Generic-Group Lower Bounds via Reductions Between Geometric-Search Problems: With and Without Preprocessing*

Benedikt Auerbach , Charlotte Hoffmann, and Guillermo Pascual-Perez

Institute of Science and Technology Austria
{bauerbac, choffman, gpascual}@ista.ac.at

Abstract

The generic-group model (GGM) aims to capture algorithms working over groups of prime order that only rely on the group operation, but do not exploit any additional structure given by the concrete implementation of the group. In it, it is possible to prove information-theoretic lower bounds on the hardness of problems like the discrete logarithm (DL) or computational Diffie-Hellman (CDH). Thus, since its introduction, it has served as a valuable tool to assess the concrete security provided by cryptographic schemes based on such problems. A work on the related algebraic-group model (AGM) introduced a method, used by many subsequent works, to adapt GGM lower bounds for one problem to another, by means of conceptually simple reductions.

In this work, we propose an alternative approach to extend GGM bounds from one problem to another. Following an idea by Yun (Eurocrypt '15), we show that, in the GGM, the security of a large class of problems can be reduced to that of geometric search-problems. By reducing the security of the resulting geometric-search problems to variants of the search-by-hypersurface problem, for which information theoretic lower bounds exist, we give alternative proofs of several results that used the AGM approach.

The main advantage of our approach is that our reduction from geometric search-problems works, as well, for the GGM with preprocessing (more precisely the bit-fixing GGM introduced by Coretti, Dodis and Guo (Crypto '18)). As a consequence, this opens up the possibility of transferring preprocessing GGM bounds from one problem to another, also by means of simple reductions. Concretely, we prove novel preprocessing bounds on the hardness of the d-strong discrete logarithm, the d-strong Diffie-Hellman inversion, and multi-instance CDH problems, as well as a large class of Uber assumptions. Additionally, our approach applies to Shoup's GGM without additional restrictions on the query behavior of the adversary, while the recent works of Zhang, Zhou, and Katz (Asiacrypt '22) and Zhandry (Crypto '22) highlight that this is not the case for the AGM approach.

*A preliminary version of this paper appeared in the proceedings of TCC 2023. Refer to the published version via 10.1007/978-3-031-48621-0_11. This is the full version.

Contents

1 Introduction

The generic group model. The concrete security provided by a cryptographic scheme is typically assessed following the reductionist approach: one first shows that its security is implied by the hardness of a problem, and then analyzes the best running times of algorithms solving said problem. Regarding the second step, for schemes defined over a group $\mathbb{G} = \langle g \rangle$ of prime order p, the generic-group model (GGM) has proven itself a valuable tool. It is an idealized model that, on the one hand, is assumed to be meaningful for elliptic-curve groups, which are heavily relied on in practice. On the other hand, it allows to derive information-theoretic lower bounds on the number of group operations required to solve problems, like the discrete logarithm and Diffie-Hellman problems (as well as many of their variants).

The model aims to capture algorithms that are generic in the sense of being applicable to any group $\mathbb{G}$ of prime order p. Algorithms of this type only make use of the group operation, but do not not exploit any additional structure given by the concrete implementation of the group. There have been several efforts of formalize this requirement. In Shoup's definition [Sho97] of the model, the adversary gets access to group elements via abstract labels, i.e., uniformly random bitstrings, and to the group operation via an oracle. The variant by Maurer [Mau05], on the other hand, gives access to group elements using abstract handles. All problems that are definable in Maurer's GGM are also definable in Shoup's, but the other direction does not hold. In fact, Zhandry [Zha22] recently showed that Maurer's GGM (and the more commonly used extension thereof, which Zhandry calls type-safe model) fails to capture many textbook techniques that are captured by Shoup's GGM. An additional difference is that Maurer's model does not capture preprocessing algorithms. For more details on the differences between the models we refer to [Zha22].

In this work we focus on Shoup's model, which we will simply refer to as the *generic-group model*, or GGM, from here on. In it, group elements g^x are represented by labels $\sigma \in \{0,1\}^\ell$. A generic algorithm receives as input some labels and, typically, either has to compute a discrete logarithm or the label of a certain group element. To do so, it has access to a group operation oracle GrpOp. This takes as input two labels and returns the label of the product of the corresponding group elements.

As an example of how one typically argues hardness of problems in the GGM, we briefly sketch the bound on the discrete logarithm (DL) problem, as proven in [Sho97]. Here, a secret exponent $x \leftarrow_\$ \mathbb{Z}_p$ is sampled and the adversary receives as input labels σ_g, σ_{g^x} corresponding to g and g^x. In the proof, each label σ is associated to a linear function $F_\sigma \in \mathbb{Z}_p[X]$ as follows. The adversary's inputs σ_g, σ_{g^x} are associated to 1 and X, and whenever the group-operation oracle is queried on labels σ, σ', their product $\mathrm{GrpOp}(\sigma, \sigma')$ is associated to the function $F_{\sigma''} = F_\sigma + F_{\sigma'}$. Then, one checks whether $F_{\sigma''}$ equals any of the functions defined previously. If so, the corresponding label is used; if not, a fresh label σ'' is sampled. The idea being that, in this way, group elements which are equal can be identified, as every label σ corresponds to a group element of the form $g^{F_\sigma(x)}$. However, this simulation of the GGM works only as long as there exists no $F_\sigma \neq F_{\sigma'}$ with $F_\sigma(x) = F_{\sigma'}(x)$, in which case the adversary would receive two different labels for the same group element. Accordingly, the proof bounds the probability of this event happening, and, in case that it does not, the probability of the adversary winning.

In [Yun15], Yun considers a natural generalization of the discrete-logarithm problem, namely the task to solve several DL instances. In the proof of his bound, he shows that one can *perfectly* simulate the GGM group-operation oracle in a reduction from the so-called search-by-hyperplane problem (SHP). In this problem, the adversary has to find a hidden value $\vec{x} \in \mathbb{Z}_p^m$ (here m is the number of discrete logarithm instances that have to be solved) by using hyperplane queries that, on input an affine function F, return whether $F(\vec{x}) = 0$ or not; exactly what is needed in the GGM to check whether a group operation query should be answered with an already defined label. By proving an information-theoretic lower bound on the hardness of SHP, one then is able to obtain bounds on the hardness of the original problem in the GGM. This approach was later generalized by Auerbach, Giacon, and Kiltz [AGK20] to allow the function F to be a multivariate polynomial of bounded degree. This is needed, for example, if one wants to argue about problems involving decisional Diffie-Hellman oracles, or "higher-degree" problems like the d-strong discrete-logarithm problem.

The GGM and preprocessing. In practice most cryptosystems rely only on a few standardized groups, which makes preprocessing attacks particularly viable. The power of those attacks was demonstrated by Mihalcik [Mih10]; Lee, Cheon, and Hong [LCH11]; and Bernstein and Lange [BL13], who construct generic algorithms with preprocessing that solve the DL problem in a group of order p in time $p^{1/3}$. The authors thereby circumvented the lower bound in the GGM of $p^{1/2}$ without preprocessing established by Shoup [Sho97].

Two recent works extend the GGM to adversaries allowed to perform unbounded preprocessing before the problem instance is sampled. Both derive lower bounds on the hardness of variants of the discrete logarithm and Diffie-Hellman problems. Corrigan-Gibbs and Kogan [CK18] leverage compression arguments, Coretti, Dodis and Guo [CDG18] a pre-sampling technique by Unruh [Unr07]. The latter work defines two variants of the GGM allowing for preprocessing: the auxiliary input (AI-GGM) and bit-fixing (BF-GGM) generic-group models. In the AI-GGM, the adversary is able to perform unbounded preprocessing on the whole labeling function to generate an advice string of bounded size before receiving the problem instance. In the preprocessing phase of the BF-GGM, on the other hand, it is able to choose labels of a bounded number of group elements, but does not have access to the remainder of the labeling function. The authors show that, under certain conditions, bounds in the BF-GGM, which is typically easier to work with, also hold in the AI-GGM. To derive a preprocessing bound on the hardness of computing multiple discrete logarithms, the latter work also uses a reduction from SHP.

Generic group lower bounds via algebraic reductions. A related restricted class of algorithms working over $\mathbb{G}$ consists of so called *algebraic* algorithms, first considered by Boneh and Venkatesan [BV98], and later further formalized by Pallier and Vergnaud [PV05]. Fuchsbauer, Kiltz, and Loss [FKL18] abstract such algorithms in their *algebraic-group model* (AGM) as follows. While an algorithm with input $g_0, \ldots, g_k \in \mathbb{G}$ in the AGM gets explicit access to the group $\mathbb{G}$, it has to provide an algebraic justification for every element $h \in \mathbb{G}$ that it outputs. More precisely, together with h, it has to produce $a_0, \ldots, a_k \in \mathbb{Z}_p$ such that $h = \prod_{i=0}^{k} g_i^{a_i}$.

In the paper, the authors introduce an approach that, assuming existing generic-group lower bounds for problem P_1, allows to extend the bound to a different problem P_2 by means of a conceptually simple reduction, which they describe as follows.

(i) If adversary A against P_2 is generic, we may assume w.l.o.g. that it is algebraic.

(ii) Construct a generic reduction from P_1 to P_2 that exploits the algebraic justifications that A has to provide for all group elements it computes.

(iii) Now, since the existence of generic solver for P_2 implies a generic solver for P_1, and since P_1 is hard, P_2 must be as well.

As this approach is conceptually simpler than establishing GGM bounds for P_2 from scratch, and typically leads to cleaner proofs, the idea of analyzing problems and schemes in the algebraic group model was picked up by many subsequent works [ABB+20, AGK20, BFL20, FPS20, GT21, KLX22, MTT19].

As it is relevant to our discussion on preprocessing below, we provide some intuition on point (i). Here, the idea is that a generic reduction interacting with generic adversary A is able to compute the required algebraic justification by itself, as long as A queries the group-operation oracle only on labels it previously received as input. Indeed, in this case the justification can be computed inductively as follows. If σ_1 and σ_2 are the labels, and the reduction already recorded their algebraic justifications $\vec{a_1}$ and $\vec{a_2}$ in a previous step, then a justification of the product of the two group elements is given by $\vec{a_1} + \vec{a_2}$.

The AGM and preprocessing. Despite the fact that both the work on the algebraic-group model [FKL18] and the one on the GGM with preprocessing [CDG18] have been taken up in many subsequent works, the approach of transferring preprocessing bounds from one problem to ones for another with simple reductions has stayed elusive so far. One presumed reason for this is that, in this setting, one cannot argue that the reduction is able to compute an algebraic justification from the generic adversary's queries. Indeed, the

argument outlined above crucially relies on the adversary only querying labels of group elements it previously received as input. However, in the preprocessing setting, the adversary receives as input an advice string, computed during an unbounded precomputation phase. And, as the advice might contain labels not accessible to the reduction, e.g. encryptions of labels under a key hard coded into the adversary's code, this poses an obstacle to the reduction's ability to compute algebraic justifications for group elements computed by the adversary. Maurer's GGM does not allow for preprocessing (see e.g. [Zha22]).

The AGM and Shoup's GGM A recent work by Zhang, Zhou, and Katz [ZZK22] showed that the AGM approach of transferring lower bounds in Shoups's GGM outlined above requires caution. Concretely, they construct a problem, the so called bit-encoding problem, that is at least as hard as the discrete logarithm problem in the AGM, but can be trivially solved in the GGM. This shows that point (i) in the approach outlined above does not hold in general. As discussed above, one would like to argue that the reduction is able to compute the required justification of group elements produced by the adversary A by itself, which is possible if A never queries for group operations on labels it did not previously receive as input. However, this cannot be guaranteed in general, a fact that is exploited in the bit-encoding problem of [ZZK22], which can be won by returning such a label.

Note that the bit-encoding problem is definable in Shoup's GGM but not in Maurer's GGM. In fact, Zhandry [Zha22] formally proved that the AGM approach is valid for all problems that are definable in Maurer's GGM, so the AGM approach is valid for most "natural" problems. However, we point out that several results in prior work [ABB+20, AGK20, FKL18, FPS20, GT21, KLX22] argue about the generic-group model in the presence of a random oracle, as is often the case when analyzing cryptographic schemes, instead of problems purely defined over groups. Opposed to Shoup's model, random oracles have to be explicitly modeled in Maurer's model. However, it is unclear, as far as we know, whether one may assume generic algorithms to be algebraic given this additional oracle.

1.1 Our Contributions

In this work we present a new proof technique to derive lower bounds in the GGM that improves over the AGM approach in the following ways:

- It also applies to the bit-fixing generic-group model of [CDG18]. Since bounds in the BF-GGM can be carried over to the AI-GGM, this opens up the possibility of extending preprocessing bounds from one problem to another by means of a reduction between the problems;
- It applies to Shoup's GGM in its full generality.

Generalizing the idea introduced in [Yun15], we show that, in the GGM, the security of a large class of computational problems can be reduced to that of analogous geometric search-problems. We then propose to construct reductions between the obtained geometric search-problems. Interestingly, several reductions from prior work using the AGM approach turn out to have a geometric equivalent. Further, the geometric analogue of several discrete-logarithm type problems are special cases of the search-by-hypersurface problem [AGK20], for which information theoretic bounds exist. As a consequence, we obtain alternative proofs of several GGM bounds from prior work that relied on AGM reductions with the additional benefit, that for all considered problems that can also be expressed in the AI-GGM we obtain the corresponding preprocessing lower bounds essentially for free.

For a visualization, through a concrete example, of our proposed approach compared to the one using the AGM, see Figure 1. We now describe our results in more detail.

From generic-group problems to geometry. In Section 3 we show that, in the GGM, the security of a large class of computational problems can be reduced to the security of a corresponding geometric search-problem. We try to capture as many problems of interest as possible to prevent that this technical step has to be redone in future work. Thus, we phrase our result in terms of a family of Uber problems MI-Uber, in the

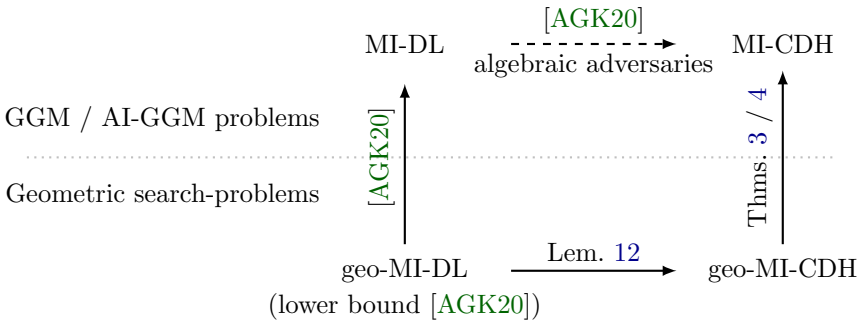

Figure 1: Our proposed way of deriving GGM and AI-GGM lower bounds at the example of the multi-instance CDH problem, compared to the approach taken in [AGK20]. Arrows indicate reductions from source to sink. The dashed arrow indicates that the reduction holds with respect to algebraic adversaries, and thus is restricted to (Maurer's) GGM, but does not apply to the AI-GGM.

style of [BBG05, RLB+08]. In this type of problems, a vector of secret exponents $\vec{x} = (x_1, \ldots, x_t)$ is sampled from $\mathbb{Z}_p^t$, and the adversary receives as input group elements of the form $g^{F(\vec{x})}$, where $F \in \mathbb{Z}_p[X_1, \ldots, X_t]$. Then, it has to compute group elements of the form $g^{F^*(\vec{x})}$, for some $F^* \in \mathbb{Z}_p(\vec{X})$. Note that our definition of MI-Uber *extends* the definitions of Uber problems in [BBG05, RLB+08]. It captures many Diffie-Hellman-type problems including e.g. the d-strong Diffie-Hellman-inversion [BB04a] problem, as we allow the target function to be rational. Further, we cover m-out-of-n multi-instance problems, in which the adversary has to produce at least m out of n target group elements, and allow access to decisional oracles, such as, for example, a decisional Diffie-Hellman (DDH) oracle.

The corresponding geometric search-problem geo-MI-Uber roughly looks as follows. A secret vector $\vec{x} = (x_1, \ldots x_t)$ is sampled uniformly at random from $\mathbb{Z}_p^t$ and the adversary has access to an evaluation oracle Eval that, on input a polynomial $\hat{F}$, returns whether the point $\vec{x}$ satisfies $\hat{F}(\vec{x}) = 0$ or not.[1] As in prior work, queries of this form are sufficient to (almost) perfectly simulate the group-operation oracle in the GGM. The problems considered in previous works were connected to variants of the discrete logarithm problem, and so the adversary's goal was to compute $\vec{x}$. In geo-MI-Uber, on the other hand, the adversary has to compute a polynomial $\hat{F}$ such that $(F^* - \hat{F})(\vec{x}) = 0$ for the challenge polynomial F^*. To prevent ending up with a trivial problem, e.g., by having the adversary simply output F^*, we have to restrict the space of admissible $\hat{F}$. Our main observation regarding this is that all solutions our reduction will obtain from a generic adversary interacting with MI-Uber will be an affine combinations of the input polynomials $\vec{F}$. Restricting the solutions in geo-MI-Uber to this form turns out to be sufficient to not end up with trivial problems. Essentially, we show the following.

> For every adversary A against MI-Uber making at most q queries, there exists an adversary B making at most $\mathcal{O}(q^2)$ queries such that
>
> $$\mathrm{Adv}^{\text{MI-Uber}}(\mathsf{B}) \geq \mathrm{Adv}^{\text{geo-MI-Uber}}(\mathsf{A}) - \frac{\mathcal{O}(d_{\max} q^2)}{p} \ ,$$
>
> where $d_{\max}$ is the highest degree of the polynomials $\vec{F}$.

The loss in our reduction stems from carefully accounting for the possibility of the adversary querying its group-operation oracle on labels it did not previously receive. We point out that our formal result allows for more flexibility regarding this error term, and shows that it can reduced exponentially, albeit at the cost

[1] We refer to these problems as geometric search-problems, since queries of this type can be seen as testing whether the hypersurface in $\mathbb{Z}_p^t$ defined by $\hat{F}$ contains $\vec{x}$ or not.

of increasing the reductions query count (see Theorem 3). When reducing from a geometric search-problem, queries of this type turn out to not be an issue, in contrast to a generic reduction interacting with a generic algorithm. The main difference is that, here, the reduction simulates the labeling function. Thus, undefined labels simply correspond to uniformly random, unused group elements, that can be sampled by the reduction itself. However, additional Eval queries are required to ensure consistency with the previous simulation and, in unlikely events, the reduction might get unlucky and not find an appropriate group element. We point out that, so far, queries of this type were either assumed to not occur [AGK20, Yun15] or not accounted for in the advantage [CDG18].

We extend our result in two ways. First, we show that an analogous result holds in the bit-fixing generic-group model (BF-GGM) of [CDG18]. As the paper uses a reduction from SHP to argue about the hardness of solving multiple discrete logarithms in the preprocessing setting, it is not surprising that our reduction from geo-MI-Uber to MI-Uber carries over to the BF-GGM. However, it requires additional queries to account for the labels chosen by the adversary.

Finally, we show an analogous result for the generic-group model for bilinear groups. We cover groups of types 1, 2, and 3. The main additional challenge in this setting is to carefully restrict the range of admissible queries to the evaluation oracle according to the bilinear group's type.

As we work in Shoup's model, our approach is compatible with nonprogrammable random oracles, which in this setting either take as input or have as output labels in $\{0,1\}^\ell$. As an example of a reduction to a geometric search-problem in the programmable ROM, we revisit the treatment of BLS signatures from [FKL18], establishing the same GGM bound of q^2/p.

Reductions between geometric search-problems and application to concrete problems. In Section 4 we derive generic group lower bounds on the hardness of several problems in the GGM, the AI-GGM, and the bilinear GGM. To do so, we construct simple reductions between the geometric analogue geo-P of the problem P and variants of the search-by-hypersurface (SHS) problem of [AGK20]. In the easiest case, both problems are defined with respect to the same oracle Eval, and the reduction can simply forward all queries and find the solution to SHS among the roots of polynomials related to the one output by the geo-P solver as a solution. In other cases, where SHS is defined with respect to secrets $(z_1, \ldots, z_s)$, and P expects input $(x_1, \ldots, x_t)$ with $s \leq t$, our reductions will implicitly set $x_i = a_0 + \sum_{i=0}^{s} a_i z_i$ with known $a_i \in \mathbb{Z}_p$. This enables them to answer $\mathrm{Eval}_{\text{geo-P}}(F)$ queries by the adversary, for $F \in \mathbb{Z}_p[X_1, \ldots, X_t]$, with the response to $\mathrm{Eval}_{\mathrm{SHS}}(F(X'_1(\vec{Z}), \ldots, X'_t(\vec{Z})))$, where $X'_i = a_0 + \sum_{i=0}^{s} a_i Z_i$. Again, the reduction will solve its SHS challenge by returning a root of a polynomial related to the one output as a solution by the geo-P solver.

We point out that our reductions are very close in concept to typical reductions in the AGM. In those, the reduction also translates an algebraic justification into a polynomial, and solves its DL challenge by finding its roots. Similarly, the processing of Eval challenges corresponds to an AGM reduction re-randomizing and expanding its challenge. An example of this would be the generation of a CDH challenge $(X = Z^{a_x} g^{b_x}, Y = Z^{a_y} g^{b_y})$ from a discrete-logarithm challenge $Z \in \mathbb{G}$ using known exponents $a_x, a_y, b_x, b_y \in \mathbb{Z}_p$.

As a consequence, some of our reductions can be seen as easy, direct translations of reductions from prior work to the geometric setting. We see this as an attractive feature of our approach. Concretely, we are able to formally justify the bounds using the AGM + RO approach for the multi-instance gap-CDH problem [AGK20] (targeted at Hashed-ElGamal key-encapsulation) and BLS signatures [FKL18]. Further, we derive new preprocessing bounds for the d-strong discrete logarithm, d-strong Diffie-Hellman inversion, and multi-instance CDH problems, as well as a large class of Uber assumptions. Regarding the latter, a recent work by Bauer, Farshim, Harasser, and O'Neill [BFHO22] proves a lower bound in the AI-GGM for a decisional Uber problem. In turn their bound holds also for the easier, corresponding computational Uber problem. However, the bound obtained with our approach substantially improves on it. For an overview on our bounds see Table 1.

Open questions and future work. Our results are limited to computational problems. So, a natural question is whether decisional problems like DDH also have a geometric equivalent; and, if so, whether reductions to SHS variants are possible, e.g., following an analogous approach to the one taken by Rotem

Model	Problem	Bound	See
GGM			
	(m,n)MI-gap-DL, (m,n)MI-gap-CDH	$\left(\frac{q^2}{mp}\right)^m$	[AGK20]*,**
		$\left(\frac{rq^2}{mp}\right)^m + q\left(\frac{q}{p}\right)^r$	Cor. 13
AI-GGM			
	d-strong-DL, d-strong-DHI	$\frac{d(sq^2+q^2)}{p}$	Cor. 7
	(m,m)MI-DL	$\left(\frac{q^2s+q^2}{mp}\right)^m$	[CDG18]**
	(m,n)MI-DL, (m,n)MI-CDH	$\left(\frac{q^2s+rq^2}{mp}\right)^m + q\left(\frac{q}{p}\right)^r$	Cor. 14
	Uber	$\frac{dq^2}{p} + \sqrt{\frac{sq^2}{p}}$	[BFHO22]
	Uber	$\frac{d(sq^2+q^2)}{p}$	Cor. 11
Bil GGM			
	Uber_ϕ	$\frac{dq^2}{p}$	[BBG05, RLB+08], Cor. 16
	BLS signatures	$\frac{q^2+q_{\text{RO}}^2}{p}$	[FKL18]*, Cor. 18

Table 1: Our GGM and AI-GGM bounds on the advantage of adversaries in groups of size p. Integer q denotes the number of queries, s the size of the advice string, the expressions are to be understood as $\tilde{\mathcal{O}}$. For problem (m,n)MI-CDH, $n \geq m$ denotes the number of challenges, m the number of required solutions, and the bounds hold for arbitrary r (see Remark 1 for a comparison to prior work). For problems Uber and Uber_ϕ we denote by d the largest degree of the input polynomials. For BLS signatures, we denote by q_{RO} the number of random oracle queries made by the adversary. Bounds without references are new and for the other ones we give alternative proofs. References marked with * proved the respective bounds using AGM + RO. In references marked with ** it is assumed that adversaries never query on labels they did not previously receive as input or such queries are not accounted for in the computation of the advantage.

and Segev [RS20], who extend the definition of the AGM to capture decisional problems. A second interesting direction would be to extend the equivalence results of BF-GGM and AI-GGM from [CDG18] to allow for decisional oracles, as this would open up the possibility of proving preprocessing bounds in the bilinear GGM via simple reductions.

Further related work. The gap-CDH problem was first introduced by Okamoto and Pointcheval [OP01]. Ying and Kunihiro [YK17] prove GGM lower bounds on the hardness of (m,n)MI-DL. Bauer, Fuchsbauer, and Plouviez [FPS20] on the hardness of the one-more-discrete logarithm problem. The latter uses techniques reminiscent of [Yun15]. Blocki and Lee [BL22] prove preprocessing GGM bounds on the hardness of $(1,n)$MI-DL.

2 Preliminaries

2.1 Notation

We use the following conventions. We denote the set of natural numbers up to n by $[n] := \{1,\ldots,n\}$ and the set including 0 by $[n]_0 := \{0,\ldots,n\}$. Typically we use lower case letters to refer to elements of $\mathbb{Z}$ or $\mathbb{R}$, and upper case letters for indeterminants or functions. For prime p and vector of indeterminants $\vec{X}$ we often work over the multivariate ring of polynomials $\mathbb{Z}_p[\vec{X}]$, which we will sometimes see as a vector space over $\mathbb{Z}_p$. For

a set of polynomials $\mathcal{F} = \{F_1, \ldots, F_k\}$, we denote by $\mathrm{Span}(\mathcal{F}) := \{F \in \mathbb{Z}_p[\vec{X}] \mid \exists a_i \in \mathbb{Z}_p : F = \sum_{i=1}^{k} a_i F_i\}$ its linear span. The ring of rational functions is denoted by $\mathbb{Z}_p(\vec{X}) := \{F_1/F_2 \mid F_1, F_2 \in \mathbb{Z}_p[\vec{X}], F_2 \neq 0\}$.

Algorithms A are typically depicted using sans-serif font. Throughout this work we assume that $p \in \mathbb{N}$ is a fixed prime, known to all adversaries and reductions. We denote the truth value of a statement E by $[E]$.

2.2 Generic-Group Model

We recall Shoup's generic group model (GGM). We consider 4 variants of it: the original one as introduced in [Sho97], its extension to bilinear groups [BB04b], and the auxiliar-input (i.e., preprocessing) and bit-fixing variants introduced in [CDG18].

Generic-group model. We consider groups $\mathbb{G} = \langle g \rangle$ of prime order p generated by g. While we use this notation for ease of exposition when giving intuitive descriptions of problems over $\mathbb{G}$, if we explicitly work in the generic group model, we identify $(\mathbb{G}, \cdot)$ with $(\mathbb{Z}_p, +)$ via the isomorphism $x \mapsto g^x$. In the GGM, adversaries get access to group elements via abstract labels, and to the group operation via an oracle. More precisely, let $\ell \geq \lceil \log(p) \rceil$ and let $\mathcal{L} : \mathbb{Z}_p \hookrightarrow \{0,1\}^\ell$ be an injection sampled uniformly at random from the set of all injections into $\{0,1\}^\ell$. We denote the range of $\mathcal{L}$ by $\mathcal{R} := \mathcal{L}(\mathbb{Z}_p) \subseteq \{0,1\}^\ell$. An adversary A in the generic group model receives as input labels $\sigma_0, \ldots, \sigma_t$, with $\sigma_i = \mathcal{L}(h_i)$ for some group elements $h_i \in \mathbb{Z}_p$. Typically, it has to compute either the label of some group element or some discrete logarithm. It has access to the group operation via the oracle $\mathrm{GrpOp}(\sigma_1, \sigma_2)$, which first checks whether both input labels σ_1 and σ_2 are in $\mathcal{R}$, returning $\perp$ if not, and then returns the label $\sigma = \mathcal{L}(\mathcal{L}^{-1}(\sigma_1) + \mathcal{L}^{-1}(\sigma_2))$ corresponding to the group operation applied to the two group elements. In some problems, A additionally will have access to decisional oracles such as, for example, a decisional Diffie-Hellman oracle. These take as input one or more labels and return 0 or 1 depending on whether a certain relation of the corresponding group elements holds. We measure the running time of A as the (worst-case) number of oracle queries made and typically denote this value by q. As this work only considers computational problems P, the advantage of adversary A in this model and any of its variants is given by $\mathrm{Adv}^{\mathrm{P}}(\mathsf{A}) := \Pr[\mathsf{A} \text{ solves P}]$.

Preprocessing and bit-fixing generic-group models We now recall the auxiliary-input (AI-GGM) and bit-fixing (BF-GGM) generic group models. Again, both models consider a group isomorphic to $(\mathbb{Z}_p, +)$. Adversaries $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ proceed in two stages, and are parameterized by both advice size s and the number of oracle queries q made by A_2. We refer to such adversaries as (s, q)-adversaries. BF-GGM is additionally parameterized by $M \leq p$, the number of values of the labeling function that can be chosen by A_1.

For problems P defined in the AI-GGM, the unbounded preprocessing phase A_1 receives as input the full description of the labeling function $\mathcal{L} : \mathbb{Z}_p \to \{0,1\}^\ell$, that is a uniformly random sampled injection, and returns a state Γ of bit-size at most s. A_2 receives as input Γ and the problem instance. It has access to the group-operation oracle $\mathrm{GrpOp}(\sigma_1, \sigma_2) = \mathcal{L}(\mathcal{L}^{-1}(\sigma_1) + \mathcal{L}^{-1}(\sigma_2))$, that it can query up to q times. As before, A's advantage is defined as $\mathrm{Adv}^{\mathrm{P}}(\mathsf{A}) := \Pr[\mathsf{A} \text{ solves P}]$.

For problems P defined in the BF-GGM, the range $\mathcal{R}$ of the labeling function is first sampled uniformly at random from all size p subsets of $\{0,1\}^\ell$. Unbounded algorithm A_1 receives $\mathcal{R}$ as input and returns a state Γ of bit size at most s, as well as a list $(\sigma_i, a_i)_i$ of at most M elements, with $\sigma_i \in \mathcal{R}$ and $a_i \in \mathbb{Z}_p$, such that all σ_i and all a_i are distinct. Then, the labeling function $\mathcal{L}$ is chosen uniformly at random from all bijections between $\mathbb{Z}_p$ and $\mathcal{R}$ that satisfy $\mathcal{L}(a_i) = \sigma_i$ for all i, and A_2 is invoked on Γ and the problem instance. It has access to group-operation oracle $\mathrm{GrpOp}(\sigma_1, \sigma_2) = \mathcal{L}(\mathcal{L}^{-1}(\sigma_1) + \mathcal{L}^{-1}(\sigma_2))$, that it can query up to q times.

We recall the following theorem, which establishes that hardness in the BF-GGM implies hardness in the AI-GGM.

Theorem 1 ([CDG18] Thm. 1)**.** *Let* P *be a single-stage computational problem defined over generic groups, and* $\gamma > 0$*. Assume that for* $M \geq 6(q + \log(\gamma^{-1})) \cdot q_{\mathrm{comb}}$ *the advantage of every* (s, q)*-adversary solving* P *in the* BF-GGM *is bounded by* ε'*, where* q_{comb} *is the combined query count of* A *and the problem environment* P*.*

Problem SHS(n,d)	**Oracle** Eval(F)
00 $\vec{a} \leftarrow_{\$} \mathbb{Z}_p^n$	03 **if** $\deg(F) > d$
01 $\vec{b} \leftarrow \mathsf{A}^{\text{Eval}}$	04 **return** $\perp$
02 **return** $[\vec{a} = \vec{b}]$	05 **return** $[F(\vec{a}) = 0]$

Figure 2: Search-by-Hypersurface problem parameterized dimension n and degree d.

Then, in the AI-GGM *every* (s,q)*-adversary has advantage bounded by*

$$\varepsilon \leq 2\varepsilon' + \gamma \ ,$$

Generic group model for bilinear groups. In the setting of bilinear groups one considers groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$, all of prime order p, equipped with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Accordingly, the extension of the GGM to bilinear groups is obtained by considering three different i.i.d. random injections $\mathcal{L}_j : \mathbb{Z}_p \hookrightarrow \{0,1\}^\ell$ for $j \in \{1,2,T\}$, with images $\mathcal{R}_j$ respectively. The group-operation oracle $\text{GrpOp}(j, \sigma_1, \sigma_2)$ can now be queried with respect to any of the label functions, and thus takes an extra input. The bilinear-map oracle $\text{Bil}(\sigma_1, \sigma_2)$ takes as input two labels $\sigma_1 \in \mathcal{R}_1$ and $\sigma_2 \in \mathcal{R}_2$, and outputs the label $\sigma = \mathcal{L}_T(\mathcal{L}_1^{-1}(\sigma_1) \cdot \mathcal{L}_2^{-1}(\sigma_2)) \in \mathcal{R}_T$.

Different types of bilinear group used in practice differ by the (non)-existence of efficiently computable isomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$. Thus, depending on type $\phi \in \{1,2,3\}$, algorithms might additionally have access to oracles $\text{Iso} : \mathbb{G}_2 \to \mathbb{G}_1$ mapping $\sigma \mapsto \mathcal{L}_1(\mathcal{L}_2^{-1}(\sigma))$; and $\text{Iso}^{-1} : \mathbb{G}_1 \to \mathbb{G}_2$ mapping $\sigma \mapsto \mathcal{L}_2(\mathcal{L}_1^{-1}(\sigma))$. If $\phi = 1$ algorithms have access to both Iso and Iso^{-1}, if $\phi = 2$ only to Iso, and if $\phi = 3$ to none of the two.

2.3 Search-by-Hypersurface Problem.

We recall the *Search-by-Hypersurface problem* (SHS(n,d)) [AGK20] for dimension n and degree d, that can be seen as a generalization of Yun's *Search-by-Hyperplane problem* [Yun15] to degrees larger than 1. In the problem, a vector $\vec{a} = (a_1, \ldots, a_n)$ sampled uniformly at random from $\mathbb{Z}_p^n$ has to be recovered by an adversary A. To do so, A receives no input, but has access to oracle Eval that, on input a hypersurface in $\mathbb{Z}_p^n$ of degree at most d, tells whether $\vec{a}$ lies on the hypersurface or not. More precisely, Eval takes as input polynomials $F \in \mathbb{Z}[X_1, \ldots, X_n]$ of degree at most d and returns 1 if $F(\vec{a}) = 0$ and 0 else. For a formal definition see Figure 2. We now recall an information theoretic lower bound on the hardness of SHS.

Lemma 2 ([AGK20], Lemma 6)**.** *Let* $n, d, q \in \mathbb{N}$*. Then, every adversary* A *that makes at most* q *queries in game* SHS(n,d) *has an advantage bounded by*

$$\text{Adv}^{\text{SHS}(n,d)}(\mathsf{A}) \leq \left(\frac{d}{p}\right)^n \cdot \sum_{i=0}^{n} \binom{q}{i} \leq \frac{1}{2} \cdot \left(\frac{e\,dq}{pn}\right)^n \ ,$$

where e *is Euler's number.*

3 From Generic Group Problems to Geometric Search-Problems

In this section we show that the hardness of a large class of problems in the generic group model(s) can be reduced to the hardness of a corresponding geometric search-problem. In prior work (plain GGM [Yun15, AGK20], bit-fixing GGM [CDG18]) this approach was taken to prove bounds on the hardness of specific discrete-logarithm type problems. We show that it can be generalized as follows.

We introduce the first geometric search-problems corresponding to problems that require the adversary to compute group elements instead of hidden exponents. For these problems the solution is going to be a polynomial/hypersurface (similar to the ones the adversary is allowed to query for) from a restricted range of

admittable solutions. The latter is necessary to not end up with a trivial problem. We define such problems both for the plain GGM and the bit-fixing GGM and then extend our approach to the setting of bilinear groups. We phrase our results in terms of Uber-assumptions in the style of [BBG05, RLB+08], where we in particular allow multi-instance problems and access to decisional oracles. The latter, as well as the bilinear map e in the case of bilinear groups, require us to carefully restrict the geometric-search problem's range of admittable evaluation-oracle inputs in a way that enables us to carry over AGM reductions to reductions between the corresponding geometric search-problems to in Section 4. This restriction is in contrast to prior work, where the only restriction was the degree of the queried polynomial.

Section 3.1 covers the plain GGM, Section 3.2 the bit-fixing GGM, and Section 3.3 the bilinear GGM.

3.1 From GGM to Geometric Search-Problems

Considered problems. Our goal is to capture as many problems in the generic group model as possible, so we state our transformation to geometric search-problems for Uber problem

$$\text{MI-Uber}(t,(m,n),F_1,\ldots,F_k,F_1^*,\ldots,F_n^*,W_1,\ldots,W_s)$$

where $t,m,n\in\mathbb{N}$ with $m\leq n$, $F_i\in\mathbb{Z}_p[X_1,\ldots,X_t]$, $F_i^*\in\mathbb{Z}_p(X_1,\ldots,X_t)$, and $W_i\in\mathbb{Z}_p[Z_1,\ldots,Z_{s_i}]$ for some s_i. The parameters have the following role.

Parameter	Role	Example: gap-CDH
t	# secrets $x_1,\ldots,x_t$ in $\mathbb{Z}_p$	$t=2$, secrets x,y
n	# target group elements	1
m	required solutions	1
$F_1,\ldots,F_k$	input group elements	$F_1=X\sim g^x$, $F_2=Y\sim g^y$
$F_1^*,\ldots,F_n^*$	target group elements	$F_1^*=XY\sim g^{xy}$
$W_1,\ldots,W_s$	decisional oracles	DDH-oracle: $W_1=X'Y'-Z'$

A MI-Uber adversary A for vector of secrets $\vec{x}=(x_1,\ldots,x_t)$ receives as input $(g,g^{F_1(\vec{x})},\ldots,g^{F_k(\vec{x})})$ and has to output an index set $I\subseteq[n]$ of size at least m as well as group elements h_i such that $h_i=g^{F_i^*(\vec{x})}$ for all $i\in I$. It has access to the group operation, as well as decisional oracles W_i which, on input s_i many group elements $h_j=g^{y_j}$, returns 1 if $g^{W_i(\vec{y_j})}=1$ (or equivalently $W_i(\vec{y_j})=0$) and 0 if not. For a formal definition of MI-Uber in the generic-group model see Figure 3. We point out that the binary encoding game of [ZZK22] used to separate the GGM and AGM does not fall under the umbrella of MI-Uber because the adversary of the binary encoding game does not get a description of the target group element via a polynomial.

Associated geometric search-problem. We now define the geometric search problem associated to $\text{MI-Uber}(t,(m,n),\vec{F},\vec{F^*},\vec{W})$, called $\text{geo-MI-Uber}(t,(m,n),\vec{F},\vec{F^*},\vec{W})$. It is parameterized by a set of integers and variables with the same restrictions as MI-Uber, some of which take different roles, as follows. A vector $\vec{x}=(x_1,\ldots,x_t)\leftarrow_{\$}\mathbb{Z}_p^t$ is sampled uniformly at random. The goal of adversary A is to return index set $I\subseteq[n]$ of size m, and polynomials $\hat{F}_i\in\mathbb{Z}_p[X_1,\ldots,X_t]$ such that $\hat{F}_i(\vec{x})-F_i^*(\vec{x})=0$ for all $i\in I$. To do so, A receives no input, but has access to oracle Eval, which on input a polynomial $F'\in\mathbb{Z}_p[X_1,\ldots,X_t]$ returns 1 if $F'(\vec{x})=0$ and 0 else. Note that this corresponds to the query, whether the hypersurface in $\mathbb{Z}_p^t$ defined by F' contains $\vec{x}$ or not. We make the additional requirement that all output solutions $\hat{F}_i$ lie in the linear span $\text{Span}(1,F_1,\ldots,F_k)$ of the input polynomials and impose the same requirement on inputs to Eval.[2] The restriction on solutions ensures that geo-MI-Uber is non-trivial as long as MI-Uber is; If MI-Uber cannot be trivially solved, we must have that (sufficiently many) $F_i^*\notin\text{Span}(\vec{F})$ as else one could compute a valid solution with a small number of group-operation queries. Accordingly, in this case geo-MI-Uber does not admit the trivial solution of simply outputting m of the F_i^*. The restriction on the inputs to Eval, on the

[2] Alternatively, one could also make this requirement explicit by changing the inputs to Eval to be a vector $(a_0,\ldots,a_k)\in\mathbb{Z}_p^k$ and return whether $\vec{x}$ lies on the hypersurface defined by $a_0+\sum_{i=1}^k a_iF_i$. The requirement for solutions $\hat{F}_i$ could be adapted accordingly.

Problem MI-Uber$(t,(m,n),\vec{F},\vec{F^*},\vec{W})$	**Oracle** $\mathrm{GrpOp}(\sigma,\sigma')$
00 $\mathcal{L} \leftarrow_\$ \mathrm{Inj}(\mathbb{Z}_p,\{0,1\}^\ell)$	08 **require** $\sigma,\sigma' \in \mathcal{L}(\mathbb{Z}_p)$
01 $\vec{x} \leftarrow_\$ \mathbb{Z}_p^t$	09 **return** $\mathcal{L}(\mathcal{L}^{-1}(\sigma)+\mathcal{L}^{-1}(\sigma'))$
02 $\sigma_0 \leftarrow \mathcal{L}(1)$	**Oracle** $\mathrm{Dec}_{W_i}(\sigma'_1,\ldots,\sigma'_{s_i})$
03 **for** $i=1,\ldots,k$	10 **require** $\sigma'_j \in \mathcal{L}(\mathbb{Z}_p)$ **for** $j\in[s_i]$
04 $\quad\sigma_i \leftarrow \mathcal{L}(F_i(\vec{x}))$	11 **return** $[W_i(\mathcal{L}^{-1}(\tilde{\sigma}_1),\ldots,\mathcal{L}^{-1}(\tilde{\sigma}_{s_i}))=0]$
05 $(I,(\hat{\sigma}_i)_{i\in I}) \leftarrow \mathsf{A}^{\mathrm{GrpOp},(\mathrm{Dec}_{W_i})_i}(\sigma_0,\ldots,\sigma_k)$	
06 **require** $I\subseteq[n]\wedge\lvert I\rvert\geq m$	
07 **return** $[\forall i\in[I]:\hat{\sigma}_i=\mathcal{L}(F_i^*(\vec{x}))]$	

Problem geo-MI-Uber$(t,(m,n),\vec{F},\vec{F^*},\vec{W})$	**Oracle** $\mathrm{Eval}(F')$
12 $\vec{x} \leftarrow_\$ \mathbb{Z}_p^t$	17 **require** $F' \in \mathrm{Span}(1,\vec{F})$
13 $(I,(\hat{F}_i)_{i\in I}) \leftarrow \mathsf{A}^{\mathrm{Eval},(\mathrm{Dec}_{W_i})_i}$	18 **return** $[F'(\vec{x})=0]$
14 **require** $\hat{F}_i \in \mathrm{Span}(1,\vec{F})$ **for all** $i\in I$	**Oracle** $\mathrm{Dec}_{W_i}(F'_1,\ldots,F'_{s_i})$
15 **require** $I\subseteq[n]\wedge\lvert I\rvert\geq m$	19 **require** $F'_j \in \mathrm{Span}(1,\vec{F})$ **for all** $j\in[s_i]$
16 **return** $[\forall i\in I:(F_i^*-\hat{F}_i)(\vec{x})=0]$	20 **return** $[W_i(F'_1(\vec{x}),\ldots,F'_{s_i}(\vec{x}))=0]$

Figure 3: Problems MI-Uber (in the GGM) and the corresponding geometric search problem geo-MI-Uber parameterized by $t,m,n\in\mathbb{N}$, and polynomials $\vec{F}=(F_1,\ldots,F_k)$, $\vec{F^*}=(F_1^*,\ldots,F_n^*)$ with $F_i\in\mathbb{Z}_p[X_1,\ldots,X_t]$, $F_i^*\in\mathbb{Z}_p(X_1,\ldots,X_t)$ for all i, in the presence of decisional oracles defined by polynomials $\vec{W}=(W_1,\ldots,W_s)$ with $W_i\in\mathbb{Z}_p[Z_1,\ldots,Z_{s_i}]$ for some $s_i\in\mathbb{N}$. $\mathrm{Inj}(\mathbb{Z}_p,\{0,1\}^\ell)$ denotes the set of injections from $\mathbb{Z}_p$ to label space $\{0,1\}^\ell$.

other hand, turns out to be useful when construction reductions between geometric search-problems. Finally, each $W_i\in\mathbb{Z}_p[Z_1,\ldots,Z_{s_i}]$ corresponds to oracle Dec_{W_i}, which on input of $F'_1,\ldots,F'_{s_i}\in\mathrm{Span}(1,F_1,\ldots,F_k)$ returns 1 if $(W_i(F'_1,\ldots,F'_{s_i}))(\vec{x})=W_i(F'_1(\vec{x}),\ldots,F'_{s_i}(\vec{x}))=0$, and 0 if not.[3] For a formal definition of the problem see Figure 3.

In the following we give the reduction from geo-MI-Uber to MI-Uber. The key observation is that, by using the oracle Eval, the reduction can simulate the view of the MI-Uber adversary A without knowledge of the secret $\vec{x}$.

Theorem 3. *Let $t,m,n,k,s\in\mathbb{N}$ with $m\leq n$, and consider vectors $\vec{F}=(F_1,\ldots,F_k)$, $\vec{F^*}=(F_1^*,\ldots,F_n^*)$ of polynomials and rational functions with $F_i\in\mathbb{Z}_p[X_1,\ldots,X_t]$, $F_i^*\in\mathbb{Z}_p(X_1,\ldots,X_t)$ for all i, and $\vec{W}=(W_1,\ldots,W_s)$ with $W_i\in\mathbb{Z}_p[Z_1,\ldots,Z_{s_i}]$ for some $s_i\in\mathbb{N}$.*

Let $r\in\mathbb{N}$ and A a MI-Uber$(t,(m,n),\vec{F},\vec{F^*},\vec{W})$*-solver in the* GGM*, which makes at most q oracle queries. Then, there exists a* geo-MI-Uber$(t,(m,n),\vec{F},\vec{F^*},\vec{W})$ *solver B that for $d_{\max}:=\max_i(\deg(F_i))$ and $s_{\max}:=\max(\max_i(s_i),2)$ makes at most*

$$q^2\cdot s_{\max}r+q\cdot\big((ks_{\max}+m)(r-1)+k+1\big)+k\big(k+1+m(r-1)\big)\approx q^2 s_{\max}r$$

queries and satisfies

$$\begin{aligned}&\mathrm{Adv}^{\text{geo-MI-Uber}(t,(m,n),\vec{F},\vec{F^*},\vec{W})}(\mathsf{B})\\ \geq&\mathrm{Adv}^{\text{MI-Uber}(t,(m,n),\vec{F},\vec{F^*},\vec{W})}(\mathsf{A})-(qs_{\max}+m)\cdot\left(\frac{(q+k)\cdot d_{\max}}{p}\right)^r\ .\end{aligned}$$

Proof of Thm 3. The geo-MI-Uber solver B receives as input the number of indeterminates t and the polynomials $\vec{F}$ rational functions $\vec{F^*}$ and has access to oracles Eval, $(\mathrm{Dec}_{W_i})_i$. To simulate the view of the MI-Uber

[3] As is the case for Eval, oracle Dec corresponds to evaluating containment in a hypersurface, albeit, one of degree possibly higher than the ones in the linear span of the input polynomials. Thus, one could incorporate Dec_{W_i} into Eval by expanding the range of admissible polynomials for the latter from $\mathrm{Span}(1,F_1,\ldots,F_k)$ to also include polynomials of the form $W_i(F'_1,\ldots,F'_{s_i})\in\mathbb{Z}_p[X_1,\ldots,X_t]$ for $F'_j\in\mathrm{Span}(1,F_1,\ldots,F_k)$. However, we decided to keep the oracles separated in order to have a clearer conceptual distinction between the group-operation oracle and decisional oracles.

solver A, B needs to construct input $(\sigma_0, \ldots, \sigma_k)$ and reply to oracle queries made by A. To do this in a consistent manner, B samples labels on the fly and maintains a table T that stores all previously recorded labels σ, each together with a corresponding polynomial $P(\vec{X}) \in \mathrm{Span}(1, \vec{F})$ in the indeterminates $X_1, \ldots, X_t$. The labels σ will correspond to a perfect simulation of MI-Uber such that we have $\mathcal{L}^{-1}(\sigma) = P(\vec{x})$ for every entry (P, σ) in T. To make sure that the simulation is consistent, B needs to check that no two polynomials P, P' such that $P(\vec{x}) = P'(\vec{x})$ get paired with different labels. This would be equivalent to a group element receiving two different labels. Hence, before sampling a new label for a polynomial P, B needs to check that there is no previously recorded polynomial P' in T such that $(P - P')(\vec{x}) = 0$ using the oracle Eval. Note that, if $P, P' \in \mathrm{Span}(1, \vec{F})$ then so is $P - P'$, and thus the oracle will not return $\perp$. In more detail, B does the following.

- It samples the range $\mathcal{R} \subseteq \{0,1\}^\ell$ uniformly at random from all subsets of $\{0,1\}^\ell$ of size p.[4]
- To create the input $(\sigma_0, \sigma_1, \ldots, \sigma_k)$ for A, B does the following: first, it samples $\sigma_0 \leftarrow_\$ \mathcal{R}$, stores $(0, \sigma_0)$ in T, and then iteratively defines σ_i as follows. To create σ_i for $i \in [k]$, it queries Eval on $F_i - P'$ for all previously recorded polynomials P' in T. If the answer is 1 for some P', it sets σ_i to the corresponding label of P'. Otherwise, it chooses a random unused value from $\mathcal{R}$ and stores (F_i, σ_i). Note that all polynomials stored so far in T trivially lie in $\mathrm{Span}(1, F_1, \ldots, F_k)$, and entries (P, σ) defined so far are consistent with the property $\mathcal{L}^{-1}(\sigma) = P(\vec{x})$.
- Then B runs $\mathsf{A}(\sigma_0, \sigma_1, \ldots, \sigma_k)$. When A makes a query (σ, σ') to GrpOp, B does the following:
  - First, it checks if σ and σ' have been recorded in T. If σ has not been recorded, checks if $\sigma \in \mathcal{R}$ and answers $\perp$ if not. Otherwise, B makes r attempts at assigning σ a constant unused element in $\mathbb{Z}_p$ that is consistent with the simulation so far. In particular, B will repeat the following steps up to r times. It starts by sampling a random unused element $a \leftarrow_\$ \mathbb{Z}_p$. If this is the $\tilde{r}$th attempt, with $\tilde{r} < r$, it queries Eval on $a - P$ for all previously recorded (non-constant) polynomials P in T. If Eval outputs 1 for some P, adversary B tries again with a new random unrecorded $a \leftarrow_\$ \mathbb{Z}_p$. If Eval does not output 1 for any of the queries or if it is the rth time of sampling a, it stores the pair (a, σ) in T, where a is to be interpreted as a constant polynomial. It then does the same if σ' has not been recorded.
  - Let P, P' be the polynomials corresponding to labels σ, σ'. For all previously recorded polynomials P'' in T, B queries $P + P' - P''$ to oracle Eval. If Eval outputs 1 for any of the P'', it looks up the corresponding label σ'' in T and sends σ'' to A. Otherwise, B samples a random σ'' from the unused values in $\mathcal{R}$ and sends it to A. Then it stores $(P + P', \sigma'')$ in T.
  - Note that, again, all newly stored polynomials are elements of $\mathrm{Span}(1, \vec{F})$ and that, if for all $(\hat{P}, \hat{\sigma}) \in T$ we had that $\mathcal{L}^{-1}(\hat{\sigma}) = \hat{P}(\vec{x})$, then the same holds for all elements added to the table during either of these steps.
- When A makes a query $(\sigma'_1, \ldots, \sigma'_{s_i})$ to one of the decisional oracles Dec_{W_i}, B does the following:
  - It first checks if for all $j \in [s_i]$, σ'_j has been recorded in T. If not, it proceeds as in the analogous case of group operation queries described above, assigning a random constant to it.
  - Then, it queries Dec_{W_i} on $(F'_1, \ldots, F'_{s_i})$, where $F'_1, \ldots, F'_r$ are the polynomials corresponding to $\sigma'_1 \ldots, \sigma'_{s_i}$ respectively. Sends the answer of Dec_{W_i} to A.
  - Note that for all j it holds that $\mathcal{L}^{-1}(\sigma'_j) = F'_j(\vec{x})$, and so the query is answered correctly, since we have that
$$W_i(\mathcal{L}^{-1}(\sigma'_1), \ldots, \mathcal{L}^{-1}(\sigma'_{s_i})) = W_i(F'_1(\vec{x}), \ldots, F'_{s_i}\vec{x}) \ .$$

[4] We measure the running time of generic algorithms by their query count. So, both sampling from $\mathcal{R}$ and checking whether $\sigma \in \mathcal{R}$ need not be efficiently computable. We use this approach for ease of exposition, but point out that these operations can easily be adapted to be done efficiently by sampling $\mathcal{R}$ on the fly.

- When A outputs $(I, (\hat{\sigma}_i)_{i\in I})$, B checks for every $i \in I$ whether $\hat{\sigma}_i$ has previously been recorded in T with corresponding polynomial $\hat{F}_i$. If not, it is treated as in the analogous case of group operation queries described above. Then B outputs $(I, (\hat{F}_i)_{i\in I})$ as its solution. Note that the check of line 14 will succeed, as $\hat{F}_i \in \mathrm{Span}(1, \vec{F})$ for all i.

We now count the number of queries made by B. First, note that to set up A's input, B adds 1 constant and k arbitrary entries to T and makes at most $k(k+1)$ queries to Eval. During the execution of A, every query to GrpOp or Dec_{W_i} adds up to $s_{\max}$ constant and one arbitrary polynomial to T. Thus, at the q'th query, T contains at most $k+1+(s_{\max}+1)q'$ entries, of which at most $k+q'$ are not constant. As the check against previously unrecorded labels needs to be done only with respect to non-constant polynomials, the q'th query requires at most $(r-1)s_{\max}(k+q') + (k+1+(s_{\max}+1)q')$ queries to Eval. Finally, to handle A's output, B makes up to $(r-1)m$ additional checks against the at most $(k+q)$ non-constant entries in T. Summing up we can bound the number of queries made by B by

$$q^2 \cdot s_{\max} r + q \cdot ((k s_{\max} + m)(r-1) + k + 1) + k\,(k + 1 + m(r-1)) \quad .$$

To show that

$$\begin{aligned} &\mathrm{Adv}^{\text{geo-MI-Uber}(t,(m,n),\vec{F},\vec{F^*},\vec{W})}(\mathsf{B}) \\ \geq &\mathrm{Adv}^{\text{MI-Uber}(t,(m,n),\vec{F},\vec{F^*},\vec{W})}(\mathsf{A}) - (q s_{\max} + m) \cdot \left(\frac{(q+k)\cdot d_{\max}}{p}\right)^r \quad , \end{aligned}$$

we define the event

$$\mathtt{bad} = \{\exists (a, \sigma_a), (P, \sigma_P) \in T \mid a \text{ constant}, P \text{ not constant} : P(\vec{x}) - a = 0\} \quad ,$$

which corresponds to B assigning a label σ it did not previously receive to a constant in a way not consistent with the simulation. Observe that in the case that $\mathtt{bad}$ does not occur, we have $\mathcal{L}^{-1}(\sigma) = P(\vec{x})$ for all (P, σ) stored in T, and the view of A is a perfect simulation of the MI-Uber game with hidden value $\vec{x}$. Thus, in this case we have that $(\hat{F}_i - F_i^*)(\vec{x}) = 0 \Leftrightarrow \mathcal{L}^{-1}(\hat{\sigma}_i) = \hat{F}_i(\vec{x}) = F_i^*(\vec{x})$ for all $i \in I$, and B wins exactly if A wins.

Finally, when assigning a constant to a previously unseen σ the probability that B does so inconsistently is at most $((q+k)d_{\max}/p)^r$. Indeed, by the Schwartz-Zippel Lemma, the probability the sampled constant is in the set of roots of any polynomial P is at most $d_{\max}/p$ and, for each of the r attempts at finding a constant, at most $(q+k)$ polynomials have to be checked. As the reduction has to sample at most $(qs_{\max}+m)$ constants, the bound follows. □

Before turning to the setting of BF-GGM we make a couple observations.

Remark 1. *(i) As opposed to prior work, our reduction does not fully preserve the advantage of* A*, but introduces an error term of order* $\sim q\left(\frac{qd_{\max}}{p}\right)^r$*. The term, as well as some of the additional queries that* B *has to make, stems from handling group-operation queries on labels, that the adversary did not previously receive. The reduction* B *handles such queries by assigning them a random, unused discrete logarithm. The number* r *corresponds to the number of attempts made for each such query to find a constant that is consistent with the simulation so far.*

Looking ahead, the loss in advantage will not cause issues when deriving lower bounds, as the bounds we obtain on the advantage of B *will, for most problems, be of order* $q^2 d_{\max}/p$*. Hence, in this case we can simply choose* $r = 1$*. The only exception are multi-instance (gap) CDH problems, for which the advantage of* B *decays exponentially in the number of instances that have to be solved. For these we end up with worse bounds than the ones from literature (However, if one only considers the number of queries required to achieve* constant *success probability the bound stays the same). We point out that in the reductions of [AGK20, Yun15] it is assumed that* A *never queries on labels it did not previously receive. In light of the work by Zhang, Zhou, and Katz [ZZK22] this seems hard to formally justify unless*

the range of labels is very sparse in $\{0,1\}^\ell$, as is assumed in [Yun15]. In [CDG18], such queries are handled in the same way that we do. However, neither the probability of failing to sample an adequate discrete logarithm, nor the additional Eval *queries to verify its consistency with the simulation, were factored in the advantage and query count, respectively, of their reduction.*

(ii) One can easily adapt Theorem 3 to discrete-logarithm variants of MI-Uber*, in which* A *receives the same input and has access to the same oracles, but instead of computing target group elements has to compute at least m of the $x_1, \ldots, x_t$. The corresponding geometric search-problem would have access to the same oracles as in* geo-MI-Uber*, and also have to compute at least m of the x_i.*

(iii) Theorem 3 also holds if in both problems the adversaries have oracle access to a random oracle, i.e., a uniformly random function $\mathsf{RO}\colon \mathcal{R} \to \{0,1\}^{\ell'}$ *for some ℓ'. In this case, the reduction can simply forward all* RO *queries. Thus, the computed* GGM *bounds also apply to cryptographic schemes making use of such oracles, as for example in [AGK20, BL22, FPS20, GT21]. In the case of a random oracle into the group, i.e.,* $\mathrm{RO}; \{0,1\}^* \to \{0,1\}^\ell$*, the reduction can simulate the random oracle, and associate previously unseen labels to a constant polynomial as discussed in point (i). For an example of this type of reduction, see our result for BLS signatures in Section 4.2 and [ABB+20, KLX22].*

3.2 Extension to the Bit-Fixing Generic-Group Model

In this section we show that the translation of problems to geometric search-problems also works in the BF-GGM. In combination with Theorem 1 and a reduction between the corresponding geometric problems, like the ones presented in Section 4, this enables us to carry over preprocessing lower-bounds from one problem to another.

We again consider Uber problems MI-Uber and geo-MI-Uber of Figure 3. However, in this section we will restrict to problems without decision oracles, i.e., we assume that $\vec{W} = \epsilon$ is the empty vector. We stress that the reason for this is that the preprocessing GGM and bit-fixing GGM, as well as the translation between the two in Theorem 1, are defined without decisional oracles. If one was able to extend both models to allow for such oracles (or to the setting of bilinear groups) and establish their equivalence, we do not see any obstacles for our translation to geometric search-problems to carry over as well.

We show that the security of MI-Uber in the bit-fixing GGM reduces to the security of geo-MI-Uber (which does not have a preprocessing phase). The proof follows the one of Theorem 3, the main difference being that, since in the preprocessing phase A_1 fixes the labels of a number of group elements, the reduction is required to add a corresponding amount of constant polynomials to its table T. This leads to a larger amount of queries to oracle Eval.

Recall that the BF-GGM is parameterized by $M \in \mathbb{N}$, the number of labels chosen by A_1 in the preprocessing phase. A_2 receives as input both the advice $\Gamma \leftarrow \mathsf{A}_1(\mathcal{R})$ and the problem instance as defined in Figure 3. We obtain the following result. As its proof is very similar to the one of Theorem 3 we defer it to Supplementary Material A.1.

Theorem 4. *Let $t, m, n, k \in \mathbb{N}$ with $m \leq n$, and consider vectors $\vec{F} = (F_1, \ldots, F_k)$, $\vec{F^*} = (F_1^*, \ldots, F_n^*)$ of polynomials and rational functions with $F_i \in \mathbb{Z}_p[X_1, \ldots, X_t]$, $F_i^* \in \mathbb{Z}_p(X_1, \ldots, X_t)$ for all i. Further, let $d_{\max} := \max_i(\deg(F_i))$. Let $r \in \mathbb{N}$ and let $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ be a* MI-Uber$(t, (m,n), \vec{F}, \vec{F^*})$ *solver in the* BF-GGM *which makes at most q queries. Using A, we can construct a* geo-MI-Uber *solver B that makes at most*

$$q\big(M + (2r+1)q\big) + kM + q\big(rk + 1 + m(r-1)\big) + k\big(k + 1 + m(r-1)\big) \approx q(M + (2r+1)q)$$

queries and satisfies

$$\mathrm{Adv}^{\text{geo-MI-Uber}(t,(m,n),\vec{F},\vec{F^*})}(\mathsf{B}) \geq \mathrm{Adv}^{\text{MI-Uber}(t,(m,n),\vec{F},\vec{F^*})}(\mathsf{A}) - (2q+m)\left(\frac{(q+k)\cdot d_{\max}}{p}\right)^r .$$

3.3 Extension to the GGM for Bilinear Groups

In this section we generalize the result of Section 3.1 to the setting of bilinear groups. In Figure 4 we define the

$$\text{MI-Uber}_\phi(t,(m,n_1,n_2,n_T),\vec{F}_1,\vec{F}_2,\vec{F}_T,\vec{F}_1^*,\vec{F}_2^*\vec{F}_T^*,W_1,\ldots,W_s)$$

problem for bilinear groups of type ϕ and the corresponding geometric problem

$$\text{geo-MI-Uber}_\phi(t,(m,n_1,n_2,n_T),\vec{F}_1,\vec{F}_2,\vec{F}_T,\vec{F}_1^*,\vec{F}_2^*\vec{F}_T^*,W_1,\ldots,W_s)\ .$$

Again, $t \in \mathbb{N}$ is the number of secrets in $\mathbb{Z}_p$ and $m \in \mathbb{N}$ is the number of solutions the adversary A is required to output. The input and target polynomials are now divided into three vectors $\vec{F}_1, \vec{F}_2, \vec{F}_T$ and $\vec{F}_1^*, \vec{F}_2^*, \vec{F}_T^*$ respectively, that correspond to the three groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, where we have $\dim(\vec{F}_J^*) = n_J$ for $J \in \{1,2,T\}$. The polynomials W_i define decisional oracles which, on input s_i many group elements defined by exponents $y_{1,1},\ldots y_{1,r_1}$, $y_{2,1},\ldots y_{2,r_2}$, $y_{T,1},\ldots y_{T,r_T}$ for generators g_1, g_2, g_T, return 1 if $W_i(y_{1,1},\ldots y_{1,r_1}, y_{2,1},\ldots y_{2,r_2}, y_{T,1},\ldots y_{T,r_T}) = 0$ and 0 if not. In MI-Uber$_\phi$, the adversary A receives as input

$$\left(g_1, g_2, g_T, g_1^{\vec{F}_{1,1}(\vec{x})},\ldots,g_1^{F_{1,k_1}(\vec{x})}, g_2^{F_{2,1}(\vec{x})},\ldots,g_2^{F_{2,k_2}(\vec{x})}, g_T^{F_{T,1}(\vec{x})},\ldots,g_T^{F_{T,k_T}(\vec{x})}\right)$$

and has to output three index sets $I_1 \subseteq [n_1]$, $I_2 \subseteq [n_2]$, $I_T \subseteq [n_T]$ such that $|I_1| + |I_2| + |I_T| \geq m$, as well as group elements $h_{J,i} = g_J^{F_{J,i}^*(\vec{x})}$ for all $J \in \{1,2,T\}$ and all $i \in I_J$.

In geo-MI-Uber$_\phi$, a vector $\vec{x} = (x_1,\ldots,x_t) \leftarrow \mathbb{Z}_p^t$ is sampled uniformly at random and A has to output three sets $I_1 \subseteq [n_1]$, $I_2 \subseteq [n_2]$, $I_T \subseteq [n_T]$ such that $|I_1| + |I_2| + |I_T| \geq m$, as well as polynomials $P_{J,i}$ such that $P_{J,i}(\vec{x}) = F_{J,i}^*(\vec{x})$ for all $J \in \{1,2,T\}$ and all $i \in I_J$. The adversary A does not receive any input but has access to the oracle Eval, that takes queries of the form (J,P) and returns 1 if $P(\vec{x}) = 0$ and 0 else. Here, $P \in \mathbb{Z}_p[X_1,\ldots,X_t]$ is a polynomial that satisfies certain restrictions depending on the type ϕ of the bilinear group, which we explain in more detail below. Before we give the reduction from geo-MI-Uber$_\phi$ to MI-Uber$_\phi$ in Theorem 5, we define some useful notation.

Notation. Let $\vec{R}, \vec{S}, \vec{F}$ be vectors of polynomials. In this section we will use the following notation:

$$\text{Span}(\vec{R},\vec{S}) := \text{Span}(R_1,\ldots,R_{\dim(\vec{R})},S_1,\ldots,S_{\dim(\vec{S})})$$

denotes the linear span of the polynomials in the entries of the vectors $\vec{R}$ and $\vec{S}$. We further define three different types of spans:

$$\text{Span}_1(\vec{R},\vec{S},\vec{F},\phi) := \begin{cases}\text{Span}(\vec{R},\vec{S}) \text{ if } \phi \in \{1,2\},\\ \text{Span}(\vec{R}) \text{ if } \phi = 3;\end{cases}$$

$$\text{Span}_2(\vec{R},\vec{S},\vec{F},\phi) := \begin{cases}\text{Span}(\vec{R},\vec{S}) \text{ if } \phi = 1,\\ \text{Span}(\vec{S}) \text{ if } \phi \in \{2,3\};\end{cases}$$

$$\text{Span}_T(\vec{R},\vec{S},\vec{F},\phi) := \text{Span}(\vec{F}, \text{Span}_1(\vec{R},\vec{S},\vec{F},\phi)\cdot\text{Span}_2(\vec{R},\vec{S},\vec{F},\phi)).$$

If $\vec{R}$ defines elements in $\mathbb{G}_1$, $\vec{S}$ defines elements in $\mathbb{G}_2$ and $\vec{F}$ defines elements in $\mathbb{G}_T$, we have that the elements in $\text{Span}_J(\vec{R},\vec{S},\vec{F},\phi)$ correspond to exactly those elements in group $\mathbb{G}_J$ that can be obtained from the input elements by performing group operations, evaluating the bilinear map and applying the isomorphism between groups $\mathbb{G}_1$ and $\mathbb{G}_2$.

We obtain the following result. Its proof is in Supplementary Material A.2.

Theorem 5. *Let $t, m, n_1, n_2, n_T, k_1, k_2, k_T \in \mathbb{N}$ with $m \leq n_1 + n_2 + n_T$, and consider vectors $\vec{F}_1 = (F_{1,1},\ldots,F_{1,k_1})$, $\vec{F}_2 = (F_{2,1},\ldots,F_{2,k_2})$, $\vec{F}_T = (F_{T,1},\ldots,F_{T,k_T})$, $\vec{F}_1^* = (F_{1,1}^*,\ldots,F_{1,n_1}^*)$, $\vec{F}_2^* = (F_{2,1}^*,\ldots,F_{2,n_2}^*)$, $\vec{F}_T^* = (F_{T,1}^*,\ldots,F_{T,n_T}^*)$ of polynomials with $F_{J,i} \in \mathbb{Z}_p[X_1,\ldots,X_t]$, $F_{J,i}^* \in \mathbb{Z}_p(X_1,\ldots,X_t)$ for all $J \in \{1,2,T\}$ and all i and $\vec{W} = (W_1,\ldots,W_s)$ with $W_i \in \mathbb{Z}_p[Z_1,\ldots,Z_{s_i}]$ for some $s_i \in \mathbb{N}$. Let $k = k_1 + k_2 + k_T$.*

Problem MI-Uber$_\phi(t,(m,n_1,n_2,n_T),\vec{F}_1,\vec{F}_2,\vec{F}_T,\vec{F}_1^*,\vec{F}_2^*,\vec{F}_T^*,W_1,\ldots,W_s)$
00 $\vec{x} \leftarrow_\$ \mathbb{Z}_p^t$
01 $\sigma_{1,0} \leftarrow \mathcal{L}_1(1)$
02 $\sigma_{2,0} \leftarrow \mathcal{L}_2(1)$
03 $\sigma_{T,0} \leftarrow \mathcal{L}_T(1)$
04 **for** $J \in \{1,2,T\}$
05 **for** $i = 1,\ldots,k_J$
06 $\sigma_{J,i} \leftarrow \mathcal{L}_J(F_{J,i}(\vec{x}))$
07 $(I_1,I_2,I_T,(\hat{\sigma}_{1,i})_{i\in I_1},(\hat{\sigma}_{2,i})_{i\in I_2},(\hat{\sigma}_{T,i})_{i\in I_T}) \leftarrow \mathsf{A}^{\mathrm{Iso},\mathrm{Iso}^{-1},\mathrm{GrpOp},\mathrm{Bil},(\mathrm{Dec}_{W_i})_{i\in[s]}}((\sigma_{1,i})_{i\in[k_1]_0},(\sigma_{2,i})_{i\in[k_2]_0},(\sigma_{T,i})_{i\in[k_T]_0})$
08 **require** $I_1 \subseteq [n_1] \wedge I_2 \subseteq [n_2] \wedge I_T \subseteq [n_T] \wedge |I_1|+|I_2|+|I_T| \geq m$
09 **return** $[\forall J \in \{1,2,T\}\ \forall i \in [I_J] : \hat{\sigma}_{J,i} = \mathcal{L}_J(F_{J,i}^*(\vec{x}))]$

Oracle Iso(σ)
10 **require** $\phi \in \{1,2\} \wedge \sigma \in \mathcal{L}_2(\mathbb{Z}_p)$
11 **return** $\mathcal{L}_1(\mathcal{L}_2^{-1}(\sigma))$

Oracle Iso$^{-1}(\sigma)$
12 **require** $\phi = 1 \wedge \sigma \in \mathcal{L}_1(\mathbb{Z}_p)$
13 **return** $\mathcal{L}_2(\mathcal{L}_1^{-1}(\sigma))$

Oracle GrpOp$(J,\sigma,\hat{\sigma})$
14 **require** $J \in \{1,2,T\} \wedge \sigma,\hat{\sigma} \in \mathcal{L}_J(\mathbb{Z}_p)$
15 **return** $\mathcal{L}_J(\mathcal{L}_J^{-1}(\sigma)+\mathcal{L}_J^{-1}(\hat{\sigma}))$

Oracle Bil$(\sigma,\hat{\sigma})$
16 **require** $\sigma \in \mathcal{L}_1(\mathbb{Z}_p) \wedge \hat{\sigma} \in \mathcal{L}_2(\mathbb{Z}_p)$
17 **return** $\mathcal{L}_T(\mathcal{L}_1^{-1}(\sigma)\cdot\mathcal{L}_2^{-1}(\hat{\sigma}))$

Oracle $\mathrm{Dec}_{W_i}(\tilde{\sigma}_{1,1},\ldots,\tilde{\sigma}_{1,s_1},\tilde{\sigma}_{2,1},\ldots,\tilde{\sigma}_{2,s_2},\tilde{\sigma}_{T,1},\ldots,\tilde{\sigma}_{T,s_T})$
18 **require** $\tilde{\sigma}_{J,j} \in \mathcal{L}_J(\mathbb{Z}_p)$ **for all** $J \in \{1,2,T\}$ **and all** $j \in [s_J]$
19 **return** $[W_i((\mathcal{L}_1^{-1}(\tilde{\sigma}_{1,j}))_{j\in[s_1]},(\mathcal{L}_2^{-1}(\tilde{\sigma}_{2,j}))_{j\in[s_2]},(\mathcal{L}_T^{-1}(\tilde{\sigma}_{T,j}))_{j\in[s_T]}) = 0]$

Problem geo-MI-Uber$_\phi(t,(m,n_1,n_2,n_T),\vec{F}_1,\vec{F}_2,\vec{F}_T,\vec{F}_1^*,\vec{F}_2^*,\vec{F}_T^*,W_1,\ldots,W_s)$
00 $\vec{x} \leftarrow_\$ \mathbb{Z}_p^t$
01 $(I_1,I_2,I_T,(\hat{P}_{1,i})_{i\in I_1},(\hat{P}_{2,i})_{i\in I_2},(\hat{P}_{T,i})_{i\in I_T}) \leftarrow \mathsf{A}^{\mathrm{Iso},\mathrm{Eval},\mathrm{Bil},(\mathrm{Dec}_{W_i})_{i\in[s]}}$
02 **require** $I_1 \subseteq [n_1], I_2 \subseteq [n_2], I_T \subseteq [n_T] \wedge |I_1|+|I_2|+|I_T| \geq m$
03 **require** $\forall J \in \{1,2,T\}\ \forall i \in I_J : \hat{P}_{J,i} \in \mathrm{Span}_J(\vec{F}_1,\vec{F}_2,\vec{F}_T,\phi)$
04 **return** $[\forall J \in \{1,2,T\}\ \forall i \in I_J : (\hat{P}_i - F_{J,i}^*)(\vec{x}) = 0]$

Oracle Iso(P_1,P_2)
05 **require** $P_J \in \mathrm{Span}_J(\vec{F}_1,\vec{F}_2,\vec{F}_T,\phi)$ **for all** $J \in \{1,2\}$
06 **return** $[(P_1-P_2)(\vec{x}) = 0]$

Oracle Eval(J,P)
07 **require** $J \in \{1,2,T\}$ **and** $P \in \mathrm{Span}_J(\vec{F}_1,\vec{F}_2,\vec{F}_T,\phi)$
08 **return** $[P(\vec{x}) = 0]$

Oracle Bil(P_1,P_2,P_3)
09 **require** $P_J \in \mathrm{Span}_J(\vec{F}_1,\vec{F}_2,\vec{F}_T,\phi)$ **for all** $J \in \{1,2,T\}$
10 **return** $[(P_1\cdot P_2 - P_3)(\vec{x}) = 0]$

Oracle $\mathrm{Dec}_{W_i}(\tilde{F}_{1,1},\ldots,\tilde{F}_{1,s_1},\tilde{F}_{2,1},\ldots,\tilde{F}_{2,s_2},\tilde{F}_{T,1},\ldots,\tilde{F}_{T,s_T})$
11 **require** $\tilde{F}_{J,j} \notin \mathrm{Span}_J(\vec{F}_1,\vec{F}_2,\vec{F}_T,\phi)$ **for all** $J \in \{1,2,T\}$ **and all** $j \in [s_J]$
12 **return** $[W_i(\tilde{F}_{1,1},\ldots,\tilde{F}_{1,s_1},\tilde{F}_{2,1},\ldots,\tilde{F}_{2,s_2},\tilde{F}_{T,1},\ldots,\tilde{F}_{T,s_T})(\vec{x}) = 0]$

Figure 4: The problems MI-Uber$_\phi$ (in the bilinear GGM) and geo-MI-Uber$_\phi$ for bilinear groups of type $\phi \in \{1,2,3\}$ parameterized by t,m,n_1,n_2,n_T, and vectors of polynomials $\vec{F}_1,\vec{F}_2,\vec{F}_T,\vec{F}_1^*,\vec{F}_2^*\vec{F}_T^*$ of different dimensions with entries in $\mathbb{Z}_p[X_1,\ldots,X_t]$ and in $\mathbb{Z}_p(X_1,\ldots,X_t)$ respectively, where $\dim(\vec{F}_J^*) = n_J$ for $J \in \{1,2,T\}$. The polynomials $W_1,\ldots,W_s$ define decisional oracles Dec_{W_i}.

Let $r \in \mathbb{N}$ and A be a MI-Uber*-solver in the* GGM*, which makes at most q oracle queries. Then, there exists a* geo-MI-Uber *solver B that, for*

$$d_{\max} := \max\left(\max_i(\deg(F_{T,i})), 2\max_i(\deg(F_{1,i})), 2\max_i(\deg(F_{2,i}))\right)$$

and $s_{\max} := \max(\max_i(s_i), 2)$, makes at most $k(k+3) + q((r-1)s_{\max}(k+q) + k + 3 + (s_{\max}+1)q) + (r-1)m(k+q) \approx s_{\max}q^2r$ queries and satisfies

$$\begin{aligned}&\mathrm{Adv}^{\text{geo-MI-Uber}(t,(m,n_1,n_2,n_3),\vec{F}_1,\vec{F}_2,\vec{F}_T,\vec{F}_1^*,\vec{F}_2^*\vec{F}_T^*,W_1,\ldots,W_s)}(\mathsf{B})\\ \geq&\mathrm{Adv}^{\text{MI-Uber}(t,(m,n_1,n_2,n_3),\vec{F}_1,\vec{F}_2,\vec{F}_T,\vec{F}_1^*,\vec{F}_2^*\vec{F}_T^*,W_1,\ldots,W_s)}(\mathsf{A}) - (qs_{\max}+m)\left(\frac{(q+k)\cdot d_{\max}}{p}\right)^r\ .\end{aligned}$$

4 Reductions Between Geometric Search-Problems

In this section we derive bounds in the GGM and AI-GGM (in Section 4.1) and bilinear GGM (in Section 4.2) for several problems. Building on the results from Section 3, which show a reduction to the considered problem from its geometric version, we show that there is a reduction from a variant of the search-by-hyperplane problem to the geometric problem. Using the lower bounds on the hardness of SHS then gives us the desired GGM bounds. Interestingly, several of our reductions closely mirror generic reductions in the AGM following the approach introduced in [FKL18]. While the bounds in the GGM and bilinear GGM are not new and either have been proven directly in the GGM or by following the AGM approach, we think they serve as nice examples on how reductions between geometric-search problems can serve as a replacement of the AGM approach. The bounds in the AI-GGM, on the other hand, are novel. In particular, we point out that preprocessing bounds for the multi-instance CDH problem seem hard to obtain with a direct reduction from $\mathrm{SHS}(m, 2)$ (compare Figure 1).

As a further result, in Section 4.2 we revisit the tight AGM+RO reduction of [FKL18] between the security of BLS signatures [BLS04] and the discrete logarithm problem. We give a reduction from $\mathrm{SHS}(1, 2)$ to BLS security in the bilinear GGM + RO and thus obtain a GGM lower bound of order q^2/p, matching that of [FKL18].

4.1 Reductions between geometric search-problems in the GGM and AI-GGM

Preprocessing bounds for d-strong-DL and d-strong-DHI in the GGM. As a warm-up, in this section we give a simple reduction between the geometric search variants of the d-strong discrete logarithm (d-strong-DL) and d-strong Diffie-Hellman inversion [BB04a] (d-strong-DHI) problems. Since we identify the former with the special case $\mathrm{SHS}(1, d)$ of the search-by-hypersurface problem, for which bounds of its hardness exist, we obtain new bounds on the hardness of the two problems in the AI-GGM. While the problems are arguably more interesting in the bilinear GGM, which is unfortunately not covered in the translation between BF-GGM and AI-GGM, we think this example nicely illustrates the simplicity of our approach compared to directly proving the corresponding bounds in the BF-GGM.

Recall that in both problems the adversary receives as input group elements $(g, g^x, \ldots, g^{x^d})$ for $x \leftarrow_\$ \mathbb{Z}_p$ and has access to no decisional oracles. In d-strong-DL the goal is to compute x, in d-strong-DHI the group element $g^{1/x}$ (assuming $x \neq 0$). We define geo-d-strong-DL $= \mathrm{SHS}(1, d)$, i.e., adversary A has access to oracle Eval accepting all univariate polynomials of degree at most d, and has to return x. Note that the d-strong Diffie-Hellman inversion problem is the special case $\text{MI-Uber}(1, (1, 1), (X, \ldots, X^d), (1/X))$ of the Uber problem. We obtain the following.

Lemma 6. *Let $d \in \mathbb{N}$. Then, for every adversary A against* geo-d-strong-DHI *making at most q queries, there exist adversary B against* geo-d-strong-DL *making at most $q + d + 1$ queries such that*

$$\mathrm{Adv}^{\text{geo-}d\text{-strong-DL}}(\mathsf{B}) \geq \mathrm{Adv}^{\text{geo-}d\text{-strong-DHI}}(\mathsf{A})\ .$$

Proof. Note that the case $x = 0$ can be efficiently recognized by B. Thus assume $x \neq 0$. Further, note that the two games only differ by the expected solution and winning condition. Indeed in both games oracle Eval is defined with respect to indeterminate x and answers queries for polynomials in $\mathrm{Span}(1, X, \ldots, X^d)$, i.e., all polynomials of degree at most d. Thus, adversary B can provide A with a perfect simulation of d-strong-DHI by simply forwarding all Eval queries. Let $\hat{F} \leftarrow \mathsf{A}^{\mathrm{Eval}}$. If A wins, then we have $\sum_{i=0}^{d} a_i X_i = \hat{F} \in \mathrm{Span}(1, \vec{X})$, and $1/x = \sum_{i=0}^{d} a_i x^i = \hat{F}(x)$. As $x \neq 0$ there exists $a_i \neq 0$. Thus the polynomial $F = X \cdot \hat{F}(X) - 1$ is nontrivial, of degree at most $d+1$, and x must be one of its at most $d+1$ roots. B computes all roots y_j of F, uses at most $d+1$ queries of the form $X - y_j$ to Eval to determine x, and returns it as its solution. □

As a consequence we obtain the following preprocessing bounds.

Corollary 7. *Let* A, B *be* (s,q)*-adversaries such that* $q \geq d$ *in the* AI-GGM *against* d-strong-DL *and* d-strong-DHI*, respectively. Then we have* $\mathrm{Adv}^{d\text{-strong-DL}}(\mathsf{A}) \in \tilde{\mathcal{O}}\left(\frac{d(sq^2+q^2)}{p}\right)$ *and* $\mathrm{Adv}^{d\text{-strong-DHI}}(\mathsf{B}) \in \tilde{\mathcal{O}}\left(\frac{d(sq^2+q^2)}{p}\right)$.

Proof. By Lemma 2, every adversary against geo-d-strong-DL = SHS$(1,d)$ making at most q' queries has advantage bounded by $e\,dq'/p$. Thus, if we set $q' = q(M+3q) + Mq + q(k+1) + k(k+1) \in \mathcal{O}(q^2 + qM)$ and $r = 1$, by Theorem 4, every (s,q)-adversary against d-strong-DL in BF-GGM has advantage bounded by

$$\frac{edq'}{p} + \frac{2dq^2 + 2d^2q + dq + d^2}{p} \in \mathcal{O}\left(\frac{dq^2 + dqM}{p}\right)$$

as a larger advantage would contradict the bound for geo-d-strong-DL. Now the statement follows from Theorem 1 by observing that $q_{\mathrm{comb}} = q + d\lceil \log(p) \rceil$, and setting $\gamma = 1/p$ and $M = 6(s+\log(p))(q+d\lceil \log(p) \rceil)$.

Regarding the bound for d-strong-DHI, by Lemma 6 we can bound the advantage of every q'-adversary against geo-d-strong-DHI by $e\,d(q'+d+1)/p$. Then, the second part of the statement follows analogously to the above. □

From geo-d-strong-DL to geo-Uber. We consider $\mathrm{Uber}(t, F_1, \ldots, F_k, F^*) := \text{MI-Uber}(t, (1,1), \vec{F}, F^*)$, the subclass of single instance Uber problems without decisional oracles. As before, to make the problem nontrivial, we require that $F^* \notin \mathrm{Span}(\vec{F})$. Note that this class contains several problems considered in the AGM setting in prior work as, for example, the CDH, square Diffie-Hellman, and strong Diffie-Hellman assumptions [FKL18], as well as the CDH variants in cyclic groups and bilinear groups of [MTT19].

We give a reduction from geo-d-strong-DL = SHS$(1,d)$ to geo-Uber by translating the AGM reduction from d-strong-DL to Uber in [BFL20] to the geometric setting. A key idea of the reduction in [BFL20] is to rerandomize the element g^x obtained from the d-strong-DL game by raising it to random powers y_i and multiplying the result by g^{z_i} for random z_i. This implicitly sets the secrets for the Uber adversary to $x_i = y_i x + z_i$. We adapt this idea to our setting by letting the reduction explicitly set the secrets for the geo-Uber solver to $P_i = y_i X + z_i$ for random values y_i, z_i, where X is an indeterminate. Then, when the geo-Uber outputs a multivariate polynomial, the reduction substitutes in the P_i for the corresponding variables and solves the resulting univariate polynomial for X. In the proof of Theorem 9 we make use of the following fact.

Lemma 8 ([BFL20, Lemma 2.1])**.** *Let* $F(X_1, \ldots, X_t) \in \mathbb{Z}_p[X_1, \ldots, X_t]$ *be a multivariate polynomial of degree* d*. Then* $F(y_1X + z_1, y_2X + z_2, \ldots, y_tX + z_t)$ *is a polynomial in* $\mathbb{Z}_p([y_1, \ldots, y_t, z_1, \ldots, z_t])[X]$ *and its coefficient of maximal degree is a polynomial in* $\mathbb{Z}_p[y_1, \ldots, y_t]$ *of degree* d*.*

Theorem 9. *Let* A *be a* geo-$\mathrm{Uber}(t, F_1, ..., F_k, F^*)$ *solver which makes at most* q *queries, where* $F^*, F_1, \ldots, F_k$ *are polynomials in* t *indeterminates of degree at most* d*, such that* $F^* \notin \mathrm{Span}(1, F_1, \ldots, F_k)$*. Using* A*, we can construct a* geo-d-strong-DL *solver* B *that makes* $q + d$ *queries and satisfies*

$$\mathrm{Adv}^{\text{geo-}d\text{-strong-DL}}(\mathsf{B}) = \mathrm{Adv}^{\text{geo-Uber}(t, F_1, \ldots, F_k, F^*)}(\mathsf{A}) - \frac{d}{p}\ .$$

Proof. The geo-d-strong-DL solver B needs to find one hidden value $x \in \mathbb{Z}_p$, but construct t independent hidden values $\vec{x} = (x_1, \ldots, x_t) \in \mathbb{Z}_p^t$ for the geo-Uber solver A. To this end, B sets up polynomials $P_i = y_i X + z_i$ for all $i \in [t]$, where X is an indeterminate, and y_i, z_i are i.i.d. uniform values from $\mathbb{Z}_p$. It then provides A with a perfect simulation of geo-Uber for the choice of secrets $x_i = P_i(x) = y_i x + z_i$ as described below. Note that, indeed, $\vec{x}$ is uniformly random in $\mathbb{Z}_p^t$.

B has access to the geo-d-strong-DL oracle Eval, that takes as input univariate polynomials of degree at most d. In order to run A, it needs to answer evaluation queries made by the latter, which consist of n-dimensional polynomials spanned by $1, F_1, \ldots, F_k$. When A makes a hypersurface query P, B first checks whether P is a polynomial in t variables $X_1, \ldots, X_t$ in the span of $1, F_1, \ldots, F_k$, and outputs $\perp$ if not. Otherwise, B sets $X_i = P_i$ for all $i \in [t]$ and then queries the resulting univariate polynomial $P(P_1(X), ..., P_k(X))$ to the geo-d-strong-DL oracle Eval. Note that, since P is in the span of $1, F_1, \ldots, F_k$, which have total degree at most d, the degree of the resulting univariate polynomial is at most d, by Lemma 8. So, Eval answers the query with 0 or 1. By choice of the x_i, we have that $P(x_1, \ldots, x_t) = 0 \Leftrightarrow P(P_1(x), \ldots, P_t(x)) = 0$, so B can simply forward this answer to A.

Finally, A outputs a t-dimensional polynomial $\hat{F}(X_1, \ldots, X_t)$. Consider the polynomial $F(X_1, \ldots, X_t) = F^* - \hat{F}$. B checks whether F has at least one (non-constant) non-zero coefficient and aborts if not. Observe that, since $F^* \notin \mathrm{Span}(1, F_1, \ldots, F_k)$, if we also have $\hat{F} \in \mathrm{Span}(1, F_1, \ldots, F_k)$ (which is a condition for A to succeed), we obtain that F will indeed have at least one (non-constant) non-zero coefficient.

Now, B sets $X_i = P_i(X)$ in $F(X_1, \ldots, X_t)$ to obtain a univariate polynomial of degree at most d. By Lemma 8, this polynomial is non-zero with probability $1 - d/p$, since the highest degree coefficient depends on the y_i, which are uniformly random. B computes the roots $r_1, \ldots, r_d$ of this polynomial and then queries $X - r_i$ to Eval for all $i \in [d]$. If Eval outputs 1 for one of the roots, B outputs that root. To show that

$$\mathrm{Adv}^{\text{geo-}d\text{-strong-DL}}(\mathsf{B}) = \mathrm{Adv}^{\text{geo-Uber}(n,F_1,\ldots,F_k,F^*)}(\mathsf{A}) - \frac{d}{p} \ ,$$

we note that the view of A is a perfect simulation of the geo-Uber game. Further, by the definition of geo-Uber, if A succeeds, then $\vec{x}$ must be a root of F. □

We obtain the following bounds in the GGM and AI-GGM.

Corollary 10. *In the GGM, every adversary A against problem $\mathrm{Uber}(t, \vec{F}, F^*)$ making at most q queries, has advantage*

$$\mathrm{Adv}^{\mathrm{Uber}(t,\vec{F},F^*)}(\mathsf{A}) \in \tilde{\mathcal{O}}\left(\frac{dq^2}{p}\right) \ ,$$

where d is the maximum among the total degrees of $\vec{F}$ and F^*.

Proof. By Lemma 2, every adversary against geo-d-strong-DL $= \mathrm{SHS}(1, d)$ making at most q_2 queries has advantage bounded by $e\, dq_2/p$. Thus, if we set $q_2 = q_1 + d$ then, by Theorem 9, every adversary against geo-Uber making q_1 queries has advantage bounded by $e\, d(q_1 + d)/p + d/p \in \mathcal{O}(dq_1/p)$. Now, setting $q_1 = 2q^2 + (q + k)(k + 1) \approx \mathcal{O}(q^2)$, the statement of the corollary follows from Theorem 3, with $r = 1$. □

Corollary 11. *Every (s, q)-adversary A against problem $\mathrm{Uber}(t, \vec{F}, F^*)$ in the* AI-GGM*, has advantage*

$$\mathrm{Adv}^{\mathrm{Uber}(t,\vec{F},F^*)}(\mathsf{A}) \in \tilde{\mathcal{O}}\left(\frac{d(sq^2 + q^2)}{p}\right) \ .$$

Proof. Following the exact same argument as in the proof above, of Corollary 10, we have that any adversary against geo-Uber, making q_2 queries, has advantage bounded by $e\, d(q_2 + d)/p + d/p \in \mathcal{O}(dq_2/p)$. Now, using Theorem 4, and setting $q_2 = (k + q_1)(M + 1 + k) + q_1(2k + 5q_1) \approx \mathcal{O}(q(M + q))$, and $r = 1$, we have that any adversary in the BF-GGM against $\mathrm{Uber}(t, \vec{F}, F^*)$ making q_1 queries, has advantage bounded by

$$\frac{e\, d(q_2 + d) + d}{p} + 2q_1 \frac{(q_1 + k)d}{p} \approx \mathcal{O}\left(\frac{d(q_1^2 + q_1 M)}{p}\right) \ .$$

Now, the statement follows from Theorem 1 by observing that $q_{comb} \leq q + tk\lceil\log(p)\rceil$, and setting $\gamma = 1/p$ and $M = 6(s + \log(p))(q + tk\lceil\log(p)\rceil)$.

□

From (m, m)geo-MI-gap-DL **to** (m, n)geo-MI-gap-CDH**.** In this section we revisit the GGM lower bounds for the multi-instance gap-CDH problem from [AGK20]. We (re)establish the claimed bound in the GGM and additionally obtain new preprocessing bounds for the (m, n)MI-CDH problem. To do so, we show that, for $m, n \in \mathbb{N}$ with $m \leq n$, the algebraic reduction from (m, m)MI-gap-DL to (m, n)MI-gap-CDH in [AGK20, Thm. 5] can easily be translated to one for the corresponding geometric search problems.

Recall that the (m, m)MI-gap-DL problem requires A, on input $(g, g^{x_1} \ldots, g^{x_m})$, to return all discrete logarithms $x_1, \ldots, x_m$. In the (m, n)MI-gap-CDH problem, on the other hand, the adversary gets as input $(g, g^{x_1}, \ldots, g^{x_n}, g^{y_1}, \ldots, g^{y_n})$, and has to return an index set $I \subseteq [n]$ of size at least m, as well as the group elements $g^{x_i y_i}$ for all $i \in I$. In both problems the term "gap" refers to the fact that the adversary has access to a DDH oracle that, on input group elements $(g^{x'}, g^{y'}, g^{z'})$, returns 1 if $g^{x'y'} = g^{z'}$ and 0 else. Thus, (m, n)MI-gap-CDH corresponds to the special case of MI-Uber, where $2n$ indeterminates $x_1, \ldots, x_n$, $y_1, \ldots, y_n$ are sampled, the input polynomials are $X_1, \ldots, X_n, Y_1, \ldots, Y_n$, the target polynomials $X_1Y_1, \ldots, X_nY_n$, and there is one decisional oracle defined by the polynomial $Z_1Z_2 - Z_3$.

The corresponding geometric search problem (m, n)geo-MI-gap-CDH thus samples $\vec{x}, \vec{y} \leftarrow_{\$} \mathbb{Z}_p^n$, and requires adversary A to return an index set $I \subseteq [n]$ of size at least m, as well as polynomials $\hat{F}_i \in \mathrm{Span}(1, \vec{X}, \vec{Y})$, such that for all $i \in I$ we have $\hat{F}_i(\vec{x}, \vec{y}) - x_i y_i = 0$. To do so, A has access to oracles Eval, that on input $F \in \mathrm{Span}(1, \vec{X}, \vec{Y})$ returns $[F(\vec{x}, \vec{y}) = 0]$, and Dec, that on input $F_1, F_2, F_3 \in \mathrm{Span}(1, \vec{X}, \vec{Y})$ returns $[(F_1 \cdot F_2 - F_3)(\vec{x}, \vec{y}) = 0]$.

Problem (m, m)geo-MI-gap-DL samples $\vec{z} \leftarrow_{\$} \mathbb{Z}_p^m$, and requires adversary A to return all of $\vec{z}$. A has access to oracles Eval, that on input of $F \in \mathrm{Span}(1, \vec{Z})$ returns $[F(\vec{Z}) = 0]$, and Dec, that on input of $F_1, F_2, F_3 \in \mathrm{Span}(1, \vec{Z})$ returns $[(F_1 \cdot F_2 - F_3)(\vec{z}) = 0]$.

Regarding our reduction between the geometric search-problems we obtain the following result. It allows to formally reestablish the lower bounds from [AGK20] on the hardness of (m, n)MI-gap-CDH in the generic group model. Afterwards, we derive new preprocessing bounds for (m, n)MI-CDH.

Lemma 12. *Let $m \leq n \in \mathbb{N}$ and let A be an adversary against (m, n)*geo-MI-gap-CDH *that makes at most q queries to oracles* Eval *and* Dec*. Then, there exists an adversary B against (m, m)*geo-MI-gap-DL *making the same number of oracles queries such that*

$$\mathrm{Adv}^{(m,m)\text{geo-MI-gap-DL}}(\mathsf{B}) \geq 2^{-m}\mathrm{Adv}^{(m,n)\text{geo-MI-gap-CDH}}(\mathsf{A}) \ .$$

Proof. We follow the reduction from the proof of [AGK20, Thm. 5], adapting it to the geometric setting as follows. Solver B for (m, m)geo-MI-gap-DL needs to compute the hidden values $z_1, \ldots, z_m$. To this end, it implicitly sets up secrets $\vec{x}, \vec{y} \in \mathbb{Z}_n$, distributed as required for (m, n)geo-MI-gap-CDH, as follows. It samples random vectors $r, s \leftarrow_{\$} \mathbb{Z}_p^n$, a subset $J \subset [n]$ uniformly at random from $2^{[n]}$, and sets $(v_{i,j})_{i,j} \in \mathbb{Z}_p^{n \times m}$ such that, when seen as a matrix, every $(m \times m)$ submatrix is invertible (concretely, [AGK20] uses a Vandermonde matrix). Then, for all $i \in J$, B defines the helper polynomials $P_{x,i} = \sum_{j=1}^{m} v_{i,j} Z_j + r_i$ and $P_{y,i} = s_i$. For $i \in [n] \setminus J$, it sets $P_{x,i} = s_i$ and $P_{y,i} = \sum_{j=1}^{m} v_{i,j} Z_j + r_i$. Then it runs adversary A. Reduction B provides A with a simulation of (m, n)geo-MI-gap-CDH with secrets $x_i = P_{x,i}(\vec{z})$ and $y_i = P_{y,i}(\vec{z})$. In particular, for $i \in J$, the value x_i corresponds to a re-randomization of $\vec{z}$ and y_i is a known constant. For $i \in [n] \setminus J$, the opposite holds.

When A queries oracle Eval on polynomial $F \in \mathbb{Z}_p[\vec{X}, \vec{Y}]$, the reduction queries its own Eval oracle on the polynomial $F' = F(P_{x,1}, \ldots, P_{x,n}, P_{y,1}, \ldots, P_{y,n}) \in \mathbb{Z}_p[\vec{Z}]$, and simply forwards its response to A. Since all $P_{x,i}$ and $P_{y,i}$ are affine functions, we have that $F \in \mathrm{Span}(1, \vec{X}, \vec{Y})$ exactly if $F' \in \mathrm{Span}(1, \vec{Z})$. Thus, any invalid query to Eval by A will also trigger a corresponding invalid query by B to its oracle, who will thus forward the correct answer to A. Further, valid queries are answered correctly as well, since we have

$$F(\vec{x}, \vec{y}) = F(P_{x,1}(\vec{z}), \ldots, P_{x,n}(\vec{z}), P_{y,1}(\vec{z}), \ldots, P_{y,n}(\vec{z})) = F'(\vec{z}) \ .$$

Similarly, queries $\mathrm{Dec}(F_1, F_2, F_3)$ are answered by forwarding the response to the query $\mathrm{Dec}(F'_1, F'_2, F'_3)$, where $F'_i = F_i(P_{x,1}, \ldots, P_{x,n}, P_{y,1}, \ldots, P_{y,n})$. It follows from the same argument as above that this corresponds to a perfect simulation of the game with $x_i = P_{x,i}(\vec{z})$, $y_i = P_{y,i}(\vec{z})$.

At the end of the game A outputs an index set $I \subseteq [n]$ of size at least m, and polynomials $\hat{F}_i \in \mathbb{Z}_p[\vec{X}, \vec{Y}]$. It remains to argue that, if this is a correct solution to (m,n)geo-MI-gap-CDH, then A is able to extract the values $z_1, \ldots, z_m$ with probability at least 2^{-m}. Note that, for A to be able to win, it is necessary that all $F_i \in \mathrm{Span}(1, \vec{X}, \vec{Y})$, and thus in this case they can be seen as a system $A'\vec{X} + B'\vec{Y} + \vec{c'}$ of affine linear polynomials in $\mathbb{Z}_p[\vec{X}, \vec{Y}]$, where $A', B' \in \mathbb{Z}_p^{m \times n}$ and $\vec{c'} \in \mathbb{Z}_p^m$. Further, in order to solve (m,n)geo-MI-gap-CDH, we must have that

$$A'\vec{x} + B'\vec{y} + \vec{c'} = \vec{u} \quad \text{where } \vec{u}_j = x_{i_j} y_{i_j} \text{ for } j \in [|I|] \text{ and } i_j \in I \ . \tag{1}$$

From here on, we can copy the arguments from [AGK20]: Their reduction, on input (m,m)MI-gap-DL challenge $(g^{z_1}, \ldots, g^{z_m})$, sets up a (m,n)MI-gap-CDH challenge consisting of group elements $g^{x_i} = g^{P_{x,i}(\vec{z})}$ and $g^{y_i} = g^{P_{y,i}(\vec{z})}$. Next, it runs the algebraic adversary, forwarding all group operation and DDH oracle queries. It then receives a solution consisting of an index set I and a vector of group elements $\vec{W}$ that come with an algebraic justification $(A', B', \vec{c'})$ that, if the adversary succeeds in solving its challenge, must satisfy Equation 1. The proof then argues that the reduction is able to extract all values $\vec{z}$ with probability 2^{-m} ([AGK20, Fig. 10, lines 09-33; Prop. 1&2]). As the values $\vec{z}, \vec{x}, \vec{y}$ are distributed equally in their reduction and ours, and since they have to satisfy the same system of equations, we obtain that if A solves its (m,n)geo-MI-gap-CDH challenge, then B is able to compute all of $\vec{z}$ with probability at least 2^{-m}. □

Corollary 13. *Let $m \leq n \leq p$. For every $r \in \mathbb{N}$, in the generic group model every adversary A against problem (m,n)*MI-gap-CDH *that makes at most q oracle queries has advantage bounded by*

$$\mathrm{Adv}^{(m,m)\text{MI-gap-CDH}}(\mathsf{A}) \in \tilde{\mathcal{O}}\left(\left(\frac{rq^2}{mp}\right)^m + q\left(\frac{q}{p}\right)^r\right)$$

*In particular, to achieve constant success probability it is necessary that $q \in \Omega(\sqrt{mp})$. The same bound holds with respect to (m,n)*MI-gap-DL.

Proof. Note that (m,m)geo-MI-gap-DL is at least as hard as the search-by-hypersurface problem $\mathrm{SHS}(m,2)$ for degree 2. This holds as the $\mathrm{Eval}_{\mathrm{SHS}}$ oracle allows arbitrary polynomials up to degree 2, and thus can be directly used to answer any $\mathrm{Eval}_{\text{MI-gap-DL}}$ query. Further, any query to oracle $\mathrm{Dec}_{\text{MI-gap-DL}}(F_1, F_2, F_3)$ can be answered by forwarding the answer to $\mathrm{Eval}_{\mathrm{SHS}}(F_1 \cdot F_2 - F_3)$ where we use that all valid F_i have degree bounded by 1. Therefore, by Lemma 2, for any (m,m)geo-MI-gap-DL adversary that makes at most q' queries, it holds that

$$\mathrm{Adv}^{(m,m)\text{geo-MI-gap-DL}} \leq 1/2 \cdot ((e\,2q')/(pm))^m \ .$$

In turn, by Lemma 12, we can bound the advantage of any adversary against (m,n)geo-MI-gap-CDH by

$$\mathrm{Adv}^{(m,m)\text{geo-MI-gap-CDH}} \leq 1/2\,((e\,4q')/(pm))^m \ .$$

Now if we set $q' = 2rq^2 + q\,((2k+m)(r-1)+k+1) + k\,(k+1+m(r-1)) \approx 2rq^2$ then by Theorem 3 we obtain that every (m,m)MI-gap-CDH adversary making at most q' queries has advantage bounded by

$$\frac{1}{2} \cdot \left(\frac{e\,2q'}{pm}\right)^m + (2q+m)\left(\frac{2(q+n)}{p}\right)^r \in \tilde{\mathcal{O}}\left(\left(\frac{rq^2}{mp}\right)^m + q\left(\frac{q}{p}\right)^r\right) \ .$$

As MI-gap-CDH is at least as hard as MI-gap-DL, the same bound applies to the latter problem. □

Regarding preprocessing we obtain the following new lower bound.

Corollary 14. *Let $m \leq n \leq p$. For every $r \in \mathbb{N}$, in the generic group model with preprocessing every adversary A against problem (m,n)MI-CDH that receives advice bounded by s and makes at most q oracle queries has advantage bounded by*

$$\mathrm{Adv}^{(m,m)\text{MI-CDH}}(\mathsf{A}) \in \tilde{\mathcal{O}}\left(\left(\frac{q^2 s + rq^2}{mp}\right)^m + q\left(\frac{q}{p}\right)^r\right) \ .$$

In particular, to achieve constant success probability it is necessary that $q^2 s \in \tilde{\Omega}(mp)$. The same bound holds for (m,n)MI-DL.

Proof. Note that (m,m)geo-MI-DL = SHS$(m,1)$, and that Lemma 12 also holds with respect to non-gap problems (m,n)geo-MI-CDH and (m,m)geo-MI-DL (the only thing changing in the reduction is that one no longer has to handle Dec queries). Thus we can bound the advantage of any adversary making at most q' oracle queries by $\mathrm{Adv}^{(m,n)\text{geo-MI-CDH}} \leq 1/2\left((e\,2q')/(pm)\right)^m$. Now, if we set $q' = q(M + (2r+1)q) + kM + q\left(2rn + 1 + m(r-1)\right) + 2n\left(2n + 1 + m(r-1)\right) \approx q(M + (2r+1)q)$ then, by Theorem 4, we can bound the advantage of any adversary against (m,n)MI-CDH making at most q oracle-queries in the bit fixing model by

$$\begin{aligned}\mathrm{Adv}^{(m,m)\text{MI-gap-CDH}} &\leq \frac{1}{2}\left(\frac{2e\,q'}{pm}\right)^m + (2q+m)\left(\frac{2(q+2n)}{p}\right)^r \\ &\in \tilde{\mathcal{O}}\left(\left(\frac{q(M+rq)}{mp}\right)^m + q\left(\frac{q}{p}\right)^r\right) \ .\end{aligned}$$

Now the corollary's statement follows from observing that the MI-gap-CDH requires at most $2n\lceil\log(p)\rceil$ group-operation queries to set up the challenge group elements, and by setting $\gamma = 1/p$ and $M = 6(s + \log(p))(q + 2n\lceil\log(p)\rceil) \in \tilde{\mathcal{O}}(sq)$. As MI-CDH is at least as hard as MI-DL, the same bound also applies to the latter problem. □

4.2 Reductions between geometric search-problems corresponding to the bilinear GGM

From geo-2d-strong-DL **to** geo-MI-Uber$_\phi$ **in bilinear groups.** In this section we revisit the lower bound for the

$$\mathrm{Uber}_\phi := \text{MI-Uber}_\phi(t, (1,0,0,1), \vec{F}_1, \vec{F}_2, \vec{F}_T, F_1^*, F_2^*, F_T^*)$$

problem in the bilinear GGM from [BBG05, Boy08]. While the bound in Theorem 15 is of the same order as in [BBG05, Boy08], it demonstrates how to apply our techniques in bilinear groups. Their proofs are similar to the ones for Uber in the GGM and AI-GGM. The proof of Theorem 15 closely follows the proof of [BFL20, Theorem 3.5].

Theorem 15. *Let A be a* geo-MI-Uber$_\phi(t, (1,0,0,1), \vec{F}_1, \vec{F}_2, \vec{F}_T, F_1^*, F_2^*, F_T^*)$ *solver which makes at most q queries, where $F_J^*, F_{J,i}$ for $J \in \{1,2,T\}$ and $i \in [k_J]$ are polynomials in t indeterminates of total degree at most d; and such that there is $\tilde{J} \in \{1,2,T\}$ such that $F_{\tilde{J}}^* \notin \mathrm{Span}_{\tilde{J}}(\vec{F}_1, \vec{F}_2, \vec{F}_T, \phi)$. Using A, we can construct a* geo-2d-strong-DL *solver B in $\mathbb{G}_{\tilde{J}}$ that makes $q + 2d$ queries and satisfies*

$$\mathrm{Adv}^{\text{geo-2}d\text{-strong-DL}}(\mathsf{B}) = \mathrm{Adv}^{\text{geo-MI-Uber}(t,(1,0,0,1),\vec{F}_1,\vec{F}_2,\vec{F}_T,F_T^*)}(\mathsf{A}) - \frac{2d}{p} \ .$$

Proof. Recall that the geo-2d-strong-DL problem corresponds to the special case SHS$(1, 2d)$ of the search-by-hyperplane problem. A solver B for it, thus, needs to find one hidden value $x \in \mathbb{Z}_p$. Our solver in this proof will also need to construct t independent hidden values $\vec{x} = (x_1, \ldots, x_t) \in \mathbb{Z}_p^t$, in order to provide a simulation for the geo-MI-Uber solver A, which it will run as a subroutine. To this end, B sets up polynomials

$R_i = y_i X + z_i$ for all $i \in [t]$, where X is an indeterminate and y_i, z_i are i.i.d. uniform values from $\mathbb{Z}_p$. It will use $x_i = R_i(x) = y_i x + z_i$ as the t hidden values, which will allow it to output a perfect simulation, as described below. Note that indeed $\vec{x}$ is uniformly random in $\mathbb{Z}_p^n$.

B has access to the geo-$2d$-strong-DL oracle Eval, that takes as input univariate polynomials of degree at most $2d$. It needs to answer A's oracle queries to Iso, Eval and Bil, which consist of t-dimensional polynomials in $\mathrm{Span}_J(\vec{F}_1, \vec{F}_2, \vec{F}_T, \phi)$ for some $J \in \{1, 2, T\}$. B does so as follows.

- When A makes a query $\mathrm{Iso}(P_1, P_2)$, B checks if $P_1 \in \mathrm{Span}_1(\vec{F}_1, \vec{F}_2, \vec{F}_T, \phi)$ and $P_2 \in \mathrm{Span}_2(\vec{F}_1, \vec{F}_2, \vec{F}_T, \phi)$ and outputs $\perp$ if not. Otherwise, B sets $X_i = R_i$ for all $i \in [t]$ into the polynomial $P_1 - P_2$ and queries the resulting univariate polynomial to the geo-$2d$-strong-DL oracle Eval. Note that since P_1 and P_2 are spanned linearly by polynomials of total degree at most d, the degree of the resulting univariate polynomial is at most $d \leq 2d$. Thus, Eval answers the query without aborting, with $[(P_1 - P_2)(x) = 0] \in \{0, 1\}$. Then, B forwards the answer to A.

- When A makes a hypersurface query (J, P) to Eval, where $J \in \{1, 2, T\}$ and P is a polynomial in t variables $X_1, \ldots, X_t$, B first checks if P is in $\mathrm{Span}_J(\vec{F}_1, \vec{F}_2, \vec{F}_T, \phi)$ and outputs $\perp$ if not. Otherwise, B sets $X_i = R_i$ for all $i \in [t]$ and then queries the resulting univariate polynomial $P(R_1(X), ..., R_t(X))$ to the geo-$2d$-strong-DL oracle Eval. By the same argument as above, the resulting polynomial has degree at most d for $J \in \{1, 2\}$ and at most $2d$ for $J = T$. So, Eval answers the query without aborting. By choice of the x_i, we have that $P(x_1, \ldots, x_t) = 0 \Leftrightarrow P(R_1(x), \ldots, R_t(x)) = 0$, so B can simply forward this answer to A.

- When A makes a query (P_1, P_2, P_T) to Bil, B checks if $P_J \in \mathrm{Span}_J(\vec{F}_1, \vec{F}_2, \vec{F}_T, \phi)$ for all $J \in \{1, 2, T\}$ and outputs $\perp$ if not. Otherwise, B sets $X_i = R_i$ for all $i \in [t]$ into the polynomial $P_1 \cdot P_2 - P_T$ and queries the resulting univariate polynomial to the geo-$2d$-strong-DL oracle Eval. By the same argument as above, the resulting polynomial has degree at most $2d$, so B can just forward the oracle's output to A.

- Eventually, A outputs three t-variate polynomials $\hat{F}_1(X_1, \ldots, X_t)$, $\hat{F}_2(X_1, \ldots, X_t)$, $\hat{F}_T(X_1, \ldots, X_t)$. Let $J \in \{1, 2, T\}$ be such that $F_J^* \notin \mathrm{Span}_J(\vec{F}_1, \vec{F}_2, \vec{F}_T, \phi)$, which exists by assumption, and consider the polynomial $F(X_1, \ldots, X_t) = F_J^* - \hat{F}_J$. As $F_J^* \notin \mathrm{Span}_J(\vec{F}_1, \vec{F}_2, \vec{F}_T, \phi)$ and $\hat{F}_J \in \mathrm{Span}_J(\vec{F}_1, \vec{F}_2, \vec{F}_T, \phi)$, we obtain that F has at least one (non-constant) non-zero coefficient. Now, B sets $X_i = R_i(X)$ into $F(X_1, \ldots, X_t)$ to obtain a univariate polynomial of degree at most $2d$. By Lemma 8, this polynomial is non-zero with all but probability $2d/p$, since the highest degree coefficient depends on the y_i, and these are uniformly random.

- B computes the roots $r_1, \ldots, r_{2d}$ of this polynomial and queries $X - r_i$ to Eval for all $i \in [2d]$. If Eval outputs 1 for one of the roots, B outputs that root.

To show that

$$\mathrm{Adv}^{\text{geo-}d\text{-strong-DL}}(\mathsf{B}) = \mathrm{Adv}^{\text{geo-MI-Uber}(t,(1,0,0,1),\vec{F}_1,\vec{F}_2,\vec{F}_T,F_T^*)}(\mathsf{A}) - \frac{2d}{p-1} \ ,$$

we note that the view of A is a perfect simulation of the geo-MI-Uber game. Further, by the definition of geo-MI-Uber, if A succeeds then $\vec{x}$ must be a root of F. □

Corollary 16. *Let* A *be an adversary against* Uber_ϕ *in the bilinear* GGM *that makes at most* q *oracle queries. Then* $\mathrm{Adv}^{\mathrm{Uber}_\phi}(\mathsf{A}) \in \tilde{\mathcal{O}}\left(\frac{q^2 d + dqk + d^2 + dk^2}{p}\right)$.

Proof. By Lemma 2, every adversary against geo-d-strong-DL = SHS$(1, d)$ making at most q' queries has advantage bounded by $e\,dq'/p$. Thus, if we set $q' = q + 2d$ then, by Theorem 15, every q-adversary against geo-Uber_ϕ in the bilinear GGM has advantage bounded by

$$\frac{edq'}{p} + \frac{2d}{p} \ ,$$

BLSGen(p, $\mathbb{G}$, $\mathbb{G}_T$, g, e)	BLSSig(m, sk)	BLSVer(m, s)
00 $sk = x \leftarrow \mathbb{Z}_p$	03 $s := H(m)^{sk}$	05 return $[e(H(m), pk) = e(s, g)]$
01 $pk := g^x$	04 return s	
02 return (pk, sk)		

Problem UF-CMA$_{\text{BLS}}(p, \mathbb{G}, \mathbb{G}_T, g, e)$
06 $sk = x \leftarrow \mathbb{Z}_p$
07 $pk := g^x$
08 $Q := \emptyset$
09 $\sigma_0 \leftarrow \mathcal{L}_1(g)$
10 $\sigma_{pk} \leftarrow \mathcal{L}_1(pk)$
11 $(m^*, \sigma_{s^*}) \leftarrow \mathsf{A}^{\text{Sign,RO,GrpOp,Bil}}(\sigma_0, \sigma_{pk})$
12 $s^* := \mathcal{L}_1^{-1}(\sigma_{s^*})$
13 **return** $[m^* \notin Q \wedge \text{BLSVer}(\text{m}^*, \text{s}^*)]$

Oracle Sign(σ_m)
14 $m := \mathcal{L}_1^{-1}(\sigma_m)$
15 $Q := Q \cup \{m\}$
16 $s \leftarrow \text{RO}(m)^{sk}$
17 $\sigma_s \leftarrow \mathcal{L}_1(s)$
18 **return** σ_s

Oracle RO(m)
19 **if** $\nexists(m, \sigma_m) \in T_{\text{RO}}$
20 $\quad\sigma_m \leftarrow_{\$} \{0,1\}^\ell$
21 $\quad T_{\text{RO}} \leftarrow_{\cup} (m, \sigma_m)$
22 **return** σ_m

Oracle GrpOp$(\sigma, \hat{\sigma})$
23 **require** $\sigma, \hat{\sigma} \in \mathcal{L}_1(\mathbb{Z}_p)$
24 **return** $\mathcal{L}_1(\mathcal{L}_1^{-1}(\sigma) + \mathcal{L}_1^{-1}(\hat{\sigma}))$

Oracle Bil$(\sigma, \hat{\sigma})$
25 **require** $\sigma, \hat{\sigma} \in \mathcal{L}_1(\mathbb{Z}_p)$
26 **return** $\mathcal{L}_T(\mathcal{L}_1^{-1}(\sigma) + \mathcal{L}_1^{-1}(\hat{\sigma}))$

Figure 5: Top: BLS signature scheme. Bottom: Unforgeability-under-chosen-message-attack problem for with respect to BLS signatures in the bilinear GGM + ROM, where the random oracle returns group elements of $\mathbb{G}_1$.

as a larger advantage would contradict the bound for geo-d-strong-DL. Thus, if we set $q' = k(k+3) + q((r-1) + k + 3 + 3q)$ and $r = 1$ then, by Theorem 5, every q-adversary against Uber_ϕ in the bilinear GGM has advantage bounded by

$$\frac{edq' + 2d(ed+1)}{p} + \frac{(2q+1)(q+k)d}{p} \in \mathcal{O}\left(\frac{q^2d + dqk + d^2 + dk^2}{p}\right) ,$$

as, again, a larger advantage would contradict the bound for geo-MI-Uber. □

Security of BLS signatures in the bilinear GGM. In this section we give a tight reduction from geo-2-strong-DL = SHS$(1, 2)$ to the unforgeability of BLS signatures under chosen-message attacks defined in Figure 5. We closely follow the proof in [FKL18, Section 6]. Recall that we work in the generic group model for bilinear groups that we presented in Section 2.2. The bilinear groups are of type 1 so, to not explicitly have to work with the isomorphism oracle, we can simply set $\mathbb{G} := \mathbb{G}_1 = \mathbb{G}_2$.

Theorem 17. *Let* A *be an* UF-CMA$_{\text{BLS}}(p, \mathbb{G}, \mathbb{G}_T, g, e)$ *solver in the random-oracle model which makes at most* q *group-operation, bilinear map, and signing queries and at most* q_{RO} *random-oracle queries. Using* A*, we can construct a* geo-2-strong-DL *solver* B *that makes* $\mathcal{O}(q^2)$ *queries and satisfies*

$$\text{Adv}^{\text{geo-2-strong-DL}}(\mathsf{B}) \geq \frac{1}{2}\text{Adv}^{\text{UF-CMA}_{\text{BLS}}(p,\mathbb{G},\mathbb{G}_T,g,e)}(\mathsf{A}) - \frac{4q(3q+2)}{p} - \frac{q_{\text{RO}}(q_{\text{RO}}+q)}{p} .$$

Proof. The geo-2-strong-DL solver B needs to find the hidden value z and has access to oracle Eval, to which it can query polynomials of degree at most 2. To simulate the view of the UF-CMA$_{\text{BLS}}$ solver A, B needs to construct the input pair (σ_0, σ_z), and answer queries made by A to GrpOp and Bil, as well as to the random oracle RO and to the signing oracle Sign. To do this in a consistent manner, B samples labels on the fly and maintains tables T_1 and T_T (for $\mathbb{G}, \mathbb{G}_T$ respectively), that store previously recorded labels σ, together with corresponding polynomials $P(Z)$ in the indeterminate Z. The labels σ will correspond to a perfect simulation of UF-CMA$_{\text{BLS}}$ such that, at every point in time, we have $\mathcal{L}_J^{-1}(\sigma) = P(Z)$ for every entry (P, σ) in T_J. To make sure that the simulation is consistent, B needs to check that no two polynomials P, P' such that $P(\vec{x}) = P'(\vec{x})$ get paired with different labels. This would be equivalent to a group element receiving two different labels. Hence, before sampling a new label for a polynomial P, B checks that there

is no previously recorded polynomial P' in T_J such that $(P - P')(z) = 0$, using the oracle Eval. Similarly, the values for RO are sampled on the fly and stored in table T_{RO}.

In more detail, B does the following: To construct the input pair, B chooses σ_0, σ_Z uniformly at random from $\mathcal{R}_1$ and runs $\mathsf{A}(\sigma_0, \sigma_Z)$. B handles the queries made by A as follows:

- When A makes a query $(\sigma, \hat{\sigma})$ to GrpOp, B does the following:
  - Check if σ and $\hat{\sigma}$ have been recorded in T_1. If σ has not been recorded, B first checks if $\sigma \in \mathcal{R}_1$ returning $\perp$ if not. Otherwise, it samples a random unused element $a \leftarrow_{\$} \mathbb{Z}_p$ and stores the pair (a, σ) in T_1, where a is to be interpreted as a constant polynomial. B does the same if $\hat{\sigma}$ has not been recorded.
  - Let $P, \hat{P}$ be the polynomials corresponding to labels $\sigma, \hat{\sigma}$. For all previously recorded polynomials P' in T_1, query $P + \hat{P} - P'$ to oracle Eval. If Eval outputs 1 for one of the P', look up the corresponding label σ' in T_J and send σ' to A. Otherwise sample a random σ' from the unused values in $\mathcal{R}_1$ and send σ' to A. Then store $(P + \hat{P}, \sigma')$ in T_1.
- When A makes a query $(\sigma, \hat{\sigma})$ to Bil, B does the following:
  - Check if $\sigma, \hat{\sigma}$ have been recorded in T_1. If not, do the procedure described above.
  - Let $(P, \hat{P})$ be the polynomials corresponding to the labels $(\sigma, \hat{\sigma})$. For all P' previously recorded in T_T, query $P \cdot \hat{P} - P'$ to Eval. If Eval outputs 1 for one of the queries, look up the corresponding label σ' and send it to A. Otherwise, sample a random unused label σ' from $\mathcal{R}_T$ and send it to A. Then store $(P \cdot \hat{P}, \sigma')$ in T_T.

In the simulation, group elements σ will correspond to affine functions $F_\sigma = (a_\sigma Z + b)$. Thus, A returning a forgery (m^*, σ_{s^*}) corresponds to the equation

$$F_{\sigma_{s^*}}(z) = a_{s^*} z + b_{s^*} = F_{\sigma_x}(z) \cdot F_{\mathrm{RO}(m^*)}(z) \ , \tag{2}$$

where σ_x is the label corresponding to the public key. To simulate the signing and random oracles B chooses between one of the following two strategies, each with probability 1/2. (The first will be successful in the case $F_{\mathrm{RO}(m^*)}(z) \neq a_{s^*}$, the second in the case $F_{\mathrm{RO}(m^*)}(z) = a_{s^*}$.)

Strategy 1

- B stores (Z, σ_x) in T_1 and sets the public key to σ_x.
- When A makes a query $\mathrm{RO}(m)$, B checks whether T_{RO} already contains a pair (m, σ_m) and, if so, returns σ_m. If not, it samples $\sigma_m \leftarrow_{\$} \mathcal{R}_1$ and checks whether $\sigma_m = \sigma$ for any σ in T_{RO}, T_1, or T_T. If so, it aborts, else stores (m, σ_m) in T_{RO} and returns σ_m. Then, B samples a random $a_m \leftarrow \mathbb{Z}_p$ and stores (a_m, σ_m) in T_1.
- When A makes a query m to Sign, B calls $\sigma_m = \mathrm{RO}(m)$, and recovers the corresponding constant polynomial a_m from T_1. Then, it sets $F(Z) := a_m \cdot Z$ and, for all polynomials F' previously recorded in T_1, queries $F' - F$ to Eval. If Eval answers 1 to one of the queries, it looks up the corresponding label σ_F and sends it to A. Otherwise it samples a random unused value σ_F from $\mathcal{R}_1$, sends it to A and stores (F, σ_F) in T_1.
- When A outputs the pair (m^*, σ_{s^*}), B checks if σ_{s^*} has previously been recorded in T_1. If not, it proceeds as in the corresponding case for GrpOp or Bil. Let $F_{s^*}(z) = a_{s^*} z + b_{s^*}$ be the polynomial corresponding to σ_{s^*} in T_1, which is linear since all entries in T_1 are in $\mathrm{Span}(1, Z)$. Let $\sigma_{m^*} := \mathrm{RO}(m^*)$ and (a_{m^*}, σ_{m^*}) be the corresponding entry in T_1. If A wins, Equation 2 corresponds to

  $$a_{s^*} z + b_{s^*} = z a_{m^*} \ . \tag{3}$$

  If $a_{s^*} - F_{\mathrm{RO}(m^*)}(z) = a_{s^*} - a_{m^*} \neq 0$, B can efficiently compute z from (3) and output the result. Otherwise, B aborts.

Strategy 2

- In this strategy, B fixes a random $x \in \mathbb{Z}_p$ as the public key and stores (x, σ_x) in T_1. The indeterminate Z is only introduced in the output of the random oracle.
- When A makes a random oracle query $\mathrm{RO}(m)$, B checks whether T_{RO} already contains a pair (m, σ_m) and, if so, returns σ_m. If not, it samples $\sigma_m \leftarrow_{\$} \mathcal{R}_1$, checks whether $\sigma_m = \sigma$ for any σ in T_{RO}, T_1, or T_T. If so, it aborts, else stores (m, σ_m) in T_{RO} and returns σ_m. Then, it samples random values $a_{\sigma_m}, b_{\sigma_m} \leftarrow \mathbb{Z}_p$ and stores $(a_{\sigma_m} Z + b_{\sigma_m}, \sigma_m)$ in T_1.
- When A makes a query m to Sign, B calls $\sigma_m = \mathrm{RO}(m)$ and recovers the polynomial F_{σ_m} corresponding to σ_m from T_1. Then it sets $F(Z) := F_{\sigma_m}(Z) \cdot x$. For all polynomials F' previously recorded in T_1, it queries $F' - F$ to Eval. If Eval answers 1 to one of the queries, B looks up the corresponding label σ_F and sends it to A. Otherwise it samples a random unused value σ_F from $\mathcal{R}_1$, sends it to A and stores (F, σ_F) in T.
- When A outputs the pair (m^*, σ_{s^*}), B calls $\sigma_{m^*} = \mathrm{RO}(m)$ and recovers from T_1 the corresponding polynomial $F_{\sigma_{m^*}}(Z) = a_{m^*} Z + b_{m^*}$. Let $F_{\sigma_{s^*}} = a_{s^*} Z + b_{s^*}$ be the polynomial corresponding to σ_{s^*} in T_1, which is linear since all entries in T_1 are in $\mathrm{Span}(1, X)$. If $F_{\mathrm{RO}(m^*)}(z) = a_{s^*}$ we have

$$a_{s^*} = a_{m^*} z + b_{m^*} \ , \tag{4}$$

  so B can efficiently compute z if $a_{m^*} \neq 0$ and output the result. Otherwise, B aborts.

We now upper-bound the number of queries done by B to Eval. First, to set up A's input, B adds 2 entries to the table T_1. During the execution of A, every query to $\mathrm{GrpOp}, \mathrm{Bil}, \mathrm{Sign}$ or RO adds at most 3 polynomials to a table T_J. Thus, at the q'th a query any table T_J contains at most $3q' + 2$ entries, so the q'th query requires at most $3q' + 2$ queries to Eval. Hence, we can bound the number of queries made by B by $q(3q + 2)$.

To show that

$$\mathrm{Adv}^{\text{geo-2-strong-DL}}(\mathsf{B}) \geq \frac{1}{2}\mathrm{Adv}^{\text{UF-CMA}_{\mathrm{BLS}}(p,\mathbb{G},\mathbb{G}_T,g,e)}(\mathsf{A}) - \frac{4q(3q+2)}{p} - \frac{q_{\mathrm{RO}}(q_{\mathrm{RO}}+q)}{p} \ ,$$

we first note that A chooses the correct strategy with probability $1/2$, and the probability that B aborts is at most $q_{\mathrm{RO}}(q_{\mathrm{RO}} + q)/p$. We define the event

$$\mathtt{bad} = \{\exists (F, \sigma_F), (P, \sigma_P) \in T_1 \mid P(\vec{x}) - F(\vec{x}) = 0\} \ ,$$

which corresponds to B assigning a label σ it did not previously receive to a polynomial not consistent with the simulation. If B does not abort, the probability of the bad event is at most $4q(3q+2)/p$, since the degree of recorded polynomials is at most 2, and so the probability of a collision is $2/p$ for a fixed polynomial. Further, B needs to, per attempt, check at most $3q + 2$ polynomials and it has to sample at most $2q$ polynomials for unrecorded labels. Otherwise, B simulates the view of A perfectly in both strategies. □

Corollary 18. *In the bilinear* GGM *for groups of type* 1 *and programmable random-oracle model every adversary* A *that make at most* q *group-operation, bilinear map, and signing queries and at most* q_{RO} *random-oracle queries has advantage of order* $\mathcal{O}((q^2 + q_{\mathrm{RO}}^2)/p)$.

Proof. By Lemma 2 every adversary against geo-2-strong-DL = $\mathrm{SHS}(1, d)$ making at most q' queries has advantage bounded by $e\, 2q'/p$. Thus, if we set $q' = 3q^2 + 2q$, then by Theorem 17 every q-adversary against $\text{UF-CMA}_{\mathrm{BLS}}$ in the bilinear GGM has advantage bounded by

$$\frac{2eq'}{p} + \frac{12q^2 + 8q + q_{\mathrm{RO}}^2 + q_{\mathrm{RO}} q}{p} \in \mathcal{O}\left(\frac{q^2 + q_{\mathrm{RO}}^2}{p}\right) \ ,$$

as a larger advantage would contradict the bound for geo-2-strong-DL. □

References

[ABB+20] Michel Abdalla, Manuel Barbosa, Tatiana Bradley, Stanislaw Jarecki, Jonathan Katz, and Jiayu Xu. Universally composable relaxed password authenticated key exchange. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 278–307. Springer, Heidelberg, August 2020.

[AGK20] Benedikt Auerbach, Federico Giacon, and Eike Kiltz. Everybody's a target: Scalability in public-key encryption. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 475–506. Springer, Heidelberg, May 2020.

[BB04a] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, Heidelberg, May 2004.

[BB04b] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, Heidelberg, May 2004.

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

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

[BFL20] Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. A classification of computational assumptions in the algebraic group model. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 121–151. Springer, Heidelberg, August 2020.

[BL13] Daniel J. Bernstein and Tanja Lange. Non-uniform cracks in the concrete: The power of free precomputation. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 321–340. Springer, Heidelberg, December 2013.

[BL22] Jeremiah Blocki and Seunghoon Lee. On the multi-user security of short schnorr signatures with preprocessing. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 614–643. Springer, Heidelberg, May / June 2022.

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004.

[Boy08] Xavier Boyen. The uber-assumption family (invited talk). In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008: 2nd International Conference on Pairing-based Cryptography*, volume 5209 of *Lecture Notes in Computer Science*, pages 39–56. Springer, Heidelberg, September 2008.

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

[CDG18] Sandro Coretti, Yevgeniy Dodis, and Siyao Guo. Non-uniform bounds in the random-permutation, ideal-cipher, and generic-group models. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 693–721. Springer, Heidelberg, August 2018.

[CK18] Henry Corrigan-Gibbs and Dmitry Kogan. The discrete-logarithm problem with preprocessing. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 415–447. Springer, Heidelberg, April / May 2018.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 33–62. Springer, Heidelberg, August 2018.

[FPS20] Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind schnorr signatures and signed ElGamal encryption in the algebraic group model. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 63–95. Springer, Heidelberg, May 2020.

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

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

[LCH11] Hyung Tae Lee, Jung Hee Cheon, and Jin Hong. Accelerating id-based encryption based on trapdoor dl using pre-computation. Cryptology ePrint Archive, Paper 2011/187, 2011. https://eprint.iacr.org/2011/187.

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

[Mih10] Joseph P. Mihalcik. An analysis of algorithms for solving discrete logarithms in fixed groups. Master's thesis, Naval Postgraduate School (2010), 2010. https://calhoun.nps.edu/bitstream/handle/10945/5395/10Mar_Mihalcik.pdf.

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

[OP01] Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In Kwangjo Kim, editor, *PKC 2001: 4th International Workshop on Theory and Practice in Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 104–118. Springer, Heidelberg, February 2001.

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

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

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

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, Heidelberg, May 1997.

[Unr07] Dominique Unruh. Random oracles and auxiliary input. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 205–223. Springer, Heidelberg, August 2007.

[YK17] Jason H. M. Ying and Noboru Kunihiro. Bounds in various generalized settings of the discrete logarithm problem. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *ACNS 17: 15th International Conference on Applied Cryptography and Network Security*, volume 10355 of *Lecture Notes in Computer Science*, pages 498–517. Springer, Heidelberg, July 2017.

[Yun15] Aaram Yun. Generic hardness of the multiple discrete logarithm problem. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 817–836. Springer, Heidelberg, April 2015.

[Zha22] Mark Zhandry. To label, or not to label (in generic groups). In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part III*, volume 13509 of *Lecture Notes in Computer Science*, pages 66–96. Springer, Heidelberg, August 2022.

[ZZK22] Cong Zhang, Hong-Sheng Zhou, and Jonathan Katz. An analysis of the algebraic group model. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022, Part IV*, volume 13794 of *Lecture Notes in Computer Science*, pages 310–322. Springer, Heidelberg, December 2022.

Supplementary Material

A Omitted Proofs of Section 3

A.1 Proof of Theorem 4

Proof of Theorem 4. The geo-MI-Uber solver B receives as input t, m, n and the polynomials $\vec{F}$, rational functions $\vec{F}^*$, and has access to the oracle Eval. To simulate the view of the MI-Uber solver A, B needs to store the labels fixed by A_1, construct input labels $\sigma_0, \sigma_1 \ldots, \sigma_k$, and answer group operation queries made by A_2 to the oracle GrpOp. To do this in a consistent manner, B samples labels on the fly and maintains a table T, that stores all previously recorded labels σ, each together with a corresponding polynomial $P(\vec{X}) \in \mathrm{Span}(1, \vec{F})$ in the indeterminates $X_1, \ldots, X_t$. The labels σ will correspond to a perfect simulation of MI-Uber such that we have $\mathcal{L}^{-1}(\sigma) = P(\vec{x})$ for every entry (P, σ) in T. As in the proof of Theorem 3, to make sure that the simulation is consistent, B needs to check that no two polynomials P, P' such that $P(\vec{x}) = P'(\vec{x})$ get paired with different labels. This would be equivalent to a group element receiving two different labels. Hence, before sampling a new label for a polynomial P, B needs to check that there is no previously recorded polynomial P' in T such that $(P - P')(\vec{x}) = 0$ using the oracle Eval. Note that, if $P, P' \in \mathrm{Span}(1, F_1, \ldots, F_k)$ then so is $P - P'$, and thus the oracle will not return $\perp$. In more detail, B does the following.

- It samples the range $\mathcal{R} \subseteq \{0,1\}^\ell$ uniformly at random from all subsets of $\{0,1\}^\ell$ of size p and sends it to A_1. Recall that this corresponds to the range of the labeling function $\mathcal{L}$. B receives back a list of M pairs of the form (a_i, σ_i), where $a_i \in \mathbb{Z}_p$ and $\sigma_i \in \mathcal{R}$, as well as a state Γ. For all $i \in [M]$, it stores the pair (a_i, σ_i) in T, where a_i is viewed as a constant polynomial.
- To create the input $(\sigma_0, \sigma_1, \ldots, \sigma_k)$ for A_2, B does the following:
  - To create σ_0, check if the zero polynomial is contained in T. If so, set σ_0 to the corresponding label. Otherwise, set σ_0 to a random unused value from $\mathcal{R}$ and store $(0, \sigma_0)$.
  - To create σ_i for $i \in [k]$, query Eval on $F_i - P'$ for all previously recorded polynomials P' in T. If the answer is 1 for some P', set σ_i to the corresponding label of P'. Otherwise, choose a random unused value from $\mathcal{R}$ and store (F_i, σ_i). Note that for all entries $(P, \sigma) \in T$, P trivially lies in $\mathrm{Span}(1, F_1, \ldots, F_k)$ and additionally $\mathcal{L}^{-1}(\sigma) = P(x)$.

  Then B runs $\mathsf{A}_2(\sigma_0, \sigma_1, \ldots, \sigma_k, \Gamma)$.
- When A_2 makes a query (σ, σ') to GrpOp, B does the following:
  - First, it checks if σ and σ' have been recorded in T. If σ has not been recorded, checks if $\sigma \in \mathcal{R}$ and answers $\perp$ if not. Otherwise, B makes r attempts at assigning σ a constant unused element in $\mathbb{Z}_p$ that is consistent with the simulation so far. In particular, B will repeat the following steps up to r times. It starts by sampling a random unused element $a \leftarrow_\$ \mathbb{Z}_p$. If this is the $\tilde{r}$th attempt, with $\tilde{r} < r$, it queries Eval on $a - P$ for all previously recorded (non-constant) polynomials P in T. If Eval outputs 1 for some P, adversary B tries again with a new random unrecorded $a \leftarrow_\$ \mathbb{Z}_p$. If Eval does not output 1 for any of the queries or if it is the rth time of sampling a, it stores the pair (a, σ) in T, where a is to be interpreted as a constant polynomial. It then does the same if σ' has not been recorded.
  - Let P, P' be the polynomials corresponding to labels σ, σ'. For all previously recorded polynomials P'' in T, it queries $P + P' - P''$ to oracle Eval. If Eval outputs 1 for one of the P'', B looks up the corresponding label σ'' in T and sends σ'' to A. Otherwise, it samples a random σ'' from the unused values in $\mathcal{R}$ and sends σ'' to A_2. Then stores $(P + P', \sigma'')$ in T.
  - Note that, again, all newly stored polynomials are elements of $\mathrm{Span}(1, \vec{F})$ and that, if for all $(\hat{P}, \hat{\sigma}) \in T$ we had that $\mathcal{L}^{-1}(\hat{\sigma}) = \hat{P}(\vec{x})$ up to this point, then the same holds for all elements added to the table during either of these steps.

- When A_2 outputs $(I, (\hat{\sigma}_i)_{i\in I})$, B checks, for every $i \in I$, if $\hat{\sigma}_i$ has previously been recorded in T with corresponding polynomial $\hat{F}_i$. If not, it is treated as in the analogous case of group operation queries described above. Then B outputs $(I, (\hat{F}_i)_{i\in I})$ as its solution. Note that the check of line 14 will succeed, as $\hat{F}_i \in \mathrm{Span}(1, \vec{F})$ for all i.

We now upper-bound the number of queries made by B to Eval. First, note that to handle the bit-fixing, B adds M constant polynomials to T. To set up A_2's input, it adds 1 constant and k arbitrary entries to T, therefore making at most $k(M+1+k)$ queries to Eval. During the execution of A_2, every query to GrpOp adds up to 2 constant and one arbitrary polynomial to T. Thus, at the q'th query, T contains at most $M+1+k+3q'$ entries, of which at most $k+q'$ are not constant. As the check against previously unrecorded labels needs to be done only with respect to non-constant polynomials, the q'th GrpOp-query requires at most $2(r-1)(k+q')+(M+1+k+3q')$ queries to Eval. Finally, to handle A_2's output, B makes up to $(r-1)m$ additional checks against the at most $(k+q)$ non-constant entries in T. Summing up, we can bound the number of queries made by B by

$$\begin{aligned} &k(M+1+k)+q\big(2(r-1)(k+q)+(k+3q+M+1)\big)+m(r-1)(k+q)\\ =&q(M+(2r+1)q)+kM+q\left(rk+1+m(r-1)\right)+k\left(k+1+m(r-1)\right) \quad . \end{aligned}$$

To prove the bound on the advantage we define the event

$$\mathtt{bad} = \{\exists (a,\sigma_a),(P,\sigma_P)\in T \mid a \text{ constant}, P \text{ not constant} : P(\vec{x})-a=0\} \ ,$$

corresponding to the event where B assigned a label σ it did not previously receive a constant not consistent with the simulation. Observe that in the case that bad does not occur, the view of A is a perfect simulation of the MI-Uber game with hidden value $\vec{x}$, and, as stated above, we have $\mathcal{L}^{-1}(\sigma)=P(\vec{x})$ for all (P,σ) stored in T. Thus, in this case, we have that $(\hat{F}_i - F_i^*)(\vec{x})=0 \Leftrightarrow \mathcal{L}^{-1}(\hat{\sigma}_i)=\hat{F}_i(\vec{x})=F_i^*(\vec{x})$ for all $i\in I$ and so B wins exactly if A wins.

Finally, when assigning a constant to a previously unseen σ the probability that B does so inconsistently is at most $((q+k)d_{\max}/p)^r$. Indeed, by the Schwartz-Zippel Lemma, the probability the sampled constant is in the set of roots of any polynomial P is at most $d_{\max}/p$ and, for each of the r attempts at finding a constant, at most $(q+k)$ polynomials have to be checked. As the reduction has to sample at most $(2q+m)$ constant, the bound follows. □

A.2 Proof of Theorem 5

Proof of Theorem 5. The geo-MI-Uber solver B receives as input the number of indeterminates t, input polynomials $\vec{F}_1, \vec{F}_2, \vec{F}_T$, and a total of n target polynomials distributed over vectors $\vec{F}_1^*, \vec{F}_2^*, \vec{F}_T^*$; and has access to the oracles $\mathrm{Iso}_\phi, \mathrm{Eval}, \mathrm{Bil}, \mathrm{Dec}_{W_1}, \ldots, \mathrm{Dec}_{W_s}$. To simulate the view of the MI-Uber solver A, B needs to construct input labels $(\sigma_{1,i})_{i\in[k_1]}, (\sigma_{2,i})_{i\in[k_2]}, (\sigma_{T,i})_{i\in[k_T]}$ and answer queries made by A to the oracles $\mathrm{Iso}_\phi, \mathrm{GrpOp}, \mathrm{Bil}, \mathrm{Dec}_{W_1}, \ldots, \mathrm{Dec}_{W_s}$. To do this in a consistent manner, B samples labels on the fly and maintains three tables T_1, T_2, T_T, that correspond to the three groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$. For each $J\in\{1,2,T\}$, the table T stores all previously recorded labels σ from the range $\mathcal{R}_J$ of the labeling function $\mathcal{L}_J$, each together with a corresponding polynomial $P(\vec{X})$ in the indeterminates $X_1,\ldots,X_t$. The labels σ will correspond to a perfect simulation of MI-Uber such that we have $\mathcal{L}_J^{-1}(\sigma)=g_J^{P(\vec{x})}$ for every entry (P,σ) in T_J. To make sure that the simulation is consistent, B needs to check that no two polynomials P, P' such that $P(\vec{x})=P'(\vec{x})$ get paired with different labels. This would be equivalent to a group element receiving two different labels. Hence, before sampling a new label for a polynomial P, B needs to check that there is no previously recorded polynomial P' in T_J such that $(P-P')(\vec{x})=0$ using the oracle Eval. Note that, if $P, P' \in \mathrm{Span}_J(\vec{F}_1, \vec{F}_2, \vec{F}_T, \phi)$, then so is $P-P'$, and thus the oracle will not return $\perp$. In more detail, B does the following.

- B samples the disjoint ranges $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_T \subseteq \{0,1\}^\ell$ uniformly at random from all subsets of $\{0,1\}^\ell$ of size p.

- To create the input $(\sigma_{1,i})_{i\in[k_1]}, (\sigma_{2,i})_{i\in[k_2]}, (\sigma_{T,i})_{i\in[k_T]}$, B does the following for all $J \in \{1, 2, T\}$:
  - Set $\sigma_{J,0}$ to a random unused value from $\mathcal{R}_J$ and store $(0, \sigma_{J,0})$.
  - To create $\sigma_{J,i}$ for $i \in [k_J], i \neq 0$, query Eval on $F_{J,i} - P'$ for all previously recorded polynomials P' in T_J. If the answer is 1 for some P', set $\sigma_{J,i}$ to the corresponding label of P'. Otherwise, choose a random unused value from $\mathcal{R}_J$ and store $(F_{J,i}, \sigma_{J,i})$. Note that all polynomials stored so far in T_J trivially lie in $\mathrm{Span}_J(\vec{F}_1, \vec{F}_2, \vec{F}_T, \phi)$.

  Then B runs $\mathsf{A}((\sigma_{1,i})_{i\in[k_1]}, (\sigma_{2,i})_{i\in[k_2]}, (\sigma_{T,i})_{i\in[k_T]})$.
- When A makes a query to any of the oracles it has access to, it is allowed to query labels that it has not received before. B handles these cases as follows: Whenever A queries a label σ of an element of the group $\mathbb{G}_J$ that has not been recorded in T_J, B first checks if $\sigma \in \mathcal{R}_J$. If not, B returns $\perp$. Otherwise, B makes r attempts at assigning σ a constant unused element in $\mathbb{Z}_p$ that is consistent with the simulation so far. In particular, B will repeat the following steps up to r times. It starts by sampling a random unused element $a \leftarrow_{\$} \mathbb{Z}_p$. If this is the $\tilde{r}$th attempt, with $\tilde{r} < r$, it queries Eval on $(J, a - P')$ for all previously recorded (non-constant) polynomials P' in T_J. If Eval outputs 1 for some P', adversary B tries again with a new random unrecorded $a \leftarrow_{\$} \mathbb{Z}_p$. If Eval does not output 1 for any of the queries or if it is the rth time of sampling a, it stores the pair (a, σ) in T_J, where a is to be interpreted as a constant polynomial.
- When A makes a query σ to Iso, B does the following:
  - If $\phi = 3$, return $\perp$.
  - Check if σ has been recorded in T_2. If not, do the procedure described above.
  - Let P be the polynomial corresponding to σ in T_2. For all previously recorded polynomials $P' \in T_1$, query (P', P) to Iso. If Iso returns 1 for one of the queries, look up the label σ' corresponding to P' in T_1 and send it to A. Otherwise, sample a random unused label σ' from $\mathcal{R}_1$ and send it to A. Then store (P, σ') in T_1.
- When A makes a query σ to Iso^{-1}, B does the following:
  - If $\phi \in \{2, 3\}$, return $\perp$.
  - Check if σ has been recorded in T_1. If not, do the procedure described above.
  - Let P be the polynomial corresponding to σ in T_1. For all previously recorded polynomials $P' \in T_2$, query (P, P') to Iso. If Iso returns 1 for one of the queries, look up the label σ' corresponding to P' in T_2 and send it to A. Otherwise, sample a random unused label σ' from $\mathcal{R}_2$ and send it to A. Then store (P, σ') in T_2.
- When A makes a query $(J, \sigma, \hat{\sigma})$ to GrpOp, B does the following:
  - Check if σ and $\hat{\sigma}$ have been recorded in T_J. If not, do the procedure described above.
  - Let $P, \hat{P}$ be the polynomials corresponding to labels $\sigma, \hat{\sigma}$. For all previously recorded polynomials P' in T_J, query $(J, P + \hat{P} - P')$ to oracle Eval. If Eval outputs 1 for one of the P', look up the corresponding label σ' in T_J and send σ' to A. Otherwise sample a random σ' from the unused values in $\mathcal{R}_J$ and send σ' to A. Then store $(P + \hat{P}, \sigma')$ in T_J.
  - Note that, again, all newly stored polynomials are elements of $\mathrm{Span}_J(\vec{F}_1, \vec{F}_2, \vec{F}_T, \phi)$ and that, if for all $(P, \sigma) \in T_J$ we had that $\mathcal{L}_J^{-1}(\sigma) = g^{P(\vec{x})}$, then the same holds for all elements added to the table during either of these steps.
- When A makes a query $(\sigma, \hat{\sigma})$ to Bil, B does the following:
  - Check if σ has been recorded in T_1 and $\hat{\sigma}$ has been recorded in T_2. If not, do the procedure described above.

– Let $(P, \hat{P})$ be the polynomials corresponding to the labels $(\sigma, \hat{\sigma})$. For all P' previously recorded in T_T, query $(T, P \cdot \hat{P} - P')$ to Eval. If Eval outputs 1 for one of the queries, look up the corresponding label σ' and send it to A. Otherwise, sample a random unused label σ' from $\mathcal{R}_T$ and send it to A. Then store $(P \cdot \hat{P}, \sigma')$ in T_T.

- When A makes a query $(\tilde{\sigma}_{1,1} \ldots, \tilde{\sigma}_{1,r_1}, \tilde{\sigma}_{2,1} \ldots, \tilde{\sigma}_{2,r_2}, \tilde{\sigma}_{T,1} \ldots, \tilde{\sigma}_{T,r_T})$ to one of the decisional oracles Dec_{W_i}, B does the following:
  - Check if for all $J \in \{1, 2, T\}$ and all $j \in [r_J]$, $\tilde{\sigma}_{J,j}$ has been recorded in T_J. If not, do the procedure described above.
  - Query Dec_{W_i} on $(\tilde{F}_{1,1}, \ldots, \tilde{F}_{1,r_1}, \tilde{F}_{2,1}, \ldots, \tilde{F}_{2,r_2}, \tilde{F}_{T,1}, \ldots, \tilde{F}_{T,r_T})$, where $\tilde{F}_{J,j}$ is the polynomial corresponding to $\tilde{\sigma}_{J,j}$. Send the answer of Dec_{W_i} to A.
- When A outputs the tuple $(I_1, I_2, I_T, (\hat{\sigma}_{1,i})_{i \in I_1}, (\hat{\sigma}_{2,i})_{i \in I_2}, (\hat{\sigma}_{T,i})_{i \in I_T})$, B checks if all labels $\hat{\sigma}_{J,i}$ have previously been recorded in T_J. If not, B does the procedure described above. Then B outputs the tuple $(I_1, I_2, I_T, (\hat{P}_{1,i})_{i \in I_1}, (\hat{P}_{2,i})_{i \in I_2}, (\hat{P}_{T,i})_{i \in I_T})$, where $\hat{P}_{J,i}$ is the polynomial corresponding to $\hat{\sigma}_{J,i}$. Note that the check of line 14 will succeed, as $\hat{P}_{J,i} \in \mathrm{Span}_J(\vec{F}_1, \vec{F}_2, \vec{F}_T, \phi)$ for all $J \in \{1, 2, T\}$ and all $i \in I_J$.

We will now upper-bound the number of queries B made to Eval. Let $k := k_1 + k_2 + k_T$.First, in order to set up A's input, B adds 3 constant and k arbitrary entries to the tables T_1, T_2, T_3 and makes less than $k(k+1)$ queries to Eval. During the execution of A, every query to $\mathrm{Iso}, \mathrm{Iso}^{-1}, \mathrm{GrpOp}, \mathrm{Bil}$ or Dec_{W_i} adds at most $s_{\max}$ constant and one arbitrary polynomial to a table T_J. Thus, at the q'th query, any table T_J contains at most $k + 3 + (s_{\max} + 1)q'$ entries, of which at most $k + q'$ are not constant. As the check against previously unrecorded labels needs to be done only with respect to non-constant polynomials, the q'th query requires at most $(r-1)s_{\max}(k+q') + (k+3+(s_{\max}+1)q')$ queries to Eval. Finally, to handle A's output, B makes up to $(r-1)m$ additional checks against the at most $(k+q)$ non-constant entries in any table T_J. Summing up, we can bound the number of queries made by B by

$$k(k+3) + (r-1)s_{\max}q(k+q) + q(k+3+(s_{\max}+1)q) + (r-1)m(k+q) \ .$$

To show that

$$\begin{aligned} &\mathrm{Adv}^{\text{geo-MI-Uber}(t,(m,n_1,n_2,n_3),\vec{F}_1,\vec{F}_2,\vec{F}_T,\vec{F}_1^*,\vec{F}_2^*\vec{F}_T^*,W_1,\ldots,W_s)}(\mathsf{B}) \\ &\geq \mathrm{Adv}^{\text{MI-Uber}(t,(m,n_1,n_2,n_3),\vec{F}_1,\vec{F}_2,\vec{F}_T,\vec{F}_1^*,\vec{F}_2^*\vec{F}_T^*,W_1,\ldots,W_s)}(\mathsf{A}) \\ &- (qs_{\max} + m)\left(\frac{(q+k)\cdot d_{\max}}{p}\right)^r \ , \end{aligned}$$

we define the event

$$\mathtt{bad} = \{\exists (a, \sigma_{J,a}), (P, \sigma_{J,P}) \in T_J \mid a \text{ constant}, P \text{ not constant} : P(\vec{x}) - a = 0\} \ .$$

This corresponds to the event where B assigned a label σ it did not previously receive to a constant not consistent with the simulation. Observe that in the case that bad does not occur, the view of A is a perfect simulation of the MI-Uber game with hidden value $\vec{x}$, and, as stated above, we have $\mathcal{L}_J^{-1}(\sigma) = g^{P(\vec{x})}$ for all $J \in \{1, 2, T\}$ and all (P, σ) stored in T_J. Thus in this case, for all J and i, we have that $(\hat{P}_{J,i} - F^*_{J,i})(\vec{x}) = 0 \Leftrightarrow \mathcal{L}_J^{-1}(\hat{\sigma}_{J,i}) = g^{\hat{P}_{J,i}(\vec{x})} = g^{F^*_{J,i}(\vec{x})}$ and so B wins exactly if A wins. Finally, when assigning a constant to a previously unseen σ, the probability that B does so inconsistently is at most $((q+k)d_{\max}/p)^r$. Indeed, by the Schwartz-Zippel Lemma, the probability the sampled constant is in the set of roots of any polynomial P is at most $d_{\max}/p$ and, for each of the r attempts at finding a constant, at most $(q+k)$ polynomials have to be checked. As B has to sample at most $(qs_{\max} + m)$ constants, the bound follows. □