# The One-More Discrete Logarithm Assumption in the Generic Group Model

Balthazar Bauer[1], Georg Fuchsbauer[2], and Antoine Plouviez[3]

[1] Université de Paris, France
[2] TU Wien, Austria
[3] Inria, ENS, CNRS, PSL, Paris, France
first.last@{ens.fr,tuwien.ac.at}

**Abstract.** The one more-discrete logarithm assumption (OMDL) underlies the security analysis of identification protocols, blind signature and multi-signature schemes, such as blind Schnorr signatures and the recent MuSig2 multi-signatures. As these schemes produce standard Schnorr signatures, they are compatible with existing systems, e.g. in the context of blockchains. OMDL is moreover assumed for many results on the impossibility of certain security reductions.

Despite its wide use, surprisingly, OMDL is lacking any rigorous analysis; there is not even a proof that it holds in the generic group model (GGM). (We show that a claimed proof is flawed.) In this work we give a formal proof of OMDL in the GGM. We also prove a related assumption, the one-more computational Diffie-Hellman assumption, in the GGM. Our proofs deviate from prior GGM proofs and replace the use of the Schwartz-Zippel Lemma by a new argument.

**Keywords:** One-more discrete logarithm, generic group model, blind signatures, multi-signatures

## 1 Introduction

*Provable security* is the prevailing paradigm in present-day cryptography. To analyze the security of a cryptographic scheme, one first formally defines what it means to break it and then gives a rigorous proof that this is infeasible assuming that certain computational problems are hard.

Classical hardness assumption like RSA and the discrete logarithm assumption in various groups have received much scrutiny over the years, but there are now myriads of less studied assumptions. This has attracted criticism [35, 34], as the value of a security proof is unclear when it is by reduction from an (often newly introduced) assumption that is not well understood. A sanity check that is considered a minimum requirement for assumptions in cyclic groups is a proof in the generic group model (GGM), which guarantees that there are no efficient solvers that work for any group.

In this work we give the first proof that the *one-more discrete logarithm assumption*, a widely used hardness assumption, holds in the GGM. While prior proofs in the GGM have followed a common blueprint, the nature of OMDL differs from that of other assumptions and its proof requires a new approach, which we propose in this paper. We then extend our proof so that it also covers the *one-more Diffie-Hellman assumption.*

GGM. The *generic group model* [43, 55] is an idealized model for the security analysis of hardness assumptions (as well as cryptographic schemes themselves) that are defined over cyclic groups. It models a "generic group" by not giving the adversary any group elements, but instead abstract "handles" (or *encodings*) for them. To compute the group operation, the adversary has access to an oracle which given handles for group elements $X$ and $Y$ returns the handle of the group element $X + Y$ (we denote groups additively).

OMDL. The one-more discrete logarithm problem, introduced by Bellare et al. [6], is an extension of the discrete logarithm (DL) problem. Instead of being given one group element $X$ of which the adversary must compute the discrete logarithm w.r.t. some basis $G$, for OMDL the adversary can ask for arbitrarily many challenges $X_i$, all sampled independently and uniformly from the group. In addition, it has access to an oracle that returns the discrete logarithm of any group element submitted by the adversary. The adversary's goal is to compute the DL of all challenges $X_i$, of which there must be (at least) *one more* than the number of calls made to the DL oracle.

### Applications of OMDL

BLIND SIGNATURES. Blind signature schemes [19] let a user obtain a signature from a signer without the latter learning the message it signed. Their security is formalized by *one-more unforgeability*, which requires that after $q$ signing interactions with the signer, the user should not be able to compute signatures on more than $q$ messages.

The signatures in the *blind Schnorr signature scheme* [20] are standard Schnorr signatures [51], which, in the form of EdDSA [11] are increasingly used in practice and considered for standardization by NIST [46]. They are now used in OpenSSL, OpenSSH, GnuPG and considered to be supported by Bitcoin [58], which will enable drastic scalability improvements due to signature aggregation [15, 41] (see below). Blind Schnorr signatures will moreover enable new privacy-preserving applications such as *blind coin swaps* and *trustless tumbler services* [44].

One-more unforgeability of blind Schnorr signatures was proven by Schnorr and Jakobsson [53, 52] directly in the GGM, also assuming the random-oracle model (ROM) and that the so-called *ROS problem* is hard. While unforgeability of blind Schnorr signatures cannot, even in the ROM, be proved from standard assumptions [24, 49, 2], Fuchsbauer et al. [27] give a proof in the *algebraic group model* (AGM) [26], a model between the standard model and the GGM.

In the AGM, adversaries are assumed to be *algebraic*, meaning that for every group element $Z$ they output, they must know a "representation" $\vec{z} = (z_1, \dots, z_n)$ such that $Z = \sum_{i=1}^{n} z_i X_i$, where $X_1, \dots, X_n$ are the group elements received so far. The authors prove unforgeability of blind Schnorr in the AGM+ROM assuming ROS and OMDL [27].

While there has been evidence [57] that the ROS problem was easier than initially assumed, Benhamouda et al. [10] recently presented a polynomial-time solver for ROS. This leads to forgeries of blind Schnorr signatures when the attacker is allowed to run *concurrent* executions of the signing protocol. To overcome these issues, Fuchsbauer et al. [27] define a new signing protocol and introduce a modified ROS assumption, against which there are no known attacks. Their *Clause blind Schnorr* signature scheme is proven unforgeable in the AGM+ROM assuming hardness of their modified ROS problem and OMDL.

Multi-signatures. *Multi-signature schemes* [32] allow a group of signers, each having individual verification and signing keys, to sign a message on behalf of all of them via a single signature. In recent work, Nick et al. [45] present a (concurrently secure) two-round multi-signature scheme called *MuSig2* (a variant of the *MuSig* scheme [41]), which they prove secure under the OMDL assumption. The resulting signatures are ordinary Schnorr signatures (under an *aggregated* verification key, which is of the same form as a key for Schnorr); they are thus fully compatible with blockchain systems already using Schnorr. This will help ease scalability issues, as a single aggregate signature can replace a set of individual signatures to be stored on the blockchain.

Earlier, Bellare and Neven [7] instantiated another signature primitive called *transitive signatures* [42] assuming OMDL.

Identification schemes. Bellare and Palacio [8] assume OMDL to prove that the Schnorr identification protocol is secure against active and concurrent attacks, and Gennaro et al. [30] use it for a batched version of the scheme. Bellare and Shoup [9] prove that the Schnorr identification scheme verifies *special soundness* under concurrent attack from OMDL. Bellare et al. [5] assume OMDL to prove their ID-based identification protocol secure against impersonation under concurrent attacks.

Negative results. OMDL has also been assumed in numerous proofs of impossibility results. Paillier and Vergnaud [48] prove that unforgeability of Schnorr signatures cannot be proven under the discrete logarithm assumption. Specifically, they show that there is no *algebraic* reduction to DL in the standard model if OMDL holds. Seurin [54] shows that, assuming OMDL, the security bound for Schnorr signatures by Pointcheval and Stern [50] using the *forking lemma* is optimal in the ROM under the DL assumption. More precisely, the paper shows that if the OMDL assumption holds, then any algebraic reduction of Schnorr signatures must lose the same factor as a proof via the forking lemma. Fischlin and Fleischhacker [23] generalize this impossibility result to a large class of reductions they call *single-instance reductions*, again assuming OMDL. There are further negative results on the security of Schnorr signatures that assume OMDL [29, 25, 28].

Finally, Drijvers et al. [22] show under the OMDL assumption that many multi-signature schemes, such as *CoSi* [56], *MuSig* [41], *BCJ* [1] and *MWLD* [37], cannot be proven secure from DL or OMDL.

### The Generic Security of OMDL

Despite its wide use, surprisingly, OMDL is lacking any rigorous analysis, apart from a comparison to DL in certain groups: while clearly the OMDL problem is not harder than DL, it is strictly easier in any group for which the index calculus algorithm is the best way to solve both problems [33, 31]. This does thus not apply to elliptic-curve groups, which typically underlie contemporary instantiations of schemes relying on OMDL.

The only analysis of OMDL in the GGM is a more recent proof sketch by Coretti, Dodis, and Guo [21, eprint version], which we show is flawed.[4] (The authors confirmed this in personal communication.)

Their analysis follows the blueprint of earlier GGM proofs, which goes back to Shoup's [55] proof of the hardness of DL in the GGM. However, as we explain below, the adversary can easily make their simulation of the GGM OMDL game fail. The particularity of OMDL compared to other assumptions, which lend themselves more easily to a GGM proof, is that via its DL oracle, the adversary can obtain information about the secret values chosen by the experiment.

Bauer et al. [3] gave further evidence that the analysis of the generic security of OMDL must differ from that of other assumptions. They show that, in the algebraic group model, a large class of assumptions, captured by an extension of the *uber assumption* framework [14, 16], is implied by the hardness of $q$-DLog. In this problem the adversary is given $(xG, x^2G, \ldots, x^qG)$ and must find $x$. While in the AGM $q$-DLog implies assumptions as diverse as the strong Diffie-Hellman [13], the gap Diffie-Hellman [47], and the LRSW assumption [36], this is not the case for OMDL. Using the meta-reduction technique, Bauer et al. [3] show that it is impossible to prove OMDL from $q$-DLog, for any $q$, in the AGM.

This extends earlier results on $q$-OMDL, a parametrized variant where the adversary receives exactly $q$ challenges. For different values of $q$, these assumptions are not equivalent under black-box reductions [18] or algebraic reductions [17] (a separation under standard white-box reductions appears to be open).

PROOFS IN THE GGM. To explain the challenges in proving OMDL in the GGM, we start by recalling how GGM proofs typically proceed. In the GGM the adversary does not see actual group elements of the form $xG$, with $x \in \mathbb{Z}_p$ and $G$ a fixed generator; instead it gets encodings $\Xi(x)$ of them, where $\Xi$ is a random injective function. As the adversary cannot compute the encoding of

[4] The authors study assumptions (including OMDL) and schemes in an extension of the GGM that models preprocessing attacks. They give a proof sketch for the security of OMDL with preprocessing. While we show that their sketch is flawed (see p. 5), their preprocessing techniques can be adapted to our proof. Their result for OMDL in the preprocessing GGM thus still holds, except for a change of the security bounds.

$(x+y)G$ from encodings of $xG$ and $yG$, it is provided with an oracle that on input $(\Xi(x), \Xi(y))$ returns $\Xi(x+y)$.

When analyzing hardness assumptions in the GGM, instead of choosing secret values in the security game, the challenger represents them by indeterminates. For concreteness, consider the GGM game for the DL assumption: the adversary is given the challenge $\Xi(x)$ and must compute the discrete logarithm $x \in \mathbb{Z}_p$. In the proof, the challenger *simulates* this game by using the variable $\mathsf{X}$ instead of $x$ and encodes the *polynomial* $\mathsf{X}$ instead of $x$. That is, the challenger gives $\Xi(\mathsf{X})$ to the adversary, who is oblivious to this change. If then the adversary asks, for example, for the addition of $\Xi(1)$ and $\Xi(\mathsf{X})$, the challenger replies $\Xi(\mathsf{X}+1)$, that is, the encoding of a polynomial of degree 1.

This allows the challenger to simulate the game without actually defining a challenge. *After* the adversary output its answer, the challenger picks a value $x$ uniformly at random, which the adversary can only guess with negligible probability. This shows generic hardness of the DL problem.

There is however a caveat: $\Xi(\mathsf{X})$ represents $\Xi(x)$, and, more generally, for any polynomial $P$ that the adversary constructed via its queries, $\Xi(P)$ represents $\Xi(P(x))$. So the simulation would be inconsistent if for some polynomials $P \neq Q$ computed by the adversary we had $P(x) = Q(x)$. Indeed, if such a *collision* occurs, then the simulated game gives the adversary $\Xi(P) \neq \Xi(Q)$ instead of $\Xi(P(x)) = \Xi(Q(x))$.

In order to bound the probability that the simulation fails due to such collisions, the standard technique is to apply the Schwartz-Zippel Lemma, which states that for a non-zero degree-$d$ (multivariate) polynomial $P \in \mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_n]$ the probability that $P(x_1, \ldots, x_n) = 0$ for a uniformly chosen $\vec{x} \xleftarrow{\$} \mathbb{Z}_p^n$ is $\frac{d}{p}$.

Since $x$ is picked uniformly *after* the adversary has defined the polynomials $P$ and $Q$, the probability that $P(x) - Q(x) = 0$ is bounded by $\frac{1}{p}$. Applying this to all pairs of polynomials generated by the adversary via its group-operation oracle during the game then yields the final bound. This was precisely how Shoup [55] proved the security of DL in the GGM and it was followed by many subsequent GGM proofs. The technique easily extends to games where there are several secrets $x_1, \ldots, x_n$.

<u>Challenges in the GGM proof of OMDL.</u> We follow Shoup [55] in that we replace all challenges $x_i$ in the OMDL game by corresponding polynomials $\mathsf{X}_i \in \mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_n]$. It seems tempting to then deduce, like for DL, that the probability that $P(x_1, \ldots, x_n) = Q(x_1, \ldots, x_n)$ for any $P \neq Q$ generated during the game is at most $\frac{1}{p}$ by Schwartz-Zippel. (This is what Coretti et al. [21] do in their proof sketch.) This argument however ignores the fact that, via the discrete logarithm oracle $\textsc{DLog}(\cdot)$, the adversary can obtain (a lot of) information on the challenges $x_i$ and can thereby easily cause collisions. In more detail, such a straightforward proof has the following issues:

First, in the game simulated via polynomials, the adversary's oracle $\textsc{DLog}(\cdot)$ must be simulated carefully. For example, suppose the adversary asks for the discrete logarithm of the first challenge by querying $\textsc{DLog}(\Xi(\mathsf{X}_1))$. Since $x_1$ is not defined yet, the challenger samples it uniformly and gives it to the adversary. Now if the adversary later asks for $\Xi(\mathsf{X}_1 + 1)$ (via its group-operation oracle) and queries DLog on it, it expects the answer $x_1 + 1$. (In [21], the DLog oracle always returns random values; the adversary can thus easily detect that it is not playing the OMDL game in the GGM.)

Second, there is a more subtle issue. Again suppose that the adversary queried DLog($\Xi(\mathsf{X}_1)$) and was given $x_1$. Let $P := \mathsf{X}_1$. Using the group-operation oracle, the adversary can compute (an encoding of) the constant polynomial $Q := x_1$, that is, it can obtain $\Xi(Q)$. Since $P(x_1) = Q(x_1) = x_1$, this means that the adversary can in fact construct polynomials $P$ and $Q$ such that $P(x_1, \ldots, x_n) = Q(x_1, \ldots, x_n)$ and $P \neq Q$.

Note that this situation cannot occur in prior GGM proofs for other assumptions, because as long as there is no simulation failure, the adversary's polynomials are *independent* of $\vec{x}$, which is a prerequisite for applying Schwartz-Zippel (SZ) in the end. This standard use of SZ (followed by [21]) is thus not possible for OMDL, as the adversary can, via its DLog oracle, obtain information on the challenge $(x_1, \ldots, x_n)$ even when there is no simulation failure.

All these issues persist if instead of Shoup's GGM model [55], one uses Maurer's model [39], which is an abstraction of the former. In his model, all (logarithms of) group elements remain in a "black box", and the adversary can ask for the creation of new entries in the box that are either the sum of existing entries or values of its choice. To capture the DLog oracle in OMDL one would have to extend the model and allow the adversary to ask for values from the box to be revealed. Moreover, in proofs in this model [39] the adversary wins as soon as it creates a collision between values in the box, which is why one can assume *non-adaptive* adversaries. However, an OMDL adversary *is* adaptive and can easily create collisions (e.g., get $x_1$ from the DLog oracle, then insert the constant $x_1$ into the black box). A new approach would thus be required.

Our GGM proof of OMDL. In our proof we simulate the OMDL game in the GGM using polynomials, but we take into account all the issues just described. That is, the challenger monitors what the adversary has learned about the challenge and defines the simulation considering this knowledge, thus preventing the adversary from trivially distinguishing the real game from the simulation.

Still, there might be simulation failures due to "bad luck", which corresponds precisely to the event whose probability previous proofs bound via Schwartz-Zippel. As OMDL requires a different approach, we propose a new lemma that bounds the probability that our simulation of the OMDL game fails. After modifying the game by aborting in case of a simulation failure, we give a formally defined sequence of game hops showing that the game is *equivalent* to a game that the adversary cannot win. Given the pitfalls in previous approaches and the intricacies outlined so far (and the importance of OMDL), we believe that such a rigorous approach is justified for OMDL.

Our first step is comparable to how Yun [59] analyzed the generic security of the *multiple discrete logarithm* assumption, where the adversary must solve multiple DL challenges (but is not given a DLog oracle, which is what makes

OMDL so different from other assumptions). Like Yun, we formalize the knowledge about the challenge that the adversary accumulates by affine hyperplanes in $\mathbb{Z}_p^n$.

<u>POSSIBLE ALTERNATIVE APPROACHES.</u> One might wonder if it was possible to nonetheless rely on the Schwartz-Zippel lemma (SZ) for proving OMDL. We have already argued that applying it once and at the end of the game, as in previous proofs, is not possible. But can SZ be used earlier in the game?

A first idea could be to apply SZ at each DLOG call. Consider a call $\mathrm{DLog}(\Xi(\mathsf{X}_1 + \mathsf{X}_2))$, answered with a uniform $v \leftarrow \mathbb{Z}_p$. One could now formally replace the indeterminate $\mathsf{X}_1$ by the expression $\mathsf{X}_2 - v$ in all polynomials $P$ generated so far and use SZ to bound the probability that this creates a collision. A first issue is that since $P$ is a multivariate polynomial, SZ does not directly imply a bound on $\Pr[P(\mathsf{X}_2 - v, \mathsf{X}_2, \ldots, \mathsf{X}_n) = 0]$. Indeed, $P(\mathsf{X}_2 - v, \mathsf{X}_2, \ldots, \mathsf{X}_n)$ is the evaluation of the polynomial $\hat{P}(\mathsf{X}_1) \coloneqq P(\mathsf{X}_1, \mathsf{X}_2, \ldots, \mathsf{X}_n)$ for $\mathsf{X}_1 = \mathsf{X}_2 - v$, so we need to bound $\Pr[\hat{P}(\mathsf{X}_2 - v) = 0]$ for a polynomial $\hat{P}$ with coefficients in the ring $\mathbb{Z}_p[\mathsf{X}_2, \ldots, \mathsf{X}_n]$, whereas SZ is defined for polynomials over fields.

Moreover, when the query $\mathrm{DLog}(\Xi(P(\mathsf{X}_1, \ldots, \mathsf{X}_n))$ involves a more complex polynomial than $P = \mathsf{X}_1 + X_2$ then the substitution of one variable by a linear expression of the others is even cumbersome to describe notationally. We avoid these problems in our proof by using our lemma instead of (a variant of) SZ, which also lets us keep notation simple.

Another idea would be to apply SZ each time a new encoding is computed. Indeed, assuming no collisions have occurred so far, one could use SZ to bound the probability that the new encoding introduces a collision and then proceed by induction. But the resulting proof would require one game hop for every newly computed encoding: In the $j$-th hybrid of this game the first $j$ encodings are chosen all different independently of the real value of the challenge; the challenge $\vec{x}$ is picked by the game just before the $(j+1)$-th encoding, when $P_{j+1}$ is defined. Using SZ, we can show that the probability that $P_{j+1}(\vec{x}) = P_i(\vec{x})$ for all $i \leq j$ is negligible.

However, we need to be more cautious. To prevent the attack in which the adversary queries $\mathrm{DLog}(\Xi(\mathsf{X}_1))$, obtains $x_1$ and then generates the constant polynomial $P_{j+1} = x_1$, we need to adapt all polynomials defined so far to reflect the information revealed by $\mathrm{DLog}(\cdot)$. In this example, this is easy to formalize: update every polynomial by evaluating $X_1$ on $x_1$ and replace $P_k(x_1, \mathsf{X}_2, \ldots, \mathsf{X}_n)$ by some $P'_k(\mathsf{X}_2, \ldots, \mathsf{X}_n)$; the updated challenge $\vec{x}$ would be of size $n-1$. To generalize this, we would have to apply an affine transformation to all variables of the polynomials at each call to $\mathrm{DLog}(\cdot)$. After as many game hops as there are queries by the adversary, we would arrive at a game in which all encodings are random and the challenge is defined after the adversary output its solution.

We believe that both approaches just sketched lead to more complicated (and error-prone) proofs than the one we propose. In our proof, in the first game hop we abort if our simulation fails and we bound this probability by our new lemma. The remaining 3 game hops are purely syntactical and do not change the adversary's winning probability.

### One-More CDH

Another "one-more" assumption is the *one-more computational Diffie-Hellman* assumption [5], also known as 1-*MDHP* [33, 34], which is similar to the *chosen-target CDH* assumption [12]. Here, the adversary receives $q$ pairs of group elements $(X, Y_i)$, all with the same first component $X = xG$, and its task is to compute $xY_i$ for all $i$. It is given an oracle $\mathrm{CDH}_1(\cdot)$, which on input $Y$ returns $xY$, and which it can query fewer than $q$ times.

It turns out that this assumption can be proved to hold in the generic group model using standard techniques. Following the original GGM proof of DL [55], we modify the simulation for the adversary from encoding logarithms to encoding polynomials in $\mathbb{Z}_p[\mathsf{X}, \mathsf{Y}_1, \ldots, \mathsf{Y}_n]$. The challenges that the adversary receives are the monomials $\mathsf{X}, \mathsf{Y}_1, \ldots, \mathsf{Y}_n$, and when the adversary queries its oracle $\mathrm{CDH}_1(\cdot)$ on an encoding of a polynomial $P$, it receives an encoding of $\mathsf{X}P$, i.e., its polynomial multiplied by the indeterminate $\mathsf{X}$. To win this "ideal" game, the adversary must construct encodings of $(\mathsf{XY}_1, \ldots, \mathsf{XY}_n)$. Making $q$ calls to its $\mathrm{CDH}_1(\cdot)$ oracle and using its group-operation oracle, it can only construct (encodings of) polynomials from $\mathsf{Span}(1, \mathsf{X}, \mathsf{Y}_1, \ldots, \mathsf{Y}_n, \mathsf{X}P_1, \ldots, \mathsf{X}P_q)$.

Ignoring polynomials of degree less than 2, the adversary wins the game if $\mathsf{Span}(\mathsf{XY}_1, \ldots, \mathsf{XY}_n) \subseteq \mathsf{Span}(\mathsf{X}P_1, \ldots, \mathsf{X}P_q)$. But it must also solve more challenges than it makes $\mathrm{CDH}_1(\cdot)$ oracle queries; that is $q < n$. Using a dimension argument, we deduce that the above condition cannot be satisfied, and thus the adversary cannot win this game.

This "ideal" game is indistinguishable from the real game if the adversary does not create two distinct polynomials that agree on $x, y_1, \ldots, y_n$, the secret values of the real game. Because the degree of all polynomials is upper-bounded by $q+1$, we can use the Schwartz-Zippel Lemma (as, e.g., done in [16]) to upper-bound the statistical distance between the two games by $\mathcal{O}\left(\frac{(q+1)(m+q)^2}{p}\right)$, where $m$ is the number of group operations made by the adversary. This establishes the generic security of this assumption. (An alternative is to cast the assumption as an uber-assumption in the algebraic group model and apply [3, Theorem 4.1].)

The situation is quite different for a variant of the above problem, in which the first component of the challenge pairs is not fixed. That is, the adversary can request challenges, which are random pairs $(X_i, Y_i)$ and is provided with an oracle $\mathrm{CDH}(\cdot)$, which on input any pair $(X = xG, Y)$ returns the CDH solution of $X$ and $Y$, that is $xY$. The adversary must compute the CDH solutions of the challenge pairs while making fewer queries to $\mathrm{CDH}(\cdot)$. In this paper we will refer to this assumption as OMCDH.

For this problem the standard proof methodology in the GGM fails for the following reason. Providing the adversary with an oracle $\mathrm{CDH}_1(\cdot)$, as in the one-more Diffie-Hellman assumption with one component fixed (or a DLog oracle in OMDL) lets the adversary only construct polynomials of degree at most $q+1$. In contrast, the $\mathrm{CDH}(\cdot)$ oracle in OMCDH leads to a multiplication of the degrees, which enables the adversary to "explode" the degrees and makes arguments à la Schwartz-Zippel impossible, since they rely on low-degree polynomials.

To get around this problem, we prove the following, *stronger* assumption: as in OMCDH, the adversary still has to compute CDH solutions, but now it is given a discrete-logarithm oracle. This hybrid assumption implies both OMDL (for which the goal is harder) and OMCDH (in which the oracle is less powerful) and we prove it in the GGM by extending our proof of OMDL.

## 2 Preliminaries

GENERAL NOTATION. We denote the (closed) integer interval from $a$ to $b$ by $[a, b]$. A function $\mu \colon \mathbb{N} \to [0, 1]$ is *negligible* (denoted $\mu = \mathsf{negl}$) if for all $c \in \mathbb{N}$ there exists $\lambda_c \in \mathbb{N}$ such that $\mu(\lambda) \leq \lambda^{-c}$ for all $\lambda \geq \lambda_c$. A function $\nu$ is *overwhelming* if $1 - \nu = \mathsf{negl}$. Given a non-empty finite set $S$, we let $x \xleftarrow{\$} S$ denote sampling an element $x$ from $S$ uniformly at random. A list $\vec{z} = (z_1, \ldots, z_n)$, also denoted $(z_i)_{i \in [n]}$, is a finite sequence. The length of a list $\vec{z}$ is denoted $|\vec{z}|$. The empty list is denoted $(\,)$.

All algorithms are probabilistic unless stated otherwise. By $y \leftarrow \mathcal{A}(x_1, \ldots, x_n)$ we denote running algorithm $\mathcal{A}$ on inputs $(x_1, \ldots, x_n)$ and uniformly random coins and letting $y$ denote the output. If $\mathcal{A}$ has oracle access to some algorithm ORACLE, we write $y \leftarrow \mathcal{A}^{\textsc{Oracle}}(x_1, \ldots, x_n)$. A *security game* $\mathrm{GAME}_{par}(\lambda)$ indexed by a set of parameters *par* consists of a main procedure and a collection of oracle procedures. The main procedure, on input the security parameter $\lambda$, generates input on which an adversary $\mathcal{A}$ is run. The adversary interacts with the game by calling oracles provided by the game and returns some output, based on which the game computes its own output bit $b$, which we write $b \leftarrow \mathrm{GAME}^{\mathcal{A}}_{par}(\lambda)$. We identify **false** with 0 and **true** with 1. As all games in this paper are computational, we define the *advantage* of $\mathcal{A}$ in $\mathrm{GAME}_{par}(\lambda)$ as $\mathbf{Adv}^{\mathrm{GAME}}_{par,\mathcal{A}} := \Pr[1 \leftarrow \mathrm{GAME}^{\mathcal{A}}_{par}(\lambda)]$. We say that $\mathrm{GAME}_{par}$ is *hard* if $\mathbf{Adv}^{\mathrm{GAME}}_{par,\mathcal{A}} = \mathsf{negl}$ for any probabilistic polynomial-time (p.p.t.) adversary $\mathcal{A}$.

ALGEBRAIC NOTATION. A *group description* is a tuple $\mathit{\Gamma} = (p, \mathbb{G}, G)$ where $p$ is an odd prime, $\mathbb{G}$ is an abelian group of order $p$, and $G$ is a generator of $\mathbb{G}$. We use additive notation for the group law and denote group elements with uppercase letters. We assume the existence of a p.p.t. algorithm $\mathsf{GrGen}$ which, on input the security parameter $1^\lambda$ in unary, outputs a group description $\mathit{\Gamma} = (p, \mathbb{G}, G)$ where $p$ is of bit-length $\lambda$. For $X \in \mathbb{G}$, we let $\log_G(X)$ denote the discrete logarithm of $X$ with respect to the generator $G$, i.e., the unique $x \in \mathbb{Z}_p$ such that $X = xG$.

For multivariate polynomials $P \in \mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_n]$ we write $\vec{\mathsf{X}} := (\mathsf{X}_1, \ldots, \mathsf{X}_n)$ and $P(\vec{x}) := P(x_1, \ldots, x_n)$ for $\vec{x} \in \mathbb{Z}_p^n$. We consider subspaces of $\mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_n]$: for a set $L = \{P_1, \ldots, P_q\}$ of polynomials, $\mathsf{Span}(L) := \left\{ \sum_{i \in [1,q]} \alpha_i P_i \mid \vec{\alpha} \in \mathbb{Z}_p^q \right\}$ is the smallest vector space containing the elements of $L$. If $L = \emptyset$ then $\mathsf{Span}(L) = \{0\}$. By $\mathsf{dim}(A)$ we denote the dimension of vector spaces or affine spaces.

By $\langle \vec{x}, \vec{y} \rangle = \sum_{i \in [1,n]} x_i y_i$ we denote the scalar product of vectors $\vec{x}$ and $\vec{y}$ of length $n$. In this work, polynomials are typically of degree 1, so we can write $P = \rho_0 + \sum_{i=1}^{n} \rho_i \mathsf{X}_i$ as a scalar product: $P(\vec{\mathsf{X}}) = \rho_0 + \langle \vec{P}, \vec{\mathsf{X}} \rangle$, where we define $\vec{P} := (\rho_i)_{i \in [1,n]}$, that is the vector of *non-constant* coefficients of $P$.

**Game $\mathrm{DL}^{\mathcal{A}}_{\mathsf{GrGen}}(\lambda)$**

$(p, \mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
$x \xleftarrow{\$} \mathbb{Z}_p\,;\;\; X := xG$
$y \leftarrow \mathcal{A}(p, \mathbb{G}, G, X)$
**return** $(y = x)$

**Game $\mathrm{OMDL}^{\mathcal{A}}_{\mathsf{GrGen}}(\lambda)$**

$(p, \mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
$\vec{x} := (\,)\,;\;\; q := 0$
$\vec{y} \leftarrow \mathcal{A}^{\textsc{Chal},\textsc{DLog}}(p, \mathbb{G}, G)$
**return** $\big(\vec{y} = \vec{x} \;\wedge\; q < |\vec{x}|\big)$

**Oracle $\textsc{Chal}()$**

$x \xleftarrow{\$} \mathbb{Z}_p\,;\;\; X := xG$
$\vec{x} := \vec{x} \,\|\, (x)$
**return** $X$

**Oracle $\textsc{DLog}(X)$**

$q := q + 1\,;\;\; x := \log_G(X)$
**return** $x$

**Fig. 1.** The DL and the OMDL problem

**Game $\mathrm{CDH}^{\mathcal{A}}_{\mathsf{GrGen}}(\lambda)$**

$(p, \mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
$x, y \xleftarrow{\$} \mathbb{Z}_p$
$X := xG\,;\;\; Y := yG$
$V \leftarrow \mathcal{A}(p, \mathbb{G}, G, X, Y)$
**return** $(V = xyG)$

**Game $\mathrm{OMCDH}^{\mathcal{A}}_{\mathsf{GrGen}}(\lambda)$**

$(p, \mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
$\overrightarrow{Z} := (\,)\,;\;\; q := 0$
$\overrightarrow{V} \leftarrow \mathcal{A}^{\textsc{Chal},\mathrm{CDH}}(p, \mathbb{G}, G)$
**return** $\big(\overrightarrow{Z} = \overrightarrow{V} \;\wedge\; q < |\overrightarrow{Z}|\big)$

**Oracle $\textsc{Chal}()$**

$x, y \xleftarrow{\$} \mathbb{Z}_p$
$(X, Y) := (xG, yG)$
$\overrightarrow{Z} := \overrightarrow{Z} \,\|\, (xyG)$
**return** $(X, Y)$

**Oracle $\mathrm{CDH}(X, Y)$**

$q := q + 1$
$x := \log_G(X)\,;\;\; y := \log_G(Y)$
**return** $xyG$

**Fig. 2.** The CDH and the OMCDH problem

<u>DISCRETE LOGARITHM AND DIFFIE-HELLMAN PROBLEMS.</u> In Figures 1 and 2 we recall the discrete logarithm (DL) problem and the computational Diffie-Hellman (CDH) problem and define the one-more discrete logarithm (OMDL) problem and the one-more computational Diffie-Hellman (OMCDH) problem.

## 3 OMDL in the GGM

### 3.1 A Technical Lemma

While a standard argument in GGM proofs uses the Schwartz-Zippel lemma, it does not work for OMDL, where the adversary obtains information on the challenge $\vec{x}$ *not* only when the simulation fails. So we cannot argue that $\vec{x}$ looks uniformly random to the adversary, which is a precondition for applying Schwartz-Zippel. We therefore use a different lemma, which bounds the probability that for a given polynomial $P$, we have $P(\vec{x}) = 0$ when $\vec{x}$ is chosen uniformly from a set $\mathcal{C}$. This set $\mathcal{C} \subseteq \mathbb{Z}_p^n$ represents the knowledge the adversary has gained on the challenge $\vec{x}$ during the OMDL game.

The Schwartz-Zippel lemma applies when $\mathcal{C} = S^n$ with $S$ a subset of $\mathbb{Z}_p$, whereas our lemma is for the case that $P$ has degree 1 and $\mathcal{C}$ is defined by an intersection of affine hyperplanes $\mathcal{Q}_j$ from which we remove other affine hyperplanes $\mathcal{D}_i$, that is $\mathcal{C} \coloneqq \big(\bigcap_{j\in[1,q]} \mathcal{Q}_j\big) \setminus \big(\bigcup_{i\in[1,m]} \mathcal{D}_i\big)$.

We start with some notations. Consider $m$ polynomials $D_i \in \mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_n]$ of degree 1, and $q+1$ polynomials $Q_j \in \mathbb{Z}_p[X_1, \ldots, \mathsf{X}_n]$ also of degree 1. We can write them as

$$D_i(\vec{\mathsf{X}}) = D_{i,0} + \sum_{k=1}^{n} D_{i,k}\mathsf{X}_k = D_{i,0} + \langle \vec{D}_i, \vec{\mathsf{X}} \rangle \tag{1}$$

with $\vec{D}_i \coloneqq (D_{i,k})_{1\le k\le n}$, and similarly for $Q_j$. We define the sets of roots of these polynomials, which are hyperplanes of $\mathbb{Z}_p^n$:

$$\begin{aligned} \forall i \in [1,m] :\ \mathcal{D}_i &\coloneqq \{\vec{x} \in \mathbb{Z}_p^n \mid D_i(\vec{x}) = 0\} \\ \forall j \in [1,q+1] :\ \mathcal{Q}_j &\coloneqq \{\vec{x} \in \mathbb{Z}_p^n \mid Q_j(\vec{x}) = 0\}\ . \end{aligned} \tag{2}$$

From (1), we see that the vector $\vec{D}_i$ of non-constant coefficients defines the direction of the hyperplane $\mathcal{D}_i$. It contains the coefficients of the polynomial $D_i - D_i(0) = \sum_{k=1}^{n} D_{i,k}\mathsf{X}_k$.

We define the set

$$\mathcal{C} \coloneqq \Big(\bigcap_{j\in[1,q]} \mathcal{Q}_j\Big) \setminus \Big(\bigcup_{i\in[1,m]} \mathcal{D}_i\Big)\ , \tag{3}$$

that is, the set of points at which all $Q_i$'s vanish but none of the $D_i$'s do. The following lemma will be the heart of our proofs of one-more assumptions in the GGM.

**Lemma 1.** *Let $D_1, \ldots, D_m, Q_1, \ldots, Q_{q+1} \in \mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_n]$ be of degree 1; let $\mathcal{C}$ be as defined in (2) and (3). Assume $\mathcal{Q}_{q+1} \cap \mathcal{C} \neq \emptyset$ and $\vec{Q}_{q+1}$ is linearly independent of $(\vec{Q}_j)_{j\in[1,q]}$. If $\vec{x}$ is picked uniformly at random from $\mathcal{C}$ then*

$$\frac{p-m}{p^2} \le\ \Pr\big[Q_{q+1}(\vec{x}) = 0\big]\ \le \frac{1}{p-m}\ .$$

*Proof.* Since $\vec{x}$ is picked uniformly in $\mathcal{C}$, we have $\Pr[\,\vec{x} \in \mathcal{Q}_{q+1}] = |\mathcal{Q}_{q+1} \cap \mathcal{C}|/|\mathcal{C}|$.

We first bound $|\mathcal{C}|$. We define $\mathcal{Q} \coloneqq \bigcap_{j\in[1,q]} \mathcal{Q}_j$, which is thus an affine space, and let $d \coloneqq \mathsf{dim}(\mathcal{Q})$ denote its dimension. Thus, $\mathcal{Q}$ contains $p^d$ elements. We rewrite $\mathcal{C}$:

$$\mathcal{C} = \mathcal{Q} \setminus \Big(\bigcup_{i\in[1,m]} (\mathcal{D}_i \cap \mathcal{Q})\Big)\ .$$

Now for a fixed $i \in [1, m]$ we bound the size of $\mathcal{D}_i \cap \mathcal{Q}$. Since the polynomial $D_i$ has degree 1 by definition, $\mathcal{D}_i$ is a hyperplane. There are three cases: either $\mathcal{Q} \subseteq \mathcal{D}_i$, which means $\mathcal{C} = \emptyset$. This contradicts the premise of the lemma, namely $\mathcal{Q}_{q+1} \cap \mathcal{C} \neq \emptyset$. Since $\mathcal{D}_i$ is an hyperplane, the remaining cases are $\mathcal{Q} \cap \mathcal{D}_i = \emptyset$ and $\mathcal{Q} \cap \mathcal{D}_i$ has dimension $\mathsf{dim}(\mathcal{Q}) - 1 = d - 1$. In both cases $\mathcal{D}_i \cap \mathcal{Q}$ contains at most $p^{d-1}$ elements.

When we remove the sets $(\mathcal{D}_i)_{i \in [1,m]}$ from $\mathcal{Q}$, we remove at most $mp^{d-1}$ elements, which yields

$$p^d - mp^{d-1} \leq |\mathcal{C}| \leq p^d \ . \tag{4}$$

We now use the same method to bound $|\mathcal{C} \cap \mathcal{Q}_{q+1}|$. We define $\mathcal{Q}' = \mathcal{Q}_{q+1} \cap \mathcal{Q}$. Since $\vec{Q}_{q+1}$ is linearly independent of $(\vec{Q}_j)_{j \in [1,m]}$, we get $\mathsf{dim}(\mathcal{Q}') = d - 1$.

For a fixed $i \in [1, m]$, since by assumption $\mathcal{Q}_{q+1} \cap \mathcal{C} \neq \emptyset$, we can proceed as with $\mathcal{Q}$ above: either $\mathcal{Q}' \cap \mathcal{D}_i = \emptyset$ or $\mathcal{Q}' \cap \mathcal{D}_i$ has dimension $d - 2$, which yields

$$p^{d-1} - mp^{d-2} \leq |\mathcal{Q}_{q+1} \cap \mathcal{C}| \leq p^{d-1} \ . \tag{5}$$

Combining equations (4) and (5) we obtain the following, which concludes the proof:

$$\frac{p^{d-1} - mp^{d-2}}{p^d} \leq \frac{|\mathcal{Q}_{q+1} \cap \mathcal{C}|}{|\mathcal{C}|} \leq \frac{p^{d-1}}{p^d - mp^{d-1}} \ . \qquad \square$$

### 3.2 Proof Overview

<u>The generic game.</u> We prove a lower bound on the computational complexity of the OMDL game in generic groups in the sense of Shoup [55]. We follow the notation developed by Boneh and Boyen [13] for this proof.

In the generic group model, elements of $\mathbb{G}$ are encoded as arbitrary unique strings, so that no property other than equality can be directly tested by the adversary. The adversary performs operations on group elements by interacting with an oracle called GCmp.

To represent and simulate the working of the oracles, we model the opaque encoding of the elements of $\mathbb{G}$ using an injective function $\mathit{\Xi} \colon \mathbb{Z}_p \to \{0,1\}^{\lceil \log_2(p) \rceil}$ where $p$ is the group order. Internally, the simulator represents the elements of $\mathbb{G}$ by their discrete logarithms relative to a fixed generator $G$. This is captured by $\mathit{\Xi}$, which maps any integer $a$ to the string $\xi \coloneqq \mathit{\Xi}(a)$ representing $a \cdot G$. In the game we will use an encoding procedure Enc to implement $\mathit{\Xi}$.

We specify the game OMDL in the GGM in Fig. 3. In contrast to Fig. 1 there are no more group elements. The game instead maintains discrete logarithms $a \in \mathbb{Z}_p$ and gives the adversary their encodings $\mathit{\Xi}(a)$, which are computed by the procedure Enc. The challenger uses the variable $j$ to represent the number of created group elements, which is incremented before each call to Enc. The procedure Enc then encodes the latest scalar $a_j$. If $a_j$ has already been assigned a string $\xi$, then Enc() outputs $\xi$, else it outputs a random string different from all previous ones. For this, the game maintains a list $(a_i, \xi_i)_{0 \leq i \leq j}$ of logarithms and their corresponding encodings.

Game $\mathrm{OMDLGGM}^{\mathcal{A}}_{\mathsf{GrGen}}(\lambda)$

$\vec{x} \coloneqq (\,)\,;\ a_0 \coloneqq 1$

$j \coloneqq 0\,;\ q \coloneqq 0\,;\ n \coloneqq 0$

$\vec{y} \leftarrow \mathcal{A}^{\text{CHAL},\text{DLOG},\text{GCMP}}(\text{ENC}())$

**return** $\big(\vec{y} = \vec{x}\ \wedge\ q < n\big)$

Oracle CHAL()

$n \coloneqq n + 1$

$x_n \xleftarrow{\$} \mathbb{Z}_p$

$j \coloneqq j + 1$

$a_j \coloneqq x_n$

**return** ENC()

Oracle DLOG($\xi$)

**if** $\xi \notin \{\xi_i\}_{i \in [0,j]}$

  **then return** $\perp$

$q \coloneqq q + 1$

$i \coloneqq \min\{k \in [0,j] \mid \xi = \xi_k\}$

**return** $a_i$

ENC()

**if** $\exists\, i \in [0, j-1] : a_j = a_i$

  **then** $\xi_j \coloneqq \xi_i$

**else**

  $\xi_j \xleftarrow{\$} \{0,1\}^{\log(p)} \setminus \{\xi_i\}_{i \in [0,j-1]}$

**return** $\xi_j$

Oracle GCMP($\xi, \xi', b$)

**if** $\xi \notin \{\xi_i\}_{i \in [0,j]}$ **or** $\xi' \notin \{\xi_i\}_{i \in [0,j]}$

  **then return** $\perp$

$i \coloneqq \min\{k \in [0,j] \,|\, \xi = \xi_k\}$

$i' \coloneqq \min\{k \in [0,j] \,|\, \xi' = \xi_k\}$

$j \coloneqq j + 1\,;\ a_j \coloneqq a_i + (-1)^b a_{i'}$

**return** ENC()

**Fig. 3.** The OMDL game in the GGM

OMDLGGM initializes $j = 0$ and $a_0 = 1$, and runs the adversary on input $\xi_0 \leftarrow$ ENC() ($\xi_0$ is thus the encoding of the group generator). The oracle CHAL increments a counter of challenges $n$, samples a new value $x_n$ and returns its encoding by calling ENC(). Since it creates a new element, it first increments $j$ and defines the $a_j \coloneqq x_n$. The oracle DLOG is called with a string $\xi$ and returns $\perp$ if the string is not in the set of assigned strings $\{\xi_i\}_{i\in[0,j]}$. Else, it picks an index $i$ (concretely: the smallest such index) such that $\xi_i = \xi$ and returns $a_i$, which is the $\Xi$-preimage of $\xi$ (and thus the logarithm of the group element encoded by $\xi$).

The adversary also has access to the oracle GCMP for group operations, which takes as input two strings $\xi$ and $\xi'$ and a bit $b$, which indicates whether to compute the addition or the subtraction of the group elements. The oracle gets the (smallest) indexes $i$ and $i'$ such that $\xi = \xi_i$ and $\xi' = \xi_{i'}$, it increments $j$, sets $a_j \coloneqq a_i + (-1)^b a_{i'}$ and returns ENC(), which computes the encoding of $a_j$.

PROOF OVERVIEW. The goal of our proof is to simulate the game without ever computing scalars $a_i$ by replacing them with polynomials $P_i$ and show that with overwhelming probability this does not affect the game. $\mathsf{Game}_0$ (defined by ignoring all the boxes, except the dashed ones, in Fig. 4) is the same game as OMDLGGM, except for two syntactical changes, which will be useful in the proof. The main modification is that we now make $n$ calls to the oracle DLOG after $\mathcal{A}$ outputs its answer $\vec{y}$: for $i \in [1, n]$ we set $x_i \coloneqq \text{DLOG}(\xi_{j_i})$, where indices $j_i$ are defined in the oracle CHAL so that $a_{j_i} = x_i$; thus DLOG($\xi_{j_i}$) always outputs

$a_{j_i} = x_i$, meaning this does not affect the game. Second, as calls to DLog increase $q$, we put the condition "**if** $q < n$ **then return** 0" before those calls.

Introducing polynomials. $\mathsf{Game}_1$, defined in Fig. 4 by only ignoring the gray boxes, introduces the polynomials $P_i$, where $P_0 = 1$ represents $a_0 = 1$. In the $n$-th call to Chal, the game defines a new polynomial $P_j = \mathsf{X}_n$, which represents the value $x_n$. We thus have

$$P_i(\vec{x}) = a_i \ , \tag{6}$$

and in this sense the polynomial $P_i$ represents the scalar $a_i$ (and thus implicitly the group element $a_iG$). The group-operation oracle maintains this invariant; when computing $a_j := a_i + (-1)^b a_{i'}$, it also sets $P_j := P_i + (-1)^b P_{i'}$.

Note that there are many ways to represent a group element $aG$ by a polynomial. E.g., the first challenge $x_1G$ is represented by both the polynomial $\mathsf{X}_1$ and the constant polynomial $x_1$. Intuitively, since $x_1$ is a challenge, it is unknown to $\mathcal{A}$, and as long as $\mathcal{A}$ does not query DLog$(\xi)$, with $\xi \coloneqq \Xi(x_1)$, it does not know that the polynomials $\mathsf{X}_1$ and $x_1$ represent the same group element. $\mathsf{Game}_1$ introduces a list $L$ that represents this knowledge of $\mathcal{A}$. E.g., when $\mathcal{A}$ calls DLog$(\Xi(x_1))$, the game will append the polynomial $\mathsf{X}_1 - x_1$ to the list $L$. More generally, on call DLog$(\xi_i)$ it appends $P_i - P_i(\vec{x})$ to $L$, which represents the fact that $\mathcal{A}$ knows that the polynomial $P_i - P_i(\vec{x})$ represents the scalar 0 and the group element $0_{\mathbb{G}}$. The list $L$ will be used to ensure consistency when we replace scalars by polynomials in the game.

Recall that our goal is to have the challenger only deal with polynomials when simulating the game for $\mathcal{A}$. As this can be done without actually defining the challenge $\vec{x}$, the challenger could then select $\vec{x}$ *after* $\mathcal{A}$ gave its output, making it impossible for $\mathcal{A}$ to predict the right answer.

This is done in the final game $\mathsf{Game}_4$, defined in Fig. 6, where the challenger is in the same position as $\mathcal{A}$: it does not know that $x_1$ is the answer to the challenge represented by the polynomial $\mathsf{X}_1$ until DLog$(\xi)$ is called with $\xi \coloneqq \Xi(x_1)$. In fact, $x_1$ is not even defined before this call, and, more generally, $\vec{x}$ does not exist until the proper DLog queries are made.

To get to $\mathsf{Game}_4$, we define two intermediate games. We will modify procedure Enc so that it later deals with polynomials only (instead of their evaluations, as $\vec{x}$ will not exist). Because of this, it will be unknown whether $P_j(\vec{x}) = P_i(\vec{x})$ for some $i \in [0, j-1]$, *unless* $P_j - P_i \in \mathsf{Span}(L)$, since both the challenger and the adversary are aware that all polynomials in $L$ evaluate to 0 at $\vec{x}$.

However, it can happen that, when $\vec{x}$ is defined later, $P_j(\vec{x}) = P_i(\vec{x})$. That is, in the original game, we would have had $a_j = a_i$, but in the final game, Enc is not aware of this. This is precisely when the simulation fails, and we abort the game. We will then bound the probability of this event, using Lemma 1.

In "typical" GGM proofs an abort happens when $P_j(\vec{x}) = P_i(\vec{x})$ and $P_j \neq P_i$. For OMDL, because the adversary might have information on the $\vec{x}$ (and the challenger is aware of this), we allow that there are $P_j \neq P_i$ for which the current knowledge on $\vec{x}$ lets us deduce $P_j(\vec{x}) = P_i(\vec{x})$. With the formalism introduced above this corresponds exactly to the situation that $P_i - P_j \in \mathsf{Span}(L)$. We

$\mathsf{Game}_0$, $\boxed{\mathsf{Game}_1}$, $\boxed{\mathsf{Game}_2}$

$\vec{x} := (\,)$ ; $a_0 := 1$

$j := 0\,;\ q := 0\,;\ n := 0$

$\boxed{P_0 := 1\,;\ L := \emptyset}$

$\vec{y} \leftarrow \mathcal{A}^{\textsc{Chal},\textsc{DLog},\textsc{GCmp}}(\textsc{Enc}())$

**if** $q \geq n$ **then return** $0$

**for** $i \in [1, n]$

  $x_i := \textsc{DLog}(\xi_{j_i})$

**return** $\vec{y} = \vec{x}$

Oracle $\textsc{Chal}()$

$n := n + 1$

$x_n \xleftarrow{\$} \mathbb{Z}_p$

$j_n := j$

$j := j + 1$

$a_j := x_n$

$\boxed{P_j := \mathsf{X}_n}$

**return** $\textsc{Enc}()$

Oracle $\textsc{DLog}(\xi)$

**if** $\xi \notin \{\xi_i\}_{i \in [0,j]}$

  **then return** $\perp$

$i := \min\{k \in [0, j] \mid \xi = \xi_k\}$

$q := q + 1$

$v := a_i$ $\boxed{v := P_i(\vec{x})}$

**if** $P_i \in \mathsf{Span}(1, L)$ **then**

  **Let** $(\alpha_k)_{k=0}^{q-1} \in \mathbb{Z}_p^q$ **s.t.**

    $P_i = \alpha_0 + \sum_{k=1}^{q-1} \alpha_k Q_k$

  $v := \alpha_0$

$Q_q := P_i - v$

$\boxed{L = L \cup \{P_i - v\}}$

**return** $v$

$\textsc{Enc}(\,)$ // outputs $\xi_j := \Xi(a_j)$

// Only in $\mathsf{Game}_0$ and $\mathsf{Game}_1$

**if** $\exists i \in [0, j-1] : a_j = a_i$

  **then** $\xi_j := \xi_i$

**if** $\exists i \in [0, j-1] : P_j(\vec{x}) = P_i(\vec{x})$

  **and** $P_j - P_i \notin \mathsf{Span}(L)$

  **then abort game**

**if** $\exists i \in [0, j-1] : P_j - P_i \in \mathsf{Span}(L)$

  **then** $\xi_j := \xi_i$

**else** $\xi_j \xleftarrow{\$} \{0,1\}^{\log(p)} \setminus \{\xi_i\}_{i \in [0,j-1]}$

**return** $\xi_j$

Oracle $\textsc{GCmp}(\xi, \xi', b)$

**if** $\xi \notin \{\xi_i\}_{i \in [0,j]}$ **or** $\xi' \notin \{\xi_i\}_{i \in [0,j]}$

  **then return** $\perp$

$i := \min\{k \in [0, j] \mid \xi = \xi_k\}$

$i' := \min\{k \in [0, j] \mid \xi' = \xi_k\}$

$j := j + 1$ ; $a_j := a_i + (-1)^b a_{i'}$

$\boxed{P_j := P_i + (-1)^b P_{i'}}$

**return** $\textsc{Enc}()$

**Fig. 4.** $\mathsf{Game}_0$ (which only includes the dashed boxes) is the GGM version of OMDL. $\mathsf{Game}_1$ (including all but the gray boxes) introduces the polynomials that represent the information that $\mathcal{A}$ obtains, and aborts when $\mathsf{Game}_0$ cannot be simulated with polynomials. In $\mathsf{Game}_2$ (including all but the dashed boxes) we eliminate the use of scalars (except for the abort condition) in oracles $\textsc{Enc}$ and $\textsc{DLog}$.

introduce this abort condition in the procedure $\textsc{Enc}$ in $\mathsf{Game}_1$ (Fig. 4). Because in the "ideal" game $\mathsf{Game}_4$ (Fig. 6), there are no more values $a_i$, we will express the abort condition differently (namely in oracle $\textsc{DLog}$) and argue that the two conditions are equivalent.

Eliminating uses of scalars. Using the abort condition in $\mathsf{Game}_1$, we can replace some uses of the scalars $a_i$ by their representations as polynomials $P_i$. This is what we do in $\mathsf{Game}_2$, (Fig. 4, including all boxes *except* the dashed box), which eliminates all occurrences of $a_i$'s. In Enc, since the game aborts when $P_j(\vec{x}) = P_i(\vec{x})$ and $P_j - P_i \notin \mathsf{Span}(L)$, and because when $P_j - P_i \in \mathsf{Span}(L)$ then $P_j(\vec{x}) = P_i(\vec{x})$, we can replace the event $P_j(\vec{x}) = P_i(\vec{x})$ by $P_j - P_i \in \mathsf{Span}(L)$. Intuitively, we can now think of Enc() as encoding the polynomial $P_j$ instead of the scalar $a_j$.

We next modify the oracle DLog. The first change is that instead of returning $a_i$ the oracle uses $P_i(\vec{x})$, which is equivalent by (6). The second change is that on input $\xi$, oracle DLog checks if $\mathcal{A}$ already knows the answer to its query, in which case it computes the answer without using $\vec{x}$. E.g., assume $\mathcal{A}$ has only made one query Chal(), and thus $q = 0$ and $L = \emptyset$: if $\mathcal{A}$ now queries DLog($\xi$) with $\xi \coloneqq \varXi(x_1)$, the oracle first checks if $P_i = \mathsf{X}_1 \in \mathsf{Span}(1, L)$, (where $i$ is the current number of group elements seen by the adversary), which is not the case, and so it computes $v \coloneqq P_i(\vec{x}) = x_1$. It then adds the polynomial $Q_1 \coloneqq \mathsf{X}_1 - x_1$ to $L$ and returns $x_1$. If for example $\mathcal{A}$ makes another call DLog($\xi'$) with $\xi' \coloneqq \varXi(2x_1 + 2)$, then it knows that the answer should be $2x_1 + 2$. And indeed, the oracle DLog checks if $2\mathsf{X}_1 + 2 \in \mathsf{Span}(1, L)$, and since this is the case, it gets the decomposition

$$2\mathsf{X}_1 + 2 = (2x_1 + 2) + 2Q_1 = \alpha_0 + \alpha_1 Q_1$$

with $\alpha_0 = 2x_1 + 2$ and $\alpha_1 = 2$. The oracle uses this decomposition to compute its answer $v \coloneqq \alpha_0 = 2x_1 + 2$.

More generally, on input $\xi_i$, the oracle DLog checks if $P_i \in \mathsf{Span}(1, L)$. If so, it computes the answer using the decomposition of $P_i$ in $\mathsf{Span}(1, L)$; else it uses $\vec{x}$ and outputs $a_i = P_i(\vec{x})$.

We have now arrived at a situation close to the "ideal" game, where the challenger only uses polynomials. The only uses of scalars are the abort condition in Enc (since it compares $P_j(\vec{x})$ and $P_i(\vec{x})$) and in DLog, when computing the logarithm of an element that is not already known to $\mathcal{A}$. Towards our goal of simulating the game without defining $\vec{x}$, we modify those two parts next.

Changing the abort condition. The aim of $\mathsf{Game}_3$ is precisely to modify the abort condition so that it does not use $\vec{x}$ anymore. Fig. 5 recalls $\mathsf{Game}_2$ and defines $\mathsf{Game}_3$ by not including the dashed and the gray box. In $\mathsf{Game}_3$ the challenger does not abort in the procedure Enc. This means that if $P_j - P_i \notin \mathsf{Span}(L)$ for some $i$, the challenger creates a string $\xi_j \neq \xi_i$ even when $P_j(\vec{x}) = P_i(\vec{x})$. This means that the simulation of the game is not correct anymore; but we will catch these inconsistencies and abort in the oracle DLog.

For concreteness consider the following example: let $\vec{x} = (x_1)$ and suppose $\mathcal{A}$ built the polynomials $P_{i_1} = x_1$ using the oracle GCmp and $P_{i_2} = \mathsf{X}_1$ using the oracle Chal; suppose also that $\mathcal{A}$ has not queried DLog yet, thus $L = \emptyset$. If $i_1 < i_2$ then $\mathsf{Game}_2$ aborts on the call Enc() which encodes $P_{i_2}$, since $P_{i_1}(\vec{x}) = P_{i_2}(\vec{x})$ and $P_{i_2} - P_{i_1} \notin \mathsf{Span}(L)$. In contrast, in $\mathsf{Game}_3$ the challenger defines $\xi_{i_1} \neq \xi_{i_2}$, which is inconsistent. But the abort will now happen during a call to DLog.

Suppose $\mathcal{A}$ queries $\mathrm{DLOG}(\xi_{i_3})$, with $\xi_{i_3} = \Xi(2\mathsf{X}_1 + 2)$. $\mathsf{Game}_3$ now adds the polynomial $Q_1 = 2\mathsf{X}_1 + 2 - (2x_1 + 2) = 2(\mathsf{X}_1 - x_1)$ to $L$ and checks for an inconsistency of this answer with all the polynomials that $\mathcal{A}$ computed. Since it finds that $P_{i_1} - P_{i_2} = x_1 - \mathsf{X}_1 \in \mathsf{Span}(L)$ but $\xi_{i_1} \neq \xi_{i_2}$, the game aborts. But $\mathsf{Game}_3$ should also abort even if $\mathcal{A}$ does not query the oracle DLOG. This was precisely the reason for adding the final calls of the game to the oracle DLOG in $\mathsf{Game}_0$. Since $P_{j_i} = \mathsf{X}_i$ and the challenger calls $x_i \leftarrow \mathrm{DLOG}(\xi_{j_i})$ for $i \in [1, n]$ at the end, the challenger makes the query $\mathrm{DLOG}(\xi_{j_1})$, which adds $\mathsf{X}_1 - x_1$ to $L$, after which we have $P_{i_1} - P_{i_2} \in \mathsf{Span}(L)$ and therefore an abort.

More generally, in $\mathsf{Game}_3$ the oracle DLOG aborts if there exists $(i_1, i_2) \in [0, j]^2$ such that $P_{i_1} - P_{i_2} \in \mathsf{Span}(L)$ and $\xi_{i_1} \neq \xi_{i_2}$. In the proof of Theorem 1 we show that this abort condition is equivalent to the abort condition in $\mathsf{Game}_2$.

ELIMINATING ALL USES OF $\vec{x}$. In $\mathsf{Game}_3$ the only remaining part that uses $\vec{x}$ is the operation $v \coloneqq P_i(\vec{x})$ in oracle DLOG. Our final game hop will replace this by an equivalent operation. In $\mathsf{Game}_4$, also presented in Fig. 5, the challenger samples $v$ uniformly from $\mathbb{Z}_p$ instead of evaluating $P_i$ on the challenge. In the proof of Theorem 1, we will show that since the distribution of $P_i(\vec{x})$ is uniform for a fixed $P_i$, this change does not affect the game.

This is the only difference between $\mathsf{Game}_4$ and $\mathsf{Game}_3$, but since this modification removes all uses of $\vec{x}$ for the challenger, we rewrite $\mathsf{Game}_4$ explicitly in Fig. 6, where we define $\vec{x}$ only after $\mathcal{A}$ outputs $\vec{y}$. $\mathsf{Game}_4$ is thus easily seen to be impossible (except with negligible probability) to win for $\mathcal{A}$. The reason is that $\mathcal{A}$ cannot make enough queries to DLOG to constrain the construction of $\vec{x}$ at the end of the game and therefore cannot predict the challenge $\vec{x}$. We now make the intuition given above formal in the following theorem.

### 3.3 Formal Proof

**Theorem 1.** *Let $\mathcal{A}$ be an adversary that solves OMDL in a generic group of prime order $p$, making at most $m$ oracle queries. Then*

$$\mathbf{Adv}_{\mathcal{A}}^{\mathrm{OMDLGGM}} \leq \frac{m^2}{p - m^2} + \frac{1}{p} \ .$$

*Proof of Theorem 1.* The proof will proceed as follows: we first compute the statistical distance between $\mathsf{Game}_0$, which is OMDLGGM, and $\mathsf{Game}_1$ (Fig. 4); we then show that $\mathsf{Game}_1$, $\mathsf{Game}_2$, $\mathsf{Game}_3$ and $\mathsf{Game}_4$ (Figures 4 and 5) are equivalently distributed; and finally we upper-bound the probability of winning $\mathsf{Game}_4$ (Fig. 6).

PRELIMINARY RESULTS. We start with proving three useful invariants of the polynomials $P_i$ and the set $L$ which are introduced in $\mathsf{Game}_1$. The first one is:

$$\forall\, i \in [0, j] : \ P_i(\vec{x}) = a_i \ . \tag{7}$$

$\mathsf{Game}_2$, $\boxed{\mathsf{Game}_3}$ , $\boxed{\mathsf{Game}_4}$

$\vec{x} := ()$
$j := 0\,;\; q := 0\,;\; n := 0$
$P_0 := 1\,;\; L := \emptyset$
$\vec{y} \leftarrow \mathcal{A}^{\textsc{Chal},\textsc{DLog},\textsc{GCmp}}(\textsc{Enc}())$
**if** $q \geq n$ **then return** $0$
**for** $i \in [1, n]$
  $x_i := \textsc{DLog}(\xi_{j_i})$
**return** $\vec{y} = \vec{x}$

Oracle $\textsc{Chal}()$

$j := j + 1\,;\; n := n + 1$
$x_n \xleftarrow{\$} \mathbb{Z}_p\,;\; j_n := j$
$P_j := \mathsf{X}_n$
**return** $\textsc{Enc}()$

Oracle $\textsc{DLog}(\xi)$

**if** $\xi \notin \{\xi_i\}_{i\in[0,j]}$ **then return** $\bot$
$i := \min\{k \in [0, j] \mid \xi = \xi_k\}$
$q := q + 1$
$v := P_i(\vec{x})$ $\boxed{;\; v \xleftarrow{\$} \mathbb{Z}_p}$
**if** $P_i \in \mathsf{Span}(1, L)$ **then**
  **let** $(\alpha_k)_{k=0}^{q-1} \in \mathbb{Z}_p^q$ **s.t.** $P_i = \alpha_0 + \sum_{k=1}^{q-1} \alpha_k Q_k$
  $v := \alpha_0$
$Q_q := P_i - v\,;\; L = L \cup \{P_i - v\}$

// Abort condition in $\mathsf{Game}_3$ and $\mathsf{Game}_4$ only
**if** $\exists (i_1, i_2) \in [0, j]^2$ **:** $P_{i_1} - P_{i_2} \in \mathsf{Span}(L)$
  **and** $\xi_{i_1} \neq \xi_{i_2}$
  **then abort game**

**return** $v$

$\textsc{Enc}()$ // outputs $\xi_j$ which encodes $P_j$

// Abort condition in $\mathsf{Game}_2$ only
**if** $\exists i \in [0, j-1]$ **:** $P_j(\vec{x}) = P_i(\vec{x})$
  **and** $P_j - P_i \notin \mathsf{Span}(L)$
  **then abort game**

**if** $\exists i \in [0, j-1] : P_j - P_i \in \mathsf{Span}(L)$
  **then** $\xi_j := \xi_i$
**else**
  $\xi_j \xleftarrow{\$} \{0,1\}^{\log(p)} \setminus \{\xi_i\}_{i\in[0,j-1]}$
**return** $\xi_j$

Oracle $\textsc{GCmp}(\xi, \xi', b)$

**if** $\xi \notin \{\xi_i\}_{i\in[0,j]}$ **or** $\xi' \notin \{\xi_i\}_{i\in[0,j]}$
  **then return** $\bot$
$i := \min\{k \in [0, j] \mid \xi = \xi_k\}$
$i' := \min\{k \in [0, j] \mid \xi' = \xi_k\}$
$j := j + 1$
$P_j := P_i + (-1)^b P_{i'}$
**return** $\textsc{Enc}()$

**Fig. 5.** In $\mathsf{Game}_3$ we move the abort condition from $\textsc{Enc}$ to the oracle $\textsc{DLog}$, so it can be checked without using scalars. The only remaining use is then "$v := P_i(\vec{x})$" in oracle $\textsc{DLog}$. $\mathsf{Game}_4$ instead pick the output $x$ uniformly at random.

This holds in $\mathsf{Game}_1$ and justifies replacing all occurrences of $a_i$ by $P_i(\vec{x})$ in $\mathsf{Game}_2$ in Fig. 5. To prove this, we show that each time the games introduce a new polynomial $P_j$, we have $P_j(\vec{x}) = a_j$.

We prove this by induction. Initially, $P_0 = 1$ and $a_0 = 1$ so the statement holds for $j = 0$. Now suppose it is true for all $i \in [0, j-1]$. We show it is true for $j$. Polynomial $P_j$ can be built either by oracle $\textsc{Chal}$ or by oracle $\textsc{GCmp}$:

$\mathsf{Game}_4$

$j \coloneqq 0\,;\ q \coloneqq 0\,;\ n \coloneqq 0$
$P_0 \coloneqq 1\,;\ L \coloneqq \emptyset$
$\vec{y} \leftarrow \mathcal{A}^{\textsc{Chal},\textsc{DLog},\textsc{GCmp}}(\textsc{Enc}())$
**if** $q \geq n$ **then return** $0$
**for** $i \in [1, n]$
  $x_i \coloneqq \textsc{DLog}(\xi_{j_i})$
**return** $\vec{y} = \vec{x}$

Oracle $\textsc{Chal}()$

$j \coloneqq j+1\,;\ n \coloneqq n+1$
$P_j \coloneqq \mathsf{X}_n\,;\ j_n \coloneqq j$
**return** $\textsc{Enc}()$

Oracle $\textsc{DLog}(\xi)$

**if** $\xi \notin \{\xi_i\}_{i\in[0,j]}$ **then return** $\perp$
$i \coloneqq \min\{k \in [0,j] \mid \xi = \xi_k\}$
$q \coloneqq q+1\,;\ v \xleftarrow{\$} \mathbb{Z}_p$
**if** $P_i \in \mathsf{Span}(1, L)$ **then**
  **let** $(\alpha_k)_{k=0}^{q-1} \in \mathbb{Z}_p^q$ **s.t.** $P_i = \alpha_0 + \sum_{k=1}^{q-1} \alpha_k Q_k$
  $v \coloneqq \alpha_0$
$Q_q \coloneqq P_i - v\,;\ L = L \cup \{P_i - v\}$
**if** $\exists (i_1, i_2) \in [0,j]^2 : P_{i_1} - P_{i_2} \in \mathsf{Span}(L)$
    **and** $\xi_{i_1} \neq \xi_{i_2}$
  **then abort game**
**return** $v$

$\textsc{Enc}()$ // outputs $\xi_j$ which encodes $P_j$

**if** $\exists i \in [0, j-1] : P_j - P_i \in \mathsf{Span}(L)$
 **then** $\xi_j \coloneqq \xi_i$
**else**
  $\xi_j \xleftarrow{\$} \{0,1\}^{\log(p)} \setminus \{\xi_i\}_{i\in[0,j-1]}$
**return** $\xi_j$

Oracle $\textsc{GCmp}(\xi, \xi', b)$

**if** $\xi \notin \{\xi_i\}_{i\in[0,j]}$ **or** $\xi' \notin \{\xi_i\}_{i\in[0,j]}$
  **then return** $\perp$
$i \coloneqq \min\{k \in [0,j] \mid \xi = \xi_k\}$
$i' \coloneqq \min\{k \in [0,j] \mid \xi' = \xi_k\}$
$j \coloneqq j+1$
$P_j \coloneqq P_i + (-1)^b P_{i'}$
**return** $\textsc{Enc}()$

**Fig. 6.** Final game $\mathsf{Game}_4$ does not use $\vec{x}$ in the oracles anymore. It defines the challenge $\vec{x}$ after $\mathcal{A}$ gave its output and this is what makes it simple for us to prove it is hard to win for $\mathcal{A}$.

- In oracle $\textsc{Chal}$, $P_j \coloneqq \mathsf{X}_n$ and $a_j \coloneqq x_n$ so we have $P_j(\vec{x}) = x_n = a_j$.
- In oracle $\textsc{GCmp}$, $P_j \coloneqq P_i + (-1)^b P_{i'}$ and $a_j \coloneqq a_i + (-1)^b a_{i'}$ so we have $P_j(\vec{x}) \coloneqq P_i(\vec{x}) + (-1)^b P_{i'}(\vec{x}) = a_i + (-1)^b a_{i'} = a_j$.

This proves (7).

We next show that the following holds in $\mathsf{Game}_1$, $\mathsf{Game}_2$ and $\mathsf{Game}_3$:

$$\forall\, Q \in \mathsf{Span}(L),\ Q(\vec{x}) = 0 \tag{8}$$

(in the other games either $L$ or $\vec{x}$ are not defined). For $L = \{Q_1, \ldots, Q_q\}$ if $Q \in \mathsf{Span}(L)$ then $Q = \sum_{k=1}^{q} \alpha_k Q_k$. To show (8), it suffices to show that for all $k \in [1, q]$ we have $Q_k(\vec{x}) = 0$.

For $k \in [0, q]$, $Q_k$ is defined during the $k$-th call to $\textsc{DLog}$ on some input $\xi$. In $\mathsf{Game}_1$, the oracle finds $i$ such that $\xi_i = \xi$ and sets $v \coloneqq a_i$ and $Q_k \coloneqq P_i - v$, so we get $Q_k(\vec{x}) = P_i(\vec{x}) - a_i$. Using the first result (7), we get that (8) holds. In $\mathsf{Game}_2$ and $\mathsf{Game}_3$ the oracle sets $v \coloneqq P_i(\vec{x})$ so we directly get $Q_k(\vec{x}) = P_i(\vec{x}) - P_i(\vec{x}) = 0$

The third result we will use holds (assuming the game did not abort) in $\mathsf{Game}_1$, $\mathsf{Game}_2$, $\mathsf{Game}_3$ and $\mathsf{Game}_4$:

$$\forall j \geq 1\, \forall i \in [0, j-1] : \xi_j = \xi_i \Leftrightarrow P_j - P_i \in \mathsf{Span}(L) \ . \tag{9}$$

We first prove

$$\forall j \geq 1\, \forall i \in [0, j-1] :\ \xi_j = \xi_i \Rightarrow P_j - P_i \in \mathsf{Span}(L)$$

by induction. We show that this holds for $j = 1$ and all other $j > 0$ and suppose that for some $i^* \in [0, j-1]$, $\xi_j = \xi_{i^*}$. We show that $P_j - P_{i^*} \in \mathsf{Span}(L)$.

– In $\mathsf{Game}_2$, $\mathsf{Game}_3$ and $\mathsf{Game}_4$, since $\xi_j$ is not a new random string when it is defined, thus for some $i_1 \in [0, j-1]$ we had $P_j - P_{i_1} \in \mathsf{Span}(L)$ and so the game defined $\xi_j \coloneqq \xi_{i_1}$. This implies that $\xi_{i_1} = \xi_{i^*}$, and since $i_1 < j$, using the induction hypothesis, we get that $P_{i_1} - P_{i^*} \in \mathsf{Span}(L)$ and furthermore

$$P_j - P_{i^*} = (P_j - P_{i_1}) - (P_{i_1} - P_{i^*}) \in \mathsf{Span}(L) \ .$$

  Now the situation is simpler when $j = 1$: we must have $i_1 = i^* = 0$ so

$$P_j - P_{i_1} = P_j - P_{i^*} = P_1 - P_0 \in \mathsf{Span}(L) \ .$$

– In $\mathsf{Game}_1$ the proof is almost the same: since $\xi_j$ is not a new random string, thus for some $i_1 \in [0, j-1]$ we had $P_j(\vec{x}) = P_{i_1}(\vec{x})$, so the game defined $\xi_j \coloneqq \xi_{i_1}$. Since the game did not abort, "$P_j(\vec{x}) = P_{i_1}(\vec{x})$ **and** $P_j - P_{i_1} \notin \mathsf{Span}(L)$" does not hold, and thus $P_j - P_{i_1} \in \mathsf{Span}(L)$. From here the proof proceeds as for the other games above, and thus $P_j - P_{i^*} \in \mathsf{Span}(L)$. When $j = 1$, we have $i^* = 0$ and $P_1 - P_0 \in \mathsf{Span}(L)$, as otherwise the game aborts.

We now prove the other implication:

$$\forall j \geq 1\, \forall i \in [0, j-1] : P_j - P_i \in \mathsf{Span}(L) \Rightarrow \xi_j = \xi_i \ ,$$

again by induction. Using the same method as before we can argue that this is true for $j = 1$. For $j > 1$, when $\textsc{Enc}()$ defines $\xi_j$, if for some $i^* \in [0, j-1]$ we have $P_j - P_{i^*} \in \mathsf{Span}(L)$ then we show that $\xi_j$ is assigned $\xi_j = \xi_{i^*}$.

– In $\mathsf{Game}_2$, $\mathsf{Game}_3$ and $\mathsf{Game}_4$, since for some $i_1 \in [0, j-1] : P_j - P_{i_1} \in \mathsf{Span}(L)$, the game defines $\xi_j \coloneqq \xi_{i_1}$. And since

$$P_{i^*} - P_{i_1} = (P_{i^*} - P_j) + (P_j - P_{i_1}) \in \mathsf{Span}(L) \ ,$$

  by induction we get $\xi_{i_1} = \xi_{i^*}$ which yields $\xi_j = \xi_{i^*}$.
– In $\mathsf{Game}_1$, since we know that $(P_j - P_{i^*})(\vec{x}) = 0$ from the (8), we get that for some $i_1 \in [0, j-1] : P_j(\vec{x}) = P_{i_1}(\vec{x})$. Since the game did not abort, we know that $P_j - P_{i_1} \in \mathsf{Span}(L)$, so by the same argument as before, we get $\xi_j = \xi_{i^*}$.

$\underline{\mathsf{Game}_0 \text{ TO } \mathsf{Game}_1}$. We now compare $\mathsf{Game}_0$ to $\mathsf{Game}_1$. The only difference between the two is when $\mathsf{Game}_1$ aborts in the procedure $\textsc{Enc}()$ on event

$$\exists i \in [0, j-1] \textbf{ such that } P_j(\vec{x}) = P_i(\vec{x}) \textbf{ and } P_j - P_i \notin \mathsf{Span}(L) \ . \tag{10}$$

We call this event $F$. Since $\textsc{Enc}$ is called at most $m$ times, we get:

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_0} \leq \mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_1} + m \cdot \Pr[F] \ . \tag{11}$$

We now upper-bound $\Pr[F]$. Before a call to $\textsc{Enc}$, the oracle defines $P_j$. Consider a fixed $i \in [0, j-1]$ and define $P \coloneqq P_j - P_i$. We will upper-bound the probability that

$$P_j(\vec{x}) - P_i(\vec{x}) = P(\vec{x}) = 0$$

with $P \coloneqq P_j - P_i \notin \mathsf{Span}(L)$.

Since $\mathcal{A}$ does not know $\vec{x}$ one might consider applying the Schwartz-Zippel lemma. But we cannot, since $\mathcal{A}$ knows information on $\vec{x}$. From $\mathcal{A}$'s point of view, $\vec{x}$ is not uniformly chosen from $\mathbb{Z}_p^n$, since it satisfies $Q(\vec{x}) = 0$ for all $Q \in L$ (using (8)). We write $L = \{Q_1, \ldots, Q_q\}$, and using the notation from Lemma 1 $Q_{q+1} \coloneqq P$.

$\mathcal{A}$ also knows that if for some indexes $i_1, i_2$ it was given $\xi_{i_1} \neq \xi_{i_2}$ then $P_{i_1}(\vec{x}) \neq P_{i_2}(\vec{x})$. We can reformulate this by writing $D_{\vec{\imath}} = P_{i_1} - P_{i_2}$ for $\vec{\imath} \in I \coloneqq \{(i_1, i_2) \in [0, j-1]^2 \mid \xi_{i_1} \neq \xi_{i_2}\}$. $\mathcal{A}$ knows that $D_{\vec{\imath}}(\vec{x}) \neq 0$. Using the notation of Lemma 1 we get that

$$\vec{x} \in \mathcal{C} \coloneqq \Big( \bigcap_{j \in [1,q]} \mathcal{Q}_j \Big) \setminus \Big( \bigcup_{i \in I} \mathcal{D}_i \Big) \ .$$

Our goal is to apply Lemma 1 to upper-bound $\Pr_{\vec{x} \leftarrow \mathcal{C}}[P(\vec{x}) = 0]$. We need to verify that the three premises of the lemma are satisfied, which are: from $\mathcal{A}$'s point of view, $\vec{x} \in \mathcal{C}$ is picked uniformly at random, $\mathcal{Q}_{q+1} \cap \mathcal{C} \neq \emptyset$ and $\vec{Q}_{q+1}$ is independent of $(\vec{Q}_i)_{i \in [1,q]}$.

$\underline{\vec{x} \text{ IS CHOSEN UNIFORMLY IN } \mathcal{C}}$. To show this, we fix the randomness (of the challenger and the adversary) of the game (which means the order in which the $\xi_i$ are picked is deterministic) and we consider the transcript $\pi(\vec{x})$ of what $\mathcal{A}$ sees during the game when the secret is chosen as $\vec{x}$: $\pi(\vec{x}) = (\xi_0, \ldots, \xi_{j-1}, v_1, \ldots, v_q)$ (In this transcript, the strings $\xi_i$ are ordered and so are the $v_i$, but we implicitly suppose that before the query $v_k$ there was a query $v_{k-1}$ or $\xi_{i_k}$ and after the query $v_k$ there was either a query $v_{k+1}$ or $\xi_{i'_k}$. We do not formalize this.)

The transcript $\pi$ corresponds to all the output of the oracles that were given to $\mathcal{A}$: The $\xi_i$ are the outputs of $\textsc{GCmp}$ and $\textsc{Chal}$, and the $v_i$ are the outputs of $\textsc{DLog}$. The transcript $\pi(\vec{x})$ only depends on the challenge $\vec{x}$. What is important to notice is that for all $\vec{y} \in \mathcal{C}$: $\pi(\vec{y}) = \pi(\vec{x})$. Indeed, if we call $\pi(\vec{y}) = (\xi'_0, \ldots, \xi'_{j-1}, v'_1, \ldots, v'_q)$ we can show by induction that $\xi'_i = \xi_i$ and $v'_k = v_k$ for all $i \in [1, j-1]$ and $k \in [1, q]$.

– Let $k \in [1, q]$; we show that $v_k = v'_k$: in both challenges $\vec{x}$ and $\vec{y}$, since the transcript $\mathcal{A}$ received is the same by the induction hypothesis, it behaves the same way and calls DLog on input $\xi$. The oracle DLog then picks $i = \min\{j \mid \xi_j = \xi\}$ which is the same in both cases by the induction hypothesis. DLog computes $v_k = P_i(\vec{x})$ and defines $Q_i \coloneqq P_i - v_k$ for the challenge $\vec{x}$ while it computes $v'_k = P_i(\vec{y})$ and $Q'_i \coloneqq P_i - v'_k$ for the challenge $\vec{y}$. Now Since $\vec{y} \in \mathcal{C}$, we have in particular $\vec{y} \in \mathcal{Q}_i$, so we know that $Q_k(\vec{y}) = P_i(\vec{y}) - v_k = 0$. This gives $P_i(\vec{y}) = v'_k = v_k$ and $Q'_k = Q_k$.
– Let $k \in [1, j-1]$; we show that $\xi_k = \xi'_k$: for both challenges $\vec{x}$ and $\vec{y}$, since the transcript $\mathcal{A}$ received is the same by induction hypothesis, it behaves the same way and calls either Chal or GCmp. In both cases the the game creates a polynomial $P_k$ and calls the procedure Enc(), for which there are two cases:
  1: $\forall\, i \in [0, k-1] : P_k(\vec{x}) \neq P_i(\vec{x})$. The game with challenge $\vec{x}$ outputs a new random $\xi_k$, which means $\xi_k \neq \xi_i$ for $i \in [1, k-1]$. Since $\vec{y} \in \mathcal{C}$, we know that for all $i \in [0, k-1]$, $\vec{y} \notin \mathcal{D}_{i,k} = \{\vec{z} \;:\; (P_i - P_k)(\vec{z}) = 0$ **and** $\xi_i \neq \xi_k\}$ This means that for all $i \in [0, k-1]$, since $\xi_i \neq \xi_k$, we have $P_i(\vec{y}) \neq P_k(\vec{y})$, so the game also chooses $\xi'_k$ as a new random string. Since we fixed the randomness of the game, we get $\xi_k = \xi'_k$.
  2: $\exists\, i^* \in [0, k-1] : P_k(\vec{x}) = P_{i^*}(\vec{x})$. The game defines $\xi_k \coloneqq \xi_i$ for the challenge $\vec{x}$. Since the game did not abort for $k < j$, we know that $P_k - P_{i^*} \in \mathsf{Span}(L)$. Now since $L = (Q_i)_i$ and $\vec{y} \in \bigcap_{i \in [1,q]} \mathcal{Q}_i$, we also get $(P_k - P_{i^*})(\vec{y}) = 0$. So the game defines $\xi'_k \coloneqq \xi'_i = \xi_i = \xi_k$, by the induction hypothesis and the preliminary result (9).

  In both cases we get that $\xi_k = \xi'_k$.

Since the transcript that $\mathcal{A}$ sees is the same for all elements in $\mathcal{C}$, $\mathcal{A}$ can only make a uniform guess on which element of $\mathcal{C}$ is the challenge. Thus from $\mathcal{A}$'s point of view, $\vec{x}$ is chosen uniformly at random in $\mathcal{C}$.

$\underline{\mathcal{Q}_{q+1} \cap \mathcal{C} \neq \emptyset.}$ Since $\mathcal{Q}_{q+1} = \{\vec{x} \in \mathbb{Z}_p : P(\vec{x}) = 0\}$, if we had $\mathcal{C} \cap \mathcal{Q}_{q+1} = \emptyset$, then $P(\vec{x}) \neq 0$ for all $\vec{x} \in \mathcal{C}$, and thus $\Pr_{\vec{x} \xleftarrow{\$} \mathcal{C}}[P(\vec{x}) = 0] = 0$. In this case, there is no need to upper-bound the probability, which is why we assume that $\mathcal{Q}_{q+1} \cap \mathcal{C} \neq \emptyset$.

$\underline{\vec{Q}_{q+1} \text{ is independent of } (\vec{Q}_i)_{i \in [1,q]}.}$ Recall that $\vec{P} = (p_k)_{k \in [1,n]}$ is the vector representing the polynomial $P - P(\vec{0}) = \sum_{k=1}^{n} p_k \mathsf{X}_k$. We assume that $\vec{Q}_{q+1}$ is dependent of $(\vec{Q}_i)_{i \in [1,q]}$ and then show that this contradicts the previous premise $\mathcal{Q}_{q+1} \cap \mathcal{C} \neq \emptyset$. Assume thus that for some $\alpha$:

$$Q_{q+1} - Q_{q+1}(\vec{0}) = \sum_{k=1}^{q} \alpha_k \big(Q_k - Q_k(\vec{0})\big) \ .$$

With $\alpha \coloneqq Q_{q+1}(\vec{0}) + \sum_{k=1}^{q} \alpha_k Q_k(\vec{0})$ and $Q \coloneqq \sum_{k=1}^{q} \alpha_k Q_k$, we can write this as $Q_{q+1} = \alpha + Q$ with $\alpha \in \mathbb{Z}_p$ and $Q \in \mathsf{Span}(L)$. Now since we are in event $F$, defined in (10), we have $Q_{q+1} = P \notin \mathsf{Span}(L)$, which implies $\alpha \neq 0$ (otherwise

$P = Q \in \mathsf{Span}(L)$). Since $\mathcal{C} \subset \mathcal{Q}_i$ we have that for all $i \in [1, q]$ and all $\vec{x} \in \mathcal{C}$: $Q_i(\vec{x}) = 0$, and thus $Q(\vec{x}) = 0$. From this, we have $Q_{q+1}(\vec{x}) = \alpha + Q(\vec{x}) = \alpha$. Thus, $Q_{q+1}(\vec{x}) \neq 0$ for all $\vec{x} \in \mathcal{C}$, which implies $\mathcal{C} \cap \mathcal{Q}_{q+1} = \emptyset$, which contradicts the previous assumption. We thus proved that $\vec{Q}_{q+1}$ is independent of $(\vec{Q}_i)_{i\in[1,q]}$.

APPLYING LEMMA 1. Since all its premises are satisfied, we can apply Lemma 1 and obtain:

$$\Pr_{\vec{x}\leftarrow\mathcal{C}}\left[P(\vec{x}) = 0\right] = \Pr_{\vec{x}\leftarrow\mathcal{C}}\left[Q_{q+1}(\vec{x}) = 0\right] \leq \frac{1}{p - |I|} \ ,$$

with $|I| \leq j^2 \leq m^2$. Since we need to test this with $P = P_j - P_i$ for all $i \in [0, j-1]$, we get $\Pr[F] \leq \dfrac{m}{p - m^2}$ and from (11):

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_0} \leq \mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_1} + \frac{m^2}{p - m^2} \ . \tag{12}$$

$\mathsf{Game}_1$ TO $\mathsf{Game}_2$. There are three changes in $\mathsf{Game}_2$, which we show do not affect the distributions of the game. First, we replace $a_i$ by $P_i(\vec{x})$ in oracle DLOG, which is equivalent by (7).

Second, in ENC, we replace the condition

$$\textbf{if } \exists i \in [0, j-1] : P_j(\vec{x}) = P_i(\vec{x}) \textbf{ then } \xi_j \coloneqq \xi_i$$

by

$$\textbf{if } \exists i \in [0, j-1] : P_j - P_i \in \mathsf{Span}(L) \textbf{ then } \xi_j \coloneqq \xi_i \ .$$

We show that this new condition does not affect the output of ENC(). There are two cases for $P_j(\vec{x})$:

Case 1: $\exists i^* \in [0, j-1] : P_j(\vec{x}) = P_{i^*}(\vec{x})$. We have either
- $P_j - P_{i^*} \in \mathsf{Span}(L)$, and in this case $\mathsf{Game}_1$ and $\mathsf{Game}_2$ both set $\xi_j = \xi_{i^*}$ and output $\xi_j$ using (9); or
- $P_j - P_{i^*} \notin \mathsf{Span}(L)$, meaning that both $\mathsf{Game}_1$ and $\mathsf{Game}_2$ abort since "$P_j - P_{i^*} \notin \mathsf{Span}(L)$ **and** $P_j(\vec{x}) = P_{i^*}(\vec{x})$" is the abort condition.

Case 2: $\forall i \in [0, j-1]$ $P_j(\vec{x}) \neq P_i(\vec{x})$. Since, by 8, all polynomials in $\mathsf{Span}(L)$ vanish at $\vec{x}$, this implies $\forall i \in [0, j-1] : P_j - P_i \notin \mathsf{Span}(L)$. In this case both $\mathsf{Game}_1$ and $\mathsf{Game}_2$ output a random new string $\xi_j$.

The third change in $\mathsf{Game}_2$, in the oracle DLOG, does not change the output either: in $\mathsf{Game}_1$ the DLOG oracle always outputs $a_i = P_i(\vec{x})$. In $\mathsf{Game}_2$, when $P_i \in \mathsf{Span}(L)$, the game uses the decomposition $P_i = \alpha_0 + \sum_{k=1}^{q-1} \alpha_k Q_k$, and since $Q_k(\vec{x}) = 0$ by (8), it outputs $P_i(\vec{x}) = \alpha_0$.

Together this yields:

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_1} = \mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_2} \ . \tag{13}$$

$\mathsf{Game}_2$ TO $\mathsf{Game}_3$. In this game hop we move the abort condition from the procedure ENC to the oracle DLOG. We show that the two abort conditions are equivalent, by showing the two implications of the equivalence:

If $\mathsf{Game}_2$ aborts then $\mathsf{Game}_3$ also aborts. If $\mathsf{Game}_2$ aborts, it means that for a fixed index $j^*$ the game found $i^* \in [0, j^* - 1]$ such that $P_{j^*} - P_{i^*} \notin \mathsf{Span}(L)$ and $P_{j^*}(\vec{x}) = P_{i^*}(\vec{x})$. We show that $\mathsf{Game}_3$ also aborts in this situation. Let $P := P_{j^*} - P_{i^*}$. At the end of $\mathsf{Game}_3$ the challenger makes calls to DLog on each challenge $P_{j_i} = \mathsf{X}_i$. This adds the corresponding polynomials $\mathsf{X}_i - x_i$ to $L$ for all $i \in [1, n]$. With $P = P(\vec{0}) + \sum_{k=1}^{n} p_k \mathsf{X}_k$, we can write

$$P = \sum_{k=1}^{n} p_k(\mathsf{X}_k - x_k) + P(\vec{0}) + \sum_{k=1}^{n} p_k x_k \ .$$

Since $P(\vec{x}) = P_{j^*}(\vec{x}) - P_{i^*}(\vec{x})$, we have $P(\vec{x}) = 0$. On the other hand, by the equation above, we have $P(\vec{x}) = P(\vec{0}) + \sum_{k=1}^{n} p_k x_k$. Together, this yields $P = \sum_{k=1}^{n} p_k(\mathsf{X}_k - x_k)$, which means $P \in \mathsf{Span}(L)$ at the end of the game. At the time when $\mathsf{Game}_2$ would have aborted, we had $P \notin \mathsf{Span}(L)$ and thus the game attributed two different strings $\xi_{i^*} \neq \xi_{j^*}$ to $P_{i^*}$ and $P_{j^*}$, respectively. But at the end of $\mathsf{Game}_3$, when $L$ contains all $\mathsf{X}_i - x_i$ for $i \in [1, n]$, we have $P \in \mathsf{Span}(L)$. This means that one call to DLog updated $L$ so that $P \in \mathsf{Span}(L)$ and when this happened, since $\xi_{i^*} \neq \xi_{j^*}$, the abort condition in DLog was satisfied and the game aborted

If $\mathsf{Game}_3$ aborts then $\mathsf{Game}_2$ also aborts. If $\mathsf{Game}_3$ aborts, then on a call to DLog we have $\exists (i_1, i_2) \in [0, j]^2$ such that $P_{i_1} - P_{i_2} \in \mathsf{Span}(L)$ and $\xi_{i_1} \neq \xi_{i_2}$. From $P_{i_1} - P_{i_2} \in \mathsf{Span}(L)$, using (8) we get $P_{i_1}(\vec{x}) = P_{i_2}(\vec{x})$. Suppose $i_1 < i_2$. The challenger in $\mathsf{Game}_2$ used the procedure Enc() when the counter $j$ was equal to $i_2$ to compute $\xi_{i_2} \neq \xi_{i_1}$. This means that at that moment, $L$ contained fewer elements and we had $P_{i_2} - P_{i_1} \notin \mathsf{Span}(L)$. Since $\mathsf{Game}_2$ aborts when $P_{i_1}(\vec{x}) = P_{i_2}(\vec{x})$ and $P_{i_2} - P_{i_1} \notin \mathsf{Span}(L)$, thus $\mathsf{Game}_2$ aborts in this case.

Combining both implications yields

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_2} = \mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_3} \ . \tag{14}$$

$\mathsf{Game}_3$ to $\mathsf{Game}_4$. The only difference between these games is in the oracle DLog. Instead of computing $v \coloneqq P_i(\vec{x})$, $\mathsf{Game}_4$ picks a random $v \xleftarrow{\$} \mathbb{Z}_p$. We prove that after this modification, the distribution of the outputs of oracle DLog remains the same. The difference between the two games occurs only when $P_i \notin \mathsf{Span}(1, L)$. Let us bound $\Pr_{\vec{x} \leftarrow \mathcal{C}}\left[P_i(\vec{x}) = v \textbf{ in } \mathsf{Game}_3\right]$, where $\vec{x} \in \mathcal{C}$ represents the information that $\mathcal{A}$ knows about $\vec{x}$, which we previously used in the first game hop.

We apply Lemma 1 again to bound this probability. Now since the game does not abort immediately when the inconsistency $P_{i_1}(\vec{x}) = P_{i_2}(\vec{x})$ and $\xi_{i_1} \neq \xi_{i_2}$ occurs, the inequalities on the strings level do not give $\mathcal{A}$ any information on what the evaluation $P_i(\vec{x})$ cannot be. This means that $\mathcal{C}$ is simpler than in the first game hop, namely

$$\mathcal{C} = \bigcap_{i \in [1,q]} \mathcal{Q}_i \ .$$

We define $Q_{q+1} \coloneqq P_i - v$ and show that once again the three premises of Lemma 1 hold: $\vec{x} \in \mathcal{C}$ is picked uniformly at random, $\mathcal{Q}_{q+1} \cap \mathcal{C} \neq \emptyset$ and $\vec{Q}_{q+1}$ is independent of $(\vec{Q}_i)_{i\in[1,q]}$.

<u>$\vec{x}$ IS CHOSEN UNIFORMLY IN $\mathcal{C}$.</u> To show this, we again fix the randomness of the game and consider the transcript $\pi$ that $\mathcal{A}$ sees during the game if a particular $\vec{x}$ is chosen: $\pi(\vec{x}) = (\xi_0, \ldots, \xi_{j-1}, v_1, \ldots, v_q)$, which contains all oracle outputs given to $\mathcal{A}$. We show that for all $\vec{y} \in \mathcal{C} : \pi(\vec{y}) = \pi(\vec{x})$. Indeed, for $\pi(\vec{y}) =: (\xi'_0, \ldots, \xi'_{j-1}, v'_1, \ldots, v'_q)$ we show by induction that $\xi'_i = \xi_i$ and $v'_k = v_k$ for all $i \in [1, j-1]$ and $k \in [1, q]$.

- Let $k \in [1, q]$; then $v_k = v'_k$ is showed exactly as in the first game hop (on page 22).
- Let $k \in [1, j-1]$; we show that $\xi_k = \xi'_k$: for both challenges $\vec{x}$ and $\vec{y}$, since the transcript $\mathcal{A}$ received is the same by induction hypothesis, $\mathcal{A}$ behaves the same way and calls either CHAL or GCMP. In both cases the game creates a polynomial $P_k$ and calls ENC(), for which there are two cases:
  1: $\forall\, i \in [0, k-1] : P_k - P_i \notin \mathsf{Span}(L)$. Since this condition is independent of $\vec{x}$ and $\vec{y}$, for both the game outputs a new random string $\xi_k$ and $\xi'_k$. Since we fixed the randomness of the game, we get $\xi_k = \xi'_k$.
  2: $\exists i^* \in [0, k-1] : P_k - P_{i^*} \in \mathsf{Span}(L)$. In this case the game defines $\xi_k \coloneqq \xi_i$ and $\xi'_k \coloneqq \xi'_i$ for both challenge $\vec{x}$ and $\vec{y}$. We get $\xi'_k \coloneqq \xi'_i = \xi_i = \xi_k$ by the induction hypothesis and (9).

  In both cases we thus have $\xi_k = \xi'_k$.

As in first game hop, we conclude that $\mathcal{A}$ cannot distinguish between two different values $\vec{x} \in \mathcal{C}$ and so we can consider $\vec{x}$ to be chosen uniformly at random in $\mathcal{C}$.

<u>$\vec{Q}_{q+1}$ IS LINEARLY INDEPENDENT OF $(\vec{Q}_i)_{i\in[1,q]}$.</u> Recall that $P_i \notin \mathsf{Span}(1, L)$ and $Q_{q+1} \coloneqq P_i - v$. If $\vec{Q}_{q+1}$ were linearly dependent of $(\vec{Q}_i)_{i\in[0,j]}$, then (using the same method as in the first game hop) we would have $Q_{q+1} = P_i - v = \alpha + Q$ with $\alpha \in \mathbb{Z}_p$ and $Q \in \mathsf{Span}(L)$. As this contradicts $P_i \notin \mathsf{Span}(1, L)$, we conclude that $\vec{Q}_{q+1}$ is linearly independent of $(\vec{Q}_i)_{i\in[1,q]}$.

<u>$\mathcal{Q}_{q+1} \cap \mathcal{C} \neq \emptyset$.</u> $\mathcal{C} = \bigcup_{i\in[1,q]} \mathcal{Q}_i$ is an affine space and $\vec{Q}_{q+1}$ is linearly independent of $(\vec{Q}_i)_{i\in[0,j]}$. This implies that $\mathcal{Q}_{q+1} \cap \mathcal{C}$ has dimension $\dim(\mathcal{C}) - 1$ and thus $\mathcal{Q}_{q+1} \cap \mathcal{C} \neq \emptyset$.

<u>APPLYING LEMMA 1.</u> Since its three premises are satisfied, Lemma 1 with $M \coloneqq 0$ yields:

$$\Pr[\, Q_{q+1}(\vec{x}) = 0\,]_{\vec{x}\leftarrow\mathcal{C}} = \Pr_{\vec{x}\leftarrow\mathcal{C}}\left[P_i(\vec{x}) = v \textbf{ in } \mathsf{Game}_3\right] = \frac{1}{p}\ .$$

This means that in $\mathsf{Game}_3$ the distribution of $P_i(\vec{x})$ is uniform, so the change we make does not affect the overall distribution of the game. We thus have

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_3} = \mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_4}\ . \tag{15}$$

Analysis of $\mathsf{Game}_4$. We prove that $\mathcal{A}$ wins $\mathsf{Game}_4$ at most with negligible probability $\frac{1}{p}$. To do this, we prove that at least one component of the vector $\vec{x}$ is picked uniformly at random *after* $\mathcal{A}$ outputs $\vec{y}$.

When $\mathcal{A}$ outputs $\vec{y}$, $L$ contains $q$ elements, so $\mathsf{dim}(\mathsf{Span}(L)) \leq q$. Since $q < n$, $\mathsf{Span}(1, L)$ has dimension at most $q+1$ and therefore at most $n$ when the adversary outputs the vector $\vec{y}$. Since the dimension of $\mathsf{Span}(\mathsf{X}_1, \ldots, \mathsf{X}_n)$ is $n$ and $1 \notin \mathsf{Span}(\mathsf{X}_1, \ldots, \mathsf{X}_n)$, we get that $\mathsf{Span}(\mathsf{X}_1, \ldots, \mathsf{X}_n)$ is not contained in $\mathsf{Span}(1, L)$. This means that there will be at least one index $i \in [1, n]$ such that $\mathsf{X}_i \notin \mathsf{Span}(1, L)$. We choose the smallest index $i$ that verifies this. Then the oracle DLog outputs a randomly sampled value $x_i$ when called on $\xi_{j_i}$. This $x_i$ is sampled randomly after the $i$-th coefficient of vector $\vec{y}$ output by $\mathcal{A}$ and we obtain: $\Pr[\vec{x} = \vec{y}] \leq \frac{1}{p}$. This yields:

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_4} \leq \frac{1}{p}. \tag{16}$$

The theorem now follows from Equations (12), (13), (14), (15), and (16) $\square$

## 4 OMCDH in the GGM

The OMCDH assumption (defined in Fig. 2), though similar to OMDL, is slightly more complex. In OMDL the adversary has access to a DLog oracle and must solve DLog challenges; in OMCDH the adversary has access to a CDH oracle and must solve CDH challenges. This CDH oracle enables the adversary to construct (encodings of) group elements corresponding to high-degree polynomials: on input $(\Xi(x), \Xi(y))$, the oracle returns $\Xi(xy)$, which in the "ideal" game is encoded as the product of the polynomials representing $x$ and $y$. This makes using known proof techniques in the GGM impossible, since if their degree is not linearly bounded, $\mathcal{A}$ can build non-zero polynomials that evaluate to zero on the challenge with non-negligible probability. (E.g., $\mathsf{X}^p - \mathsf{X}$ evaluates to 0 everywhere in $\mathbb{Z}_p$.)

Given this, we can neither use the Schwartz-Zippel lemma (as it would only yield a non-negligible bound on the adversary's advantage) nor Lemma 1 (since it only applies to polynomials of degree 1). In fact, existing cryptanalysis, such as the attacks by Maurer and Wolf [40, 38]), precisely uses high-degree polynomials to break DL in groups of order $p$ (when $p-1$ is smooth) when given a CDH oracle.

Since the GGM does not handle high-degree polynomials well, we will analyze the hardness of OMCDH by considering a *stronger* assumption instead, which we call $\text{OMCDH}^{\text{DL}}$ and define in Fig. 7. This problem is analogous to OMCDH, except that the CDH oracle is replaced by a DLog oracle. As the adversary has access to the same oracles as in the OMDL game, it can only build polynomials of degree at most 1, as seen in our proof of OMDL. In the full version [4] we show that $\text{OMCDH}^{\text{DL}}$ implies OMDL and that (modulo a polynomial number of group operations) $\text{OMCDH}^{\text{DL}}$ implies OMCDH.

| Game $\mathrm{OMCDHDL}^{\mathcal{A}}_{\mathsf{GrGen}}(\lambda)$ | Oracle $\textsc{Chal}()$ | Oracle $\mathrm{D}\textsc{Log}(X)$ |
|---|---|---|
| $(p, \mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$ | $x \xleftarrow{\$} \mathbb{Z}_p\,;\ X \coloneqq xG$ | $q \coloneqq q+1$ |
| $\vec{Z} \coloneqq (\,)\,;\ q \coloneqq 0$ | $y \xleftarrow{\$} \mathbb{Z}_p\,;\ Y \coloneqq yG$ | $x \coloneqq \log_G(X)$ |
| $\vec{Z}' \leftarrow \mathcal{A}^{\textsc{Chal},\mathrm{D}\textsc{Log}}(p, \mathbb{G}, G)$ | $\vec{Z} \coloneqq \vec{Z} \,\|\, (xyG)$ | **return** $x$ |
| **return** $\big(\vec{Z} = \vec{Z}' \ \wedge\ q < \lvert\vec{Z}\rvert\big)$ | **return** $(X, Y)$ | |

**Fig. 7.** The $\mathrm{OMCDH}^{\mathrm{DL}}$ problem

In the full version [4] we formally prove the hardness of $\mathrm{OMCDH}^{\mathrm{DL}}$ in the generic group model. This is done following the same strategy as for OMDL in Theorem 1; the games hops are the same, only the final analysis of the last game is different, since the winning condition is different, which yields a different winning probability at the end. This is summarized in Theorem 2 below.

**Proposition 1 ($\mathbf{OMCDH^{DL}}$ implies OMCDH).** *In a cyclic group of order $p$, let $\mathcal{A}$ be an adversary that solves OMCDH using at most $m$ group operations and $q$ calls to* $\mathrm{D}\textsc{Log}$*. Then there exists an adversary $\mathcal{B}$ that solves $OMCDH^{DL}$ using at most $m + 2q\lceil\log(p)\rceil$ group operations.*

The proof is straightforward; the reduction answers CDH oracle queries by making queries to its $\mathrm{D}\textsc{Log}$ oracle.

**Theorem 2.** *Let $\mathcal{A}$ be an adversary that solves $OMCDH^{DL}$ in a generic group of order $p$, making at most $m$ oracle queries. Then*

$$\mathbf{Adv}^{\text{OMCDH-GGM}}_{\mathcal{A}} \le \frac{1}{p-1} + \frac{2m}{p} + \frac{m^2}{p-m^2}\ .$$

A formal proof can be found in the full version [4]. Combining this with Proposition 1, we obtain the following corollary, which proves the security of OMCDH in the generic group model.

**Corollary 1.** *Let $\mathcal{A}$ be an adversary that solves $OMCDH^{DL}$ in a generic group of order $p$, making at most $m$ oracle queries and $q$* CDH *oracle queries. Then*

$$\mathbf{Adv}^{\text{OMCDH-GGM}}_{\mathcal{A}} \le \frac{1}{p-1} + \frac{2(m+2q\lceil\log(p)\rceil)}{p} + \frac{(m+2q\lceil\log(p)\rceil)^2}{p-(m+2q\lceil\log(p)\rceil)^2}\ .$$

**Acknowledgements.** We would like to thank the reviewers for their valuable feedback. The second author is supported by the Vienna Science and Technology Fund (WWTF) through project VRG18-002. This work is funded in part by the MSR–Inria Joint Centre.

## References


[1] Ali Bagherzandi, Jung Hee Cheon, and Stanislaw Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008*, pages 449–458. ACM Press, October 2008.

[2] Foteini Baldimtsi and Anna Lysyanskaya. On the security of one-witness blind signature schemes. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 82–99. Springer, Heidelberg, December 2013.

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

[4] Balthazar Bauer, Georg Fuchsbauer, and Antoine Plouviez. The one-more discrete logarithm assumption in the generic group model. Cryptology ePrint Archive, Report 2021/866, 2021. https://ia.cr/2021/866.

[5] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security proofs for identity-based identification and signature schemes. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 268–286. Springer, Heidelberg, May 2004.

[6] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, June 2003.

[7] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, October / November 2006.

[8] Mihir Bellare and Adriana Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 162–177. Springer, Heidelberg, August 2002.

[9] Mihir Bellare and Sarah Shoup. Two-tier signatures, strongly unforgeable signatures, and Fiat-Shamir without random oracles. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 201–216. Springer, Heidelberg, April 2007.

[10] Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. On the (in)security of ROS. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 33–53. Springer, Heidelberg, October 2021.

[11] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. *Journal of Cryptographic Engineering*, 2(2):77–89, September 2012.

[12] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, Heidelberg, January 2003.

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

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

[15] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 435–464. Springer, Heidelberg, December 2018.

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

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

[18] Daniel R. L. Brown. Irreducibility to the one-more evaluation problems: More may be less. Cryptology ePrint Archive, Report 2007/435, 2007. https://eprint.iacr.org/2007/435.

[19] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982.

[20] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer, Heidelberg, August 1993.

[21] 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, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 693–721. Springer, Heidelberg, August 2018.

[22] Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. On the security of two-round multi-signatures. *IEEE Symposium on Security and Privacy*, pages 1084–110, 2019.

[23] Marc Fischlin and Nils Fleischhacker. Limitations of the meta-reduction technique: The case of Schnorr signatures. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 444–460. Springer, Heidelberg, May 2013.

[24] Marc Fischlin and Dominique Schröder. On the impossibility of three-move blind signature schemes. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 197–215. Springer, Heidelberg, May / June 2010.

[25] Nils Fleischhacker, Tibor Jager, and Dominique Schröder. On tight security proofs for Schnorr signatures. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 512–531. Springer, Heidelberg, December 2014.

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

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

[28] Masayuki Fukumitsu and Shingo Hasegawa. Impossibility of the provable security of the Schnorr signature from the one-more DL assumption in the non-programmable random oracle model. In Tatsuaki Okamoto, Yong Yu, Man Ho Au, and Yannan Li,

editors, *ProvSec 2017*, volume 10592 of *LNCS*, pages 201–218. Springer, Heidelberg, October 2017.

[29] Sanjam Garg, Raghav Bhaskar, and Satyanarayana V. Lokam. Improved bounds on security reductions for discrete log based signatures. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 93–107. Springer, Heidelberg, August 2008.

[30] Rosario Gennaro, Darren Leigh, R. Sundaram, and William S. Yerazunis. Batching Schnorr identification scheme with applications to privacy-preserving authorization and low-bandwidth communication devices. In Pil Joong Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 276–292. Springer, Heidelberg, December 2004.

[31] Robert Granger. On the static Diffie-Hellman problem on elliptic curves over extension fields. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 283–302. Springer, Heidelberg, December 2010.

[32] K. Itakura and K. Nakamura. A public-key cryptosystem suitable for digital multisignatures. *NEC Research and Development*, 71:1–8, 1983.

[33] Neal Koblitz and Alfred Menezes. Another look at non-standard discrete log and Diffie-Hellman problems. *J. Math. Cryptol.*, 2(4):311–326, 2008.

[34] Neal Koblitz and Alfred Menezes. The brave new world of bodacious assumptions in cryptography. *Notices of the American Mathematical Society*, 57(3):357–365, 2010.

[35] Neal Koblitz and Alfred J. Menezes. Another look at "provable security". *Journal of Cryptology*, 20(1):3–37, January 2007.

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

[37] Changshe Ma, Jian Weng, Yingjiu Li, and Robert H. Den. Efficient discrete logarithm based multi-signature scheme in the plain public key mode. *Des. Codes Cryptography*, 54(2):121–133, 2010.

[38] Ueli M. Maurer. Information-theoretic cryptography (invited lecture). In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 47–64. Springer, Heidelberg, August 1999.

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

[40] Ueli M. Maurer and Stefan Wolf. Diffie-Hellman oracles. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 268–282. Springer, Heidelberg, August 1996.

[41] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple Schnorr multi-signatures with applications to Bitcoin. *Des. Codes Cryptogr.*, 87(9):2139–2164, 2019.

[42] Silvio Micali and Ronald L. Rivest. Transitive signature schemes. In Bart Preneel, editor, *CT-RSA 2002*, volume 2271 of *LNCS*, pages 236–243. Springer, Heidelberg, February 2002.

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

[44] Jonas Nick. Blind signatures in scriptless scripts. Presentation given at *Building on Bitcoin* 2019, 2019. Slides and video available at https://jonasnick.github.io/blog/2018/07/31/blind-signatures-in-scriptless-scripts/.

[45] Jonas Nick, Tim Ruffing, and Yannick Seurin. MuSig2: Simple two-round Schnorr multi-signatures. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 189–221, Virtual Event, August 2021. Springer, Heidelberg.

[46] NIST. Digital signature standard (DSS), FIPS PUB 186-5 (draft), 2019. https://csrc.nist.gov/publications/detail/fips/186/5/draft.

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

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

[49] Rafael Pass. Limits of provable security from standard assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 109–118. ACM Press, June 2011.

[50] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398. Springer, Heidelberg, May 1996.

[51] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991.

[52] Claus-Peter Schnorr. Security of blind discrete log signatures against interactive attacks. In Sihan Qing, Tatsuaki Okamoto, and Jianying Zhou, editors, *ICICS 01*, volume 2229 of *LNCS*, pages 1–12. Springer, Heidelberg, November 2001.

[53] Claus-Peter Schnorr and Markus Jakobsson. Security of discrete log cryptosystems in the random oracle and the generic model, 1999. Available at https://core.ac.uk/download/pdf/14504220.pdf.

[54] Yannick Seurin. On the exact security of Schnorr-type signatures in the random oracle model. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 554–571. Springer, Heidelberg, April 2012.

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

[56] Ewa Syta, Iulia Tamas, Dylan Visher, David Isaac Wolinsky, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ismail Khoffi, and Bryan Ford. Keeping authorities "honest or bust" with decentralized witness cosigning. In *2016 IEEE Symposium on Security and Privacy*, pages 526–545. IEEE Computer Society Press, May 2016.

[57] David Wagner. A generalized birthday problem. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 288–303. Springer, Heidelberg, August 2002.

[58] Pieter Wuille, Jonas Nick, and Tim Ruffing. Schnorr signatures for secp256k1. Bitcoin Improvement Proposal 340, 2020. See https://github.com/bitcoin/bips/blob/master/bip-0340.mediawiki.

[59] Aaram Yun. Generic hardness of the multiple discrete logarithm problem. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 817–836. Springer, Heidelberg, April 2015.