about computational problems in such groups. A problem in the family is parameterized by three tuples of multivariate polynomials $\vec{F}, \vec{H}, \vec{K}$ and three polynomials Q_1, Q_2, Q_T and is defined by the following task: Given the group elements $\{g_1^{F_i(\vec{x})}\}_i, \{g_2^{H_i(\vec{x})}\}_i, \{g_T^{K_i(\vec{x})}\}_i$ for a random vector $\vec{x}$, compute $g_1^{Q_1(\vec{x})}$, $g_2^{Q_2(\vec{x})}$ and $g_T^{Q_T(\vec{x})}$, where g_1, g_2 and g_T are the generators of the source groups and the target group, respectively. Bauer, Fuchsbauer and Loss [BFL20] recently showed that in the AGM, the hardness of any problem in this family, as long as it does not admit a trivial solution, is implied by the

[1] Following Abdalla, Benhamouda and Mackenzie [ABM15] and Bernhard, Fischlin and Warinschi [BFW16]. Additionally, the earlier works of Boneh and Venkatesan [BV98] and Paillier and Vergnaud [PV05] considered algebraic *reductions*, rather than algebraic adversaries.

[2] The AGM may also be instantiated in the standard model from falsifiable assumptions, as demonstrated by the elegant work of Agrikola, Hofheinz and Kastner [AHK20]. This is in contrast to the GGM [Den02].

[3] Another recent result [RS20b], proving the equivalence of speeding-up repeated squaring and factoring within the generic ring model is discussed in Section 1.2.

hardness of the q-DLOG problem in one of the source groups.[4] Their result provides a clean and succinct characterization of the Uber family, reducing its hardness to that of a seemingly simpler and better-understood family of problems. However, the parameter q in their result is lower bounded by the *maximal total degree* of the polynomials in $\vec{F}, \vec{H}, \vec{K}$, which may not be optimal. To see why that is, consider the following toy problem for any integer n: Given a generator g and a tuple $(g^{x_1}, g^{x_2}, g^{x_3}, g^{x_1 x_3}, g^{x_2 x_3})$ of group elements for randomly-chosen x_1, x_2, x_3, compute $g^{x_1 x_2 x_3}$. On the one hand, the result of Bauer et al. can be used to conclude that the hardness of this problem in the AGM is implied by the hardness of the 2-DLOG problem. On the other hand, it is not hard to see that this problem is actually equivalent to the Computational Diffie-Hellman (CDH) problem. The CDH problem was proven equivalent to the DLOG problem (i.e., 1-DLOG) in the AGM [FKL18], suggesting that the bound of Bauer et al. might not be optimal with respect to the parameter q.[5]

1.1 Our Results

In this work, we provide stronger hardness results within the AGM, both for a new fine-grained Uber family of problems in hidden-order groups that we put forth, and for the Uber family of problems in bilinear groups.

1.1.1 Our Results for Fine-Grained Computations in Hidden-Order Groups

A fine-grained Uber family. As our first contribution, we present a univariate and fine-grained variant of the Uber family of problems in hidden-order groups. Our family of problems generalizes the repeated squaring function [RSW96], as well as well as a computational variant[6] of the generalized BBS problem underlying Boneh and Naor's timed commitments [BBS86, BN00] and refinements thereof [GMP+06] (see also [GJ02, GP03]). A problem in this family is parameterized by integers $u_1, \ldots, u_\ell$ and an integer w, and requires that the adversary computes x^w for a uniformly-chosen group element x, given $(x^{u_1}, \ldots, x^{u_\ell})$ as input. Of course, if one can efficiently express w as a linear combination of $u_1, \ldots, u_\ell$ with integer coefficients, then one can trivially compute x^w from $(x^{u_1}, \ldots, x^{u_\ell})$ using a polynomial number of group operations. Therefore, we carefully define what it means for a strongly-algebraic algorithm to non-trivially solve a problem in our new Uber family, also accounting for the possibility of parallel computations. Looking ahead, the repeated squaring function is obtained by setting $u_1 = 1$ and $w = 2^T$, whereas the generalized BBS problem is obtained by setting $u_1 = 0$, $u_i = 2^{2^{i-2}}$ for $i = 2, \ldots, k+2$ and $w = 2^{2^{k+1}}$. Moreover, the repeated squaring function cannot be trivially solved (according to our triviality notion) in less than T steps, whereas the generalized BBS problem cannot be trivially solved is less than 2^k steps.

The algebraic hardness of the fine-grained Uber problem. Within the strong AGM of Katz, Loss and Xu [KLX20], we provide evidence for the hardness of our new family of problems. Firstly, we present a general hardness result which may be applied in any cryptographic group, and prove that the hardness of any problem in the fine-grained Uber family in a group $\mathbb{G}$ (i.e., the hardness

[4] The q-DLOG problem in a cyclic group $\mathbb{G}$ is defined as: Given a generator g and the q group elements $g^x, \ldots, g^{x^q}$ for a randomly chosen x, compute x. Actually, in type 3 bilinear groups (see Section 2), Bauer et al. considered the related (q_1, q_2)-DLOG problem (which we formally define in Section 5).

[5] Though in the specific case of the toy problem considered above, the result of Fuchsbauer, Kiltz and Loss [FKL18] already shows it to be equivalent to the DLOG problem, this is not the case for general instances of the Uber family in bilinear groups, motivating Theorem 1.3 below.

[6] In practice, one can always achieve pseudorandomness from this computational variant heuristically by applying a cryptographic hash function (e.g., SHA) onto the output of the problem [BR93].

of computing the target group element x^w in a less-than-trivial number of steps), is implied by the RSA assumption in the same group.

Theorem 1.1 (informal)**.** *Let $\mathbb{G}$ be a group, let $\ell \in \mathbb{N}$ and let $u_1, \ldots, u_\ell, w \in \mathbb{Z}$. Let A be a strongly-algebraic algorithm for the $(u_1, \ldots, u_\ell, w)$-univariate fine-grained Uber problem in the group $\mathbb{G}$, which makes a less-than-trivial number of steps. Then, there exists an algorithm B for the RSA problem in $\mathbb{G}$ whose running time and success probability are polynomially-related to those of A.*

Theorem 1.1 immediately implies that any strongly-algebraic algorithm that computes the repeated squaring function in less than T steps in some group $\mathbb{G}$, or solves the generalized BBS problem in less than 2^k steps, can be used in order to solve the RSA problem in the group. Note that Theorem 1.1 assumes nothing about the group $\mathbb{G}$, and in particular can be applied in any group in which the RSA assumption is believed to hold, such as RSA groups, class groups of imaginary quadratic number fields, and the group QR_N with respect to arbitrary bi-prime moduli. Importantly, Theorem 1.1 provides evidence for the sequentiality of repeated squaring in class groups, as the RSA problem has been considered and studied in these groups for a while now (see for example [BH01, BBH$^+$02, DK02] and the references therein), whereas the sequentiality of repeated squaring in these groups is a much newer assumption [BBF18, Wes19]. As far as we are aware, this is the first result supporting the sequentiality of repeated squaring in class groups.

Our second hardness result for the fine-grained Uber problem considers RSA groups with a modulus N which is the product of two safe primes. Informally, within the strong AGM, we prove that in such groups, the hardness of any problem in the fine-grained Uber family is implied by the hardness of factoring N.

Theorem 1.2 (informal)**.** *Let N be the product of two safe primes and let $\ell \in \mathbb{N}$ and let $u_1, \ldots, u_\ell, w \in \mathbb{Z}$. Let A be a strongly-algebraic algorithm for the $(u_1, \ldots, u_\ell, w)$-univariate fine-grained Uber problem in $\mathbb{Z}_N^*$ which makes a less-than-trivial number of steps. Then, there exists an algorithm B for factoring N whose running time and success probability are polynomially-related to those of A.*

Observe that Theorem 1.2 strictly strengthens the result of Katz, Loss and Xu [KLX20] in two respects. Firstly, by considering our new fine-grained Uber family, which captures in particular the sequentiality of the repeated squaring function (considered by Katz, Loss and Xu), but also other problems, such as the generalized BBS problem [BN00]. Secondly, Theorem 1.2 considers RSA groups with respect to moduli which are the product of two safe primes, whereas Katz, Loss and Xu consider the group QR_N with respect to the same family of moduli. Since a uniformly-random element in $\mathbb{Z}_N^*$ is in QR_N with probability $1/4$, the hardness of any problem within the fine-grained Uber family with respect to $\mathbb{Z}_N^*$ implies in particular its hardness with respect to QR_N.

Interestingly, to the best of our knowledge, Theorems 1.1 and 1.2 are the first applications of the algebraic group model (or a variant thereof) in non-cyclic groups. As far as we are aware, all previous reductions in the AGM were either in cyclic groups of prime order or in the cyclic group QR_N where N is the product of two safe primes. Hence, our work exemplifies for the first time the applicability of the AGM beyond cyclic groups.

1.1.2 Our Results for Bilinear Groups

The Algebraic Hardness of the Uber problem in bilinear groups. We strengthen the characterization of Bauer, Fuchsbauer and Loss [BFL20] of the Uber family framework in bilinear groups. Concretely, let $\vec{F}, \vec{H}, \vec{K}$ be vectors of polynomials and let Q_1, Q_2, Q_T be polynomials. We prove that within the AGM, as long as these polynomials do not admit a trivial solution, the hardness of their respective problem in the Uber family is implied by the q-DLOG assumption, where q is lower bounded by the maximal degree in which *a variable appears* in $\vec{F}, \vec{H}, \vec{K}$.

Theorem 1.3 (informal)**.** *Let $\mathcal{G}$ be a bilinear group, let $\vec{F}, \vec{H}, \vec{K}$ be vectors of m-variate polynomials and let Q_1, Q_2, Q_T be m-variate polynomials that do not admit a trivial solution to the $(\vec{F}, \vec{H}, \vec{K}, Q_1, Q_2, Q_T)$-Uber problem. Let q be the maximal degree in which a variable appears in $\vec{F}, \vec{H}, \vec{K}$. Then, for any algebraic algorithm A for the $(\vec{F}, \vec{H}, \vec{K}, Q_1, Q_2, Q_T)$-Uber problem in $\mathcal{G}$, there exists an algorithm B for the q-DLOG problem in one of the source groups, whose running time and success probability are polynomially-related to those of A.*

Theorem 1.3 strengthens the result by Bauer, Fuchsbauer and Loss, since the total degree of a polynomial is always at least the degree of each variable in it, and in many typical cases it is indeed strictly greater. For example, consider again our toy example from before: Given $g, g^{x_1}, g^{x_2}, g^{x_3}, g^{x_1 x_3}$ and $g^{x_2 x_3}$, compute $g^{x_1 x_2 x_3}$. Since the polynomials defining the input elements of this problem are all multilinear,[7] Theorem 1.3 implies that within the AGM, its hardness is implied by the hardness of the discrete logarithm problem. Recall that the result of Bauer et al. bases the hardness of the aforesaid problem only on the hardness of the seemingly easier 2-DLOG problem.

Application: Group Key Exchange. The toy-problem example might seem contrived at first sight, but it is actually a special case of the Group Computational Diffie-Hellman (G-CDH) problem, which underlies the highly-influential group key-exchange protocols of Bresson, Chevassut, Pointcheval and Quisquater [BCP+01b, BCP01a, BCP07]. This problem is parameterized by an integer n (which in group key-exchange applications represents the number of users in the group) and a collection Γ of subsets of $\{1, \ldots, n\}$. The adversary is given a generator g of the group, alongside the group elements $\left\{g^{\prod_{i \in \mathcal{S}} x_i}\right\}_{\mathcal{S} \in \Gamma}$ for uniformly-chosen $x_1, \ldots, x_n$, and is asked to compute $g^{x_1 \cdots x_n}$. Typically, in group key-exchange protocols Γ includes at least one subset of size $n-1$. In such cases, Theorem 1.3 reduces the hardness of the (n, Γ)-G-CDH problem to the hardness of the discrete logarithm problem, whereas the previous bound of Bauer et al. reduces it to the hardness of the $(n-1)$-DLOG problem, where n may be a very large integer and perhaps not even a-priori bounded.

The decisional Uber problem in bilinear groups. As our second contribution in bilinear groups, we extend Theorem 1.3 to the decisional setting. Within the decisional algebraic group model (DAGM) of Rotem and Segev [RS20a] which we extend to accommodate analysis in asymmetric bilinear groups, we prove that the hardness of the *decisional* Uber problem in bilinear groups is implied by the hardness of the q-DLOG problem. Again, the parameter q in our results is the maximal degree in which some variable appears in polynomials defining the problem.

Theorem 1.4 (informal)**.** *Let $\mathcal{G}$ be a bilinear group, let $\vec{F}, \vec{H}, \vec{K}$ be vectors of m-variate polynomials and let Q_T be an m-variate polynomial which does not admit a trivial solution to the $(\vec{F}, \vec{H}, \vec{K}, Q_T)$-Uber problem. Let q be the maximal degree in which a variable appears in $\vec{F}, \vec{H}, \vec{K}, Q_T$. Then, for any algorithm A for the decisional $(\vec{F}, \vec{H}, \vec{K}, Q_T)$-Uber problem in $\mathcal{G}$, there exists an algorithm B for the q-DLOG problem in one of the source groups, whose running time and success probability that are polynomially-related to those of A.*

The parameters $\vec{F}, \vec{H}, \vec{K}, Q_T$ play a similar role in the decisional setting to their role in the computational setting; for a formal definition of the decisional Uber problem, see Appendix B.1. Theorem 1.4 improves upon a result of Rotem and Segev [RS20a], who showed a similar result, but in their work the parameter q is strictly greater than the maximal total degree of the polynomials defining the problem.

[7] These are the polynomials $X_1, X_2, X_3, X_1 X_3$ and $X_2 X_3$.

1.2 Additional Related Work

Bauer, Fuchsbauer and Loss [BFL20] also considered additional variants of the (computational) Uber problem in bilinear groups. Concretely, they proved that their result extends to: The flexible Uber problem, in which the adversary can choose the target polynomials Q_1, Q_2, Q_T; the Uber problem for rational functions, in which the polynomials $\vec{F}, \vec{H}, \vec{K}, Q_1, Q_2$ and Q_T are replaced by rational functions; the Uber problem with decisional oracles, in which the adversary is given access to an oracle for checking whether tuples of group elements satisfy a polynomial relation in the exponent; and the flexible "GeGenUber" problem in which the adversary may choose the generators with respect to which the problem is defined. It seems that the techniques used by Bauer et al. to extend their results to all of the aforesaid variants can be used essentially as is in order to extend our results (Theorems 1.3 and 1.4) to accommodate these variants as well, but we leave this task to future work. We refer the reader to [BFL20] for a formal definition of these variants of the Uber problem and for the techniques used by Bauer et al. in order to extend their result to these variants.

The concurrent work of van Baarsen and Stevens [vBS21] also considered reductions supporting the sequentiality of repeated squaring in hidden-order groups. They propose a strengthening of the strong algebraic group model and within it, they reduce the task of speeding up repeated squaring to that of finding a multiple of the group's order. It seems that the results of van Baarsen and Stevens are incomparable to our results in hidden-order groups. On the one hand, we consider the more general fine-grained Uber problem, which we put forth, while they only consider the repeated squaring problem. Additionally, the model of van Baarsen and Stevens seems more restrictive for general hidden-order groups. It assumes that either every algorithm receives as input a set of generators or the ability to uniformly sample group elements (as discussed in their paper, the two assumptions are essentially equivalent). This is essentially the case for $\mathbb{Z}_N^*$, but in general groups, this poses an additional assumption. In contrast, our reduction from the RSA problem is algebraic and does not require the ability to publicly sample uniformly-random group elements. It also works if the group element x specifying the fine-grained Uber problem instance is sampled uniformly by the challenger in a private-coin manner, or if it comes from a non-uniform distribution. In the latter case, we solve that RSA problem for the same distribution. Finally, the reduction of van Baarsen and Stevens loses a factor of n in running time, where n is an upper bound on the number of generators of the group. Potentially, n can be as large as logarithmic in the order of the group. In comparison, our reductions are essentially tight. On the other hand, the reduction of van Baarsen and Steve is from the problem of finding the group's order, which is harder than the RSA problem we consider in general groups. In particular, their result establishes the equivalence, within the strong AGM (which is equivalent to their strengthened model in $\mathbb{Z}_N^*$, where sampling is easy), of speeding up repeated squaring in RSA groups and factoring the RSA modulus, for general bi-prime moduli (whereas we only establish this equivalence when the modulus is the product of two safe primes).

Rotem and Segev [RS20b] put forth the notion of generic-ring delay functions within the generic ring model of Aggarwal and Maurer [AM09], and the notion of a *sequentiality depth* of such functions. Informally, they proved that in the ring $\mathbb{Z}_N$ where N is an RSA modulus, generically computing a generic-ring delay function in a number of steps which is less than its sequentiality depth is equivalent to factoring the modulus N. In particular, they showed that this implies that within the generic ring model, computing the repeated squaring function [RSW96] in $\mathbb{Z}_N$ with respect to delay parameter T in less than T sequential steps is equivalent to factoring. Their results are incomparable to ours for several reasons. First, the generic ring model is incomparable to the strong AGM: On the one hand, the generic ring model withholds the elements' representation from the adversary (which the strong AGM does not do); but on the other hand, it allows the adversary to apply all of the ring operation onto pairs of ring elements, whereas the strong AGM requires that the adversary explains how its

output was computed solely using the group operation. Secondly, Rotem and Segev only consider RSA groups (within the ring $\mathbb{Z}_N$), whereas Theorem 1.1 may be applied in any group. Finally, when considering the specific case of RSA groups, our result (Theorem 1.2) is restricted to RSA groups with respect to a modulus N which is the product of two safe primes, whereas the result of Rotem and Segev is not.

1.3 Overview of Our Contributions

In this section we provide an informal overview of the main technical ideas underlying our contributions. We start by presenting the techniques we use to derive Theorems 1.1 and 1.2 in hidden-order groups, and then move on to describe our techniques in bilinear groups for deriving Theorem 1.3. For brevity, we do not discuss here how we extend Theorem 1.3 to the decisional setting to obtain Theorem 1.4, and the reader is referred to Appendices A and B for details.

1.3.1 Our Reductions in Hidden-Order Groups

For simplicity of presentation in this informal overview, when presenting our reduction from the factoring problem to the fine-grained Uber problem in RSA group (Theorem 1.2), and our reduction from the RSA problem to the fine-grained Uber problem in general groups (Theorem 1.1), we restrict our attention to the problem of speeding-up the repeated squaring function of Rivest, Shamir and Wagner [RSW96]. The reader is referred to Section 4 for a formal definition of the fine-grained Uber problem, and for our theorem statements and reductions in their full generality (applying to all problems within the fine-grained Uber family, and not just to speeding-up repeated squaring).

The strong AGM. We prove our results for the fine-grained Uber problem in hidden-order groups in the strong algebraic group model (the SAGM) put forth by Katz, Loss and Xu [KLX20]. Informally speaking, the SAGM strengthens the AGM, by requiring that whenever a strongly-algebraic algorithm A outputs a group element y, it outputs alongside it not only a representation of it in the basis of the input group elements, but also the entire sequence of group operations used to derive this representation. An important feature of this model, is that the length of this sequence is dominated by the running time of the algorithm. Hence, if we view the input elements to a strongly-algebraic algorithm A as polynomials of degree at most d in some underlying indeterminates, then the output y can be viewed as a polynomial of degree at most d^{2^t} in these indeterminates, where t is the running time of A. This is the case since each group operation at most doubles the degree of the highest degree polynomial in the computation up to it. Note that this observation remains true even if A runs in time t on many processors that may perform group operations in parallel. See Section 3.2 for a formal definition of the model.

The reduction of Katz, Loss and Xu in QR_N. Recall the reduction of Katz et al. from the factoring problem to the problem of speeding-up the repeated squaring. They focused on the group QR_N of quadratic residues modulo N, for a modulus N which is the product of two safe primes; that is $N = (2p+1)\cdot(2q+1)$, where $p, q, 2p+1$ and $2q+1$ are all primes. Let $T \in \mathbb{N}$ and consider a strongly-algebraic algorithm A, that given a uniformly random group element $x \leftarrow QR_N$ computes x^{2^T} in time $t < T$. The reduction samples such an element x and invokes A on it. Since A is strongly-algebraic, it produces alongside its output y an integer $\alpha \leq 2^t$ such that $y = x^\alpha$. Whenever A succeeds in computing x^{2^T}, it means that $x^\alpha = x^{2^T}$, or equivalently, $x^{2^T-\alpha} = 1$ (all equalities are in the group QR_N). Since $\alpha \leq 2^t < 2^T$, this implies that $\omega = 2^T - \alpha$ is a non-zero multiple of the order of x in QR_N. The analysis of Katz et al. proceeds by observing that when N is the product

of two safe primes, then almost all elements in the group are generators, and that the order of the group is $\varphi(N)/4 = p \cdot q$, where $\varphi(\cdot)$ is Euler's totient function. Thus, if A succeeds, then 4ω is almost surely a multiple of $\varphi(N)$, and the reduction is completed by invoking a well-known algorithm for factoring N given a multiple of $\varphi(N)$ (e.g., [KL14, Theorem 8.50]).

Our reduction in RSA groups (Theorem 1.2). Unlike in the group QR_N, when moving to consider the RSA group $\mathbb{Z}_N^*$, the group is no longer cyclic and hence a random element in the group is never a generator. However, our generalization of the result of Katz et al. to the RSA group $\mathbb{Z}_N^*$ is based on the observation that their reduction actually does not rely on the fact that the sampled x is a generator of QR_N. Instead, it only uses the fact that with overwhelming probability over the choice of x, its order satisfies a certain relation with $\varphi(N)$. We use this observation to prove that the reduction of Katz et al. can be applied not only in QR_N when N is the product of two safe primes, but also in $\mathbb{Z}_N^*$ when N is of this form. Concretely, denoting $N = (2p+1) \cdot (2q+1)$, we use the isomorphism of $\mathbb{Z}_N^*$ to the product group $\mathbb{Z}_2^2 \times \mathbb{Z}_p \times \mathbb{Z}_q$ in order to argue that almost all elements in $\mathbb{Z}_N^*$ have order either $p \cdot q = \varphi(N)/4$ or order $2p \cdot q = \varphi(N)/2$. Therefore, it is still the case that that whenever A succeeds in computing x^{2^T}, it must be that $4\omega = 4(2^T - \alpha)$ is a multiple of $\varphi(N)$ and the correctness of the reduction follows.

Our reduction in general groups (Theorem 1.1). Let $\mathbb{G}$ be an abelian group. The goal of our reduction is to solve the RSA problem in $\mathbb{G}$, where the problem is parameterized by an integer e which is coprime to the order of $\mathbb{G}$. That is, given a uniformly random $u \in \mathbb{G}$, we need to find a group element $w \in \mathbb{G}$ such that $w^e = u$ (meaning, w is the eth root of u). Let $T \in \mathbb{N}$ and consider a strongly-algebraic algorithm A, that given a uniformly random group element $x \leftarrow \mathbb{G}$ computes x^{2^T} in time $t < T$. Our reduction starts by invoking A on input u (the input to the RSA problem). As before, since A is strongly-algebraic, it produces alongside its output y an integer $\alpha \leq 2^t$ such that $y = u^\alpha$. Following a similar argument as in the previous reduction, if y is indeed equal to u^{2^T}, then $\omega = 2^T - \alpha$ is a non-zero multiple of the order of u in $\mathbb{G}$. The new idea underlying our reduction from the RSA problem, is that we can use this information about the order of u to find its eth root. Suppose that we could find an inverse d of e modulo ω; i.e., an integer d satisfying $ed = n \cdot \omega + 1$ for some integer n. Then, we would be done, since u^d would be the eth root of u:

$$\left(u^d\right)^e = u^{ed} = u^{n \cdot \omega + 1} = (u^\omega)^n \cdot u = 1 \cdot u = u, \tag{1.1}$$

where $u^\omega = 1$ because ω is a multiple of u's order. The problem is that e might not have an inverse modulo ω, as the two integers might not be relatively prime. To remedy this situation, we factor out from ω any common divisors that it shares with e, by computing $\omega' = \omega / \mathsf{gcd}(\omega, e)$. On the one hand, we are now guaranteed that ω' and e are coprime, and we can find an inverse d' of e modulo ω'. On the other hand, the key observation is that ω' must still be a multiple of u's order in $\mathbb{G}$. This is because e is coprime to the order of $\mathbb{G}$ and thus, by Lagrange's theorem, it is also coprime to the order of u. This means that if we write $\omega = \mathsf{order}(u) \cdot c$ for some integer c, then

$$\omega' = \frac{\omega}{\mathsf{gcd}(\omega, e)} = \frac{\mathsf{order}(u) \cdot c}{\mathsf{gcd}(\mathsf{order}(u) \cdot c, e)} = \mathsf{order}(u) \cdot \frac{c}{\mathsf{gcd}(c, e)}$$

and ω' is indeed a multiple of $\mathsf{order}(u)$. Hence, our reduction may simply output $u^{d'}$, and the same analysis from Eq. (1.1) still applies, replacing ω and d with ω' and d' respectively.

1.3.2 Our Reduction in Bilinear Groups

For simplicity of presentation, we focus here on the case of symmetric bilinear groups. In this setting, we consider a single source group $\mathbb{G}$ of prime order $p \in \mathbb{N}$ that is equipped with a bilinear map e, mapping pairs of elements in $\mathbb{G}^2$ to elements in a target group $\mathbb{G}_T$. We also restrict our attention to a simple problem within the Uber family, referred to below as the (f,h)-Uber problem, as the reduction in this case already captures the gist of the techniques used in our proof of Theorem 1.3. In the (f,h)-Uber problem the adversary is given g and $g^{f(\vec{x})}$ and is required to compute $g_T^{h(\vec{x})}$, where g is a generator of $\mathbb{G}$, $g_T = e(g,g)$ is a generator of $\mathbb{G}_T$, f and h are polynomials parameterizing the problem, and $\vec{x} = (x_1, \ldots, x_m)$ is a m-tuple of elements in $\mathbb{Z}_p$ chosen independently and uniformly at random. We assume that there are no integers $\alpha, \beta, \gamma \in \mathbb{Z}_p$ such that $h = \alpha + \beta \cdot f + \gamma \cdot f^2$ over $\mathbb{Z}_p$, as otherwise the problem is trivial to solve and we cannot hope to reduce the q-DLOG problem (or any other problem which we believe to be hard) to it.[8] We start by recalling the reduction of Bauer et al. [BFL20] and then move to describe our reduction and how it improves upon it.

The reduction of Bauer, Fucshbauer and Loss. Let A be an algebraic algorithm for the (f,h)-Uber problem. The reduction receives as input $g, g^x, \ldots, g^{x^q}$ for a uniformly sampled $x \leftarrow \mathbb{Z}_p$, and its goal is to find x. The idea of Bauer et al. was to have the reduction "plant" m independent randomized versions of the secret exponent x as the m secret exponents in a random instance of the (f,h)-Uber problem and then invoke A on this instance. This is done by sampling $\alpha_i, \beta_i \leftarrow \mathbb{Z}_p$ for each $i \in [m]$ and invoking A on input $(g, g^{f(\vec{x'})})$ where $\vec{x'} = (\alpha_1 \cdot x + \beta_1, \ldots, \alpha_m \cdot x + \beta_m)$. Since A is an algebraic algorithm, it provides alongside its output $y \in \mathbb{G}_T$ three integers $\alpha, \beta, \gamma \in \mathbb{Z}_p$ such that $y = e(g,g)^\alpha \cdot e(g, g^{f(\vec{x'})})^\beta \cdot e(g^{f(\vec{x'})}, g^{f(\vec{x'})})^\gamma$. Whenever A succeeds, it means that $y = g_T^{h(\vec{x'})}$, and hence, since g_T is a generator of $\mathbb{G}_T$, we obtain that

$$h(\vec{x'}) = \alpha + \beta \cdot f(\vec{x'}) + \gamma \cdot \left(f(\vec{x'})\right)^2. \tag{1.2}$$

Observe that Eq. (1.2) is a univariate equation over the finite field $\mathbb{F}_p$. Roughly speaking, the non-triviality of the (f,h)-Uber problem guarantees that with overwhelming probability, Eq. (1.2) is not the trivial equation. Therefore, the reduction simply uses any efficient polynomial factorization algorithm (e.g., Berlekamp's algorithm [Ber70, Rab80]) to find all solutions to Eq. (1.2) and then checks each of them to see if it is the secret exponent x. Note that in order to compute $g^{f(\vec{x'})}$ given as input to A, the reduction needs access to $g, g^x, \ldots, g^{x^q}$ for q that is the *total* degree of f.

Our improved reduction (Theorem 1.3). In order to reduce the parameter q needed by the reduction, our idea is to plant the secret exponent x in place of just one of the exponents $x_1, \ldots, x_m$ in a random instance of the (f,h)-Uber problem, and sample the rest of the exponents uniformly at random. Concretely, our reduction samples a random index $i^* \leftarrow [m]$ and plants x instead of x_{i^*}: It samples $m-1$ additional values $x_1, \ldots, x_{i^*-1}, x_{i^*+1}, \ldots, x_m \leftarrow \mathbb{Z}_p$ and invokes A on input $(g, g^{f(\vec{x''})})$ where $\vec{x''} = (x_1, \ldots, x_{i^*-1}, x, x_{i^*+1}, \ldots, x_m)$. Observe that indeed, in order to compute the group element $g^{f(\vec{x''})}$ given as input to A, all the reduction needs is access to $g, g^x, \ldots, g^{x^q}$ for a parameter q which is at least the degree of x_{i^*} in f. In particular, the reduction can be efficiently implemented as long as q is at least the maximal degree in which some variable appears in f. As before, since A is algebraic, it outputs alongside its output $y \in \mathbb{G}_T$ three integers $\alpha, \beta, \gamma \in \mathbb{Z}_p$ such

[8] If such integers existed, the adversary could simply compute and output $e(g,g)^\alpha \cdot e(g, g^{f(\vec{x})})^\beta \cdot e(g^{f(\vec{x})}, g^{f(\vec{x})})^\gamma$.

that $y = e(g,g)^{\alpha} \cdot e(g, g^{f(\vec{x''})})^{\beta} \cdot e(g^{f(\vec{x''})}, g^{f(\vec{x''})})^{\gamma}$, and whenever $y = g_T^{h(\vec{x''})}$, it holds that

$$h(\vec{x''}) = \alpha + \beta \cdot f(\vec{x''}) + \gamma \cdot \left(f(\vec{x''})\right)^2 . \tag{1.3}$$

Alas, we can no longer simply solve Eq. (1.3) for x. The problem is that for our choice of $x_1, \ldots, x_{i^*-1}, x_{i^*+1}, \ldots, x_m$, Eq. (1.3) might be a trivial equation, yielding no information about x. Worse still, this situation may arise with high probability over the choice of i^* and of $x_1, \ldots, x_{i^*-1}$, $x_{i^*+1}, \ldots, x_m$. So instead, our reduction uses the following observation by Rotem and Segev [RS20a], which in turn generalizes the previous work of Fuchsbauer, Kiltz and Loss [FKL18]. For a non-zero polynomial $\ell(X_1, \ldots, X_m)$ in the indeterminates $X_1, \ldots, X_m$, we define a cascade of multivariate polynomials recursively: We set $\ell_1 = \ell$; and for every $i \in \{2, \ldots, m\}$, we let $\ell_i(X_i, \ldots, X_m)$ be the first non-zero coefficient of $\ell_{i-1}(X_{i-1}, \ldots, X_m)$, when ℓ_{i-1} is written as a univariate polynomial in the indeterminate X_{i-1} with coefficients which are polynomials in $X_i, \ldots, X_m$. Rotem and Segev proved that for every vector $\vec{z} = (z_1, \ldots, z_m) \in \mathbb{Z}_p^m$ such that $\ell(\vec{z}) = 0$, there exists $t^* \in [m]$ such that: The univariate polynomial $v(X) = \ell_{t^*}(X, z_{t^*+1}, \ldots, z_m)$ is not the zero polynomial and additionally $v(z_{t^*}) = 0$.[9] Using this observation, our reduction considers the m-variate polynomial $\ell(\vec{X}) = \alpha + \beta \cdot f(\vec{X}) + \gamma \cdot \left(f(\vec{X})\right)^2 - h(X)$, and computes the polynomial $\ell_{i^*}(X_{i^*+1}, \ldots, X_m)$ from it as described by the above recursive procedure, where i^* is the index sampled by the reduction when preparing the (f,h)-Uber instance to A. It then factors the univariate polynomial $v(X) = \ell_{i^*}(X, x_{i^*+1}, \ldots, x_m)$ to find all of its roots, where $x_{i^*+1}, \ldots, x_m$ are the random $\mathbb{Z}_p$ values chosen by the reduction for generating the (f,h)-Uber instance. As for the success probability of the reduction, whenever A succeeds in computing the $g_T^{h(\vec{x''})}$, it holds that $\vec{x''}$ is a root of the m-variate polynomial ℓ. By the observation of Rotem and Segev, this means that there is an index $t^* \in [m]$ such that if the index i^* guessed by the reduction matches it, then $v(X) = \ell_{i^*}(X, x_{i^*+1}, \ldots, x_m)$ is not the zero polynomial, and x is a root of it. Hence, if A succeeds and $i^* = t^*$, then the reduction will successfully retrieve the secret exponent x of the q-DLOG problem.

1.4 Paper Organization

The remainder of this paper is organized as follows. First, in Section 2 we present the basic notation and algebraic background that are used throughout the paper. In Section 3 we present the algebraic group model and its extension to bilinear groups and to the strong AGM. In Section 4 we present our fine-grained Uber problem in hidden-order groups, and present and prove our security reductions for it; and in Section 5 we present and prove our security reduction for the Uber problem in bilinear groups. Our extension of the decisional AGM to asymmetric bilinear groups and our security reduction for the decisional Uber problem within this extension, can be found in Appendices A and B, respectively.

2 Preliminaries

In this section we briefly review the basic notions and definitions that are used in this work. For a distribution X we denote by $x \leftarrow X$ the process of sampling a value x from the distribution X. Similarly, for a set $\mathcal{X}$ we denote by $x \leftarrow \mathcal{X}$ the process of sampling a value x from the uniform distribution over $\mathcal{X}$. For an integer $n \in \mathbb{N}$, we use the notation $[n]$ to denote the set $\{1, \ldots, n\}$.

[9] Fuchsbauer, Kiltz and Loss [FKL18] essentially observed that this holds for bi-linear polynomials to reduce the hardness of the Computational Diffie-Hellman (CDH) problem to that of the discrete log problem within the AGM.

Group notation. For a group $\mathbb{G}$ we denote by $\mathsf{order}(\mathbb{G})$ the order of the group. As all groups considered in this paper are finite, this is simply the number of elements in the group. We will assume (often without noting it explicitily) that in all groups considered in this paper, the group operation can be implemented in time polylogarithmic in $\mathsf{order}(\mathbb{G})$. We also abuse notation and for a group element $\mathbf{X} \in \mathbb{G}$, we denote by $\mathsf{order}(\mathbf{X})$ the order of $\mathbf{X}$ in $\mathbb{G}$; that is, $\mathsf{order}(\mathbf{X})$ is the minimal integer m such that $\mathbf{X}^m = 1_{\mathbb{G}}$ where $1_{\mathbb{G}}$ is the identity of the group.

2.1 Bilinear Groups

In considering bilinear groups we follow the notation [BFL20]. A description of a bilinear group $\mathcal{G}$ consists of a tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e, \phi, \psi, p)$ such that

- $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are all cyclic groups of prime order $p \in \mathbb{N}$.
- g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$.
- e is a non-degenerate bilinear map from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$: For every $g \in \mathbb{G}_1, h \in \mathbb{G}_2$ and $x, y \in \mathbb{Z}_p$, it holds that $e(g^x, h^y) = e(g, h)^{xy}$. Moreover, if g generates $\mathbb{G}_1$ and h generates $\mathbb{G}_2$, then $e(g, h)$ generates $\mathbb{G}_T$.
- ϕ is an isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$, and ψ is an isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$.

We always assume that the group operations in $\mathbb{G}_1, \mathbb{G}_2$ and in $\mathbb{G}_T$ are computable in time polylogarithmic in p, and so is the bilinear map e. As is standard, we differentiate between three types of bilinear groups according to whether the isomorphisms between the two source groups are efficiently-computable (i.e., in time polylogarithmic in p) or not:

- **Type 1:** In this case, both $\phi : \mathbb{G}_1 \to \mathbb{G}_2$ and $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ are efficiently-computable. Note that this type includes in particular the case of *symmetric* bilinear groups, in which $\mathbb{G}_1 = \mathbb{G}_2$.
- **Type 2:** The isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ is efficiently-computable, but it is assumed that there is no efficiently computable isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$. In this case $\phi = \perp$ in the description $\mathcal{G}$.
- **Type 3:** It is assumed that there are no efficiently computable isomorphisms from $\mathbb{G}_1$ to $\mathbb{G}_2$ and vice versa. In this case $\phi = \psi = \perp$ in the description $\mathcal{G}$.

In actual instantiation of cryptographic primitives that rely on cyclic groups, a description $\mathcal{G}$ of a bilinear group is usually generated via a group-generation algorithm $\mathsf{GroupGen}(1^\lambda)$, where $\lambda \in \mathbb{N}$ is the security parameter that determines the bit-length of the prime p. However, we will abstract this fact away in the paper, since our reductions hold even when underlying group is fixed. We will only assume that in cases where we consider groups of type 1 or 2 it is the case that $g_1 = \psi(g_2)$. Throughout the paper, we will also use the notation $g_T = e(g_1, g_2)$.

2.2 RSA Groups and Factoring

The description of the RSA group $\mathbb{Z}_N^*$ for a bi-prime modulus N is comprised solely of the modulus N. We will use the following formalization in order to reason about ensembles of RSA moduli and the hardness of finding their factorizations. Let ModGen be a probabilistic polynomial-time algorithm, which takes as input the security parameter $\lambda \in \mathbb{N}$, and outputs a bi-prime modulus $N = p \cdot q$.

Definition 2.1. The factoring assumption holds with respect to modulus generation algorithm ModGen if for every probabilistic polynomial time algorithm A, there exists a negligible function $\nu(\cdot)$ such that

$$\Pr\left[\begin{array}{l|l} p' \cdot q' = N & N \leftarrow \mathsf{ModGen}(1^\lambda) \\ p', q' \in \{2, \ldots, N-1\} & (p', q') \leftarrow \mathsf{A}(N) \end{array}\right] \leq \nu(\lambda),$$

for all sufficiently large $\lambda \in \mathbb{N}$.

Our reductions will work per modulus, and hence we will abstract away the generation of the modulus via ModGen. For a bi-prime integer N and an algorithm A we will use the notation

$$\mathbf{Adv}^{\mathbf{Factor}_N}_{\mathsf{A}} \stackrel{\mathsf{def}}{=} \Pr\left[\begin{array}{l|l} p' \cdot q' = N & (p', q') \leftarrow \mathsf{A}(N) \\ p', q' \in \{2, \ldots, N-1\} & \end{array}\right].$$

A useful lemma. The order of an RSA group $\mathbb{Z}_N^*$ is $\varphi(N)$ where $\varphi(\cdot)$ is Euler's totient function. Lemma 2.2 formalizes a well-known result, stating that N can be efficiently factored given any positive multiple of $\varphi(N)$ (see for example [KL14, Theorem 8.50]).

Lemma 2.2. *There exists an algorithm F such that for every primes p, q and $N = p \cdot q$, and for every $\alpha \in \mathbb{Z}^+$, it holds that when invoked on input $(N, \alpha \cdot \varphi(N))$, F runs in time $O(\log \alpha \cdot \mathsf{poly}(\log N))$ and outputs the factorization of N with probability at least $1/2$.*

Using safe primes. We will focus on the case in which the RSA modulus N is the product of two *safe* primes. That is, $N = p' \cdot q'$, such that p' and q' are primes and there exist primes p and q for which $p' = 2p + 1$ and $q' = 2q + 1$. In this case, the order of the RSA group $\mathbb{Z}_N^*$ is $\varphi(N) = 4 \cdot p \cdot q$. Looking ahead, this fact imposes a relatively simple subgroup structure on the group $\mathbb{Z}_N^*$; a fact that will prove useful in Section 4.3.

3 The Algebraic Group Model

In this Section we define the algebraic group model. We begin with the simplest definition of Fuchsbauer, Kiltz and Loss [FKL18], and then move on to consider a generalization of the model to bilinear groups [MTT19, BFL20], and a strengthening of the model which allows one to reason about fine-grained complexity [KLX20].

Algebraic security games. Notions of security within the algebraic-group model are formalized using "security games", following the classic framework of Bellare and Rogaway [BR06]. A game $\mathbf{G}$ is parameterized by a set *par* of public parameters, and is comprised of an adversary A interacting with a challenger via oracle access. Such a game is described by a main procedure and possibly additional oracle procedures, which describe the manner in which the challenger replies to oracle queries issued by the adversary. We denote by $\mathbf{G}_{par}$ a game $\mathbf{G}$ with public parameters *par*, and we denote by $\mathbf{G}^{\mathsf{A}}_{par}$ the output of $\mathbf{G}_{par}$ when executed with an adversary A (note that $\mathbf{G}^{\mathsf{A}}_{par}$ is a random variable defined over the randomness of both A and the challenger). We emphasize that the parameters *par* are always given to the adversary as input, and we will not note this explicitly in the games we will consider. For example, if the parameters *par* include generators g_1, g_2 of the source groups of a bilinear group, then the adversary is always given these generators as input, even when this is not noted explicitly. We denote by $\mathbf{Time}^{\mathbf{G}_{par}}_{\mathsf{A}}$ the worst-case running time of $\mathbf{G}_{par}$ when executed with an adversary A. An adversary A participating in a game $\mathbf{G}_{par}$ is said to win whenever $\mathbf{G}^{\mathsf{A}}_{par} = 1$, and the advantage of A in $\mathbf{G}_{par}$ is defined as $\mathbf{Adv}^{\mathbf{G}_{par}}_{\mathsf{A}} \stackrel{\mathsf{def}}{=} \Pr\left[\mathbf{G}^{\mathsf{A}}_{par} = 1\right]$.

All security games in this paper are *algebraic*, which means that their public parameters consist of a cryptographic group $\mathbb{G}$, and potentially additional parameters. We already covered what $\mathcal{G}$ consists of in the case of bilinear groups and RSA groups, but in Section 4.2 we will not restrict ourselves to any particular group. In these sections, as in the case of bilinear groups and RSA groups, our reductions will work per group, and hence we will not consider the group generation algorithm explicitly.

Similarly to Fuchsbauer et al. we use boldface upper-case letters (e.g., $\mathbf{Z}$) to denote group elements in algebraic games, in order to distinguish them from other variables in the game. Figure 1 exemplifies the notion of an algebraic game by describing the games associated with the RSA problem and with the q-Discrete Logarithm problem that we consider in subsequent sections.

$\underline{\mathbf{RSA}^{\mathsf{A}}_{\mathbb{G},e}}$	$\underline{q\text{-}\mathbf{DLOG}^{\mathsf{A}}_{\mathcal{G}}}$
1. $\mathbf{Y} \leftarrow \mathbb{G}$ 2. $\mathbf{X} \leftarrow \mathsf{A}(\mathbf{Y})$ 3. If $\mathbf{X}^e = \mathbf{Y}$ output 1, and otherwise output 0	1. $x \leftarrow \mathbb{Z}_p$ 2. $\mathbf{X}_i := g^{x^i}$ for all $i \in [q]$ 3. $x' \leftarrow \mathsf{A}(g, \mathbf{X}_1, \ldots, \mathbf{X}_q)$ 4. If $x' = x$ output 1, and otherwise output 0

Figure 1: Examples of algebraic games relative to an adversary A. The game $\mathbf{RSA}^{\mathsf{A}}_{\mathcal{G},e}$ (on the left) captures the RSA problem relative to a group $\mathbb{G}$ and a public exponent e. The game q-$\mathbf{DLOG}^{\mathsf{A}}_{\mathcal{G}}$ (on the right) captures the q-Discrete Logarithm problem in a cyclic group $\mathbb{G}$ whose description $\mathcal{G} = (\mathbb{G}, g, p)$ is comprised of a description of the group itself, a generator g and the group's order p.

Algebraic algorithms. Fuchsbauer et al. [FKL18] presented the following notion of *algebraic algorithms*. Roughly speaking, an algorithm A is algebraic if whenever it outputs a group element $\mathbf{Z}$, it also outputs a representation of this element in the basis comprised of all group elements A has observed (i.e., received as input or as an incoming message from the challenger) so far.

Definition 3.1 ([FKL18])**.** Let $\mathbb{G}$ be a group. An algorithm A participating in an algebraic game relative to $\mathbb{G}$ is said to be *algebraic* if whenever A outputs a group element $\mathbf{Z} \in \mathbb{G}$, it also outputs a vector $\vec{z} = (z_1, \ldots, z_k) \in \mathbb{N}^k$ such that $\mathbf{Z} = \prod_{i=1}^{k} \mathbf{X}_i^{z_i}$, where $\mathbf{X}_1, \ldots, \mathbf{X}_k$ are the group elements that A has received so far.

Observe that when working over groups with publicly-known order p (i.e., p is included in the description of the group), then we can assume without loss of generality that all integers in the vector $\vec{z}$ are in fact in $\mathbb{Z}_p$. We will sometimes implicitly make this assumption when discussing our results within the bilinear algebraic group model. This assumption is indeed without loss of generality since by basic group theory, if $\vec{z}$ satisfies the condition in Definition 3.1, then it would also satisfy it when reducing all elements of z modulo p.

3.1 The Bilinear Algebraic Group Model

Bauer, Fuchsbauer and Loss [BFL20] extended the algebraic group model to algebraic algorithms which are executed in algebraic games with respect to bilinear groups (following Mizuide, Takayasu and Takagi who considered symmetric bilinear groups [MTT19]). Informally, an algorithm in such games is said to be algebraic, if whenever it outputs a group element $\mathbf{Z}$ (in either one of the source groups or in the target group), it also outputs a representation of this element in terms of group elements which it received so far in the game, explaining how $\mathbf{Z}$ was computed via the group operation and possibly the bilinear map e.

Definition 3.2. Let $\tau \in \{1,2,3\}$, and let $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e, \phi, \psi, p)$ be a description of a bilinear group. An algorithm A participating in an algebraic game $\mathbf{G}$ with parameters $\mathcal{G}$ is said to be *algebraic* if whenever A outputs a group element $\mathbf{Z} \in \mathbb{G}_1 \cup \mathbb{G}_2 \cup \mathbb{G}_T$, it also provides an additional output as follows. Let $\mathbf{X}_1, \ldots, \mathbf{X}_\ell \in \mathbb{G}_1$, $\mathbf{Y}_1, \ldots, \mathbf{Y}_m \in \mathbb{G}_2$ and $\mathbf{W}_1, \ldots, \mathbf{W}_t \in \mathbb{G}_T$ be the group elements received by A in $\mathbf{G}$ so far. Then, A also outputs vectors $\vec{v}, \vec{w}, \vec{u}, \vec{s}, \vec{r} \in \mathbb{Z}_p^*$ and matrices $A = (a_{i,j}), B = (b_{i,j})$ and $C = (c_{i,j})$ such that:

- If $\mathbf{Z} \in \mathbb{G}_1$:
  - If $\tau \in \{1,2\}$, then $\mathbf{Z} = \prod_i \mathbf{X}_i^{v_i} \cdot \prod_i \psi(\mathbf{Y}_i)^{w_i}$.
  - If $\tau = 3$, then $\mathbf{Z} = \prod_i \mathbf{X}_i^{v_i}$.
- If $\mathbf{Z} \in \mathbb{G}_2$:
  - If $\tau = 1$, then $\mathbf{Z} = \prod_i \phi(\mathbf{X}_i)^{u_i} \cdot \prod_i \mathbf{Y}_i^{s_i}$.
  - If $\tau \in \{2,3\}$, then $\mathbf{Z} = \prod_i \mathbf{Y}_i^{s_i}$.
- If $\mathbf{Z} \in \mathbb{G}_T$:
  - If $\tau = 1$, then $\mathbf{Z} = \prod_{i,j} e(\mathbf{X}_i, \mathbf{Y}_j)^{a_{i,j}} \cdot \prod_{i,j} e(\mathbf{X}_i, \phi(\mathbf{X}_j))^{b_{i,j}} \cdot \prod_{i,j} e(\psi(\mathbf{Y}_i), \mathbf{Y}_j)^{c_{i,j}} \cdot \prod_i \mathbf{W}_i^{r_i}$.
  - If $\tau = 2$, then $\mathbf{Z} = \prod_{i,j} e(\mathbf{X}_i, \mathbf{Y}_j)^{a_{i,j}} \cdot \prod_{i,j} e(\psi(\mathbf{Y}_i), \mathbf{Y}_j)^{c_{i,j}} \cdot \prod_i \mathbf{W}_i^{r_i}$.
  - If $\tau = 3$, then $\mathbf{Z} = \prod_{i,j} e(\mathbf{X}_i, \mathbf{Y}_j)^{a_{i,j}} \cdot \prod_i \mathbf{W}_i^{r_i}$.

3.2 The Strong Algebraic Group Model

In order to reason about fine-grained computations, Katz, Loss and Xu [KLX20] introduced the strong algebraic group model. This model strengthens the standard algebraic group model by requiring that whenever a strongly-algebraic algorithm outputs a group element $\mathbf{Z}$, it outputs alongside it a complete account of how this element was algebraically computed from the input elements. The definition also accounts for the possibility of algorithms with parallel processors in order to reason about the *sequentiality* of fine-grained computations.

Definition 3.3. Let $\mathbb{G}$ be a group and let $p \in \mathbb{N}$. An algorithm A participating in an algebraic game relative to $\mathbb{G}$ is said to be *strongly algebraic* with parallelism p if whenever A outputs a group element $\mathbf{Z} \in \mathbb{G}$, it also outputs an ordered list $U = (\vec{u_1}, \ldots, \vec{u_t})$ such that the following holds:

1. For each $i \in [t]$, the vector $\vec{u_i}$ contains at most p tuples of the following form:
   (a) $(\mathbf{X}, \mathbf{X}_1, \mathbf{X}_2) \in \mathbb{G}^3$, where $\mathbf{X} = \mathbf{X}_1 \cdot \mathbf{X}_2$ and each of $\mathbf{X}_1$ and $\mathbf{X}_2$ was either given as input to A, or appeared in a tuple in $\vec{u_j}$ for some $j < i$.
   (b) $(\mathbf{X}, \mathbf{X}_1) \in \mathbb{G}^2$, where $\mathbf{X} = \mathbf{X}_1^{-1}$ and $\mathbf{X}_1$ was either given as input to A, or appeared in a tuple in $\vec{u_j}$ for some $j < i$.
2. $\vec{u_t}$ contains a tuple of the form $(\mathbf{Z}, \mathbf{X}_1, \mathbf{X}_2)$ or $(\mathbf{Z}, \mathbf{X}_1)$.

Running time in the strong algebraic group model. Following Katz, Loss and Xu, we will refer to the length of the list U outputted by a strongly-Algebraic algorithm (the integer t in Definition 3.3) as the number of sequential algebraic "steps" that it makes. We differentiate between the number of algebraic steps and any additional computation that the algorithm might make, measured in some underlying computational model. To this end, we will express the running time of such algorithms as a pair, and say that a strongly-algebraic algorithm runs in time (t_A, t_C) if it makes at most

t_A algebraic steps, and runs in time at most t_C measured in the underlying computational model. For a strongly-algebraic algorithm A participating in a game $\mathbf{G}$ with respect to a group $\mathbb{G}$, we will denote this by $(t_A, t_C) = \mathbf{Time}^{\mathbf{G}_{\mathbb{G}}}_{\mathsf{A}}$. In some of the security games that we consider, an adversary will be comprised of two algorithms – a preprocessing algorithm and an online algorithm. In such cases, for an adversary $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$, we will use the notation $(t_A, t_C) = \mathbf{Time}^{\mathbf{G}_{\mathbb{G}}}_{\mathsf{A}}$ to imply that t_A is the total number of algebraic steps that A_0 and A_1 make in the game, and t_C is the total running time of A_0 and A_1 in the underlying computational model. Since in all of these cases, the preprocessing algorithm A_0 will not receive group elements as input, t_A will actually measure the number of algebraic steps made solely by A_1.

4 A Fine-Grained Uber Assumption in Hidden-Order Groups

In this section we present our fine-grained variant of the uber assumption in groups of hidden order. We then discuss how this assumption generalizes the repeated squaring assumption of Rivest, Shamir and Wagner [RSW96], and the generalized BBS assumption put forth by Boneh and Naor [BN00]. Then, we show that within the strong algebraic group model, this general and strong assumption is in fact implied by the RSA assumption; and that in the specific case of certain RSA groups, it is implied by the hardness of factoring.

4.1 The Univariate Fine-Grained Uber Problem

The univariate fine-grained Uber problem is parameterized by a vector $\vec{u} \in \mathbb{Z}^n$ (which correspond to the input elements) and an integer $w \in \mathbb{Z}$ (which corresponds to the target element). Informally, the adversary is comprised of two algorithms: A preprocessing algorithm and an online algorithm. The preprocessing algorithm receives the public parameters (which include the group's representation) and outputs some state st. The online algorithm receives as input the state st and n group elements of the form $\mathbf{X}^{u_i}$ (for $i = 1, \ldots, n$) and her goal is to compute $\mathbf{X}^w$. The problem is formally defined in Figure 2.

$\underline{(\vec{u}, w)\textbf{-UFGUBER}^{\mathsf{A}}_{\mathbb{G}}}$

1. $\mathsf{st} \leftarrow \mathsf{A}_0$
2. $\mathbf{X} \leftarrow \mathbb{G}$
3. For $i = 1, \ldots, n$, $\mathbf{Y}_i = \mathbf{X}^{u_i}$
4. $\mathbf{Z} = \mathbf{X}^w$
5. $\mathbf{Z}' \leftarrow \mathsf{A}_1(\mathsf{st}, \mathbf{Y}_1, \ldots, \mathbf{Y}_n)$
6. If $\mathbf{Z}' = \mathbf{Z}$ output 1, and otherwise output 0

Figure 2: The game $(\vec{u}, w)$-$\mathbf{UFGUBER}^{\mathsf{A}}_{\mathbb{G}}$ which captues the univariate fine-grained uber problem. The game is defined with respect to a group $\mathbb{G}$ and a pair $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$ of algorithms, where A_0 is the preprocessing algorithm and A_1 is the online algorithm.

We emphasize that $\vec{u}$ and w are publicly known as part of the public parameters of the problem. Hence, there are choices of $\vec{u}$ and of w for which the adversary can always compute the target element from the input elements (indeed, these will be the cases of interest in this section). However, we will be interested in the exact time (or, more precisely, the number of algebraic steps) that the adversary makes in order to compute the target element. What we wish to show is that an adversary that computes the target element in less than the "trivial" time in which this can be done, can be turned into an adversary for breaking the RSA assumption in the group. So we first have to define what we mean by "trivial".

Intuitively, for integers w, p and t and a vector $\vec{u}$ of integers, we will say that the pair $(\vec{u}, w)$ is (p,t)-trivial, if it is possible to obtain w from $\vec{u}$ in depth t with parallelism p, using binary addition and unitary sign change operations. This is formally captured by Definition 4.1.

Definition 4.1. Let $\vec{u} \in \mathbb{Z}^n$ and $w \in \mathbb{Z}$ and let $p, t \in \mathbb{N}$. We say that the pair $(\vec{u}, w)$ is (p,t)-trivial if there exists an ordered list $V = (\vec{v_1}, \ldots, \vec{v_t})$ such that the following holds:

1. For each $i \in [t]$, the vector $\vec{v_i}$ contains at most p tuples of the following form:
   (a) $(a, a_1, a_2) \in \mathbb{Z}^3$, where $a = a_1 + a_2$ and each of a_1 and a_2 is either equal to u_j for some $j \in [n]$, or appeared in a tuple in $\vec{v_m}$ for some $m < i$.
   (b) $(a, a_1) \in \mathbb{Z}^2$, where $a = -a_1$ and a_1 is either equal to u_j for some $j \in [k]$, or appeared in a tuple in $\vec{v_m}$ for some $m < i$.
2. $\vec{v_t}$ contains a tuple of the form (w, a_1, a_2) or (w, a_1).

We say that the game $(\vec{u}, w)$-**UFGUBER** is (p,t)-trivial if $(\vec{u}, w)$ is (p,t)-trivial. Roughly speaking, with this terminology in mind, the univariate fine-grained uber assumption with respect to some group $\mathbb{G}$ is that no t-time algorithm with parallelism p can win in the game $(\vec{u}, w)$-**UFGUBER** if $(\vec{u}, w)$ is not (p,t)-trivial.

Relation to other fine-grained problems. In Figure 3 we define the games T-**RSW** and T-**BBS** which capture the repeated squaring problem of Rivest, Shamir and Wagner [RSW96], and a strengthening thereof considerd by Boneh and Naor [BN00], respectively. In both games, the adversary consists of a preprocessing algorithm who receives the group representation and outputs some state st, as well as an online algorithm. In T-**RSW** the online algorithm receives the state st and a uniformly sampled group element $\mathbf{X}$ and needs to compute $\mathbf{X}^{2^T}$. Observe that this is indeed a special case of $(\vec{u}, w)$-**UFGUBER**, in which $\vec{u} = (1)$ and $w = 2^T$. Further note that the T-**RSW** problem is $(1, T)$-trivial per Definition 4.1; but for every $p \in \mathbb{N}$, T-**RSW** is not (p,t)-trivial for any $t < T$.

In T-**BBS**, the online adversary receives the state st and $T+1$ group elements $\mathbf{X}, \mathbf{X}^2, \mathbf{X}^4, \ldots, \mathbf{X}^{2^{2^i}}$, $\ldots, \mathbf{X}^{2^{2^T}}$ for a uniformly chosen group element $\mathbf{X}$, and her goal is to compute $\mathbf{X}^{2^{2^{T+1}}}$.[10] Observe that this is indeed a special case of $(\vec{u}, w)$-**UFGUBER**, in which $\vec{u} = (1, 2, 4, \ldots, 2^{2^T})$ and $w = 2^{2^{T+1}}$. Additionally, observe that the T-**BBS** problem is $(1, 2^T)$-trivial per Definition 4.1; but for every $p \in \mathbb{N}$, T-**BBS** is not (p,t)-trivial for any $t < 2^T$.

4.2 From RSA to Univariate Fine-Grained Uber in General Groups

In this section we prove that within the strong algebraic group model, the hardness of the univariate fine-grained uber problem (for non-trivial parameters) with respect to a group $\mathbb{G}$ is implied by the hardness of the RSA problem in this group (recall Figure 1). That is, given any strongly-algebraic algorithm A which makes t algebraic steps with parallelism p and wins in the game $(\vec{u}, w)$-**UFGUBER** (for any $(\vec{u}, w)$ which is is not (p,t)-trivial) in a group $\mathbb{G}$, we construct an algorithm which runs in roughly the same time and wins in the game $\mathbf{RSA}_{\mathbb{G},e}$ (for any exponent e which is co-prime to the order of the group).

[10] Boneh and Naor actually considered a decisional variant of this problem, in which the adversary needs to distinguish between $\mathbf{X}^{2^{2^{T+1}}}$ and the square of a uniformly-random element in the group. As mentioned in Section 1, in practice, such indistinguishability-based hardness can be achieved heuristically from the computational variant that we consider, by applying a cryptographic hash function onto $\mathbf{X}^{2^{2^{T+1}}}$ and squaring the result [BR93].

$\underline{T\text{-}\mathbf{RSW}^{\mathsf{A}}_{\mathbb{G}}}$	$\underline{T\text{-}\mathbf{BBS}^{\mathsf{A}}_{\mathbb{G}}}$
1. $\mathsf{st} \leftarrow \mathsf{A}_0$ 2. $\mathbf{X} \leftarrow \mathbb{G}$ 3. $\mathbf{W}' \leftarrow \mathsf{A}_1(\mathsf{st}, \mathbf{X})$ 4. $\mathbf{W} := \mathbf{X}^{2^T}$ 5. If $\mathbf{W}' = \mathbf{W}$ output 1, and otherwise output 0	1. $\mathsf{st} \leftarrow \mathsf{A}_0$ 2. $\mathbf{X} \leftarrow \mathbb{G}$ 3. $\mathbf{W}' \leftarrow \mathsf{A}_1(\mathsf{st}, \mathbf{X}^2, \ldots, \mathbf{X}^{2^{2^i}}, \ldots, \mathbf{X}^{2^{2^T}})$ 4. $\mathbf{W} := \mathbf{X}^{2^{2^{T+1}}}$ 5. If $\mathbf{W}' = \mathbf{W}$ output 1, and otherwise output 0

Figure 3: The games T-$\mathbf{RSW}^{\mathsf{A}}_{\mathbb{G}}$ and T-$\mathbf{BBS}^{\mathsf{A}}_{\mathbb{G}}$ capturing the RSW problem and the BBS problem with respect to a parameter T, a group $\mathbb{G}$ and an adversary $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$.

Theorem 4.2. *Let $\mathbb{G}$ be a group, let $e < \mathsf{order}(\mathbb{G})$ be an integer co-prime to $\mathsf{order}(\mathbb{G})$, let $n, p \in \mathbb{N}$, let $\vec{u} \in \mathbb{Z}^n$ and let $w \in \mathbb{Z}$. For every pair $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$ of strongly-algebraic algorithms with parallelism p, there exists an algorithm B such that the following holds: If $(\vec{u}, w)$ is* not *(p, t_A)-trivial for $(t_A, t_C) = \mathbf{Time}^{(\vec{u},w)\text{-}\mathbf{UFGUBER}_{\mathbb{G}}}_{\mathsf{A}}$, then $\mathbf{Adv}^{\mathbf{RSA}_{\mathbb{G},e}}_{\mathsf{B}} \geq \epsilon$ and $\mathbf{Time}^{\mathbf{RSA}_{\mathbb{G},e}}_{\mathsf{B}} = t_C + t_A \cdot p \cdot \mathsf{poly}(\log(\mathsf{order}(\mathbb{G})))$, where $\epsilon = \mathbf{Adv}^{(\vec{u},w)\text{-}\mathbf{UFGUBER}_{\mathbb{G}}}_{\mathsf{A}}$.*

We stress that since A_0 does not take any group elements as input, t_A in the statement of Theorem 4.2 is the number of sequential algebraic steps taken by A_1. Before presenting the proof of Theorem 4.2, we make the following observation. Consider a strongly-algebraic algorithm A which receives as input n group elements $\mathbf{X}_1, \ldots, \mathbf{X}_n$ and outputs a group element $\mathbf{Z}$ within t_A algebraic steps. Since A is strongly algebraic, it also outputs an ordered list $V = (\vec{v_1}, \ldots, \vec{v_{t_A}})$, where each $\vec{v_i}$ is a vector of tuples as defined in Definition 3.3. The observation is that given V, one can efficiently compute integers $z_1, \ldots, z_n \in \mathbb{Z}$ such that $\mathbf{Z} = \prod_{i=1}^{n} \mathbf{X}_i^{z_i}$. This is done via the following recursive process that attributes a vector $\vec{y} \in \mathbb{Z}^n$ to every group element in every tuple in $\vec{v}_i$ for every $i \in [t_A]$:

- To every input element $\mathbf{X}_j$ we attribute the vector $\vec{y_j}$ with 1 in its j-th entry and 0 everywhere else.
- To every tuple of group elements in $\vec{v}_i$ (for $i \in [t_A]$): If the tuple is of the form $(\mathbf{Y}, \mathbf{Y}_1, \mathbf{Y}_2)$, then we attribute to $\mathbf{Y}$ the vector $\vec{y} = \vec{y_1} + \vec{y_2}$, where $\vec{y_1}$ and $\vec{y_2}$ are the vectors attributed to $\mathbf{Y}_1$ and to $\mathbf{Y}_2$, respectively. If the tuple is of the form $(\mathbf{Y}, \mathbf{Y}_1)$, then we attribute to $\mathbf{Y}$ the vector $\vec{y} = -\vec{y_1}$, where $\vec{y_1}$ is the vector attributed to $\mathbf{Y}_1$.

Since by Definition 3.3, the vector $\vec{v}_{t_A}$ must contain a tuple of the form $(\mathbf{Z}, \mathbf{Y}_1, \mathbf{Y}_2)$ or $(\mathbf{Z}, \mathbf{Y}_1)$, then we obtain a representation $\vec{z} = (z_1, \ldots, z_n)$ as required (if $\vec{v}_{t_A}$ contains more than one tuple of these forms, we pick the first such tuple in $\vec{v}_{t_A}$). We denote the above process of extracting from V the vector $\vec{z}$ for $\mathbf{Z}$ by $\vec{z} = \mathsf{Ext}(\mathbf{Z}, V)$. We now turn to the proof of Theorem 4.2.

Proof. Let $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$ be a pair of strongly-algebraic algorithms as in the statement of Theorem 4.2. Consider the algorithm B participating in $\mathbf{RSA}_{\mathbb{G},e}$.

Algorithm B

Input: A group elements $\mathbf{S}$ uniformly sampled by the challenger from $\mathbb{G}$.

1. Invoke $\mathsf{st} \leftarrow \mathsf{A}_0$
2. For $i = 1, \ldots, n$, compute $\mathbf{Y}_i = \mathbf{S}^{u_i}$.
3. Invoke $\mathsf{A}_1(\mathsf{st}, \mathbf{Y}_1, \ldots, \mathbf{Y}_n)$ to obtain a group element $\mathbf{Z}$ and an ordered list $V = (\vec{v_1}, \ldots, \vec{v_t})$, where each $\vec{v_i}$ is a vector of tuples as defined in Definition 3.3. Compute $\vec{z} = \mathsf{Ext}(\mathbf{Z}, V)$.

4. Compute $a = |w - \sum_{i=1}^{n} u_i \cdot z_i|$.
5. Compute $b = \mathsf{gcd}(a, e)$ and $a' = a/b$.
6. Compute $d = e^{-1} \mod a'$.
   [that is, d is the inverse of e modulo a'; meaning $d \cdot e = 1 \mod a'$]
7. Output $\mathbf{X} = \mathbf{S}^d$.

Let $\epsilon = \mathbf{Adv}_{\mathsf{A}}^{(\vec{u},w)\text{-}\mathbf{UFGUBER}_{\mathbb{G}}}$ and $(t_A, t_C) = \mathbf{Time}_{\mathsf{A}}^{(\vec{u},w)\text{-}\mathbf{UFGUBER}_{\mathbb{G}}}$. By definition of B, whenever A_1 succeeds in outputting $\mathbf{Z}$ which is equal to $\mathbf{S}^w$ (in the simulated instance of the $(\vec{u}, w)$-$\mathbf{UFGUBER}_{\mathbb{G}}$ problem), it holds that

$$\mathbf{S}^w = \prod_{i=1}^{n} \mathbf{Y}_i^{z_i} = \prod_{i=1}^{n} (\mathbf{S}^{u_i})^{z_i} = \mathbf{S}^{\sum_{i=1}^{n} u_i \cdot z_i}. \tag{4.1}$$

Rearranging equality (4.1), we get that

$$\mathbf{S}^{w - \sum_{i=1}^{n} u_i \cdot z_i} = 1_{\mathbb{G}}, \tag{4.2}$$

where $1_{\mathbb{G}}$ is the identity element of $\mathbb{G}$. By assumption, $(\vec{u}, w)$ is not (p, t_A)-trivial, and hence it holds that $w \neq \sum_{i=1}^{n} u_i \cdot z_i$. Assume without loss of generality that $w > \sum_{i=1}^{n} u_i \cdot z_i$ (the proof in case that $\sum_{i=1}^{n} u_i \cdot z_i > w$ is symmetric). Then, $a = w - \sum_{i=1}^{n} u_i \cdot z_i$ (where a is as computed by B in Step 4), and it holds that

$$a \neq 0 \quad \wedge \quad \mathbf{S}^a = 1_{\mathbb{G}}. \tag{4.3}$$

This means that $\mathsf{order}(\mathbf{S})$ divides a, so we can write $a = c \cdot \mathsf{order}(\mathbf{S})$ for some positive integer c. Since e is co-prime to $\mathsf{order}(\mathbb{G})$, by Lagrange's theorem, it is also co-prime to $\mathsf{order}(\mathbf{S})$. Hence, $b = \mathsf{gcd}(a, e) = \mathsf{gcd}(c, e)$, and hence

$$a' = a/b = \mathsf{order}(\mathbf{S}) \cdot c/b = \mathsf{order}(\mathbf{S}) \cdot c/\mathsf{gcd}(c, e) = c' \cdot \mathsf{order}(\mathbf{S}) \tag{4.4}$$

for some positive integer c', where b and a' are as computed by B in Step 5. Moreover, e is co-prime to a', and hence e is indeed invertible modulo a'. This means that d (as computed by B in Step 6) is well-defined, and that there exists an integer m so that $e \cdot d = m \cdot a' + 1 = m \cdot c' \cdot \mathsf{order}(\mathbf{S}) + 1$. Hence, in case A succeeds, the output $\mathbf{X} = \mathbf{S}^d$ of B indeed satisfies:

$$\mathbf{X}^e = \mathbf{S}^{d \cdot e} = \mathbf{S}^{d \cdot e \mod \mathsf{order}(\mathbf{S})} = \mathbf{S}^{m \cdot c' \cdot \mathsf{order}(\mathbf{S}) + 1 \mod \mathsf{order}(\mathbf{S})} = \mathbf{S}^1 = \mathbf{S}. \tag{4.5}$$

We showed that whenever A wins in the simulated instance of the game $(\vec{u}, w)$-$\mathbf{UFGUBER}_{\mathbb{G}}$, B wins in $\mathbf{RSA}_{\mathbb{G},e}$. Observe that B perfectly simulates the game $(\vec{u}, w)$-$\mathbf{UFGUBER}_{\mathbb{G}}$ to A, and hence $\mathbf{Adv}_{\mathsf{B}}^{\mathbf{RSA}_{\mathbb{G},e}} \geq \epsilon$, as required. ■

4.3 From Factoring to Univariate Fine-Grained Uber in RSA groups

In this section we restrict our attention to RSA groups of the form $\mathbb{Z}_N^*$ where the modulus N is the product of two safe primes $p' = 2p + 1$ and $q' = 2q + 1$ (recall our discussion in Section 2.2). Concretely, we prove that within the strong algebraic group model, the hardness of the univariate fine-grained uber problem (for non-trivial parameters) with respect to a group $\mathbb{Z}_N^*$ is implied by the hardness of factoring the modulus N. That is, given any strongly-algebraic algorithm A which makes t algebraic steps with parallelism p and wins in the game $(\vec{u}, w)$-$\mathbf{UFGUBER}$ (for any $(\vec{u}, w)$ which is is not (p, t)-trivial) in $\mathbb{Z}_N^*$, we construct an algorithm which runs in roughly the same time and factors N.

Theorem 4.3. *Let p', q', p, q be primes such that $p' = 2p+1$ and $q' = 2q+1$, and let $N = p' \cdot q'$. Let $n, k \in \mathbb{N}$, let $\vec{u} \in \mathbb{Z}^n$ and let $w \in \mathbb{Z}$. For every pair $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$ of strongly-algebraic algorithm with parallelism k, there exists an algorithm B such that the following holds: If $(\vec{u}, w)$ is* not *(k, t_A)-trivial for $(t_A, t_C) = \mathbf{Time}_{\mathsf{A}}^{(\vec{u},w)\text{-}\mathbf{UFGUBER}_N}$, then $\mathbf{Adv}_{\mathsf{B}}^{\mathbf{Factor}_N} \geq \epsilon/2 - (3p+3q)/8pq$ and $\mathbf{Time}_{\mathsf{B}}^{\mathbf{Factor}_N} = t_C + t_A \cdot k \cdot \mathsf{poly}(\log(pq))$, where $\epsilon = \mathbf{Adv}_{\mathsf{A}}^{(\vec{u},w)\text{-}\mathbf{UFGUBER}_N}$.*

The proof of Theorem 4.3 relies on Lemma 2.2 and on Lemma 4.4 which can be found below. In the latter, we prove that when N is the product $(2p+1) \cdot (2q+1)$ of two safe primes, almost all elements in $\mathbb{Z}_N^*$ have order at least $p \cdot q$.

Lemma 4.4. *Let p', q', p, q be primes such that $p' = 2p+1$ and $q' = 2q+1$, and let $N = p' \cdot q'$. Then,*

$$\Pr_{\mathbf{X} \leftarrow \mathbb{Z}_N^*} [\mathsf{order}(\mathbf{X}) < p \cdot q] = \frac{3 \cdot p + 3 \cdot q - 2}{4 \cdot p \cdot q}.$$

Proof. We first note that

$$\begin{aligned} \mathbb{Z}_N^* &\cong \mathbb{Z}_{p'}^* \times \mathbb{Z}_{q'}^* && (4.6) \\ &\cong \mathbb{Z}_{2p} \times \mathbb{Z}_{2q} && (4.7) \\ &\cong \mathbb{Z}_2 \times \mathbb{Z}_p \times \mathbb{Z}_2 \times \mathbb{Z}_q, && (4.8) \end{aligned}$$

where Eq. (4.6) and Eq. (4.8) follow from the Chinese remainder theorem, and Eq. (4.7) holds since p' and q' are prime; hence, the groups $\mathbb{Z}_{p'}^*$ and $\mathbb{Z}_{q'}^*$ are cyclic and therefore isomorphic to the additive groups $\mathbb{Z}_{p'-1}$ and $\mathbb{Z}_{q'-1}$, respectively.

So in order to count the number of elements of order $d \in \mathbb{N}$ in $\mathbb{Z}_N^*$, we can count the number of quadruples $(x_1, x_2, x_3, x_4) \in \mathbb{Z}_2 \times \mathbb{Z}_p \times \mathbb{Z}_2 \times \mathbb{Z}_q$ such that $\mathsf{LCM}(d_1, d_2, d_3, d_4) = d$, where LCM stands for the least common multiple, and for each $i \in [4]$, d_i is the order of x_i in its respective additive group. Using this analysis, an element of order p in $\mathbb{Z}_N^*$ must correspond to a quadruple of the form $(0, y, 0, 0)$ for some $y \in \mathbb{Z}_p \setminus \{0\}$. There are $p-1$ such quadruples and therefore there are $p-1$ elements of order p in $\mathbb{Z}_N^*$. Similarly an element of order $2p$ in $\mathbb{Z}_N^*$ must correspond to a quadruple of the form $(1, y, 0, 0)$ or $(0, y, 1, 0)$ for some $y \in \mathbb{Z}_p \setminus \{0\}$. There are $2p-2$ such quadruples and therefore there are $2p-2$ elements of order $2p$ in $\mathbb{Z}_N^*$. The same analysis shows that $\mathbb{Z}_N^*$ contains $q-1$ elements of order q; $2q-2$ elements of order $2q$; 3 elements of order 2; 1 element of order 1; and no elements of order $4p$ or $4q$. By Lagrange's theorem, all remaining elements of order at least $p \cdot q$.

Overall, we counted a total of $3p+3q-2$ elements of orders strictly less than $p \cdot q$, out of a total of $\varphi(N) = 4 \cdot p \cdot q$ group elements in $\mathbb{Z}_N^*$. ∎

Equipped with these two lemmata, we turn to prove Theorem 4.3. The reduction follows the two-step approach of Katz, Loss and Xu [KLX20]: First extract a multiple of $\varphi(N)$ from a successful adversary A, and then use Lemma 2.2 to factor N. In our case, however, the analysis of the first (and main) step is more involved and relies on Lemma 4.4, which Katz et al. do not require.

Proof of Theorem 4.3. Let $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$ be a pair of strongly-algebraic algorithm taking part in $(U, \vec{w})$-$\mathbf{FGUBER}_N$ as in the statement of Theorem 4.3. Consider the following algorithm B attempting to factor N.

Algorithm B

Input: The integer N.

1. Sample $x \leftarrow \mathbb{Z}_N \setminus \{0\}$.
2. If x divides N, output $\{x, N/x\}$ and terminate. Otherwise, set $\mathbf{X} = x$.
   [in case x does not divide N it is indeed an element of the group $\mathbb{Z}_N^*$ and so we switch to a boldface letter to describe it]
3. Invoke $\mathsf{st} \leftarrow \mathsf{A}_0$
4. For $i = 1, \ldots, n$, compute $\mathbf{Y}_i = \mathbf{X}^{u_i}$.
5. Invoke $\mathsf{A}(\mathsf{st}, \mathbf{Y}_1, \ldots, \mathbf{Y}_n)$ to obtain a group element $\mathbf{Z}$ and an ordered list $V = (\vec{v_1}, \ldots, \vec{v_t})$, where each $\vec{v_i}$ is a vector of tuples as defined in Definition 3.3. Compute $\vec{z} = \mathsf{Ext}(\mathbf{Z}, V)$.
6. Compute $a = 4 \cdot |w - \sum_{i=1}^{n} u_i \cdot z_i|$.
7. Invoke $\mathsf{F}(N, a)$, where F is the algorithm guaranteed by Lemma 2.2. If F succeeds in outputting a pair $\{r, s\}$, verify that $N = r \cdot s$, and if that is the case, output $\{r, s\}$. Otherwise, output $\perp$ and terminate.

Let AWin denote the event in which A_1 outputs $\mathbf{Z}$ satisfying $\mathbf{Z} = x^w$ (in the execution of $(\vec{u}, w)$-**UFGUBER**$_N$ simulated to it by B). Note that whenever x (sampled by B in Step 1) is in $\mathbb{Z}_N \setminus (\mathbb{Z}_N^* \cup \{0\})$, B factors N with probability 1. Hence,

$$\begin{aligned}
\Pr_{x \leftarrow \mathbb{Z}_N}\left[\mathbf{Factor}_N^{\mathsf{B}} = 1\right] &\geq \Pr_{\mathbf{X} \leftarrow \mathbb{Z}_N^*}\left[\mathbf{Factor}_N^{\mathsf{B}} = 1\right] \\
&\geq \Pr_{\mathbf{X} \leftarrow \mathbb{Z}_N^*}\left[\mathbf{Factor}_N^{\mathsf{B}} = 1 \,\middle|\, \begin{matrix}\mathsf{AWin} \\ \mathsf{order}(\mathbf{X}) \geq p \cdot q\end{matrix}\right] \cdot \Pr_{\mathbf{X} \leftarrow \mathbb{Z}_N^*}\left[\begin{matrix}\mathsf{AWin} \\ \mathsf{order}(\mathbf{X}) \geq p \cdot q\end{matrix}\right] \\
&\geq \frac{1}{2} \cdot \Pr_{\mathbf{X} \leftarrow \mathbb{Z}_N^*}\left[\begin{matrix}\mathsf{AWin} \\ \mathsf{order}(\mathbf{X}) \geq p \cdot q\end{matrix}\right], \qquad (4.9)
\end{aligned}$$

where all probabilities are also over the randomness of A_0, A_1 and the randomness of F. Eq. (4.9) holds since conditioned on AWin, it holds that

$$\mathbf{X}^w = \mathbf{X}^{\sum_{i=1}^{n} u_i \cdot z_i}. \qquad (4.10)$$

This in turn means that

$$\mathbf{X}^{w - \sum_{i=1}^{n} u_i \cdot z_i} = 1_{\mathbb{G}}. \qquad (4.11)$$

By assumption, $(\vec{u}, w)$ is not (k, t_A)-trivial, and hence it holds that $w \neq \sum_{i=1}^{n} u_i \cdot z_i$. Assume without loss of generality that $w > \sum_{i=1}^{n} u_i \cdot z_i$ (the proof in case that $\sum_{i=1}^{n} u_i \cdot z_i > w$ is symmetric). Then, $w - \sum_{i=1}^{n} u_i \cdot z_i$ is a multiple of the order of $\mathbf{X}$. Since we also conditioned on $\mathsf{order}(\mathbf{X}) \geq p \cdot q$, the order of $\mathbf{X}$ is either $p \cdot q$ or $2 \cdot p \cdot q$. Hence, in this case it holds that $a = 4 \cdot (w - \sum_{i=1}^{n} u_i \cdot z_i)$ (as computed by B in Step 6) is a multiple of $\mathsf{order}(\mathbb{Z}_N^*) = 4 \cdot p \cdot q$. Therefore, Lemma 2.2 implies that in this case, in Step 7 of B, the algorithm F successfully outputs the factorization of N with probability at least $1/2$, implying Eq. (4.9) above.

We are left with bounding $\Pr[\mathsf{AWin} \wedge \mathsf{order}(\mathbf{X}) \geq p \cdot q]$ from Eq. (4.9). By total probability and

the union bound, it holds that

$$\begin{aligned}\Pr_{\mathbf{X}\leftarrow\mathbb{Z}_N^*}\left[\begin{array}{c}\mathsf{AWin}\\ \mathsf{order}(\mathbf{X})\geq p\cdot q\end{array}\right] &\geq \Pr_{\mathbf{X}\leftarrow\mathbb{Z}_N^*}[\mathsf{AWin}]-\Pr_{\mathbf{X}\leftarrow\mathbb{Z}_N^*}[\mathsf{order}(\mathbf{X})<p\cdot q]\\ &\geq \epsilon-\Pr_{\mathbf{X}\leftarrow\mathbb{Z}_N^*}[\mathsf{order}(\mathbf{X})<p\cdot q] \qquad (4.12)\\ &\geq \epsilon-\frac{3\cdot p+3\cdot q-2}{4\cdot p\cdot q}, \qquad (4.13)\end{aligned}$$

where Eq. (4.12) follows from the fact that conditioned on $\mathbf{X}$ being in $\mathbb{Z}_N^*$, B perfectly simulates $(U,\vec{w})$-**FGUBER**$_N$ to A; and Eq. (4.13) holds by Lemma 4.4. This concludes the proof the theorem. ■

5 The Algebraic Hardness of the Uber Problem in Bilinear Groups

In this section we present our improved bounds for the uber assumption in the bilinear algebraic group model. We start by formally defining the uber problem in bilinear groups and then move on to present and prove our reduction from the q-DLOG problem.

5.1 The Uber Problem in Bilinear Groups

Boneh, Boyen and Goh [BBG05] introduced the Uber family of computational assumptions in bilinear groups, which was later extended by Boyen [Boy08]. We follow the more general presentation of Bauer, Fuchsbauer and Loss [BFL20]. Roughly speaking, an assumption in the Uber family is parameterized by an integer m, three vectors $\vec{F}=(F_1,\ldots,F_k),\vec{H}=(H_1,\ldots,H_k)$ and $\vec{K}=(K,\ldots,K_k)$ of m-variate polynomials over $\mathbb{Z}_p$, and target polynomials Q_1,Q_2 and Q_T. The adversary is given $g_1^{F_1(\vec{x})},\ldots,g_1^{F_k(\vec{x})},g_2^{H_1(\vec{x})},\ldots,g_2^{H_k(\vec{x})}$ and $g_T^{K_1(\vec{x})},\ldots,g_T^{K_k(\vec{x})}$ for a uniformly chosen $\vec{x}\leftarrow\mathbb{Z}_p^m$, and her goal is to compute $g_1^{Q_1(\vec{x})}$, $g_2^{Q_2(\vec{x})}$ and $g_T^{Q_T(\vec{x})}$. An assumption in the family is defined via the algebraic game $(\vec{F},\vec{H},\vec{K},Q_1,Q_2,Q_T)$-**UBER** in Figure 4.

We emphasize that throughout Section 5, even when not noting it explicitly, all polynomials are viewed as polynomials over the ring $\mathbb{Z}_p$. This means that all of their coefficients can be reduced modulo p; that polynomial arithmetic is done modulo p; and that polynomial evaluation is done modulo p. In particular, a polynomial is said to be the zero polynomial if all of its coefficients are 0 modulo p.

Note that there are choices of $(\vec{F},\vec{H},\vec{K},Q_1,Q_2,Q_T)$ for which the $(\vec{F},\vec{H},\vec{K},Q_1,Q_2,Q_T)$-**UBER** game can be easily won. If given access to $g_1^{F_1(\vec{X})},\ldots,g_1^{F_k(\vec{X})},g_2^{H_1(\vec{X})},\ldots,g_2^{H_k(\vec{X})}$ and $g_T^{K_1(\vec{X})},\ldots,g_T^{K_k(\vec{X})}$, one can obtain $g_1^{Q_1(\vec{X})}$, $g_2^{Q_2(\vec{X})}$ and $g_T^{Q_T(\vec{X})}$ through a sequence of group operations and bilinear map operations (where X_i is a indeterminate replacing x_i and $\vec{X}=(X_1,\ldots,X_m)$), then one can in particular compute the target group elements for any concrete choice of $\vec{x}$. To rule out such trivial attacks, we consider the following definitions.

Definition 5.1 captures the case in which a polynomial can be computed in the exponent of a group element (in either the target group or one of the source groups), using only the group operation and possibly the isomorphisms ϕ and ψ (if these are efficiently-computable).

Definition 5.1. Let $p\in\mathbb{N}$ be a prime, let $m,k\in\mathbb{N}$, let $\vec{F}\in(\mathbb{Z}_p[X_1,\ldots,X_m])^k$ be a tuple of polynomials, and let $Q\in\mathbb{Z}_p[X_1,\ldots,X_m]$ be a polynomial. We say that Q is *linearly dependent* on

$\underline{(\vec{F},\vec{H},\vec{K},Q_1,Q_2,Q_T)\textbf{-UBER}^{\mathsf{A}}_{\mathcal{G}}}$

1. $x_1,\ldots,x_m \leftarrow \mathbb{Z}_p$
2. $\vec{\mathbf{X}} := g_1^{\vec{F}(x_1,\ldots,x_m)}$
3. $\vec{\mathbf{Y}} := g_2^{\vec{H}(x_1,\ldots,x_m)}$
4. $\vec{\mathbf{Z}} := g_2^{\vec{K}(x_1,\ldots,x_m)}$
5. $\mathbf{W}_1 := g_1^{Q_1(x_1,\ldots,x_m)}$, $\mathbf{W}_2 := g_1^{Q_2(x_1,\ldots,x_m)}$ and $\mathbf{W}_T := g_T^{Q_T(x_1,\ldots,x_m)}$
6. $(\mathbf{W}'_1,\mathbf{W}'_2,\mathbf{W}'_T) \leftarrow \mathsf{A}(\vec{\mathbf{X}},\vec{\mathbf{Y}},\vec{\mathbf{Z}})$
7. If $(\mathbf{W}'_1,\mathbf{W}'_2,\mathbf{W}'_T) = (\mathbf{W}_1,\mathbf{W}_2,\mathbf{W}_T)$ output 1, and otherwise output 0

Figure 4: The game $(\vec{F},\vec{H},\vec{K},Q_1,Q_2,Q_T)$-$\mathbf{UBER}^{\mathsf{A}}_{\mathcal{G}}$ capturing the uber problem in bilinear groups with respect to parameters $(\vec{F},\vec{H},\vec{K},Q_1,Q_2,Q_T)$, a bilinear group $\mathcal{G} = (\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,g_1,g_2,e,\phi,\psi,p)$ and an adversary A. The notation $\vec{\mathbf{X}} := g_1^{\vec{F}(x_1,\ldots,x_m)}$ is a shorthand for $\mathbf{X}_i = g_1^{F_i(x_1,\ldots,x_m)}$ for each $i \in [k]$ and $\vec{\mathbf{X}} = (\mathbf{X}_1,\ldots,\mathbf{X}_k)$. The vectors $\vec{\mathbf{Y}}$ and $\vec{\mathbf{Z}}$ are defined analogously.

$\vec{F}$ if there exist integers $\alpha_1,\ldots,\alpha_k$ such that

$$Q = \sum_{i=1}^{k} \alpha_i \cdot F_i.$$

If Q is not linearly dependent on $\vec{F}$, then we say that Q is *linearly independent* of $\vec{F}$.

Definition 5.2 captures the case in which a polynomial can be computed in the exponent of an element in the target group, using the group operation and the bilinear map e (and possibly the isomorphisms ϕ and ψ if these are efficiently-computable).

Definition 5.2. Let $p \in \mathbb{N}$ be a prime, let $m,k \in \mathbb{N}$ be integers, let $\vec{F},\vec{H},\vec{K} \in (\mathbb{Z}_p[X_1,\ldots,X_m])^k$ be tuples of polynomials, and let $Q \in \mathbb{Z}_p[X_1,\ldots,X_m]$ be a polynomial. Let $\tau \in \{1,2,3\}$. We say that Q is *type-τ dependent on* $(\vec{F},\vec{H},\vec{K})$ if there exist integers $\{\alpha_{i,j}\}_{i,j}$, $\{\beta_{i,j}\}_{i,j}$, $\{\gamma_{i,j}\}_{i,j}$ and $\{\delta_\ell\}_\ell$ such that

- If $\tau = 1$, then $Q = \sum_{i,j} \alpha_{i,j} \cdot F_i \cdot H_j + \sum_{i,j} \beta_{i,j} \cdot F_i \cdot F_j + \sum_{i,j} \gamma_{i,j} \cdot H_i \cdot H_j + \sum_\ell \delta_\ell \cdot K_\ell$.
- If $\tau = 2$, then $Q = \sum_{i,j} \alpha_{i,j} \cdot F_i \cdot H_j + \sum_{i,j} \gamma_{i,j} \cdot H_i \cdot H_j + \sum_\ell \delta_\ell \cdot K_\ell$.
- If $\tau = 3$, then $Q = \sum_{i,j} \alpha_{i,j} \cdot F_i \cdot H_j + \sum_\ell \delta_\ell \cdot K_\ell$.

If Q is not type-τ dependent on $(\vec{F},\vec{H},\vec{K})$, we say that it is type-τ *independent* of $(\vec{F},\vec{H},\vec{K})$.

The following definition captures when a problem within the Uber family cannot be trivially solved.

Definition 5.3. Let $p \in \mathbb{N}$ be a prime, let $m,k \in \mathbb{N}$ be integers, let $\vec{F},\vec{H},\vec{K} \in (\mathbb{Z}_p[X_1,\ldots,X_m])^k$ be tuples of polynomials, and let $Q_1,Q_2,Q_T \in \mathbb{Z}_p[X_1,\ldots,X_m]$ be polynomials. We say that the parameters $(\vec{F},\vec{H},\vec{K},Q_1,Q_2,Q_T)$ are *non-trivial for groups of type τ* for $\tau \in \{1,2,3\}$ if:

- If $\tau = 1$, at least one of the following holds:
  - Case (1.1): Q_1 is linearly independent of $(\vec{F},\vec{H})$.

- Case (1.2): Q_2 is linearly independent of $(\vec{F}, \vec{H})$.
- Case (1.T): Q_T is Type-1 independent of $(\vec{F}, \vec{H}, \vec{K})$

- If $\tau = 2$, at least one of the following holds:
  - Case (2.1): Q_1 is linearly independent of $(\vec{F}, \vec{H})$.
  - Case (2.2): Q_2 is linearly independent of $\vec{H}$.
  - Case (2.T): Q_T is Type-2 independent of $(\vec{F}, \vec{H}, \vec{K})$
- If $\tau = 3$, at least one of the following holds:
  - Case (3.1): Q_1 is linearly independent of $\vec{F}$.
  - Case (3.2): Q_2 is linearly independent of $\vec{H}$.
  - Case (3.T): Q_T is Type-3 independent of $(\vec{F}, \vec{H}, \vec{K})$

Observe, that in bilinear groups of type $\tau \in \{1, 2, 3\}$, if $(\vec{F}, \vec{H}, \vec{K}, Q_1, Q_2, Q_T)$ is trivial for type τ, then the induced problem can be easily solved (see [BFL20] for concrete details). Thus, within bilinear groups of type τ, we can only hope to reduce the hardness of $(\vec{F}, \vec{H}, \vec{K}, Q_1, Q_2, Q_T)$-**UBER** to the hardness of q-**DLOG** (or to the hardness of any other problem which we believe is hard) for tuples $(\vec{F}, \vec{H}, \vec{K}, Q_1, Q_2, Q_T)$ which are non-trivial for type τ.

5.2 From q-DLOG to UBER

Our main result in bilinear groups is a reduction from the hardness of $(\vec{F}, \vec{H}, \vec{K}, Q_1, Q_2, Q_T)$-**UBER** with non-trivial parameters to the hardness of q-**DLOG** or a variant thereof for certain choices of q. This improves upon a result of Bauer, Fuchsbauer and Loss [BFL20], who showed a similar reduction, in the following manner. Roughly speaking, in their work, the parameter q needs to be at least the *total degree* of the highest degree polynomial among the polynomials in $\vec{F}, \vec{H}, \vec{K}$. In our result below, q only has to be the maximal degree in which *a variable* appears in the polynomials in $\vec{F}, \vec{H}, \vec{K}$. Similarly to Bauer, Fuchsbauer and Loss, when considering type-3 bilinear groups, we consider the (q_1, q_2)-**DLOG** problem rather then the q-**DLOG** problem. This problem is defined in Figure 5, and our result exhibits a similar improvement to the one discussed above over the result of Bauer, Fuchsbauer and Loss when determining the necessary lower bound for the parameters q_1 and q_2.

$\underline{(q_1, q_2)\text{-}\mathbf{DLOG}^{\mathsf{A}}_{\mathcal{G}}}$

1. $x \leftarrow \mathbb{Z}_p$
2. $\mathbf{X}_i := g_1^{x^i}$ for all $i \in [q_1]$
3. $\mathbf{Y}_j := g_2^{x^j}$ for all $j \in [q_2]$
4. $x' \leftarrow \mathsf{A}(\mathbf{X}_1, \ldots, \mathbf{X}_{q_1}, \mathbf{Y}_1, \ldots, \mathbf{Y}_{q_2})$
5. If $x' = x$ output 1, and otherwise output 0

Figure 5: The game (q_1, q_2)-$\mathbf{DLOG}^{\mathsf{A}}_{\mathcal{G}}$ capturing the (q_1, q_2)-discrete logarithm problem in type 3 bilinear groups with respect to parameters q_1 and q_2, a type 3 bilinear group $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e, \phi, \psi, p)$ and an adversary A.

Theorem 5.4 below uses the following notation. For a polynomial $F \in \mathbb{Z}_p[X_1, \ldots, X_m]$ and for every $i \in [m]$, we denote by $\mathsf{deg}_{X_i}(F)$ the degree of F in the variable X_i. For a bilinear group $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$, for $i \in \{1, 2\}$ and for a parameter $q \in \mathbb{N}$, we denote by q-$\mathbf{DLOG}_{\mathcal{G}_i}$, the game q-**DLOG** with respect to the source group $\mathbb{G}_i$ (i.e., the parameters of the game are the description $\mathcal{G}_i = (\mathbb{G}_i, p, g_i)$).

Theorem 5.4. *Let $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$ be a description of a bilinear group of type $\tau \in \{1,2,3\}$ and let $k, m \in \mathbb{N}$ be integers. Let $\vec{F}, \vec{H}, \vec{K} \in (\mathbb{Z}_p[X_1, \ldots, X_m])^k$ be k-tuples of polynomials, let $d_{\vec{F}} = \max\{\mathsf{deg}_{X_j}(F_i)\}_{i\in[k],j\in[m]}$, $d_{\vec{H}} = \max\{\mathsf{deg}_{X_j}(H_i)\}_{i\in[k],j\in[m]}$ and $d_{\vec{K}} = \max\{\mathsf{deg}_{X_j}(K_i)\}_{i\in[k],j\in[m]}$, and let $Q_1, Q_2, Q_T \in \mathbb{Z}_p[X_1, \ldots, X_m]$ be polynomials. Let $q \geq \max\{d_{\vec{F}}, d_{\vec{H}}, d_{\vec{K}}/2\}$, and let $q_1 \geq d_{\vec{F}}$ and $q_2 \geq d_{\vec{H}}$ such that $q_1 + q_2 \geq d_{\vec{K}}$. If $(\vec{F}, \vec{H}, \vec{K}, Q_1, Q_2, Q_T)$ is non-trivial for type τ, then for any algebraic algorithm A there exist algebraic algorithms $\mathsf{B}_1, \mathsf{B}_2$, B_3 and B_4 such that*

$$\mathbf{Time}^{q\text{-}\mathbf{DLOG}_{\mathcal{G}}}_{\mathsf{B}_i} \leq \mathbf{Time}^{(\vec{F},\vec{H},\vec{K},Q_1,Q_2,Q_T)\text{-}\mathbf{UBER}_{\mathcal{G}}}_{\mathsf{A}} + \mathsf{poly}(m, k, q, q_1, q_2, \log p)$$

for each $i \in \{1,2,3,4\}$, and:

- *If $\tau = 2$: $\mathbf{Adv}^{q\text{-}\mathbf{DLOG}_{\mathcal{G}_2}}_{\mathsf{B}_1} \geq \epsilon/2m$;*
- *If $\tau = 1$: $\mathbf{Adv}^{q\text{-}\mathbf{DLOG}_{\mathcal{G}_2}}_{\mathsf{B}_2} \geq \epsilon/2m$ and $\mathbf{Adv}^{q\text{-}\mathbf{DLOG}_{\mathcal{G}_1}}_{\mathsf{B}_3} \geq \epsilon/2m$;*
- *If $\tau = 3$: $\mathbf{Adv}^{(q_1,q_2)\text{-}\mathbf{DLOG}_{\mathcal{G}}}_{\mathsf{B}_4} \geq \epsilon/2m$;*

where $\epsilon = \mathbf{Adv}^{(\vec{F},\vec{H},\vec{K},Q_1,Q_2,Q_T)\text{-}\mathbf{UBER}_{\mathcal{G}}}_{\mathsf{A}}$.

The proof of Theorem 5.4 relies on an approach introduced by Rotem and Segev [RS20a] in the context of the algebraic hardness of the k-linear problem, and generalizes it to full-fledged Uber problem. Concretely, the proof uses the following notation: Let f be a non-zero multivariate polynomial over $\mathbb{Z}_p$ in the indeterminates $X_1, \ldots, X_m$. Then, we denote by $\mathcal{S}(f)$ the sequence $h_1, \ldots, h_m$ of polynomials defined recursively by:

- $h_1 = f$; and
- For every $i \in \{2, \ldots, m\}$: If $h_{i-1} = 0$, set $h_i = 0$. Otherwise, write h_{i-1} as a polynomial in $(\mathbb{Z}_p[X_i, \ldots, X_m])[X_{i-1}]$. That is, write

$$h_{i-1} = \sum_{j=0}^{d} g_j(X_i, \ldots, X_m) \cdot X_{i-1}^j,$$

  where d is the degree of X_{i-1} in h_{i-1}. Let j^* be the minimal index for which $g_{j^*}(X_i, \ldots, X_m)$ is not the zero polynomial over $\mathbb{Z}_p$. Set $h_i = g_{j^*}$. If no such index j^* exist, set $h_{i+1} = 0$.

The proof of Theorem 5.4 relies on the following lemma, adapted from [RS20a] and proven below for completeness.

Lemma 5.5. *Let f be a non-zero m-variate polynomials in the indeterminates $X_1, \ldots, X_m$ over $\mathbb{Z}_p$, and let $\mathcal{S}(f) = (h_1, \ldots, h_m)$. Then, the following two conditions hold:*

1. *$h_i \neq 0$ for every $i \in [m]$.*
2. *For every vector $\vec{\alpha} = (\alpha_1, \ldots, \alpha_m) \in \mathbb{Z}_p^m$ such that $f(\vec{\alpha}) = 0$, there exists $i^* \in [m]$ such that the following holds:*
   The univariate polynomial $v(X_{i^*}) = h_{i^*}(X_i, \alpha_{i^*+1}, \ldots, \alpha_m)$ is not the zero polynomial and additionally $v(\alpha_{i^*}) = 0$.

Proof. The proof is by induction on the number m of variables. For $m = 1$, both conditions hold trivially since by definition $h_1 = f$. For $m > 1$, we can write $h_1 = \sum_{j=0}^{d} g_j(X_2, \ldots, X_m) \cdot X_1^j$. Since h_1 is not the zero polynomial (since $h_1 = f$), there exists some index $j \in \{0, \ldots, d\}$ for which $g_j(X_2, \ldots, X_m)$ is not the zero polynomial; let j^* be the minimal such index. By construction,

$h_2 = g_{j^*}$, and hence h_2 is not the zero polynomial. Since h_2 is a polynomial in at most $m-1$ variables, the induction hypothesis implies that $h_3, \ldots, h_m$ are non-zero polynomials as well. This proves the first condition.

As for the second condition, let $\vec{\alpha} = (\alpha_1, \ldots, \alpha_m) \in \mathbb{Z}_p$ such that $f(\vec{\alpha}) = 0$ and consider two cases:

1. **Case 1:** If $v_1(X_1) = h_1(X_1, \alpha_2, \ldots, \alpha_m)$ is not the zero polynomial, then the second condition is satisfied for the index $i^* = 1$ since
$$v_1(\alpha_1) = h_1(\vec{\alpha}) = f(\vec{\alpha}) = 0.$$
2. **Case 2:** If $v_1(X_1) = h_1(X_1, \alpha_2, \ldots, \alpha_m)$ is the zero polynomial, then when we write $h_1 = \sum_{j=0}^{d} g_j(X_2, \ldots, X_m) \cdot X_1^j$, it is necessarily the case that $g_j(\alpha_2, \ldots, \alpha_m) = 0$ for every $j \in [d]$. This is true in particular for the minimal index $j^* \in [d]$ for which g_{j^*} is not the zero polynomial (we already established above that such an index exists), and by definition $h_2 = g_{j^*}$. Hence, since h_2 is a polynomial in at most $m-1$ indices, the second condition follows from the induction hypothesis.

■

We now turn to prove Theorem 5.4.

Proof of Theorem 5.4. We first prove Theorem 5.4 in type-2 bilinear groups, and then review the necessary amendments in order to adapt the proof to groups of types 1 and 3.

Let A be an algebraic algorithm participating in $(\vec{F}, \vec{H}, \vec{K}, Q_1, Q_2, Q_T)$-**UBER**$_{\mathcal{G}}$. We construct an algorithm B_1 participating in q-**DLOG**$_{\mathcal{G}_2}$. Observe that for any $x \in \mathbb{Z}_p$, any index $i^* \in [m]$, any values $\alpha_1, \ldots, \alpha_{i^*-1}, \alpha_{i^*+1}, \ldots, \alpha_m$, and any polynomial $f(X_1, \ldots, X_m)$ such that the degree of X_{i^*} in f is at most q, the following is true: Given the values $\alpha_1, \ldots, \alpha_{i^*-1}, \alpha_{i^*+1}, \ldots, \alpha_m$ and the group elements $g_1, \mathbf{S}_0 = g_2, \mathbf{S}_1 = g_2^x, \ldots, \mathbf{S}_q = g_2^{x^q}$, the algorithm B_1 can efficiently compute the elements $g_1^{f(\alpha_1,\ldots,\alpha_{i^*-1},x,\alpha_{i^*+1},\ldots,\alpha_m)}$ and $g_2^{f(\alpha_1,\ldots,\alpha_{i^*-1},x,\alpha_{i^*+1},\ldots,\alpha_m)}$. To compute $g_2^{f(\alpha_1,\ldots,\alpha_{i^*-1},x,\alpha_{i^*+1},\ldots,\alpha_m)}$:

1. Write the univariate polynomial $u(X) = f(\alpha_1, \ldots, \alpha_{i^*-1}, X, \alpha_{i^*+1}, \ldots, \alpha_m)$ as $\sum_{j=0}^{q} \beta_j \cdot X^j$.
2. For $j = 0, \ldots, q$: Compute $\mathbf{U}_j = \mathbf{S}_j^{\beta_j}$.
3. Compute and output $\prod_{j=0}^{q} \mathbf{U}_j$.

B_1 can similarly compute $g_1^{f(\alpha_1,\ldots,\alpha_{i^*-1},x,\alpha_{i^*+1},\ldots,\alpha_m)}$, by first computing $g_1^x = \psi(\mathbf{S}_1), \ldots, g_1^{x^q} = \psi(\mathbf{S}_q)$ and then proceeding as above with $g_1, g_1^x, \ldots, g_1^{x^q}$ instead of $g_2, g_2^x, \ldots, g_2^{x^q}$. Additionally, B_1 can compute $g_T^{f(\alpha_1,\ldots,\alpha_{i^*-1},x,\alpha_{i^*+1},\ldots,\alpha_m)}$ as long as the degree of X_{i^*} in f is at most $2q$, by first computing $g_T = e(g_1, g_2), g_T^x = e(g_1, \mathbf{S}_1), \ldots, g_T^{x^q} = e(g_1, \mathbf{S}_q), g_T^{x^{q+1}} = e(\psi(\mathbf{S}_1), \mathbf{S}_q), \ldots, g_T^{x^{2q}} = e(\psi(\mathbf{S}_q), \mathbf{S}_q)$, and then proceeding as above with $g_T, g_T^x, \ldots, g_T^{x^{2q}}$ instead of $g_2, g_2^x, \ldots, g_2^{x^q}$.

Observe that the fact that tuple $(\vec{F}, \vec{H}, \vec{K}, Q_1, Q_2, Q_T)$ is non-trivial for type $\tau = 2$ needs to rely on at least one of the conditions (2.1),(2.2) or $(2.T)$ of Definition 5.3. Each of these conditions endows a different reduction. We start with assuming that condition $(2.T)$ holds, and consider the algorithm B_1 participating in q-**DLOG**$_{\mathcal{G}_2}$.

Algorithm B_1

Input: q elements $\mathbf{S}_1 = g_2^x, \ldots, \mathbf{S}_q = g_2^{x^q}$ in $\mathbb{G}_2$ for $x \leftarrow \mathbb{Z}_p$.

1. Sample $i^* \leftarrow [m]$ and $\alpha_1, \ldots, \alpha_{i^*-1}, \alpha_{i^*+1}, \ldots, \alpha_m \leftarrow \mathbb{Z}_p$.

2. For $i = 1, \ldots, k$, compute $\mathbf{X}_i := g_1^{F_i(\vec{x})}$, $\mathbf{Y}_i := g_2^{H_i(\vec{x})}$ and $\mathbf{Z}_i := g_T^{K_i(\vec{x})}$, where $\vec{x} = (\alpha_1, \ldots, \alpha_{i^*-1}, x, \alpha_{i^*+1}, \ldots, \alpha_m)$.
[since the degree of X_{i^*} is at most q in each polynomial in $\{F_i\}, \{H_i\}$ and is at most $2q$ in each polynomial in $\{K_i\}$, these group elements can be computed efficiently as discussed above]
3. Invoke $\mathsf{A}(\mathbf{X}_1, \ldots, \mathbf{X}_k, \mathbf{Y}_1, \ldots, \mathbf{Y}_k, \mathbf{Z}_1, \ldots, \mathbf{Z}_k)$ to obtain $\mathbf{W}_1 \in \mathbb{G}_1$, $\mathbf{W}_2 \in \mathbb{G}_2$ and $\mathbf{W}_T \in \mathbb{G}_T$. Since A is algebraic, it also outputs vectors $\vec{v}, \vec{w}, \vec{u}, \vec{s}, \vec{r}$ and matrices $A = (a_{i,j}), B = (b_{i,j})$ and $C = (c_{i,j})$ such that:
   - $\mathbf{W}_1 = \prod_i \mathbf{X}_i^{v_i} \cdot \prod_i \psi(\mathbf{Y}_i)^{w_i}$.
   - $\mathbf{W}_2 = \prod_i \mathbf{Y}_i^{s_i}$.
   - $\mathbf{W}_T = \prod_{i,j} e(\mathbf{X}_i, \mathbf{Y}_j)^{a_{i,j}} \cdot \prod_{i,j} e(\psi(\mathbf{Y}_i), \mathbf{Y}_j)^{c_{i,j}} \cdot \prod_i \mathbf{Z}_i^{r_i}$.
4. Define the m-variate polynomial

$$f_{\vec{r},A,C}(X_1, \ldots, X_m) = Q_T(\vec{X}) - \sum_{i,j} a_{i,j} \cdot F_i(\vec{X}) \cdot H_j(\vec{X}) - \sum_{i,j} c_{i,j} \cdot H_i(\vec{X}) \cdot H_j(\vec{X}) - \sum_i r_i \cdot K_i(\vec{X}),$$

and compute the sequence of polynomials $\mathcal{S}(f_{\vec{r},A,C})$ and denote these polynomials by $f_1, \ldots, f_m$.

5. Find all roots $x_1^*, \ldots, x_\ell^*$ of the univariate polynomial $h_{i^*}(X_{i^*}) = f_{i^*}(X_{i^*}, \alpha_{i^*+1}, \ldots, \alpha_m)$.
6. For every $i = 1, \ldots, \ell$, check if $g^{x_i^*} = \mathbf{X}$. If so, output x_i^* and terminate; otherwise, continue.
7. If reached, output $\perp$ and terminate.

We start by analyzing the success probability of B_1. Let the vector $\vec{r}$ and the matrices A and C be those outputted by A in Step 3 of B_1, and let $\alpha_1, \ldots, \alpha_m$ be integers in $\mathbb{Z}_p$ such that for each $i \neq i^*$ (where i^* is the index sampled by A in Step 1) α_i is the value sampled by A in Step 1, and $\alpha_{i^*} = x$ (where x is the $\mathbb{Z}_p$ exponent used to generate B_1's input).

Consider the sequence of polynomials $f_1, \ldots, f_m$ computed by B_1 in Step 4 as $\mathcal{S}(f_{\vec{r},A,C})$. We make following claim about $f_1, \ldots, f_m$.

Claim 5.6. *The following conditions hold:*

1. *For every $i \in [m]$, f_i is not the zero polynomial; and*
2. *If the output $\mathbf{W}_T$ of A (when invoked in Step 3 of B_1) is equal to $g_T^{Q_T(\alpha_1, \ldots, \alpha_m)}$, then there exists an index $j^* \in [m]$ such that the univariate polynomial $h_{j^*} = f_{j^*}(X_{j^*}, \alpha_{j^*+1}, \ldots, \alpha_m)$ is not the zero polynomial and $h_{j^*}(\alpha_{j^*}) = 0$.*

Proof. We wish to use Lemma 5.5 in order to prove the two statements made in Claim 5.6. In order to do so, we argue that:

- $f_{\vec{r},A,C}$ is not the zero polynomial. This follows immediately from the definition of $f_{\vec{r},A,C}$ and the fact that Q_T is type-2 independent of $(\vec{F}, \vec{H}, \vec{K})$. The latter follows from our assumption that condition $(2.T)$ of Definition 5.3 holds.
- If $\mathbf{W}_T = g_T^{Q_T(\alpha_1, \ldots, \alpha_m)}$, then $f_{\vec{r},A,C}(\alpha_1, \ldots, \alpha_m) = 0$. This is the case since by the definition of the algebraic group model it holds that

$$\mathbf{W}_T = g_T^{\sum_{i,j} a_{i,j} \cdot F_i(\vec{\alpha}) \cdot H_j(\vec{\alpha}) + \sum_{i,j} c_{i,j} \cdot H_i(\vec{\alpha}) \cdot H_j(\vec{\alpha}) + \sum_i r_i \cdot K_i(\vec{\alpha})} = g_T^{Q_T(\vec{\alpha}) - f_{\vec{r},A,C}(\vec{\alpha})},$$

where $\vec{\alpha} = (\alpha_1, \ldots, \alpha_m)$. This implies that

$$g_T^{Q_T(\alpha_1, \ldots, \alpha_m)} = g_T^{Q_T(\vec{\alpha}) - f_{\vec{r},A,C}(\vec{\alpha})},$$

which in turn implies that $f_{\vec{r},A,C}(\alpha_1,\ldots,\alpha_m)=0$.

Hence, Claim 5.6 follows from Lemma 5.5. ∎

Denote by AWin the event in which the group element $\mathbf{W}_T$ outputted by A in Step 3 of B_1 is indeed equal to $g_T^{Q_T(\alpha_1,\ldots,\alpha_m)}$. Since B_1 perfectly simulates the game $(\vec{F},\vec{H},\vec{K},Q_1,Q_2,Q_T)$-$\mathbf{UBER}^{\mathsf{A}}_{\mathcal{G}}$ to A, and since $\mathbf{W}_T = g_T^{Q_T(\alpha_1,\ldots,\alpha_m)}$ is a necessary condition in order for the output of this (simulated) game to be 1, it holds that $\Pr[\mathsf{AWin}] \geq \epsilon$. Therefore,

$$\begin{aligned}\mathbf{Adv}^{q\text{-}\mathbf{DLOG}_{\mathcal{G}_2}}_{\mathsf{B}_1} &= \Pr\left[q\text{-}\mathbf{DLOG}^{\mathsf{B}_1}_{\mathcal{G}_2}=1\right]\\ &\geq \Pr\left[q\text{-}\mathbf{DLOG}^{\mathsf{B}_1}_{\mathcal{G}_2}=1\middle|\mathsf{AWin}\right]\cdot\Pr[\mathsf{AWin}]\\ &\geq \epsilon\cdot\Pr\left[q\text{-}\mathbf{DLOG}^{\mathsf{B}_1}_{\mathcal{G}_2}=1\middle|\mathsf{AWin}\right].\end{aligned}$$

Let $\mathsf{IndexHit}$ denote the event in which the univariate polynomial h_{i^*} considered by B_1 in Step 5 is not the zero polynomial, and in addition it holds that $h_{i^*}(\alpha_{i^*})=0$. Observe that Claim 5.6 implies that whenever AWin occurs, there exists an index $j^*\in[m]$ such that if $i^*=j^*$ then $\mathsf{IndexHit}$ occurs. We thus infer that $\Pr[\mathsf{IndexHit}|\mathsf{AWin}]\geq 1/m$. Hence,

$$\begin{aligned}\mathbf{Adv}^{q\text{-}\mathbf{DLOG}_{\mathcal{G}_2}}_{\mathsf{B}_1} &\geq \epsilon\cdot\Pr\left[q\text{-}\mathbf{DLOG}^{\mathsf{B}_1}_{\mathcal{G}_2}=1\middle|\mathsf{IndexHit}\wedge\mathsf{AWin}\right]\cdot\Pr[\mathsf{IndexHit}|\mathsf{AWin}]\\ &\geq \frac{\epsilon}{m}\cdot\Pr\left[q\text{-}\mathbf{DLOG}^{\mathsf{B}_1}_{\mathcal{G}_2}=1\middle|\mathsf{IndexHit}\wedge\mathsf{AWin}\right].\end{aligned}$$

Recall that $\alpha_{i^*}=x$. Therefore, whenever $\mathsf{IndexHit}$ occurs, it means that the secret exponent x is a root of the polynomial h_{i^*}. The algorithm B_1 can use a randomized algorithm which finds all the roots of h_{i^*} – the secret exponent x included – with probability at least $1/2$, and runs in time polynomial in q and in $\log p$ (for example, the Berlekamp algorithm [Ber70, Rab80]). Overall, we obtain that

$$\mathbf{Adv}^{q\text{-}\mathbf{DLOG}_{\mathcal{G}}}_{\mathsf{B}_1}\geq\frac{\epsilon}{2m}.$$

This concludes the analysis assuming condition $(2.T)$ holds.

If condition (2.1) holds, the algorithm B_1 is defined similarly, but in Step 4 it computes the polynomials $f_1,\ldots,f_m$ differently. Concretely, if defines the polynomial

$$f_{\vec{v},\vec{w}}(X_1,\ldots,X_m)=Q_1(\vec{X})-\sum_i v_i\cdot F_i(\vec{X})-\sum_i w_i\cdot H_i(\vec{X}),$$

and then computes $f_1,\ldots,f_2=\mathcal{S}(f_{\vec{v},\vec{w}}(X_1,\ldots,X_m))$. Condition (2.1) guarantees that we can again use Claim 5.6, replacing the condition $\mathbf{W}_T=g_T^{Q_T(\alpha_1,\ldots,\alpha_m)}$ in the second statement of the claim, with the condition $\mathbf{W}_1=g_1^{Q_1(\alpha_1,\ldots,\alpha_m)}$. Condition (2.2) is handled similarly, replacing the polynomial $f_{\vec{v},\vec{w}}$ above with the polynomial $f_{\vec{s}}(\vec{X})=Q_2(\vec{X})-\sum_i s_i\cdot H_i(\vec{X})$.

Type 1 groups. The reduction to q-$\mathbf{DLOG}_{\mathcal{G}_2}$ in bilinear groups of type 1 is almost identical to the one above. The algorithm B_2 is defined identically to B_1 until Step 3. In this step, A now outputs elements $\mathbf{W}_1,\mathbf{W}_2,\mathbf{W}_T$, alongside vectors $\vec{v},\vec{w},\vec{u},\vec{s},\vec{r}$ and matrices $A=(a_{i,j}),B=(b_{i,j})$ and $C=(c_{i,j})$ such that:

- $\mathbf{W}_1=\prod_i\mathbf{X}_i^{v_i}\cdot\prod_i\psi(\mathbf{Y}_i)^{w_i}$.
- $\mathbf{W}_2=\prod_i\phi(\mathbf{X}_i)^{u_i}\cdot\prod_i\mathbf{Y}_i^{s_i}$.

- $\mathbf{W}_T = \prod_{i,j} e(\mathbf{X}_i, \mathbf{Y}_j)^{a_{i,j}} \cdot \prod_{i,j} e(\mathbf{X}_i, \phi(\mathbf{X}_j))^{b_{i,j}} \cdot \prod_{i,j} e(\psi(\mathbf{Y}_i), \mathbf{Y}_j)^{c_{i,j}} \cdot \prod_i \mathbf{W}_i^{r_i}$.

In type 1 groups, the fact that tuple $(\vec{F}, \vec{H}, \vec{K}, Q_1, Q_2, Q_T)$ is non-trivial for type $\tau = 1$ needs to rely on at least one of the conditions (1.1),(1.2) or $(1.T)$ of Definition 5.3. Step 4 of the algorithm B_2 is then modified as follows, depending on which of the aforesaid conditions hold. If condition $(1.T)$, the polynomial $f_{\vec{r},A,C}$ is replaced with the polynomial

$$f_{\vec{r},A,B,C}(\vec{X}) = Q_T(\vec{X}) - \sum_{i,j} a_{i,j} \cdot F_i(\vec{X}) \cdot H_j(\vec{X}) - \sum_{i,j} b_{i,j} \cdot F_i(\vec{X}) \cdot F_j(\vec{X}) - \sum_{i,j} c_{i,j} \cdot H_i(\vec{X}) \cdot H_j(\vec{X}) - \sum_i r_i \cdot K_i(\vec{X}).$$

If condition (1.2), then the polynomial $f_{\vec{s}}(\vec{X})$ is replaced with the polynomial $f_{\vec{u},\vec{s}}(\vec{X}) = Q_2(\vec{X}) - \sum_i u_i \cdot F_i(\vec{X}) - \sum_i s_i \cdot H_i(\vec{X})$.

The reduction to q-$\mathbf{DLOG}_{\mathcal{G}_1}$ is defined identically, except for the way in which B_3 generates the instance $(\mathbf{X}_1, \ldots, \mathbf{X}_k, \mathbf{Y}_1, \ldots, \mathbf{Y}_k, \mathbf{Z}_1, \ldots, \mathbf{Z}_k)$ of the Uber problem on which A is then invoked in Step 3. Now, B_3 is given as input $g_1^x, \ldots, g_1^{x^q}$ instead of $g_2^x, \ldots, g_2^{x^q}$. Nevertheless, B_3 can still compute the Uber problem instance as explained in the beginning of the proof, while switching the roles of $\mathbb{G}_1$ and $\mathbb{G}_2$ and using the isomorphism ϕ instead of ψ.

Type 3 groups. The reduction to (q_1, q_2)-$\mathbf{DLOG}_{\mathcal{G}}$ in groups of type 3 is also very similar, the two differences being the way B_4 computes the Uber instance in Step 2 and the way in which the polynomial in Step 4 is defined. First, observe that in Step 2, B_4 can still compute $(\mathbf{X}_1, \ldots, \mathbf{X}_k, \mathbf{Y}_1, \ldots, \mathbf{Y}_k, \mathbf{Z}_1, \ldots, \mathbf{Z}_k)$ since for each $i \in [k]$ it holds that $\mathbf{X}_i = g_1^{F_i(\alpha_1, \ldots, \alpha_{i^*-1}, x, \alpha_{i^*+1}, \ldots, \alpha_m)}$ where F_i has degree at most q_1 in X_{i^*}; $\mathbf{Y}_i = g_2^{H_i(\alpha_1, \ldots, \alpha_{i^*-1}, x, \alpha_{i^*+1}, \ldots, \alpha_m)}$ where H_i has degree at most q_2 in X_{i^*}; and $\mathbf{Z}_i = g_T^{K_i(\alpha_1, \ldots, \alpha_{i^*-1}, x, \alpha_{i^*+1}, \ldots, \alpha_m)}$ where K_i has degree at most $q_1 + q_2$ in X_{i^*}.

Secondly, in Step 3, A outputs elements $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_T$, alongside vectors $\vec{v}, \vec{w}, \vec{u}, \vec{s}, \vec{r}$ and matrices $A = (a_{i,j}), B = (b_{i,j})$ and $C = (c_{i,j})$ such that:

- $\mathbf{W}_1 = \prod_i \mathbf{X}_i^{v_i}$.
- $\mathbf{W}_2 = \prod_i \mathbf{Y}_i^{s_i}$.
- $\mathbf{W}_T = \prod_{i,j} e(\mathbf{X}_i, \mathbf{Y}_j)^{a_{i,j}} \cdot \prod_i \mathbf{W}_i^{r_i}$.

Then Step 4 of B_4 is changed as follows. In case condition $(3.T)$ holds the polynomial $f_{\vec{r},A,C}$ is replaced with polynomial $f_{\vec{r},A}$ which is obtained from $f_{\vec{r},A,C}$ be setting A to be the all zero matrix. In case condition (3.1) holds, the polynomial $f_{\vec{v},\vec{w}}$ is replaced with the polynomial $f_{\vec{v}}$ which is obtained from $f_{\vec{v},\vec{w}}$ by setting $\vec{w}$ to be the all zero vector. In case condition (3.2) holds, the polynomial $f_{\vec{s}}$ remains unchanged. ■

References

[ABB+20] M. Abdalla, M. Barbosa, T. Bradley, S. Jarecki, J. Katz, and J. Xu. Universally composable relaxed password authenticated key exchange. In *Advances in Cryptology – CRYPTO '20*, pages 278–307, 2020.

[ABK+21] M. Abdalla, M. Barbosa, J. Katz, J. Loss, and J. Xu. Algebraic adversaries in the universal composability framework. In *Advances in Cryptology – ASIACRYPT '21*, pages 311–341, 2021.

[ABM15] M. Abdalla, F. Benhamouda, and P. MacKenzie. Security of the J-PAKE password-authenticated key exchange protocol. In *IEEE Symposium on Security and Privacy*, pages 571–587, 2015.

[AGK20] B. Auerbach, F. Giacon, and E. Kiltz. Everybody's a target: Scalability in public-key encryption. In *Advances in Cryptology – EUROCRYPT '20*, pages 475–506, 2020.

[AHK20] T. Agrikola, D. Hofheinz, and J. Kastner. On instantiating the algebraic group model from falsifiable assumptions. In *Advances in Cryptology – EUROCRYPT '20*, pages 96–126, 2020.

[AM09] D. Aggarwal and U. Maurer. Breaking RSA generically is equivalent to factoring. In *Advances in Cryptology – EUROCRYPT '09*, pages 36–53, 2009.

[BBB+18] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch. Verifiable delay functions. In *Advances in Cryptology – CRYPTO '18*, pages 757–788, 2018.

[BBF18] D. Boneh, B. Bünz, and B. Fisch. A survey of two verifiable delay functions. Cryptology ePrint Archive, Report 2018/712, 2018.

[BBG05] D. Boneh, X. Boyen, and E. Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology – EUROCRYPT '05*, pages 440–456, 2005.

[BBH+02] L. Biehl, J. Buchmann, S. Hamdy, and A. Meyer. A signature scheme based on the intractability of computing roots. *Designs, Codes and Cryptography*, 25(3):223–236, 2002.

[BBS86] L. Blum, M. Blum, and M. Shub. A simple unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15(2):364–383, 1986.

[BCP01a] E. Bresson, O. Chevassut, and D. Pointcheval. Provably authenticated group diffie-hellman key exchange – the dynamic case. In *Advances in Cryptology – ASIACRYPT '01*, pages 290–309, 2001.

[BCP+01b] E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater. Provably authenticated group diffie-hellman key exchange. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pages 255–264, 2001.

[BCP07] E. Bresson, O. Chevassut, and D. Pointcheval. Provably secure authenticated group diffie-hellman key exchange. *ACM Transactions on Information and System Security*, 10(3):10:1–10:45, 2007.

[BDF+20] D. Boneh, J. Drake, B. Fisch, and A. Gabizon. Efficient polynomial commitment schemes for multiple points and polynomials. Cryptology ePrint Archive, Report 2020/081, 2020.

[Ber70] E. R. Berlekamp. Factoring polynomials over large finite fields. *Mathematics of Computation*, 24(111):713–735, 1970.

[BFL20] B. Bauer, G. Fuchsbauer, and J. Loss. A classification of computational assumptions in the algebraic group model. In *Advances in Cryptology – CRYPTO '20*, pages 121–151, 2020.

[BFW16] D. Bernhard, M. Fischlin, and B. Warinschi. On the hardness of proving CCA-security of signed ElGamal. In *Public-Key Cryptography – PKC ,16*, pages 47–69, 2016.

[BH01] J. Buchmann and S. Hamdy. A survey on IQ cryptography. In *In Proceedings of Public Key Cryptography and Computational Number Theory*, pages 1–15, 2001.

[BN00] D. Boneh and M. Naor. Timed commitments. In *Advances in Cryptology – CRYPTO '00*, pages 236–254, 2000.

[Boy08] X. Boyen. The Uber-assumption family – A unified complexity framework for bilinear groups. In *Proceedings of the 2nd International Conference on Pairing-based Cryptography*, pages 39–56, 2008.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BR06] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *Advances in Cryptology – EUROCRYPT '98*, pages 409–426, 2006.

[BV98] D. Boneh and R. Venkatesan. Breaking RSA may not be equivalent to factoring. In *Advances in Cryptology – EUROCRYPT '98*, pages 59–71, 1998.

[CH20] G. Couteau and D. Hartmann. Shorter non-interactive zero-knowledge arguments and ZAPs for algebraic languages. In *Advances in Cryptology – CRYPTO '20*, pages 768–798, 2020.

[CHM+20] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In *Advances in Cryptology – EUROCRYPT '20*, pages 738–768, 2020.

[Den02] A. W. Dent. Adapting the weaknesses of the random oracle model to the generic group model. In *Advances in Cryptology – ASIACRYPT '02*, pages 100–109, 2002.

[DK02] I. Damgård and M. Koprowski. Generic lower bounds for root extraction and signature schemes in general groups. In *Advances in Cryptology – EUROCRYPT '02*, pages 256–271, 2002.

[FKL18] G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In *Advances in Cryptology – CRYPTO '18*, pages 33–62, 2018.

[FPS20] G. Fuchsbauer, A. Plouviez, and Y. Seurin. Blind Schnorr signatures and signed ElGamal encryption in the algebraic group model. In *Advances in Cryptology – EUROCRYPT '20*, pages 63–95, 2020.

[GJ02] J. A. Garay and M. Jakobsson. Timed release of standard digital signatures. In *Financial Cryptography*, pages 190–207, 2002.

[GMP+06] J. A. Garay, P. D. MacKenzie, M. Prabhakaran, and K. Yang. Resource fairness and composability of cryptographic protocols. In *Proceedings of the 4th Theory of Cryptography Conference*, pages 404–428, 2006.

[GP03] J. A. Garay and C. Pomerance. Timed fair exchange of standard signatures. In *Financial Cryptography*, pages 190–207, 2003.

[GRW+20] S. Gorbunov, L. Reyzin, H. Wee, and Z. Zhang. Pointproofs: Aggregating proofs for multiple vector commitments. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 2007–2023, 2020.

[GT21] A. Ghoshal and S. Tessaro. Tight state-restoration soundness in the algebraic group model. In *Advances in Cryptology – CRYPTO '21*, pages 64–93, 2021.

[KL14] J. Katz and Y. Lindell. Introduction to modern cryptography (2nd edition). Chapman & Hall/CRC press, 2014.

[KLX20] J. Katz, J. Loss, and J. Xu. On the security of time-lock puzzles and timed commitments. In *Proceedings of the 18th Theory of Cryptography Conference*, pages 390–413, 2020.

[KLX22] J. Kastner, J. Loss, and J. Xu. On pairing-free blind signature schemes in the algebraic group model. To appear in *Public-Key Cryptography — PKC '22* (available at `https://eprint.iacr.org/2020/1071.pdf`), 2022.

[MBK+19] M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2111–2128, 2019.

[MTT19] T. Mizuide, A. Takayasu, and T. Takagi. Tight reductions for Diffie-Hellman variants in the algebraic group model. In *Topics in Cryptology – CT-RSA '19*, pages 169–188, 2019.

[Pie19] K. Pietrzak. Simple verifiable delay functions. In *Proceedings of the 10th Conference on Innovations in Theoretical Computer Science*, pages 60:1–60:15, 2019.

[PV05] P. Paillier and D. Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In *Advances in Cryptology – ASIACRYPT '05*, pages 1–20, 2005.

[Rab80] M. O. Rabin. Probabilistic algorithms in finite fields. *SIAM Journal on Computing*, 9(2):273–280, 1980.

[RS20a] L. Rotem and G. Segev. Algebraic distinguishers: From discrete logarithms to decisional uber assumptions. In *Proceedings of the 18th Theory of Cryptography Conference*, pages 366–389, 2020.

[RS20b] L. Rotem and G. Segev. Generically speeding-up repeated squaring is equivalent to factoring: Sharp thresholds for all generic-ring delay functions. In *Advances in Cryptology – CRYPTO '20*, pages 481–509, 2020.

[RSW96] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto, 1996.

[vBS21] A. van Baarsen and M. Stevens. On time-lock cryptographic assumptions in abelian hidden-order groups. In *Advances in Cryptology – ASIACRYPT '21*, pages 367–397, 2021.

[Wes19] B. Wesolowski. Efficient verifiable delay functions. In *Advances in Cryptology – EUROCRYPT '19*, pages 379–407, 2019.

A Extending the Decisional Algebraic Group Model

In this section we present the decisional AGM (the DAGM for short) of Rotem and Segev [RS20a], and then present a novel extension thereof which considers asymmetric bilinear groups.

A.1 The Basic DAGM

We review the DAGM of Rotem and Segev [RS20a] with respect to cyclic groups described by a tuple of the form $\mathcal{G} = (\mathbb{G}, p, g)$, where $\mathbb{G}$ is the group, p is its order, and g is a generator of the group. Our review of the DAGM of Rotem and Segev is taken mutatis mutandis from [RS20a].

Decisional Algebraic Games. Decisional games are aimed at capturing decisional cryptographic problems (e.g., the decisional Diffie-Hellman problem) and indistinguishability-based security properties of cryptographic primitives (e.g., semantic security of an encryption scheme). At the end of a decisional game, the adversary outputs either the acceptance symbol Acc, in which case the output of the game is 1, or the rejection symbol Rej, in which case the output of the game is 0. The advantage of an adversary A in distinguishing between two decisional games $\mathbf{G}_{par}$ and $\mathbf{G'}_{par'}$ is defined as

$$\mathbf{Adv}_{\mathsf{A}}^{\mathbf{G}_{par},\mathbf{G'}_{par'}} \stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathbf{G}_{par}^{\mathsf{A}} = 1\right] - \Pr\left[\mathbf{G'}_{par'}^{\mathsf{A}} = 1\right]\right|.$$

Typically, a decisional security definition will be obtained by a single decisional game $\mathbf{G}$ with an additional parameter bit b, where the adversary needs to distinguish between the cases $b = 0$ and $b = 1$. For brevity, we will refer to the advantage of an adversary A in distinguishing between $\mathbf{G}_{par,0}$ and $\mathbf{G}_{par,1}$ simply as the advantage of A in G_{par}, and we will use the notation $\mathbf{Adv}_{\mathsf{A}}^{\mathbf{G}_{par}} \stackrel{\mathsf{def}}{=} \mathbf{Adv}_{\mathsf{A}}^{\mathbf{G}_{par,0},\mathbf{G}_{par,1}}$. The running time of $\mathbf{G}_{par}^{\mathsf{A}}$ is defined as the maximum of the running times of $\mathbf{G}_{par,0}^{\mathsf{A}}$ and of $\mathbf{G}_{par,1}^{\mathsf{A}}$. Figure 6 exemplifies the notion of a decisional algebraic game by presenting the game associated with the Decisional Diffie-Hellman problem.

$\underline{\mathbf{DDH}}_{\mathcal{G},b}^{\mathsf{A}}$

1. $x, y, z \leftarrow \mathbb{Z}_p$
2. $\mathbf{X} := g^x, \mathbf{Y} := g^y, \mathbf{Z} := g^{xy+(1-b)z}$
3. $\mathsf{Sym} \leftarrow \mathsf{A}(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$
4. If $\mathsf{Sym} = \mathsf{Acc}$ then output 1, and otherwise output 0

Figure 6: An example of a decisional algebraic game relative to a cyclic group $\mathcal{G} = (\mathbb{G}, p, g)$ and an adversary A. The game $\mathbf{DDH}_{\mathcal{G},b}^{\mathsf{A}}$ captures the Decisional Diffie-Hellman problem.

Algebraic distinguishers. Roughly, a distinguisher A taking part in a decisional algebraic security game is algebraic, if together with its decision bit it also outputs a sequence of $\mathbb{Z}_p$ elements which describe a zero test (in the exponent) that is passed by the group elements given as input to the algorithm. Moreover, the definition requires that if A distinguishes between two games $\mathbf{G}_0$ and $\mathbf{G}_1$ with advantage ϵ, the zero test it outputs needs to be a "good separator" between the two games with probability related to ϵ (where the probability is taken over the randomness of A and the random choices of either $\mathbf{G}_0$ or $\mathbf{G}_1$).

More formally, the definition of algebraic distinguishers (i.e., algebraic algorithms participating in decisional games) uses to following notation. For an algebraic game $\mathbf{G}$, a group description $\mathcal{G}$

and an algorithm A, the notation $\mathsf{View}_{\mathsf{A}}^{\mathbf{G}_{\mathcal{G}}}$ denotes the random variable which is the view of A in the game $\mathbf{G}_{\mathcal{G}}$. The view of A consists of its randomness, its input, and all incoming messages that it receives throughout the game. For a vector $\vec{w}$ of integers, we denote by $\left[\mathsf{View}_{\mathsf{A}}^{\mathbf{G}_{\mathcal{G}}}\right]_{supp(\vec{w})}$ the random variable obtained from $\mathsf{View}_{\mathsf{A}}^{\mathbf{G}_{\mathcal{G}}}$ by omitting all group elements whose corresponding entry in $\vec{w}$ is 0 (where the ith group element observed by A is naturally associated with the ith entry of $\vec{w}$). That is, for a fixed vector $\vec{w}$, the distribution associated with $\left[\mathsf{View}_{\mathsf{A}}^{\mathbf{G}_{\mathcal{G}}}\right]_{supp(\vec{w})}$ is defined by first sampling a view V according to $\mathsf{View}_{\mathsf{A}}^{\mathbf{G}_{\mathcal{G}}}$; and then for each $i \in [\min\{k, m\}]$ for which $w_i = 0$, replacing the ith group element in V with the unique erasure symbol $\perp$, where m is the number of group elements in V. Hence, fixing $\vec{w}$, the random variable $\left[\mathsf{View}_{\mathsf{A}}^{\mathbf{G}_{\mathcal{G}}}\right]_{supp(\vec{w})}$ is defined over the randomness of A and of the challenger in $\mathbf{G}_{\mathcal{G}}$. For two random variables X_1 and X_2, we use the notation $X_1 \not\equiv X_2$ to indicate that X_1 and X_2 are *not* identically distributed.

Definition A.1. Let $\mathcal{G} = (\mathbb{G}, p, g)$ be a description of a cyclic group. An algorithm A participating in an algebraic game with parameters $\mathcal{G}$ is said to be *algebraic* if it is computationally-algebraic (per Definition 3.1) and in addition, whenever A outputs either the Acc or the Rej symbols, it also outputs a vector $\vec{w}$ of elements in $\mathbb{Z}_p$ such that the following conditions hold:

1. $\prod_{i=0}^{k} \mathbf{X}_i^{w_i} = 1_{\mathbb{G}}$, where $\mathbf{X}_1, \ldots, \mathbf{X}_k$ are the group elements that A has received so far in the game, $\mathbf{X}_0 = g$ and $1_{\mathbb{G}}$ is the identity element of $\mathbb{G}$.

2. For any two decisional algebraic games $\mathbf{G}$ and $\mathbf{G}'$, there exists $\mathbf{H} \in \{\mathbf{G}, \mathbf{G}'\}$ such that

$$\Pr_{\vec{w}}\left[\left[\mathsf{View}_{\mathsf{A}}^{\mathbf{G}_{\mathcal{G}}}\right]_{supp(\vec{w})} \not\equiv \left[\mathsf{View}_{\mathsf{A}}^{\mathbf{G}'_{\mathcal{G}}}\right]_{supp(\vec{w})}\right] \geq \frac{\epsilon}{t^2},$$

   where $\epsilon = \mathbf{Adv}_{\mathsf{A}}^{\mathbf{G}_{\mathcal{G}}, \mathbf{G}'_{\mathcal{G}}}$, $t = \mathbf{Time}_{\mathsf{A}}^{\mathbf{H}_{\mathcal{G}}}$, and the probability is taken over the choice of $\vec{w}$ induced by a random execution of $\mathbf{H}_{\mathcal{G}}$ with A.

We clarify that the probability in the second condition of Definition A.1 is over the choice of vector $\vec{w}$ in a random execution of $\mathbf{H}_{\mathcal{G}}$ with A; meaning, it is taken over the randomness of A and of the challenger in $\mathbf{H}_{\mathcal{G}}$. The event inside the probability means that for the chosen $\vec{w}$, the random variable $\left[\mathsf{View}_{\mathsf{A}}^{\mathbf{G}_{\mathcal{G}}}\right]_{supp(\vec{w})}$ is distributed differently than the random variable $\left[\mathsf{View}_{\mathsf{A}}^{\mathbf{G}'_{\mathcal{G}}}\right]_{supp(\vec{w})}$. For a discussion on the rationale underlying the DAGM, we refer the reader to [RS20a].

A.2 The Bilinear DAGM

Our definition of algebraic distinguishers with respect to asymmetric bilinear groups extends the intuition underlying that of Definition A.1. The main difference is the following: In cyclic groups (which are not equipped with a bilinear map), the zero test that an algorithm describes to explain its decision is linear in the exponents of the input elements. In contrast, in bilinear groups we permit zero tests of degree 2, in accordance with the specific structure of the group (that is, of which type it is).

For a vector $\vec{r}$ and matrices A, B, C, we denote by $\left[\mathsf{View}_{\mathsf{A}}^{\mathbf{G}_{\mathcal{G}}}\right]_{supp(\vec{r},A,B,C)}$ the random variable obtained from $\mathsf{View}_{\mathsf{A}}^{\mathbf{G}_{\mathcal{G}}}$ by omitting from it all group elements for which *all* corresponding entries of $\vec{r}, A, B$ and C are 0. Formally, for a game $\mathbf{G}$, a group $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$ of type $\tau \in \{1, 2, 3\}$, an algorithm A, a vector $\vec{r}$ and matrices $A = (a_{i,j})_{i,j}, B = (b_{i,j})_{i,j}, C = (c_{i,j})_{i,j}$, the

distribution of $\left[\mathsf{View}_{\mathsf{A}}^{\mathbf{G}_{\mathcal{G}}}\right]_{supp(\vec{r},A,B,C)}$ is defined by first sampling a view V according to $\mathsf{View}_{\mathsf{A}}^{\mathbf{G}_{\mathcal{G}}}$, then erasing (replacing with $\perp$) the ith $\mathbb{G}_T$ element in V for each i for which $r_i = 0$, and then erasing $\mathbb{G}_1$ elements and $\mathbb{G}_2$ elements according to the following rules:

- If $\tau = 1$:
  - We erase the ith $\mathbb{G}_1$ element in V if for every j it holds that $a_{i,j} = 0$, for every ℓ it holds that $b_{i,\ell} = 0$ and for every k it holds that $b_{k,i} = 0$.
  - We erase the ith $\mathbb{G}_2$ element in V if for every j it holds that $a_{j,i} = 0$, for every ℓ it holds that $c_{\ell,i} = 0$ and for every k it holds that $c_{i,k} = 0$.
- If $\tau = 2$:
  - We erase the ith $\mathbb{G}_1$ element in V if for every j it holds that $a_{i,j} = 0$.
  - We erase the ith $\mathbb{G}_2$ element in V if for every j it holds that $a_{j,i} = 0$, for every ℓ it holds that $c_{\ell,i} = 0$ and for every k it holds that $c_{i,k} = 0$.
- If $\tau = 3$:
  - We erase the ith $\mathbb{G}_1$ element in V if for every j it holds that $a_{i,j} = 0$.
  - We erase the ith $\mathbb{G}_2$ element in V if for every j it holds that $a_{j,i} = 0$.

Equipped with this notation, we now present our definition for the DAGM in asymmetric bilinear groups.

Definition A.2. Let $\tau \in \{1, 2, 3\}$, and let $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$ be a description of a bilinear group. An algorithm A participating in an algebraic game $\mathbf{G}$ with parameters $\mathcal{G}$ is said to be *algebraic* if whenever A outputs either Acc or Rej, it also provides an additional output as follows. Let $\mathbf{X}_1, \ldots, \mathbf{X}_\ell \in \mathbb{G}_1$, $\mathbf{Y}_1, \ldots, \mathbf{Y}_m$ and $\mathbf{W}_1, \ldots, \mathbf{W}_t$ be the group elements received by A in $\mathbf{G}$ so far. Then, A also outputs a vector $\vec{r} \in \mathbb{Z}_p^*$ and matrices $A = (a_{i,j}), B = (b_{i,j})$ and $C = (c_{i,j})$ such that the following conditions hold:

1. It holds that:
   - If $\tau = 1$ then $\prod_{i,j} e(\mathbf{X}_i, \mathbf{Y}_j)^{a_{i,j}} \cdot \prod_{i,j} e(\mathbf{X}_i, \phi(\mathbf{X}_j))^{b_{i,j}} \cdot \prod_{i,j} e(\psi(\mathbf{Y}_i), \mathbf{Y}_j)^{c_{i,j}} \cdot \prod_i \mathbf{W}_i^{r_i} = 1_{\mathbb{G}_T}$.
   - If $\tau = 2$ then $\prod_{i,j} e(\mathbf{X}_i, \mathbf{Y}_j)^{a_{i,j}} \cdot \prod_{i,j} e(\psi(\mathbf{Y}_i), \mathbf{Y}_j)^{c_{i,j}} \cdot \prod_i \mathbf{W}_i^{r_i} = 1_{\mathbb{G}_T}$.
   - If $\tau = 3$ then $\prod_{i,j} e(\mathbf{X}_i, \mathbf{Y}_j)^{a_{i,j}} \cdot \prod_i \mathbf{W}_i^{r_i} = 1_{\mathbb{G}_T}$.
2. For any two decisional algebraic games $\mathbf{G}$ and $\mathbf{G}'$, there exists $\mathbf{H} \in \{\mathbf{G}, \mathbf{G}'\}$ such that

$$\Pr_{(\vec{r},A,B,C)}\left[\left[\mathsf{View}_{\mathsf{A}}^{\mathbf{G}_{\mathcal{G}}}\right]_{supp(\vec{r},A,B,C)} \not\equiv \left[\mathsf{View}_{\mathsf{A}}^{\mathbf{G}'_{\mathcal{G}}}\right]_{supp(\vec{r},A,B,C)}\right] \geq \frac{\epsilon}{t^2},$$

   where $\epsilon = \mathbf{Adv}_{\mathsf{A}}^{\mathbf{G}_{\mathcal{G}},\mathbf{G}'_{\mathcal{G}}}$, $t = \mathbf{Time}_{\mathsf{A}}^{\mathbf{H}_{\mathcal{G}}}$, and the probability is taken over the choice of $(\vec{r}, A, B, C)$ induced by a random execution of $\mathbf{H}_{\mathcal{G}}$ with A.

Intuitively, Definition A.2 considers only zero tests in the target group. This is without loss of generality, since any zero test in one of the source groups can be converted into a zero test in the target group using the bilinear map e. As a concrete example, consider a bilinear group $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$ of type 2, group elements $\mathbf{X}_1, \ldots, \mathbf{X}_k \in \mathbb{G}_1$ and $\mathbf{Y}_1, \ldots, \mathbf{Y}_m$ and vectors $\vec{v}, \vec{w}$ such that $\prod_{i\in[k]} \mathbf{X}_i^{v_i} \cdot \prod_{j\in[m]} \psi(\mathbf{Y}_j)^{w_j} = 1_{\mathbb{G}_1}$. In this case, it also holds that $\prod_{i\in[k]} e(\mathbf{X}_i, g_2)^{v_i} \cdot \prod_{j\in[m]} e(g_1, \mathbf{Y}_j)^{w_j} = 1_{\mathbb{G}_T}$.

B The Algebraic Hardness of the Decisional Uber Problem in Bilinear Groups

In this section we extend our bounds from Section 5 to the decisional Uber problem in the bilinear groups, within the bilinear DAGM defined above. We start by formally defining the decisional Uber problem in bilinear groups and then prove our reduction.

B.1 The Decisional Uber Problem in Bilinear Groups

Similarly to the computational case, a decisional problem in the Uber family is parameterized by an integer m, three vectors $\vec{F} = (F_1, \ldots, F_k), \vec{H} = (H_1, \ldots, H_k)$ and $\vec{K} = (K, \ldots, K_k)$ of m-variate polynomials over $\mathbb{Z}_p$, and a polynomial Q. The adversary is given $g_1^{F_1(\vec{x})}, \ldots, g_1^{F_k(\vec{x})}, g_2^{H_1(\vec{x})}, \ldots, g_2^{H_k(\vec{x})}$ and $g_T^{K_1(\vec{x})}, \ldots, g_T^{K_k(\vec{x})}$ for a uniformly chosen $\vec{x} \leftarrow \mathbb{Z}_p^m$, along with an additional challenge element $\mathbf{W}$ in $\mathbb{G}_T$, and her goal is to distinguish between the case in which $\mathbf{W} = g_T^{Q(\vec{x})}$ and the case in $\mathbf{W}$ is a uniformly-random element of $\mathbb{G}_T$. An assumption in the family is defined via the algebraic game $(\vec{F}, \vec{H}, \vec{K}, Q)$-**DUBER** in Figure 7.[11]

$\underline{(\vec{F}, \vec{H}, \vec{K}, Q)\text{-}\mathbf{DUBER}^{\mathsf{A}}_{\mathcal{G},b}}$

1. $x_1, \ldots, x_m, r \leftarrow \mathbb{Z}_p$
2. $\vec{\mathbf{X}} := g_1^{\vec{F}(x_1,\ldots,x_m)}$
3. $\vec{\mathbf{Y}} := g_2^{\vec{H}(x_1,\ldots,x_m)}$
4. $\vec{\mathbf{Z}} := g_2^{\vec{K}(x_1,\ldots,x_m)}$
5. $\mathbf{W} := g_T^{Q(x_1,\ldots,x_m)+(1-b)\cdot r}$
6. $\mathsf{Sym} \leftarrow \mathsf{A}(\vec{\mathbf{X}}, \vec{\mathbf{Y}}, \vec{\mathbf{Z}}, \mathbf{W})$
7. If $\mathsf{Sym} = \mathsf{Acc}$ then output 1 and otherwise output 0

Figure 7: The game $(\vec{F}, \vec{H}, \vec{K}, Q)$-$\mathbf{DUBER}^{\mathsf{A}}_{\mathcal{G},b}$ capturing the decisional Uber problem in bilinear groups with respect to parameters $(\vec{F}, \vec{H}, \vec{K}, Q)$, a bilinear group $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e, \phi, \psi, p)$, a bit $b \in \{0, 1\}$ and an adversary A. The case $b = 0$ corresponds to the case in which $\mathbf{W}$ is a uniformly sampled $\mathbb{G}_T$ element, and the case $b = 1$ corresponds to the case in which $\mathbf{W} = g_T^{Q(x_1,\ldots,x_m)}$.

Note that if one can efficiently compute $g_T^{Q(x_1,\ldots,x_m)}$ through a sequence of group operations and bilinear map operations, given access to $g_1^{F_1(\vec{X})}, \ldots, g_1^{F_k(\vec{X})}, g_2^{H_1(\vec{X})}, \ldots, g_2^{H_k(\vec{X})}$ and $g_T^{K_1(\vec{X})}, \ldots, g_T^{K_k(\vec{X})}$, then the related problem can be trivially solved: The distinguisher will simply compute $g_T^{Q(x_1,\ldots,x_m)}$ and check if the result is equal to the challenge element $\mathbf{W}$. To rule out such trivial distinguishing attacks, we define a non-triviality condition for the parameters of a problem in the decisional Uber family. The definition relies on Definition 5.2, which defined type-τ (bilinear) dependence.

Definition B.1. Let $p \in \mathbb{N}$ be a prime, let $m, k \in \mathbb{N}$ be integers, let $\vec{F}, \vec{H}, \vec{K} \in (\mathbb{Z}_p[X_1, \ldots, X_m])^k$ be tuples of polynomials, and let $Q \in \mathbb{Z}_p[X_1, \ldots, X_m]$ be a polynomial. We say that the parameters $(\vec{F}, \vec{H}, \vec{K}, Q)$ are *non-trivial for groups of type* τ for $\tau \in \{1, 2, 3\}$ if Q is type-τ independent of $\vec{F}, \vec{H}, \vec{K}$.

[11] Note that there is no reason to consider challenge elements in the source groups, since such elements can always be mapped into challenge elements in the target group using the bilinear map. Hence, for each $i \in \{1, 2\}$, an assumption about the hardness of distinguishing between $g_i^{Q(x_1,\ldots,x_m)}$ and a uniformly-sampled element in $\mathbb{G}_i$, can always be expressed as an assumption about the hardness of distinguishing between $g_T^{Q(x_1,\ldots,x_m)}$ and a uniformly-sampled element in $\mathbb{G}_T$.

B.2 From q-DLOG to DUBER

Our main result in this section is that within the bilinear DAGM, the hardness of the $(\vec{F},\vec{H},\vec{K},Q)$-**UBER** problem with non-trivial parameters is implied by the hardness of q-**DLOG**. This strengthens a prior result by Rotem and Segev [RS20a] who proved a similar result, but only considering symmetric bilinear groups, and for much larger values of q. In our result below, q will be lower bounded by the maximal degree in which a variable appears in a polynomial among $\vec{F},\vec{H},\vec{K}$ and Q, whereas in the work of Rotem and Segev, q even greater than the maximal total degree among these polynomial.

We recall our notation from Section 5. For a polynomial $F \in \mathbb{Z}_p[X_1,\ldots,X_m]$ and for every $i \in [m]$, we denote by $\mathsf{deg}_{X_i}(F)$ the degree of F in the variable X_i. For a bilinear group $\mathcal{G} = (\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,p,g_1,g_2,e)$, for $i \in \{1,2\}$ and for a parameter $q \in \mathbb{N}$, we denote by q-$\mathbf{DLOG}_{\mathcal{G}_i}$, the game q-**DLOG** with respect to the source group $\mathbb{G}_i$ (i.e., the parameters of the game are the description $\mathcal{G}_i = (\mathbb{G}_i,p,g_i)$).

Theorem B.2. *Let $\mathcal{G} = (\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,p,g_1,g_2,e)$ be a description of a bilinear group of type $\tau \in \{1,2,3\}$ and let $k,m \in \mathbb{N}$ be integers. Let $\vec{F},\vec{H},\vec{K} \in (\mathbb{Z}_p[X_1,\ldots,X_m])^k$ be k-tuples of polynomials, let $d_{\vec{F}} = \max\{\mathsf{deg}_{X_j}(F_i)\}_{i\in[k],j\in[m]}$, $d_{\vec{H}} = \max\{\mathsf{deg}_{X_j}(H_i)\}_{i\in[k],j\in[m]}$ and $d_{\vec{K}} = \max\{\mathsf{deg}_{X_j}(K_i)\}_{i\in[k],j\in[m]}$, let $Q \in \mathbb{Z}_p[X_1,\ldots,X_m]$ be a polynomial and let $d_Q = \max_{j\in[n]}\{\mathsf{deg}_{X_j}(Q)\}$. Let $q \geq \max\{d_{\vec{F}},d_{\vec{H}},d_{\vec{K}}/2, d_Q/2\}$, and let $q_1 \geq d_{\vec{F}}$ and $q_2 \geq d_{\vec{H}}$ such that $q_1+q_2 \geq \max\{d_{\vec{K}},d_Q\}$. If $(\vec{F},\vec{H},\vec{K},Q)$ is non-trivial for type τ, then for any algebraic algorithm A there exist algebraic algorithms $\mathsf{B}_1,\mathsf{B}_2,\mathsf{B}_3$ and B_4 such that $\mathbf{Time}_{\mathsf{B}_i}^{q\text{-}\mathbf{DLOG}_{\mathcal{G}}} \leq t$ for each $i \in \{1,2,3,4\}$, and:*

- *If $\tau = 2$: $\mathbf{Adv}_{\mathsf{B}_1}^{q\text{-}\mathbf{DLOG}_{\mathcal{G}_2}} \geq \epsilon/4t^2(m+1)$;*
- *If $\tau = 1$: $\mathbf{Adv}_{\mathsf{B}_2}^{q\text{-}\mathbf{DLOG}_{\mathcal{G}_2}} \geq \epsilon/4t^2(m+1)$ and $\mathbf{Adv}_{\mathsf{B}_3}^{q\text{-}\mathbf{DLOG}_{\mathcal{G}_1}} \geq \epsilon/4t^2(m+1)$;*
- *If $\tau = 3$: $\mathbf{Adv}_{\mathsf{B}_4}^{(q_1,q_2)\text{-}\mathbf{DLOG}_{\mathcal{G}}} \geq \epsilon/4t^2(m+1)$;*

where $t = \mathbf{Time}_{\mathsf{A}}^{(\vec{F},\vec{H},\vec{K},Q)\text{-}\mathbf{DUBER}_{\mathcal{G}}} + \mathsf{poly}(m,k,q,q_1,q_2,\log p)$ and $\epsilon = \mathbf{Adv}_{\mathsf{A}}^{(\vec{F},\vec{H},\vec{K},Q)\text{-}\mathbf{DUBER}_{\mathcal{G}}}$.

The proof of Theorem B.2 uses similar ideas as the proof of 5.4, extending them to work with the definition of the bilinear DAGM. Hence, in the proof of Theorem B.2, we focus mainly on the differences from the proof of Theorem 5.4.

Proof of Theorem B.2. We focus on the case of type-2 bilinear groups. The extension of the proof to type 1 and to type 3 bilinear groups is obtained as in the proof of Theorem 5.4. Let A be an algebraic algorithm participating in $(\vec{F},\vec{H},\vec{K},Q)$-$\mathbf{DUBER}_{\mathcal{G},b}$, and consider the following algorithm B_1 participating in q-$\mathbf{DLOG}_{\mathcal{G}_2}$

Algorithm B_1

Input: q elements $\mathbf{S}_1 = g_2^x,\ldots,\mathbf{S}_q = g_2^{x^q}$ in $\mathbb{G}_2$ for $x \leftarrow \mathbb{Z}_p$.

1. Sample $b \leftarrow \{0,1\}$, $i^* \leftarrow [m+1]$, and $\alpha_1,\ldots,\alpha_{i^*-1},\alpha_{i^*+1},\ldots,\alpha_{m+1} \leftarrow \mathbb{Z}_p$.
2. Compute $\mathbf{W} = g_T^{Q(\vec{x})+(1-b)\cdot\alpha_{m+1}}$, and for $i = 1,\ldots,k$, compute $\mathbf{X}_i := g_1^{F_i(\vec{x})}$, $\mathbf{Y}_i := g_2^{H_i(\vec{x})}$ and $\mathbf{Z}_i := g_T^{K_i(\vec{x})}$, where $\vec{x} = (\alpha_1,\ldots,\alpha_{i^*-1},x,\alpha_{i^*+1},\ldots,\alpha_m)$.
   [these elements can be computed in accordance with our discussion in the proof of Theorem 5.4]
3. Invoke $\mathsf{A}(\mathbf{X}_1,\ldots,\mathbf{X}_k,\mathbf{Y}_1,\ldots,\mathbf{Y}_k,\mathbf{Z}_1,\ldots,\mathbf{Z}_k,\mathbf{W})$ to obtain a decision symbol $\mathsf{Sym} \in \{\mathsf{Acc},\mathsf{Rej}\}$.

Since A is algebraic, it also outputs a vector $\vec{r}$ and matrices $A = (a_{i,j}), B = (b_{i,j})$ and $C = (c_{i,j})$ such that:

$$\prod_{i,j} e(\mathbf{X}_i, \mathbf{Y}_j)^{a_{i,j}} \cdot \prod_{i,j} e(\psi(\mathbf{Y}_i), \mathbf{Y}_j)^{c_{i,j}} \cdot \prod_{i \in [k]} \mathbf{Z}_i^{r_i} \cdot \mathbf{W}^{r_{k+1}} = 1_{\mathbb{G}_T}.$$

4. Define the $(m+1)$-variate polynomial

$$\begin{aligned} &f_{\vec{r},A,C}(X_1, \ldots, X_{m+1}) \\ &\quad = r_{k+1} \cdot \Big(b \cdot Q(\vec{X}) + (1-b) \cdot X_{m+1}\Big) + \sum_{i,j} a_{i,j} \cdot F_i(\vec{X}) \cdot H_j(\vec{X}) \\ &\qquad + \sum_{i,j} c_{i,j} \cdot H_i(\vec{X}) \cdot H_j(\vec{X}) + \sum_i r_i \cdot K_i(\vec{X}), \end{aligned}$$

   where $\vec{X} = (X_1, \ldots, X_m)$, compute the sequence of polynomials $\mathcal{S}(f_{\vec{r},A,C})$ and denote these polynomials by $f_1, \ldots, f_{m+1}$.
5. Find all roots $x_1^*, \ldots, x_\ell^*$ of the univariate polynomial $h(X_{i^*}) = f_{i^*}(X_{i^*}, \alpha_{i^*+1}, \ldots, \alpha_m)$.
6. For every $i = 1, \ldots, \ell$, check if $g^{x_i^*} = \mathbf{X}$. If so, output x_i^* and terminate; otherwise, continue.
7. If reached, output $\perp$ and terminate.

Since A is an algebraic distinguisher (in accordance with Definition A.2), there exists a bit $b^* \in \{0,1\}$, such that

$$\Pr_{(\vec{r},A,B,C)} \left[\begin{array}{c} \left[\mathsf{View}_{\mathsf{A}}^{(\vec{F},\vec{H},\vec{K},Q)\text{-}\mathbf{DUBER}_{\mathcal{G},0}}\right]_{supp(\vec{r},A,B,C)} \\ \not\equiv \\ \left[\mathsf{View}_{\mathsf{A}}^{(\vec{F},\vec{H},\vec{K},Q)\text{-}\mathbf{DUBER}_{\mathcal{G},1}}\right]_{supp(\vec{r},A,B,C)} \end{array} \right] \geq \frac{\epsilon}{t^2}, \tag{B.1}$$

where the probability is taken over the choice of $(\vec{r}, A, B, C)$ induced by the randomness of A and of the challenger in $(\vec{F}, \vec{H}, \vec{K}, Q)$-$\mathbf{DUBER}^{\mathsf{A}}_{\mathcal{G},b^*}$. Observe that the predicate in the probability in Eq. (B.1) is satisfied if and only if $r_{k+1} \neq 0$, and hence

$$\Pr_{(\vec{r},A,B,C)} [r_{k+1} \neq 0] \geq \frac{\epsilon}{t^2}, \tag{B.2}$$

where again, the probability is taken over $(\vec{F}, \vec{H}, \vec{K}, Q)$-$\mathbf{DUBER}^{\mathsf{A}}_{\mathcal{G},b^*}$.

The probability that $b = b^*$, where b is the bit sampled by B_1 in Step 1 is $1/2$. Therefore, the probability that A (when invokes in Step 3 of B_1) outputs $\vec{r}$ such that $r_{k+1} \neq 0$ is at least $\epsilon/2t^2$; denote this event by Good. Denote $\vec{\alpha} = (\alpha_1, \ldots, \alpha_{i^*-1}, x, \alpha_{i^*+1}\alpha_m)$, where $i^* \in [m+1]$ is the index sampled by B_1 in Step 1. We wish to argue that conditioned on Good it holds that: (1) $f_{\vec{r},A,C}(\vec{\alpha}) = 0$; and that (2) the polynomial $f_{\vec{r},A,C}$ is non zero. Claim (1) immediately follows from the fact that

$$\prod_{i,j} e(\mathbf{X}_i, \mathbf{Y}_j)^{a_{i,j}} \cdot \prod_{i,j} e(\psi(\mathbf{Y}_i), \mathbf{Y}_j)^{c_{i,j}} \cdot \prod_{i \in [k]} \mathbf{Z}_i^{r_i} \cdot \mathbf{W}^{r_{k+1}} = 1_{\mathbb{G}_T}.$$

In order to prove claim (2), we consider two cases. If $b = 0$, then this holds since there is nothing that can cancel out the term $r_{k+1} \cdot X_{m+1}$, as all other monomials are only in $X_1, \ldots, X_m$. If $b = 1$, this holds since Q is 2-type independent of $(\vec{F}, \vec{H}, \vec{K})$, and hence the term $\sum_{i,j} a_{i,j} \cdot F_i(\vec{X}) \cdot H_j(\vec{X}) + \sum_{i,j} c_{i,j} \cdot H_i(\vec{X}) \cdot H_j(\vec{X}) + \sum_i r_i \cdot K_i(\vec{X})$, cannot cancel out the term $r_{k+1} \cdot Q(\vec{X})$.

The proof continues essentially identically to the proof of Theorem 5.4, to show that conditioned on Good, B_1 outputs the correct exponent x with probability at least $1/2(m+1)$. Hence, overall, we obtain that $\mathbf{Adv}^{q\text{-}\mathbf{DLOG}_{\mathcal{G}_2}}_{\mathsf{B}_1} \geq \epsilon/4t^2(m+1)$. ■