Sufficient Conditions for Intractability over Black-Box Groups: Generic Lower Bounds for Generalized DL and DH Problems

Andy Rupp[1], Gregor Leander[1], Endre Bangerter[2], Ahmad-Reza Sadeghi[1], Alexander W. Dent[3]

[1] Horst Görtz Institute for IT-Security (Germany)
{arupp, sadeghi}@crypto.rub.de, gregor.leander@rub.de
[2] Bern University of Applied Sciences (Switzerland)
endre.bangerter@bfh.ch
[3] Royal Holloway, University of London (United Kingdom)
a.dent@rhul.ac.uk

Abstract. The generic (aka. black-box) group model is a valuable methodology for analyzing the computational hardness of number-theoretic problems used in cryptography. Since the properties ensuring generic hardness have not been well-studied and formalized yet, for each newly proposed problem an entire hardness proof has to be done from scratch. In this work we identify criteria that guarantee the hardness of generalized DL and DH problems in an extended generic group model where algorithms are allowed to perform any operation representable by a polynomial function. Assuming our conditions are satisfied, we are able to provide negligible upper bounds on the success probability of such algorithms. As useful means for the formalization of definitions and conditions we explicitly relate the concepts of generic algorithms and straight-line programs that have only been used independently in cryptography so far.

Keywords: Generic Group Model, Straight-Line Programs, Hardness Conditions, Lower Bounds.

1 Introduction

The *generic group model* was introduced by Nechaev [Nec94] and Shoup [Sho97a]. In this model one considers algorithms (so-called *generic algorithms*) that given a group G as black box, may only perform a set of basic operations on the elements of G such as applying the group law, inversion of group elements and equality testing. Since in this model the group is treated as black box, the algorithms cannot exploit any special properties of a concrete group representation. Natural examples of this class of algorithms are the Pohlig-Hellman [PH78] and Pollard-Rho [Pol78] algorithm for computing discrete logarithms.

Many fundamental cryptographic problems were proven to be computationally intractable in the generic model, most notably the discrete logarithm problem (DLP), the computational and decisional Diffie-Hellman problem (DHP and DDHP) [Sho97a] and the root extraction problem (in groups of hidden order) [DK02]. These intractability results are considered to be evidence supporting cryptographic assumptions of number-theoretic nature which underly the security of a vast number of systems of applied cryptography.

To be precise, a generic group algorithm can only perform a subset of the operations that can be performed by an algorithm that may exploit specific properties of the representation of group elements. This implies that proving a problem to be intractable in the generic group model is a *necessary, but not sufficient* condition for the problem to be intractable in any concrete group. A generically intractable problem that is easy in any concrete group has been considered in [Den02].

Nonetheless, when making new intractability assumptions it is, loosely speaking, considered to be good practice to prove the underlying problem to be hard in the generic model. Many novel assumptions rely on more complex algebraic settings than standard assumptions like the DL, DH or root assumption. They involve multiple groups and operations on group elements additional to the basic operations. Examples are the numerous assumptions based on bilinear pairings (e.g., see [BF01,BB04a,Yac02]). For an adequate analysis of computational and decisional problems over such algebraic settings and their proper reflection, the basic generic group model needs to be extended. For instance, in the case of the bilinear Diffie-Hellman problem [BF01], the model is extended to two (or three) groups and the set of operations a generic algorithm may perform in addition to the basic ones contains an operation for applying the bilinear map between these groups.

As a result, one ends up with various flavors of the generic model. Since the properties ensuring generic hardness had not been well-studied and formalized before this work, for each novel problem an entire hardness proof had to be done from scratch. This can easily become a complicated and cumbersome task since a rigorous and complete proof in the generic model is quite technical. In fact, the proofs are often only sketched or even entirely omitted in the literature.

1.1 Our Contributions

In a nutshell, we introduce comprehensive classes of computational problems and identify the core aspects making these problems hard in the generic model. We provide a set of conditions which, given the description of a cryptographic problem, allow to easily check whether the problem at hand is intractable with respect to generic algorithms performing certain operations. In this way we aim at (i) providing a useful tool to structure and analyze the rapidly growing set of cryptographic assumptions as motivated in [Bon07] and (ii) making the first steps towards automatically checkable hardness conditions in the generic model.

More concretely, we formalize quite general classes of computational problems over *known order* groups, we call DL-type and DH-type problems. Numerous cryptographic problems currently known are covered by these classes and, due to their broadness, we expect that many future computational problems will also comply with them. A considerable list of covered problems is given in Section 2. In particular, we also support problems like the w-strong Bilinear Diffie-Hellman problem [BB04b] where an algorithm is asked to output an element being described by a rational function.

We show the existence of a set of conditions which, in the generic group model, are *sufficient* for the hardness of DL-type and DH-type problems. An important issue to point out is that we do not just consider the basic generic group model, i.e., where an oracle provides operations to apply the group law and compute inverses. Rather, we consider a broader and more flexible variant where one can additionally make available a rich class of operations. Examples of such operations are multilinear mappings, arbitrary homomorphisms between groups and also oracles solving other computational or decisional problems. Correspondingly, our conditions not only consider the computational problem at hand but also explicitly the operations that are available. We were able to derive non-trivial (generic) bounds for all problems and sets of operations satisfying our conditions.

In this context, we explore and formalize, for the first time, the relationship of generic algorithms and so-called straight-line programs (SLPs) which are non-probabilistic algorithms performing a static sequence of operations on their inputs without branching or looping. The concept of SLPs is widely-used in computational algebra and also proved its usefulness in the area of cryptography (see, e.g., [BV98,Bro06]). Its relation to generic algorithms has not been explicitly analyzed before.

1.2 Related Work

In [Kil01] the author analyzes a generalization of the Diffie-Hellman problem, the P-Diffie-Hellman problem: given group elements (g, g^{x_1}, g^{x_2}) the challenge is to compute $g^{P(x_1,x_2)}$, where P is a (non-linear) polynomial and g is a generator of some group G. Besides other results, it is shown there that the computational and decisional variant of this problem class is hard in the (standard) generic model.

Recently, another general problem class has been introduced in [BBG05b] to cover DH related problems over bilinear groups. The authors show that decisional problems belonging to this class which satisfy a certain condition are hard in a generic bilinear group setting.

An interesting work by Maurer [Mau05] presents an abstract model of computation, which generalizes the generic model. This model captures so-called extraction, computation and distinction problems. In contrast to our framework, this model does not cover the case of multiple groups. Maurer proves lower bounds on the computation complexity for several specific examples within this setting, however the paper does not give sufficient nor necessary conditions for the hardness of problem classes in the framework.

Recent work by Bresson et al. [BLMW07] independently analyzes generalized decisional problems over a single prime order group in the plain model. They showed that under several restrictions a so-called (P, Q)-DDH problem is efficiently reducible to the standard DDH problem. This result has also some consequences for the hardness of (P, Q)-CDH problems in the generic group model: since standard DDH is known to be hard in the generic model also a (P, Q)-DDH problem (over a single prime order group) satisfying the restrictions is hard and thus the respective computational version of this problem. However, one important requirement for applying their results is that the P and Q polynomials describing the problem need to be *power-free*, i.e., variables are only allowed to occur with exponents being equal to zero or one. This is not a necessary condition for a computational problem to be hard in the generic model (e.g., consider the square exponent problem) and excludes a large class of problems.

To summarize, the works cited above consider quite restricted classes of cryptographic problems and/or fixed algebraic settings. For instance, the w-strong family of problems is covered by neither approach. Our framework is much more flexible: we provide sufficient hardness conditions for large classes of DL- and DH-type problems defined over multiple groups that are not necessarily of prime order. Moreover, in contrast to existing solutions our conditions take variable sets of operations into account.

1.3 Basic Notation

Let $\mathsf{poly}(x)$ denote the class of univariate polynomials in x with non-negative integer coefficients. We call a function $f : \mathbb{N} \rightarrow \mathbb{R}^+$ *polynomial bounded* if there exists $poly \in \mathsf{poly}(x)$ s.t. for all κ: $f(\kappa) \leq poly(\kappa)$. We call a function f *negligible* if the inverse of any $poly \in \mathsf{poly}(x)$ is asymptotically an upper bound of f, i.e., $\forall poly \in \mathsf{poly}(x)\, \exists \kappa_0\, \forall \kappa \geq \kappa_0 : f(\kappa) \leq \frac{1}{poly(\kappa)}$.

Throughout the paper we are concerned with multivariate *Laurent polynomials* over the ring $\mathbb{Z}_n$. Informally speaking, Laurent polynomials are polynomials whose variables may also have negative exponents. More precisely, a Laurent polynomial P over $\mathbb{Z}_n$ in indeterminates $X_1, \ldots, X_\ell$ is a finite sum $P = \sum a_{\alpha_1,\ldots,\alpha_\ell} X_1^{\alpha_1} \cdots X_\ell^{\alpha_\ell}$ where $a_{\alpha_1,\ldots,\alpha_l} \in \mathbb{Z}_n$ and $\alpha_i \in \mathbb{Z}$. The set of Laurent polynomials over $\mathbb{Z}_n$ forms a ring with the usual addition and multiplication (where the identity $X_j^{-\ell} X_j^{\ell} = 1$ holds). By $\deg(P) = \max\{\sum_i |\alpha_i| \mid a_{\alpha_1,\ldots,\alpha_l} \not\equiv 0 \bmod n\}$ (and $\deg(0) = -\infty$) we denote the *(absolute) total degree* of a Laurent polynomial $P \neq 0$. Furthermore, we denote by $\mathfrak{L}_n^{(\ell,c)}$ (where $0 \leq c \leq l$) the subring of Laurent polynomials over $\mathbb{Z}_n$ where only the variables $X_{c+1}, \ldots, X_\ell$ can appear with negative exponents. Note that for any $P \in \mathfrak{L}_n^{(\ell,c)}$ and $\mathbf{x} = (x_1, \ldots, x_\ell) \in \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c}$ the evaluation $P(\mathbf{x})$, where we set $X_1 = x_1, \ldots, X_\ell = x_\ell$ in P, is well-defined.

If $\mathcal{A}$ is a probabilistic algorithm, then $y \overset{\mathrm{R}}{\leftarrow} \mathcal{A}(x)$ denotes the assignment to y of the output of $\mathcal{A}$'s run on x with fresh random coins; we write $y \leftarrow \mathcal{A}(x)$ if $\mathcal{A}$ is deterministic. Furthermore, by $[\mathcal{A}(x)]$ we denote the set of all possible outputs of a probabilistic algorithm $\mathcal{A}$ on input of a fixed value x and variable random coins. If S is a set, then $x \overset{\mathrm{R}}{\leftarrow} S$ denotes the random generation of an element $x \in S$ using the uniform distribution.

1.4 Outline

In Section 2 we formalize the classes of computational problems under consideration. Section 3 introduces our extended generic group model based on Shoup's model. In particular, we define the notion of a generic operation set as well as a problem's intractability against generic algorithms only allowed to perform operations from this set. Section 4 presents abstract hardness conditions linking the concepts of generic algorithms and straight-line programs (SLPs). Loosely speaking, we show that a problem is intractable with respect to generic algorithms if (i) not too much information about the secret choices is leaked due to equality relations between computable group elements (leak-resistance) and (ii) the considered problem is not solvable in a trivial way (SLP-intractability). In Section 5 we introduce easily checkable conditions ensuring leak-resistance and SLP-intractability. Additionally, concrete bounds for these properties are given. Finally, Section 6 extends our framework in different directions. We describe how the conditions can be used to prove the non-existence of generic reductions thereby adding operations implementing decision oracles to the framework. Furthermore, we extend our basic problem class definition from Section 2 as well as our conditions in order to cover the w-strong problem family.

2 Problem Classes

In this section we formally define the classes of computational problems under consideration. In order to get the key ideas of our formalization, we start with an informal and simplified description of an important class of computational problems over a single (known order) group. Such problems often comply to the following general form: Let G be a cyclic group with corresponding generator g. Then given a tuple $(a_1, \ldots, a_z)$ of (public) elements from G which are related to some secret random choices $\mathbf{x} = (x_1, \ldots, x_\ell) \overset{\text{R}}{\leftarrow} \mathbb{Z}_{|G|}^\ell$, the challenge is to compute some element $a \in G$ (*DH-type problem*) or some discrete logarithm $x \in \mathbb{Z}_{|G|}$ (*DL-type problem*) which is also related to $\mathbf{x}$. The relation between $\mathbf{x}$ and a public element a_j belonging to a problem instance can be represented by a function of the form

$$f : \mathbb{Z}_{|G|}^\ell \to \mathbb{Z}_{|G|} \, .$$

That means, we have $a_j = g^{f(\mathbf{x})}$. Similarly, the relation between the solution $a \in G$ resp. $x \in \mathbb{Z}_{|G|}$ and $\mathbf{x}$ can be specified by such a function. In this work, we restrict ourselves to functions given by Laurent polynomials. As an example, consider the description of the CDH problem: Given $(a_1 = g^1, a_2 = g^{x_1}, a_2 = g^{x_2}) \in G^3$, where $\mathbf{x} = (x_1, x_2) \in_U \mathbb{Z}_{|G|}^2$ the challenge is to compute the element $a = g^{x_1 x_2}$. Thus, the relational structure between $\mathbf{x}$ and the public group elements can be captured by polynomials $1, X_1, X_2$. The relation to the solution is given by the polynomial $X_1 X_2$.

Definition 1 generalizes this approach to the case of multiple groups and mixed secret choices over $\mathbb{Z}_n$ and $\mathbb{Z}_n^*$. The former is required to include the important class of problems over bilinear (and more generally, multilinear) groups while the latter is necessary to include the class of so-called exponent inversion problems. For our formalization we adapted and extended the framwework in [SS01].

Definition 1 (DL-/DH-type problem). *A* DL-/DH-type problem $\mathcal{P}$ *is characterized by*

- *A tuple of parameters*
$$Param_{\mathcal{P}} = (k, \ell, c, z)$$
consisting of some constants $k, \ell \in \mathbb{N}, c \in \mathbb{N}_0$ *where* $c \leq \ell$ *and a polynomial* $z \in \mathsf{poly}(x)$*.*
- *A structure instance generator* $\mathsf{SIGen}_{\mathcal{P}}(\kappa)$ *that on input of a security parameter* κ *outputs a tuple of the form*
$$((\mathbf{G}, \mathbf{g}, n), (\mathbf{I}, Q)) \, ,$$
where

- $(\mathbf{G}, \mathbf{g}, n)$ *denotes the* algebraic structure instance *consisting of descriptions of cyclic groups* $\mathbf{G} = (G_1, \ldots, G_k)$ *of order* n *and corresponding generators* $\mathbf{g} = (g_1, \ldots, g_k)$*,*
- $(\mathbf{I}, Q)$ *denotes the* relation structure instance *consisting of the* input polynomials $\mathbf{I} = (\mathbf{I_1}, \ldots, \mathbf{I_k})$*, with* $\mathbf{I_i} \subset \mathfrak{L}_n^{(\ell,c)}$*,* $|\mathbf{I_i}| \leq z(\kappa)$*, and the* challenge polynomial $Q \in \mathfrak{L}_n^{(\ell,c)}$*.*

Then a problem instance *of* $\mathcal{P}$ *consists of a structure instance* $((\mathbf{G}, \mathbf{g}, n), (\mathbf{I}, Q)) \overset{R}{\leftarrow} \mathsf{SIGen}_{\mathcal{P}}(\kappa)$ *as well as group elements*

$$\mathbf{g}^{\mathbf{I}(\mathbf{x})} = \left(g_i^{P(\mathbf{x})} | P \in \mathbf{I_i}, 1 \leq i \leq k \right),$$

where $\mathbf{x} \overset{R}{\leftarrow} \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c}$ *are secret values. Given such a problem instance, the challenge is to compute*

$$\begin{cases} Q(\mathbf{x}), & \textit{for a DL-type problem} \\ g_1^{Q(\mathbf{x})}, & \textit{for a DH-type problem} \end{cases}.$$

Numerous computational problems of cryptographic interest fall into the class of DL-type or DH-type problems. Examples are problems such as the DLP, DHP, a variant of the representation problem [SS01], generalized DHP [SS01], GAP-DHP [OP01], square and inverse exponent problem [SS01], divisible DHP [BDZ03], w-weak and w-strong DHP [Che06], bilinear DHP [BF01], w-bilinear DH inversion problem [BB04a], w-bilinear DH exponent problem [BBG05a], co-DHP [BGLS03], co-bilinear DHP [BF01], bilinear pairing inversion problem [Yac02] and many more. In Section 6.2 we extend our definitions and conditions to also include problems like the w-strong DH and w-strong BDH problem where the challenge is specified by a *rational function.* As an illustration of the definition, we consider several problems in more detail.

Example 1 (Discrete Logarithm Problem). For the DL problem we have parameters

$$Param_{\text{DL}} = (1, 1, 1, 2)$$

and a structure instance generator $\mathsf{SIGen}_{\text{DL}}$ that on input κ returns

$$((\mathbf{G} = G_1, \mathbf{g} = g_1, n), (\mathbf{I} = \mathbf{I_1} = \{1, X_1\}, Q = X_1)) .$$

Note that we have the "same" input polynomials and challenge polynomial for all κ and random coins of $\mathsf{SIGen}_{\text{DL}}$. A problem instance of the DL problem additionally comprises group elements

$$\left(g_1^{P(\mathbf{x})} | P \in \{1, X_1\} \right) = (g_1, g_1^{x_1}),$$

where $\mathbf{x} = x_1 \overset{\text{R}}{\leftarrow} \mathbb{Z}_n$, and the task is to compute $Q(\mathbf{x}) = x_1$.

Example 2 (Computational Diffie-Hellman Problem). For the DH problem we have parameters

$$Param_{\text{DH}} = (1, 2, 2, 3)$$

and a structure instance generator $\mathsf{SIGen}_{\text{DH}}$ that on input κ returns

$$((\mathbf{G} = G_1, \mathbf{g} = g_1, n), (\mathbf{I} = \mathbf{I_1} = \{1, X_1, X_2\}, Q = X_1 X_2)) .$$

Again, we always have the "same" input polynomials and challenge polynomial. A problem instance additionally comprises group elements

$$\left(g_1^{P(\mathbf{x})} | P \in \{1, X_1, X_2\} \right) = (g_1, g_1^{x_1}, g_1^{x_2}),$$

where $\mathbf{x} = (x_1, x_2) \overset{\text{R}}{\leftarrow} \mathbb{Z}_n^2$, and the task is to compute $g_1^{Q(\mathbf{x})} = g_1^{x_1 x_2}$.

Example 3 (w-Bilinear Diffie-Hellman Inversion Problem). For the w-BDHI problem we have parameters

$$Param_{w\text{-BDHI}} = (3, 1, 0, w + 1)$$

and a structure instance generator $\mathsf{SIGen}_{w\text{-BDHI}}$ that on input κ returns

$$((\mathbf{G} = (G_1, G_2, G_3), \mathbf{g} = (g_1, g_2, g_3), p), (\mathbf{I} = (\mathbf{I_1} = \{1\}, \mathbf{I_2} = \{1, X_1, X_1^2, \dots, X_1^{w(\kappa)}\}, \mathbf{I_3} = \{1\}), Q = X_1^{-1}))$$

such that p is a prime, there exists a non-degenerated, efficiently computable bilinear mapping $e : G_2 \times G_3 \to G_1$ with $e(g_2, g_3) = g_1$ and an isomorphism $\psi : G_2 \to G_3$ with $\psi(g_2) = g_3$). Note that the number of input polynomials in $\mathbf{I_2}$ grows with the security parameter. A problem instance additionally comprises group elements

$$\left(g_i^{P(\mathbf{x})} | P \in \mathbf{I_i}, 1 \le i \le 3\right) = \left(g_1, g_2, g_2^{x_1}, \dots, g_2^{x_1^{w(\kappa)}}, g_3\right),$$

where $\mathbf{x} = x_1 \overset{R}{\leftarrow} \mathbb{Z}_n^*$, and the task is to compute $g_1^{Q(\mathbf{x})} = g_1^{x_1^{-1}}$.

In the remainder of this paper, we are often only interested in individual parts of the output of $\mathsf{SIGen}_{\mathcal{P}}$. To this end, we introduce the following simplifying notation: By $\$ \overset{R}{\leftarrow} \mathsf{SIGen}_{\mathcal{P}}^{\$}(\kappa)$, where $\$$ is a wildcard character, we denote the projection of SIGen's output to the part $\$$. For instance, $(n, \mathbf{I}, Q) \overset{R}{\leftarrow} \mathsf{SIGen}_{\mathcal{P}}^{(n,\mathbf{I},Q)}(\kappa)$ denotes the projection of the output to the triple consisting of the group order, the input polynomials, and the challenge polynomial. Furthermore, by $[\mathsf{SIGen}_{\mathcal{P}}^{\$}(\kappa)]$ we denote the set of all possible outputs $\$$ for a given fixed security parameter κ.[1]

3 Extending Shoup's Generic Group Model

We need a slightly more flexible definition of the generic group model than usual since we consider large classes of problems with varying *algebraic setting* which refers to the underlying algebraic groups and operations available on these groups. Thus, also the generic group setting needs to take the varying number of groups and their structures into account. To make the results in the generic model as strong as possible the operations made available through the generic oracle should closely correspond to those of the algebraic setting in the plain model. That means the generic algorithms considered in our model should be allowed to apply all operations that are known to be efficiently computable in this algebraic setting and do not make intrinsic use of the encoding of group elements.

3.1 Generic Operations

Given a problem and its underlying algebraic setting, we call the operations being made available in the generic group model the (*generic*) *operation set* of the problem. In the following we formally define the notion of an operation set and describe what operations can be part of it.

For our framework we restrict to consider operations of the form

$$\begin{gathered} \circ : G_{s_1} \times \ldots \times G_{s_u} \to G_d \\ (a_1, \dots, a_u) \mapsto \circ(a_1, \dots, a_u) \end{gathered}$$

where $u \ge 1$, $s_1, \dots, s_u, d \in \{1, \dots, k\}$ are some fixed constants (that does not depend on κ). Furthermore, we demand that the action of $\circ$ on the group elements can be represented by a fixed

[1] In [SS01], the authors call this kind of sets "siblings".

regular polynomial. That means, there exists a fixed $F \in \mathbb{Z}[Y_1, \ldots, Y_u]$ (also not depending on κ) such that for any generators $g_1, \ldots, g_u, g_d$ given as part of a problem instance hold

$$\circ(a_1, \ldots, a_u) = g_d^{F(y_1,\ldots,y_u)}$$

where $a_1 = g_1^{y_1}, \ldots, a_u = g_u^{y_u}$. For instance, the bilinear mapping $e : G_2 \times G_3 \to G_1$ which is part of the algebraic setting of the w-BDHIP is such an operation: for any g_2, g_3 and $g_1 = e(g_2, g_3)$ it holds that $e(a_1, a_2) = e(g_2^{y_1}, g_3^{y_2}) = g_1^{F(y_1,y_2)}$ where $F = Y_1Y_2$. In fact, to the best of our knowledge, virtually any deterministic operation considered in the context of the generic group model in the literature so far belongs to this class of operations. In Section 6.1 we extend the operation set to additionally include decision oracles.

We represent an operation of the above form by a tuple $(\circ, s_1, \ldots, s_u, d, F)$, where the first component is a symbol serving as a unique identifier of the operation. The set of allowed operations can thus be specified by a set of such tuples as done in Definition 2. In the following we may assume that an operation set always contains at least operations for performing the group law and inversion of group elements (for each group).

Definition 2 (Operation Set). *An operation set Ω is a finite set of tuples of the form $(\circ, s_1, \ldots, s_u, d, F)$, where the components of these tuples are defined as above.*

Example 4 (Operation Set for w-BDHIP). The operation set $\Omega = \{(\circ_1, 1, 1, 1, Y_1 + Y_2), (\circ_2, 2, 2, 2, Y_1 + Y_2), (\circ_3, 3, 3, 3, Y_1 + Y_2), (inv_1, 1, 1, -Y_1), (inv_2, 2, 2, -Y_1), (inv_3, 3, 3, -Y_1), (\psi, 2, 3, Y_1), (e, 2, 3, 1, Y_1 \cdot Y_2)\}$ specifies operations for performing the group law ($\circ_i$) and inversion (inv_i) over each group as well as the isomorphism $\psi : G_2 \to G_3$ and the bilinear map $e : G_2 \times G_3 \to G_1$.

3.2 Generic Group Algorithms and Intractability

In this section, we formally model the notion of generic (aka. black-box) algorithms for our setting. We adapt Shoup's standard generic group model [Sho97b] for this purpose.

Let $S_n \subset \{0,1\}^{\lceil \log_2(n) \rceil}$ denote a set of bit strings of cardinality n and Σ_n the set of all bijective functions from $\mathbb{Z}_n$ to S_n. Furthermore, let $\sigma = (\sigma_1, \ldots, \sigma_k) \in \Sigma_n^k$ be a k-tuple of randomly chosen encoding functions for the groups $G_1, \ldots, G_k \cong \mathbb{Z}_n$, i.e., for each $1 \leq i \leq k$

$$\sigma_i : \mathbb{Z}_n \to S_n$$

is a bijective encoding function assigning elements from $\mathbb{Z}_n$ to bit strings, chosen at random among all possible bijections.

A *generic algorithm* $\mathcal{A}$ in our setting is a probabilistic algorithm that is given access to a *generic (multi-) group oracle* $\mathcal{O}_\Omega$ allowing $\mathcal{A}$ to perform operations from Ω on encoded group elements. Since any cyclic group of order n is isomorphic to $(\mathbb{Z}_n, +)$, we will always use $\mathbb{Z}_n$ with generator 1 for the internal representation of a group G_i. As internal state $\mathcal{O}_\Omega$ maintains two types of lists namely *element lists* $L_1, \ldots, L_k$, where a $L_i \subset \mathfrak{L}_n^{(\ell,c)}$, and *encoding lists* $E_1, \ldots, E_k$, where $E_i \subset S_n$. For an index j let $L_{i,j}$ and $E_{i,j}$ denote the j-th entry of L_i and E_i, respectively. Each list L_i is initially populated with the corresponding input polynomials given as part of a problem instance of a DL-/DH-type problem $\mathcal{P}$, i.e., $L_i = (P | P \in \mathbf{I_i})$ (and the polynomials P are taken from $\mathbf{I_i}$ in some fixed publicly known order). A list E_i contains the encodings of the group elements corresponding to the entries of L_i, i.e., $E_{i,j} = \sigma_i(L_{i,j}(\mathbf{x}))$. E_i is initialized with $E_i = (\sigma_i(P(\mathbf{x})) | P \in \mathbf{I_i})$. $\mathcal{A}$ is given (read) access to all encodings lists. In order to be able to perform operations on the randomly encoded elements, the algorithm may query $\mathcal{O}_\Omega$. Let $(\circ, s_1, \ldots, s_u, d, F)$ be an operation from Ω. Upon receiving a query $(\circ, j_1, \ldots, j_u)$, where $j_1, \ldots j_u$ are pointers into the encoding lists $E_{j_1}, \ldots E_{j_u}$, the oracle computes $P := F(L_{s_1,j_1}, \ldots, L_{s_u,j_u})$, appends P to L_d and $\sigma_d(P(\mathbf{x}))$ to the encoding list E_d. After having issued a number of queries, $\mathcal{A}$ eventually provides its final output. In the case that $\mathcal{P}$ is a DL-type problem, we say that $\mathcal{A}$ has solved the problem instance of $\mathcal{P}$ if its output a satisfies $Q(\mathbf{x}) - a \equiv 0 \bmod n$. In the case that $\mathcal{P}$ is a DH-type problem, $\mathcal{A}$ has solved the problem instance if for its output $\sigma_1(a)$ holds that $Q(\mathbf{x}) - a \equiv 0 \bmod n$.

Let a DL-/DH-type problem over cyclic groups $G_1, \ldots, G_k$ of order n be given. We can write the group order as $n = p^e \cdot s$ with $\gcd(p, s) = 1$ where p be the largest prime factor of n. Then for each i it holds that

$$G_i \cong G_i^{(p^e)} \times G_i^{(s)}$$

where $G_i^{(p^e)}$ and $G_i^{(s)}$ are cyclic groups of order p^e and s, respectively. It is easy to see that solving an instance of a DL-/DH-type over groups G_i of order n means for a generic algorithm to solve it *separately* over the subgroups $G_i^{(p^e)}$ *and* the subgroups $G_i^{(s)}$. In particular, the inputs, the generic operations, and a solution over $G_i^{(s)}$ are of no help to find a solution over $G_i^{(p^e)}$. Thus, computing a solution to a DL-type or DH-type problem over the groups G_i is a least as hard for generic algorithms as computing a solution over groups the $G_i^{(p^e)}$ given only inputs over these subgroups. Hence, to assess the intractability of DL-/DH-type problems we can restrict to consider these problems over groups of prime power order. In the following we always assume that $\mathsf{SIGen}_{\mathcal{P}}$ on input κ generates groups of prime power order $n = p^e$ with $p > 2^\kappa$ and $e > 0$ as well as challenge and input polynomials over $\mathbb{Z}_{p^e}$.

Our notion of a problem's intractability against generic algorithms allowed to perform certain operations Ω is formalized in Definition 4 and 5, respectively. Note that the two definitions only differ with respect to the output of the generic algorithm. In our formalizations we consider classes of generic algorithms exhibiting the same runtime where runtime is measured by the number of oracle queries. This class is captured by Definition 3.

Definition 3 (q-GGA). *A q-GGA $\mathcal{A}$ is a generic group algorithm that for any $\kappa \in \mathbb{N}$, which it receives as part of its input, makes at most $q(\kappa)$ queries to the generic group oracle.*

Definition 4 (GGA-intractability of DL-type Problems). *A DL-type problem $\mathcal{P}$ is called (Ω, q, ν)-GGA-intractable if for all q-GGA $\mathcal{A}$ and $\kappa \in \mathbb{N}$ we have*

$$\begin{aligned}
\Pr[Q(\mathbf{x}) \equiv a \bmod n \ :& \\
&(n, \mathbf{I}, Q) \xleftarrow{R} \mathsf{SIGen}_{\mathcal{P}}^{(n,\mathbf{I},Q)}(\kappa);\\
&\sigma \xleftarrow{R} \Sigma_n^k;\\
&\mathbf{x} \xleftarrow{R} \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c};\\
&a \xleftarrow{R} \mathcal{A}^{\mathcal{O}_\Omega}(\kappa, n, \mathbf{I}, Q, (\sigma_1(P(\mathbf{x}))|P \in \mathbf{I_1}), \ldots, (\sigma_k(P(\mathbf{x}))|P \in \mathbf{I_k}))\\
] \le \nu(\kappa)&
\end{aligned}$$

Definition 5 (GGA-intractability of DH-type Problems). *A DH-type problem $\mathcal{P}$ is called (Ω, q, ν)-GGA-intractable if for all q-GGA $\mathcal{A}$ and $\kappa \in \mathbb{N}$ we have*

$$\begin{aligned}
\Pr[Q(\mathbf{x}) \equiv a \bmod n \ :& \\
&(n, \mathbf{I}, Q) \xleftarrow{R} \mathsf{SIGen}_{\mathcal{P}}^{(n,\mathbf{I},Q)}(\kappa);\\
&\sigma \xleftarrow{R} \Sigma_n^k;\\
&\mathbf{x} \xleftarrow{R} \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c};\\
&\sigma_1(a) \xleftarrow{R} \mathcal{A}^{\mathcal{O}_\Omega}(\kappa, n, \mathbf{I}, Q, (\sigma_1(P(\mathbf{x}))|P \in \mathbf{I_1}), \ldots, (\sigma_k(P(\mathbf{x}))|P \in \mathbf{I_k}))\\
] \le \nu(\kappa)&
\end{aligned}$$

4 Abstract Hardness Conditions: Linking GGA and SLP Intractability

Informally speaking, we will see that the grade of intractability of a DL-/DH-type problem with respect to generic algorithms can be "measured" by means of two "quantities":

1. The probability of gaining information about the secret choices $\mathbf{x}$ in the course of a computation by means of non-trivial equalities between group elements. This quantity is called *leak-resistance*.
2. The probability to solve problem instances using a trivial strategy, i.e., by taking actions independently of (in)equalities of computed group elements and thus independent of the specific problem instance. This quantity is called *SLP-intractability*.

For formalizing both quantities we make use of (a generalized form of) so-called straight-line program (SLP) generators specified by Definition 6. SLPs are a very common concept in the field of computational algebra that has also proven its usefulness in the area of cryptography (e.g., see [BV98,Bro06]). However, the SLP model and the GGA model have not been explicitly related in the literature so far. It is important to note that an SLP-generator is *not* given any group elements (which are part of a problem instance) as input but is only aware of relation structure instance in terms of certain Laurent polynomials.

Definition 6 ((Ω, q)-SLP-generator). *A (Ω, q)-SLP-generator $\mathcal{S}$ is a probabilistic algorithm that on input $(\kappa, n, \mathbf{I}, Q)$ where $\kappa, n \in \mathbb{N}$, $\mathbf{I} = (\mathbf{I_1}, \dots, \mathbf{I_k})$ with $\mathbf{I_i} \subset \mathfrak{L}_n^{(\ell,c)}$, and $Q \in \mathfrak{L}_n^{(\ell,c)}$, outputs lists $(L_1, \dots, L_k)$ where $L_i \subset \mathfrak{L}_n^{(\ell,c)}$. Each list L_i is initially populated with $L_i = (P | P \in \mathbf{I_i})$. The algorithm can append a polynomial to a list by applying an operation from Ω to polynomials already contained in the lists, i.e., for an operation $(\circ, s_1, \dots, s_u, d, F) \in \Omega$ and existing polynomials $P_1 \in L_{s_1}, \dots, P_u \in L_{s_u}$ the algorithm can append $F(P_1, \dots, P_u)$ to L_d. In this way, the algorithm may add up to $q(\kappa)$ polynomials in total to the lists. Occasionally, we assume in the following that the algorithm additionally outputs an element $a \in \mathbb{Z}_n$ in the case of a DL-type problem and a polynomial $P \in L_1$ in the case of DH-type problem, respectively.*

Let us first formalize the leak-resistance of a problem. When do group elements actually leak information due to equality realtions? To see this, reconsider the definition of the generic oracle in Section 3.2 and observe that two encodings $E_{i,j}$ and $E_{i,j'}$ are equal if and only if the evaluation $L_{i,j} - L_{i,j'}(\mathbf{x})$ yields zero. However, it is clear that such an equality relation yields no information about particular choices $\mathbf{x}$ if it holds for *all* elements of $\mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c}$. Thus, denoting the ideal of $\mathfrak{L}_n^{(\ell,c)}$ containing all Laurent polynomials that are effectively zero over $\mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c}$ by

$$\mathcal{I}_n = \{P \in \mathfrak{L}_n^{(\ell,c)} \mid \forall \mathbf{x} \in \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c} : \; P(\mathbf{x}) \equiv 0 \bmod n\}$$

an equality yields no information at all if $(L_{i,j} - L_{i,j'}) \in \mathcal{I}_n$. Otherwise, a *non-trivial collision* occurred and $\mathcal{A}$ learns that $\mathbf{x}$ is a modular root of $L_{i,j} - L_{i,j'}$.

By Definition 7 we capture the chance that information about the secret choices $\mathbf{x}$ is leaked in the course of a computation due to non-trivial equalities between group elements. For this purpose we can make use of (Ω, q)-SLP-generators since they generate all possible sequences of polynomials that may occur as internal state of the generic oracle $\mathcal{O}_\Omega$ during an interaction with an q-GGA.

Definition 7 (Leak-resistance). *A DL-/DH-type problem $\mathcal{P}$ is called (Ω, q, ν)-leak-resistant if for all (Ω, q)-SLP-generators $\mathcal{S}$ and $\kappa \in \mathbb{N}$ we have*

$$\begin{aligned}
\Pr[\exists i \textit{ and } P, P' \in L_i \textit{ s.t. } (P - P')(\mathbf{x}) \equiv 0 \bmod n \,\wedge\, P - P' \notin \mathcal{I}_n \;:\\
&(n, \mathbf{I}, Q) \xleftarrow{R} \mathsf{SIGen}_{\mathcal{P}}^{(n,\mathbf{I},Q)}(\kappa);\\
&(L_1, \dots, L_k) \xleftarrow{R} \mathcal{S}(n, \kappa, \mathbf{I}, Q);\\
&\mathbf{x} \xleftarrow{R} \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c}\\
] \le \nu(\kappa)
\end{aligned}$$

Now assume that no information about $\mathbf{x}$ can be gained. In this case, we can restrict to consider algorithms applying trivial solution strategies to solve instances of a problem. That means, we can restrict our considerations to the subclass of generic algorithms that, when fixing all inputs except for the choice of $\mathbf{x}$, always apply the *same fixed sequence of operations* from Ω and provide the same output in order to solve an arbitrary problem instance. Thus, the algorithm actually acts as a straight-line program in this case. More precisely, assuming the absence of non-trivial collisions, we can show that a generic algorithm is essentially equivalent to an SLP-generator.

Definitions 8 and 10 formalize the chance of solving a DL-type or DH-type problem in such a trivial way. For DL-type problems this essentially means the chance to blindly guess $Q(\mathbf{x})$.

Definition 8 (SLP-intractability of DL-Type Problems). *A DL-type problem $\mathcal{P}$ is called (Ω, q, ν)-SLP-intractable if for all $\kappa \in \mathbb{N}$ we have*

$$\begin{aligned}\Pr[Q(\mathbf{x}) \equiv a \bmod n\ :\ & \\ & (n, \mathbf{I}, Q) \xleftarrow{R} \mathsf{SIGen}_{\mathcal{P}}^{(n,\mathbf{I},Q)}(\kappa); \\ & (a, L_1, \ldots, L_k) \xleftarrow{R} \mathcal{S}(\kappa, n, \mathbf{I}, Q); \\ & \mathbf{x} \xleftarrow{R} \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c} \\ ] \le \nu(\kappa) &\end{aligned}$$

Definition 9 formulates the seemingly stronger notion of ν-non-triviality that is independent of Ω and q-SLP-generators. Clearly, a DL-type problem $\mathcal{P}$ that is ν-non-trivial for some function ν is also (Ω, q, ν)-SLP-intractable for any Ω and any q. While Definition 8 reflects what is actually needed to prove GGA-intractability, our practical conditions presented in Section 5 ensures that a DL-type problem is ν-non-trivial where ν is a negligible function.

Definition 9 (Non-Triviality of DL-Type Problems). *A DL-type problem $\mathcal{P}$ is called ν-non-trivial if for all (computationally unbounded) algorithms Guess that on input $(\kappa, n, \mathbf{I}, Q)$ output some element $a \in \mathbb{Z}_n$ and for all $\kappa \in \mathbb{N}$ we have*

$$\begin{aligned}\Pr[Q(\mathbf{x}) \equiv a \bmod n\ :\ & \\ & (n, \mathbf{I}, Q) \xleftarrow{R} \mathsf{SIGen}_{\mathcal{P}}^{(n,\mathbf{I},Q)}(\kappa); \\ & a \xleftarrow{R} \mathit{Guess}(\kappa, n, \mathbf{I}, Q); \\ & \mathbf{x} \xleftarrow{R} \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c} \\ ] \le \nu(\kappa) &\end{aligned}$$

For DH-type problems, we consider the probability to obtain $Q(\mathbf{x})$ when evaluating the output P of an q-SLP-generator with $\mathbf{x} \xleftarrow{R} \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c}$. Remember that the SLP-generator cannot simply output Q that is given as input to him, but the output P must be generatable from $\mathbf{I}$ by only using (at most q) operations from Ω.

Definition 10 (SLP-intractability of DH-type Problems). *A DH-type problem $\mathcal{P}$ is called (Ω, q, ν)-SLP-intractable if for all (Ω, q)-SLP-generators $\mathcal{S}$ and all $\kappa \in \mathbb{N}$ we have*

$$\begin{aligned}\Pr[(P - Q)(\mathbf{x}) \equiv 0 \bmod n\ :\ & \\ & (n, \mathbf{I}, Q) \xleftarrow{R} \mathsf{SIGen}_{\mathcal{P}}^{(n,\mathbf{I},Q)}(\kappa); \\ & (P, L_1, \ldots, L_k) \xleftarrow{R} \mathcal{S}(\kappa, n, \mathbf{I}, Q); \\ & \mathbf{x} \xleftarrow{R} \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c} \\ ] \le \nu(\kappa) &\end{aligned}$$

Finally, Theorem 1 relates GGA-intractability to leak-resistance and SLP-intractability.

Theorem 1 (GGA-intractability of DL-/DH-type Problems). *If a DL-type problem is (Ω, q, ν_1)-leak-resistant and (Ω, q, ν_2)-SLP-intractable then it is $(\Omega, q, \nu_1 + \nu_2)$-GGA-intractable. If a DH-type problem is (Ω, q, ν_1)-leak-resistant and (Ω, q, ν_2)-SLP-intractable then it is $(\Omega, q, \frac{1}{2^\kappa-(q(\kappa)+z(\kappa))} + \nu_1 + \nu_2)$-GGA-intractable.*

Proof. We restrict to the more complex case of DH-type problems. The proof for DL-type problems works analogously.

In the following we assume a DH-type problem $\mathcal{P}$ according to Definition 1 and an operation set Ω according to Definition 2 are given. Considering a fixed operation set Ω, we denote the corresponding generic oracle simply by $\mathcal{O} = \mathcal{O}_\Omega$. Furthermore, we consider an arbitrary but fixed $\kappa \in N$ and output

$$(n, \mathbf{I}, Q) \overset{R}{\leftarrow} \mathsf{SIGen}_{\mathcal{P}}^{(n,\mathbf{I},Q)}(\kappa)$$

of the problem instance generator.

Let $\mathcal{A}$ be an arbitrary q-GGA. Let $\mathbf{S}$ denote the event that $\mathcal{A}$ on input

$$(\kappa, n, \mathbf{I}, Q, (\sigma_1(P(\mathbf{x}))|P \in \mathbf{I_1}), \ldots, (\sigma_k(P(\mathbf{x}))|P \in \mathbf{I_k})),$$

where $\mathbf{x} \overset{R}{\leftarrow} \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c}$ and $\sigma \overset{R}{\leftarrow} \Sigma_n^k$, outputs an encoding $s = \sigma_1(a)$ such that $Q(\mathbf{x}) - a \equiv 0 \bmod n$. We can split the success event $\mathbf{S}$ into the event $\mathbf{S}'$ that $\mathcal{A}$ wins with an encoding that occurred during computation, i.e., $s \in E_1$, and the event $\mathbf{S}''$ that $\mathcal{A}$ wins with a new encoding, i.e., $s \notin E_1$. Since a new encoding is associated with an uniformly random element from $\mathbb{Z}_n \setminus L_1$ we have

$$\Pr[\mathbf{S}''] \leq \frac{1}{n - (q(\kappa) + z(\kappa))} \leq \frac{1}{2^\kappa - (q(\kappa) + z(\kappa))}\,.$$

In the remainder of this proof we are concerned with bounding the probability of the event $\mathbf{S}'$. From now on we assume that $\mathcal{A}$ always outputs an encoding contained in E_1.

Introducing a Simulation Game. Starting from the real game we first introduce an intermediate game before we specify the actual simulation game.

An Equivalent Oracle. Let us define an equivalent oracle, denoted by $\mathcal{O}'$, by doing a slight change to the original oracle $\mathcal{O}$: instead of choosing the encoding functions σ up front, we define them just-in-time, i.e., we apply the well-known lazy sampling technique. More precisely, each time a new polynomial P should be appended to a list L_i the following computation is triggered to determine the encoding of $P(\mathbf{x})$: $\mathcal{O}'$ checks if there exists any index $1 \leq j \leq |L_i|$ such that

$$(P - L_{i,j})(\mathbf{x}) \equiv 0 \bmod n\,.$$

If this equation holds for some j, then the respective encoding $E_{i,j}$ is appended to E_i again. Otherwise, the oracle chooses a new encoding $s \overset{R}{\leftarrow} S_n \setminus E_i$ and appends it to E_i. It is easy to see that $\mathcal{O}'$ and $\mathcal{O}$ are equivalent, i.e., their behavior is perfectly indistinguishable.

The definition of success in this intermediate game is adapted accordingly: Let $\mathbf{S}^*$ denote the event that $\mathcal{A}$ outputs an encoding $E_{1,j}$ such that

$$(Q - L_{1,j})(\mathbf{x}) \equiv 0 \bmod n\,.$$

Certainly, we have $\Pr[\mathbf{S}^*] = \Pr[\mathbf{S}']$.

Remark 1. Note that the output encoding can clearly occur several times in E_1 and thus is associated with several polynomials in L_1. However, by the definition of the element encoding procedure, all of these polynomials evaluate to the same value. Hence, for the definition of success it does not matter which of these polynomials is considered.

The Simulation Oracle. Now we replace $\mathcal{O}'$ by a simulation oracle $\mathcal{O}_{\mathsf{sim}}$. The simulation oracle behaves exactly like $\mathcal{O}'$ except for the encodings of elements in order to be independent of $\mathbf{x}$: Each time a new polynomial P should be appended to some list L_i, $\mathcal{O}_{\mathsf{sim}}$ checks whether there exists an index $1 \leq j \leq |L_i|$ such that

$$P - L_{i,j} \in \mathcal{I}_n \tag{1}$$

In other words, it checks if L_i contains a polynomial $L_{i,j}$ that evaluates to the same value as P for *all* possible choices and thus describes the same function as P. If this is the case, the encoding $E_{i,j}$ is appended to E_i again, where j is the smallest index for which Equation (1) is satisfied. Observe that if this equation is satisfied for two different indices j and j' then we also have $E_{i,j} = E_{i,j'}$. Thus, actually it does not matter which of the corresponding encodings is taken if more than one polynomial matches. If Equation (1) cannot be satisfied, the $\mathcal{O}_{\mathsf{sim}}$ chooses a new encoding $s \overset{\mathrm{R}}{\leftarrow} S_n \backslash E_i$ and appends it to E_i.

In this way, the determined encoding and thus the behavior of $\mathcal{O}_{\mathsf{sim}}$ is independent of the particular secret choices $\mathbf{x}$ given as input to it. Obviously, the applied modification also leads to a behavior differing from that of $\mathcal{O}'$ in the case that there exists an index i and $P, P' \in L_i$ such that

$$(P - P')(\mathbf{x}) \equiv 0 \bmod n \ \wedge \ P - P' \notin \mathcal{I}_n \ .$$

We denote this simulation failure event by $\mathbf{F}$. Note that if and only if $\mathbf{F}$ occurs, the simulation oracle computes different encodings for the same group element. The success event in the simulation game is defined as before. We denote this event by $\mathbf{S}_{\mathsf{sim}}$.

Let us consider the oracles $\mathcal{O}'$ and $\mathcal{O}_{\mathsf{sim}}$ as deterministic Turing machines and assume both receive the same inputs and random strings. Furthermore, consider the algorithm $\mathcal{A}$ as a deterministic Turing machine with identical input and random tape when interacting with both oracles. Assuming that $\mathbf{F}$ does not occur, the algorithm receives the same sequence of encodings and thus issues the same sequence of queries (leading to the same lists of Laurent polynomials maintained by $\mathcal{O}'$ and $\mathcal{O}_{\mathsf{sim}}$) in both games. Furthermore, it outputs the same encoding in the end of both games. Hence, we have the following relation between the considered events:

$$\mathbf{S}^* \wedge \neg \mathbf{F} \iff \mathbf{S}_{\mathsf{sim}} \wedge \neg \mathbf{F}$$

That means, both games proceed identically unless a simulation failure occurs. Using this relation we immediately obtain the desired upper bound on the success probability in the original game in terms of the success probability and the failure probability in the simulation game:

$$\begin{aligned}
\Pr[\mathbf{S}] &= \Pr[\mathbf{S}''] + \Pr[\mathbf{S}'] \\
&\leq \frac{1}{2^\kappa - (q(\kappa) + z(\kappa))} + \Pr[\mathbf{S}'] \\
&= \frac{1}{2^\kappa - (q(\kappa) + z(\kappa))} + \Pr[\mathbf{S}^*] \\
&\leq \frac{1}{2^\kappa - (q(\kappa) + z(\kappa))} + \Pr[\mathbf{S}_{\mathsf{sim}}] + \Pr[\mathbf{F}]
\end{aligned}$$

The last inequality follows from applying the Difference Lemma [Sho06].

Bounding Probabilities in the Simulation Game. Remember that the behavior of $\mathcal{O}_{\mathsf{sim}}$ is independent of the secret $\mathbf{x}$. That means, assuming fixed random coins for $\mathcal{O}_{\mathsf{sim}}$ and $\mathcal{A}$, the simulation game proceeds *identically for all possible choices of* $\mathbf{x} \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell - c}$. So the lists of polynomials $L_1, \ldots, L_k$ maintained by $\mathcal{O}_{\mathsf{sim}}$ stay the same and $\mathcal{A}$ outputs the same encoding $E_{1,j}$ for any $\mathbf{x}$. In other words, $\mathcal{A}$ computes a straight-line program in this case. The choice of $\mathcal{A}$'s random tape can therefore be thought of as selecting an SLP from a set of alternatives. Hence, assuming arbitrary but fixed random coins only for $\mathcal{O}_{\mathsf{sim}}$, $\mathcal{A}$ is equivalent to some (Ω, q)-SLP-generator. This is the key observation for bounding the success and failure probability in the simulation game as done in the following.

Provided that the considered DH-type problem $\mathcal{P}$ is (Ω, q, ν_1)-leak-resistant there is a common function ν_1 bounding the probability of any (Ω, q)-SLP-generator in computing two polynomials $P, P' \in L_i$ (for arbitrary $1 \leq i \leq k$) such that

$$(P - P')(\mathbf{x}) \equiv 0 \bmod n \ \wedge \ P - P' \notin \mathcal{I}_n \ .$$

From this it follows immediately that ν_1 also upper bounds the probability of a simulation failure during an interaction of the q-GGA $\mathcal{A}$ and the simulation oracle $\mathbf{S}_{\mathsf{sim}}$, i.e.,

$$\Pr[\mathbf{F}] \leq \nu_1(\kappa) \, .$$

Similarly, assume that $\mathcal{P}$ is (Ω, q, ν_2)-SLP-intractable and let $(P, L_1, \ldots, L_k)$ be the output of an arbitrary (Ω, q)-SLP-generator. Then we know that the probability of $P \in L_1$ in colliding with the challenge polynomial, i.e.,

$$(Q - P)(\mathbf{x}) \equiv 0 \bmod n \, ,$$

for $\mathbf{x} \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell - c}$ is upper bounded by ν_2. Thus, it follows that also the probability of $\mathcal{A}$ in finding an encoding $E_{1,j}$ such that $L_{1,j}$ collides with Q is bounded by this function, i.e.,

$$\Pr[\mathbf{S}_{\mathsf{sim}}] \leq \nu_2(\kappa) \, .$$

This concludes the proof of our theorem. □

5 Practical Conditions

In this section, we present easily checkable conditions ensuring that a DH-type (DL-type) problem is (Ω, ν_1, q)-leak-resistant and (Ω, ν_2, q)-SLP-intractable (ν_2-SLP-intractable) with q being polynomial and ν_1 and ν_2 being negligible functions in the security parameter. Reviewing the corresponding definitions, we see that the probabilities ν_1 and ν_2 are closely related to the probability of randomly picking roots of certain multivariate Laurent polynomials. Section 5.1 shows that the probability of finding such a root is small for non-zero polynomials in $\mathfrak{L}_n^{(\ell,c)}$ having low total absolute degrees. This observation is the main line along which we develop our set of practical conditions. More precisely, in Section 5.2 we deduce upper bounds on the the degree of polynomials that can be generated from the input polynomials applying q operations from Ω. Based on these bounds, we finally derive the conditions for leak-resistance in Section 5.3 and for SLP-intractability in Section 5.4 and 5.5.

5.1 Randomly Picking Roots of Laurent Polynomials

The following auxiliary lemmas provide the desired link between the degrees of a multivariate Laurent polynomial and the probability of randomly picking one of its roots.

We start with Lemma 1 being an extended version of Lemma 1 in [Sho97a]. It upper bounds the probability of randomly picking a root over $\mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell - c}$ of a non-zero multivariate polynomial in terms of its degree.

Lemma 1. *Let $p \in \mathbb{P}$ be a prime, $e \in \mathbb{N}$, and $n = p^e$. Let $P \in \mathbb{Z}_n[X_1, \ldots, X_\ell]$ be a non-zero polynomial of total degree d. Then for random $\mathbf{x} \in_U \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell - c}$, where $0 \leq c \leq \ell$, we have*

$$\Pr[P(\mathbf{x}) \equiv 0 \bmod n] \leq \frac{d}{p-1} \, .$$

Proof. We can write the polynomial P as $P = p^s \cdot P'$ for some integer $s \geq 0$ and polynomial $P' \in \mathbb{Z}_{p^e}[X_1, \ldots, X_\ell]$ such that $P' \not\equiv 0 \pmod{p}$ and $\deg(P') = d$. Since we assume that $P \not\equiv 0 \pmod{p^e}$ it follows that $p^s < p^e$. Let $(x_1, \ldots, x_\ell) \in \mathbb{Z}_{p^e}^\ell$ be a root of the polynomial P. Then we have $p^e | P(x_1, \ldots, x_\ell) \Leftrightarrow p^e | p^s P'(x_1, \ldots, x_\ell)$. Since $p^s < p^e$ it follows that $p^{e-s} | P'(x_1, \ldots, x_\ell)$ and thus $P'(x_1, \ldots, x_\ell) \equiv 0 \pmod{p}$. That means every root $(x_1, \ldots, x_\ell) \in \mathbb{Z}_{p^e}^\ell$ of P over $\mathbb{Z}_{p^e}$ is also a root of P' over the prime field $\mathbb{Z}_p$. Thus, if $(x_1, \ldots, x_\ell) \in \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c}$ is a root of P modulo p^e then $(x'_1, \ldots, x'_\ell) = (x_1 \bmod p, \ldots, x_\ell \bmod p) \in \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c}$ is a root of P' modulo p. A simple consequence of a lemma by Schwartz (Lemma 1 in [Sch80]) tells us that P' has at most

$$\left|\mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c}\right| \frac{d}{|\mathbb{Z}_p^*|} = dp^c(p-1)^{\ell-c-1}$$

roots in $\mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c}$ modulo p. Thus, P has at most

$$dp^c(p-1)^{\ell-c-1}p^{(e-1)\ell} \tag{2}$$

roots in $\mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c}$ modulo p^e. So if $(x_1, \ldots, x_\ell)$ is chosen uniformly from $\mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c}$ we get

$$\begin{aligned}
\Pr[P(x_1, \ldots, x_\ell) \equiv 0 \bmod p^e] &\leq \frac{dp^c(p-1)^{\ell-c-1}p^{(e-1)\ell}}{p^{ec}(p^{e-1}(p-1))^{\ell-c}} \\
&= \frac{dp^c p^{(e-1)\ell}}{p^{e\ell-ec-\ell+c+ec}(p-1)} \\
&= \frac{dp^{(e-1)\ell+c}}{p^{(e-1)\ell+c}(p-1)} \\
&= \frac{d}{p-1}
\end{aligned}$$

□

Finally, Lemma 2 provides an upper bound on the probability of picking a root of a Laurent polynomials of degree d. We make use of Lemma 1 to prove this result.

Lemma 2. *Let $p \in \mathbb{P}$ be a prime, $e \in \mathbb{N}$, and $n = p^e$. Let $P \in \mathfrak{L}_n^{(\ell,c)}$, where $0 \leq c \leq \ell$, be a non-zero Laurent polynomial of total absolute degree d. Then for random $\mathbf{x} \in_U \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c}$ we have*

$$\Pr[P(\mathbf{x}) \equiv 0 \bmod n] \leq \frac{(\ell - c + 1)d}{p-1}.$$

Proof. Actually, we like to upper bound the probability of randomly picking a root of P by applying Lemma 1. However, P is a Laurent polynomial and may exhibit negative exponents in some variables preventing us from using this lemma directly. Therefore, let us consider the polynomial

$$R \cdot P, \text{ where } R := \prod_{c<j\leq\ell} X_j^d.$$

Since we have $R(\mathbf{x}) \in \mathbb{Z}_n^*$, i.e., it is no zero-divisor, for all $\mathbf{x} \in \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c}$, we know that $R \cdot P$ has exactly the same roots as P (and is in particular a non-zero polynomial). Thus, we can determine the probability for randomly picking a root of $R \cdot P$ instead.

Now observe that in the product polynomial $R \cdot P$ all variables only occur with positive exponents and so it can be seen as a "regular" polynomial over $\mathbb{Z}_n$: by multiplying with R, a term of the form

$X_j^{\pm d'}$, where $d' \leq d$, occurring in P is reduced to the term $X_j^{d \pm d'}$. It is easy to see that the total degree of the product polynomial is upper bounded by

$$d + (\ell - c)d = (\ell - c + 1)d\,.$$

Since $R \cdot P$ is not the zero polynomial we can apply Lemma 1 yielding

$$\begin{aligned}\Pr[P(\mathbf{x}) \equiv 0 \bmod n] &= \Pr[(R \cdot P)(\mathbf{x}) \equiv 0 \bmod n] \\ &\leq \frac{(\ell - c + 1)d}{p - 1}\,.\end{aligned}$$

□

5.2 Representing Operation Sets as Graphs: Bounding Polynomial Degrees

We aim at formalizing the class of operation sets that only allow for a small raise of the degrees of polynomials that can be generated by any (Ω, q)-SLP-generator. Remember, these are the polynomials that can be generated from the input polynomials by applying operations from Ω at most $q(\kappa)$ times. Moreover, for this kind of operation sets concrete upper bounds on the polynomial degrees should be derived.

To this end, we introduce a special type of graph, called *operation set graph* (Definition 11), modelling an operation set and reflecting the corresponding raise of degrees.

Definition 11 (Operation Set Graph). *An operation set graph $\mathcal{G} = (V, E)$ is a directed multi-edge multi-vertex graph. There are two types of vertices, namely group and product vertices. The vertex set V contains at least one group vertex and zero or more product vertices. Each group vertex in V is labeled with an unique integer. All product vertices are labeled by $\prod$. E contains zero or more directed edges, where an edge may connect two group vertices or a group and a product vertex in both directions.*

Let Ω be an operation set involving k groups. Then the operation set graph $\mathcal{G}_\Omega = (V, E)$ corresponding to Ω is constructed as follows: V is initialized with k group vertices representing the k different groups, where these vertices are labeled with the numbers that are used in the specification of Ω, say the numbers 1 to k. For each operation $(\circ, s_1, \ldots, s_u, d, F) \in \Omega$ we add additional product vertices to V and edges to E. Let the polynomial F describing the operation $\circ$ be represented as sum of non-zero monomials M_i, i.e., $F = \sum_i M_i$. Then for each M_i we do the following:

1. We add a product vertex and an edge from this vertex to the group vertex with label d.
2. For each variable Y_j $(1 \leq j \leq u)$ occurring with non-zero exponent ℓ in M_i we add ℓ edges from the group vertex labeled with the integer s_j to the product vertex just added before.

In order to embed the notion of increasing polynomial degrees by applying operations into the graph model, we introduce the following graph terminology: We associate each group vertex in a graph with a number, called *weight*. The weight may change by doing walks through the graph. Taking a *walk* through the graph means to take an arbitrary path that contains exactly two group vertices (that are not necessarily different) where one of these vertices is the start point and the other is the end point of the path. Note that such a path may contain at most one product vertex.[2] A walk modifies the weight of the end vertex in the following way:

[2] For the graphs obtained by applying the construction rules to an operation set it holds that each such path always contains a product vertex. However, we need this relaxed definition in the subsequent proof.

– If the path contains only the two group vertices, the new weight is set to be the maximum of the weights of the start and end vertex.
– If the path contains a product vertex, the new weight is set to be the maximum of the old weight and $\sum_{j=1}^{u} w_j$, where u is the indegree and w_j is the weight of the j-th predecessor of this product vertex.

We define a *free walk* to be a walk on a path that only consists of the two group vertices and no other vertex. A *non-free walk* is a walk on a path containing a product vertex. It is important to observe that

– a non-free walk can actually increase the maximum vertex weight of a graph in contrast to a free-walk.
– after each non-free walk the weight of any vertex can be changed at most finitely many times by doing free walks.

Hence, the following definition of the *q-weight* makes sense: Let q be a fixed positive number. We consider finite sequences of walks through a graph, where each sequence consists of exactly q non-free walks and an arbitrary finite number of free walks. We define the q-weight of a (group) vertex to be the maximum weight of this vertex over all such sequences. Similarly, we define the q-weight of an operation set graph to be the maximum of the q-weights of all its vertices. Considering the above observations, we see that the q-weight of any vertex and thus the q-weight of the graph is well-defined and a finite number.

Obviously, the q-weights of the vertices $1, \dots, k$ of an operation set graph $\mathcal{G}_\Omega$ can be used to upper bound the degrees of the output polynomials $L_1, \dots, L_k$ of any (Ω, q)-SLP-generator $\mathcal{S}$ when setting the initial vertex weights appropriately. This is captured more formaly in Lemma 3.

Lemma 3. *Let $\mathbf{I} = (\mathbf{I_1}, \dots, \mathbf{I_k})$, where $\mathbf{I_i} \subset \mathfrak{L}_n^{(\ell,c)}$, and $Q \in \mathfrak{L}_n^{(\ell,c)}$ be given. Let Ω be an operation set involving k groups and $\mathcal{G}_\Omega$ be the corresponding operation set graph, where we define the initial weight of any group vertex i to be the maximal degree of the polynomials in $\mathbf{I_i}$. Let $\mathcal{S}$ be a (Ω, q)-SLP-generator. Then for any output $(L_1, \dots, L_k) \in [\mathcal{S}(\kappa, n, \mathbf{I}, Q)]$, where κ is some fixed positive number, the degrees of the polynomials in L_i are upper bounded by the $q(\kappa)$-weight of the group vertex i.*

Remark 2. Assume we set the initial weights of any group vertex i to be the the maximal positive (respectively negative) degree, i.e., the maximal sum of the positive (respectively negative) exponents in any monomial, of the input polynomials $\mathbf{I_i}$. Then the $q(\kappa)$-weight of the group vertex i can be used to bound the positive (and negative) degrees of any Laurent polynomial in L_i. Similarly, it could be used the to bound the positive or negative degree of a single variable X_j occurring in any Laurent polynomial in L_i. Thus, the following results on the q-weight are also helpful to bound these degrees, which is especially interesting with regard to Condition 5 in Section 5.5.

With regard to the definition of the q-weight, we can immediately simplify the structure of operation set graphs: Clearly, we do not change the q-weight of a graph if we remove self-loops and product vertices with indegree 1, where in the latter case the two edges entering and leaving the vertex are replaced by a single edge going from the predecessor vertex to the successor vertex. In the following we only consider operation set graphs reduced in this way and thus being compliant to Definition 12.

Definition 12 (Reduced Operation Set Graph). *A reduced operation set graph is an operation set graph containing any product vertex with indegree one or self-loops.*

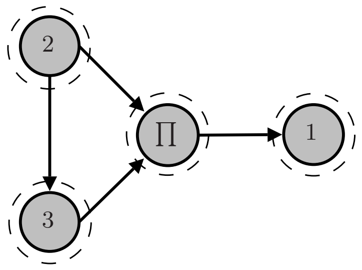

Fig. 1. $\mathcal{G}_\Omega$ for $\Omega = \{(\circ_1, 1, 1, 1, Y_1 + Y_2), (\circ_2, 2, 2, 2, Y_1 + Y_2), (\circ_3, 3, 3, 3, Y_1 + Y_2), (inv_1, 1, 1, -Y_1), (inv_2, 2, 2, -Y_1), (inv_3, 3, 3, -Y_1), (\psi, 2, 3, Y_1), (e, 2, 3, 1, Y_1 \cdot Y_2)\}$ of w-BDHIP. Strongly connected components are marked by dashed borders.

As an illustrating example, consider the reduced operation set graph depicted in Figure 1, which belongs to the operation set for the w-BDHI problem (cf. Example 4).

The following condition characterizes graphs that do not allow for a super-polynomial grow of vertex weights. Intuitively, it prohibits any kind of repeated doubling.

Condition 1 *Let $\mathcal{G}_\Omega$ be a reduced operation set graph. Then for every strongly connected component*[3] *$\mathcal{C}$ of $\mathcal{G}_\Omega$ it holds that every product vertex contained in $\mathcal{C}$ has at most one incoming edge from a vertex that is also contained in $\mathcal{C}$.*

Furthermore, for the q-weight of operation set graphs satisfying Condition 1, it is possible to derive non-trivial upper bounds as given in Theorem 2.

Theorem 2. *Let $\mathcal{G}_\Omega$ be a reduced operation set graph satisfying Condition 1. Let n_1 denote the number of product vertices contained in $\mathcal{G}_\Omega$, $u_{\max}$ the maximal indegree and $d_{\max}$ the maximal initial weight of these product vertices, and n_2 the number of SCCs containing at least one product and one group vertex. Then the q-weight of $\mathcal{G}_\Omega$ is upper bounded by*

$$D(n_1, n_2, u_{\max}, d_{\max}, q) = \begin{cases} d_{\max}(u_{\max})^{n_1}, & n_2 = 0 \\ d_{\max}\mathsf{e}^{n_1}, & n_2 > 0 \text{ and } q < \frac{\mathsf{e}}{u_{\max}} n_1 \\ d_{\max}\left(\frac{u_{\max} q}{n_1}\right)^{n_1}, & n_2 > 0 \text{ and } q \geq \frac{\mathsf{e}}{u_{\max}} n_1 \end{cases},$$

where e denotes Euler's number.

Example 5. Condition 1 is satisfied for $\mathcal{G}_\Omega$ depicted in Figure 1 since the strongly connected component containing the product vertex contains no other vertices. We have $n_1 = 1$, $n_2 = 0$, and $u_{\max} = 2$. When setting the initial weight of any group vertex in $\mathcal{G}_\Omega$ to some value w, then Theorem 2 yields that for any q the q-weight of the graph is bounded by $2w$.

Note that the factor by which the (maximal) initial weight of the vertices can be increased only depends on the particular operation set graph. Hence, with regard to Lemma 3, once we have shown that an operation set only allows to increase degrees by a low (i.e., polynomial) factor, this certainly holds for all problems involving this operation set and does not need to be reproven (as it is currently done in the literature).

Proof. The operation set graph $\mathcal{G}_\Omega$ can have manifold forms. Table 1 summarizes which connections between vertices are possible in $\mathcal{G}_\Omega$. In order to determine an upper bound on the q-weight of $\mathcal{G}_\Omega$ we apply a sequence of slight transformations to the graph that simplify its structure and can only *increase* its q-weight. Finally, we will end up with a graph for the case $n_2 = 0$ and $n_2 > 0$, respectively, exhibiting a very simple structure that allows us to determine the desired bound.

[3] A strongly connected component of a directed graph $\mathcal{G}_\Omega = (V, E)$ is a maximal set of vertices $U \subset V$ s.t. every two vertices in U are reachable from each other. The strongly connected components of a graph can be computed in time $O(V + E)$ (e.g., see [CLR90]).

Table 1. Possible connections between different types of vertices in $\mathcal{G}_\Omega$

Vertex Type	**Indegree**	**Possible Predecessors**	**Outdegree**	**Possible Successors**
group	$0 \ldots *$	group, product	$0 \ldots *$	group, product
product	$2 \ldots *$	group	1	group

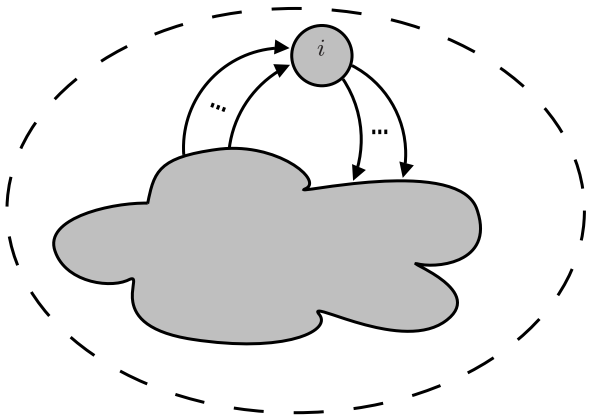

Fig. 2. Structure of an SCC containing more than one vertex after first modification

The following transformations simplify the structure of the strongly connected components of the graph so that we end up with only a few different forms. In the course of these modifications we sometimes add new group vertices to the graph receiving edges from other group vertices. This is the reason why the notion of free-walks has been introduced in Section 5.2. When adding such a vertex, we always assume that it is labeled with a new unique integer and initially receives the weight $d_{\max}$.

Modification 1. First of all, we set the initial weight of each group vertex to $d_{\max}$. Clearly, it will not change the q-weight of the graph if we replace directly connected group vertices within each SCC by a joined single vertex. Incoming and outgoing edges of the individual group vertices are simply connected to the joined vertex. Finally, we remove all self-loops in the graph, i.e., edges that have the same vertex as start and end point.

Now, the graph may contain three different types of SCCs. We have SCCs consisting of a single group vertex or a single product vertex. Moreover, there may exist SCCs that contain more than one vertex. Such SCCs comply with the structure depicted in Figure 2. The cloud shown in this figure is composed of one or more product vertices and zero or more group vertices that are connected as follows:

- each group vertex is connected to one or more product vertices also contained in the SCC.
- each group vertex has zero or more incoming edges from outside the SCC and zero or more outgoing edges to other SCCs (omitted in Figure 2).
- each product vertex has exactly one incoming edge from exactly one group vertex also contained in the SCC and it has exactly one outgoing edge to a group vertex also contained in the SCC.
- each product vertex has at least one incoming edge from outside the SCC (omitted in Figure 2).

Note that there might be more than one path through the cloud since group vertices might have edges to more than one product vertex inside the cloud.

Modification 2. The following modification only needs to be applied if $n_2 > 0$. We reduce all SCCs containing more than two vertices to SCCs containing exactly two vertices. Certainly, it can only increase the q-weight of a graph if we connect each group vertex in such an SCC with each other by direct edges in both directions. Thus, we can join all group vertices within the same

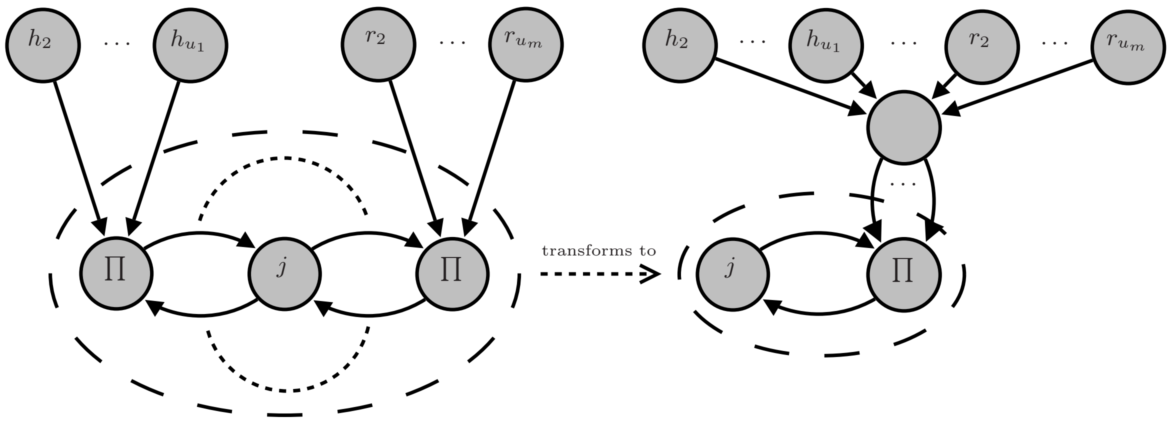

Fig. 3. Transformation of a group vertex with connections to more than one product vertex

SCC as we did in the course of the first modification. In this way, we get one group vertex that is connected to more than one product vertex in the same SCC. This situation is depicted at the left hand side of Figure 3.

Assume that a joined group vertex has outgoing edges to m product vertices. The m product vertices have $u_1 - 1, \ldots, u_m - 1$ incoming edges from outside the SCC. These edges stem from group vertices. For the sake of simplicity, we omitted additional incoming and outgoing edges of the joined group vertex shown in Figure 3. We transform such a structure as follows: we remove all m product vertices and connect their incoming edges to an extra group vertex added to the graph (outside the SCC). Furthermore, we add a product vertex to the SCC and connect the newly added group vertex via $u_{\max} - 1$ edges to this product vertex. Finally, we put an edge from the joined group vertex to the product vertex and another edge in the opposite direction. Hence, we receive an SCC containing exactly one group and one product vertex that form a cycle as shown on the right hand side of Figure 3. We apply the transformation just described also to SCCs containing exactly one group and one product vertex (i.e., $m = 1$) if the product vertex has an indegree smaller than $u_{\max}$.

To summarize, our graph can now be composed of the following types of strongly connected components: SCCs consisting of a single group vertex (Type 1), or a single product vertex (Type 2) or a group and a product vertex (Type 3).

Modification 3. We apply the following modification provided that $n_2 > 0$: The structure of the graph is further simplified by replacing any SCC of Type 2 by an SCC of Type 3 as shown in Figure 4. The product vertex of the SCC of Type 3 (on the right hand side of the figure) has $u_{\max} - 1$ incoming edges. The group vertex in this SCC has been newly added to the graph. Note that we cannot simply use the group vertex j for this purpose since this may introduce additional cycles to the graph. Moreover, observe that the group vertices $i_1, \ldots, i_u$ that were connected to the SCC of Type 2 are now connected to another new group vertex added outside the SCC. This new vertex has $u_{\max} - 1$ edges to the product vertex in the SCC.

Modification 4. Let us consider each SCC of the graph as a single vertex. The resulting graph is a directed acyclic graph (DAG). We perform a topological sort on this DAG, i.e., we arrange the vertices of the DAG in different layers such that Layer 0 contains only the sources of the DAG, Layer 1 only the vertices that receive edges from vertices in Layer 0, Layer 2 only the vertices that receive edges from Layer 1 or Layer 0 and so on. In general, vertices in Layer j can only receive edges from vertices contained in Layers $i < j$.

Now observe that paths through the DAG starting at the same source may split at some vertex and join again at another vertex. To analyze the q-weight of the graph we like to prevent this. We

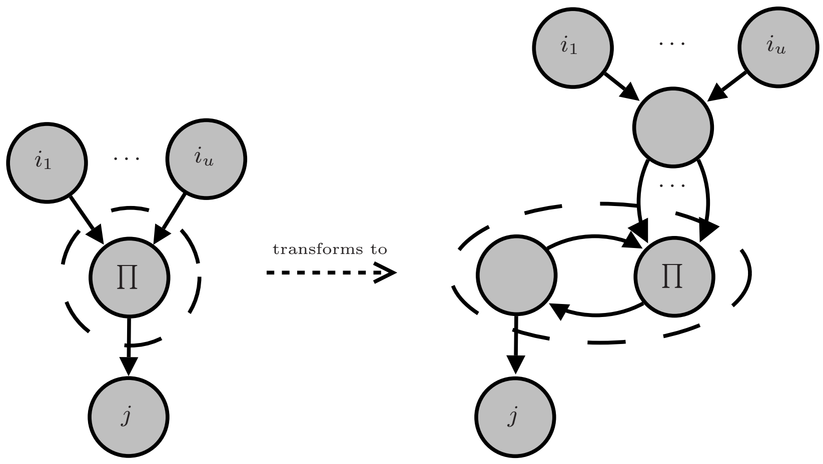

Fig. 4. Replace SCC consisting of a single product vertex

further simplify the DAG such that if a path branches at some vertex the splitted paths never join again.

To this end, we proceed as follows: We first describe the modifications for the case $n_2 > 0$. We start at Layer 1 and consider all SCCs having incoming edges from multiple different sources at Layer 0. If this is the case for an SCC of Type 1, we can simply keep exactly one edge from one arbitrary source and remove all other edges. Note that we do not change the q-weight of the graph in this way since all sources are group vertices and are associated with the same initial weight $d_{\max}$. Now assume that an SCC of Type 3 has this property. Remember that the product vertex of such an SCC has $u_{\max} - 1 \geq 1$ incoming edges from vertices of Layer 0 and the group vertex may have $m \geq 0$ incoming edges from outside the SCC. In this case we remove all incoming edges, choose an arbitrary source at Layer 0 and connect this source via $u_{\max} - 1$ edges to the product vertex and via one edge to the group vertex of this SCC.

After that, we consider the SCCs in Layer 2. Such an SCC may have incoming edges from different vertices of Layer 0 and Layer 1. We have to choose the predecessor vertex providing the maximal increase of weight among all predecessors. After having selected such a predecessor, we remove the edges from all other predecessors and connect the chosen predecessor to the SCC of Layer 2 in the way described before. The right predecessor is chosen as follows: If the SCC of Layer 2 receives only edges from Layer 0, we can pick an arbitrary predecessor of this layer. If it has one or more edges from Layer 1, then we only need to consider the predecessors from Layer 1. If one or more of them are SCCs of Type 3, we pick one of these as source. If all of them are SCCs of Type 1, then we choose an arbitrary of these as source.

In general, we handle one layer after another in this way. If we are at a certain layer and find an SCC receiving edges from multiple predecessors, we do the following to select a single predecessor: we count the number of SCCs of Type 3 along each path from a source of the DAG in Layer 0 to this SCC. and choose the predecessor being part of the path that exhibits the maximal number. (If more than one path has a maximal number of SCCs of Type 3, we can choose an arbitrary of the corresponding predecessors.)

In the case $n_2 = 0$, we apply the same transformations where the predecessor is selected according to the maximal number of SCCs of Type 2.

Analyzing the q-weight. Now, it is clear that the q-weight of the graph will be the weight at a vertex belonging to the path from a source to a sink of the DAG that contains the *maximal* number of SCCs of Type 2 if $n_2 = 0$ or of Type 3 if $n_2 > 0$. Moreover, it is generated by doing

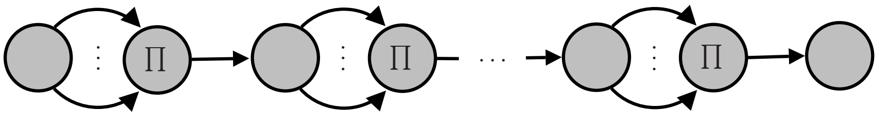

Fig. 5. Paths containing the q-weight for the case $n_2 = 0$

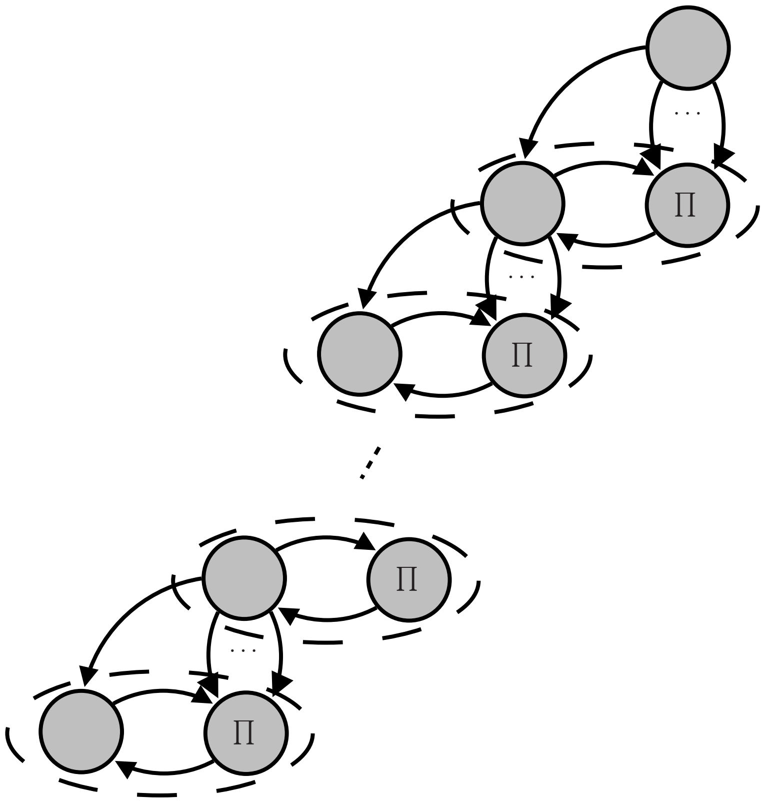

Fig. 6. Paths containing the q-weight for the case $n_2 > 0$

walks only along this path. This is because splitted paths will never join again and so in order to obtain a maximal vertex weight it is not "worthwhile" to do walks along different paths.

Let us first consider the case $n_2 = 0$. The structure such a path is shown in Figure 5. For the sake of simplicity, we omitted group vertices that may exist between the depicted group vertices. Note that the presence of such additional group vertices does not affect the q-weight of a graph. It is easy to see that the best strategy in order to achieve a maximal vertex weight by doing a number of q walks along the shown path is to first apply a walk from the source of the path to the second group vertex via the product vertex, then to apply a walk from this group vertex to the third group vertex via the second product vertext, and so on. Since the path may contain at most n_1 product vertices each of indegree $u_{\max}$, the q-weight is upper bounded by

$$d_{\max} u_{\max}^{q}$$

in the special case $q \leq n_1$ and in general by

$$d_{\max} u_{\max}^{n_1} .$$

Now, let us consider the case $n_2 > 0$. Figure 5 shows the path structure for this case, where additional group vertices between the SCCs of Type 3 are omitted again. In order to achieve a maximal vertex weight, we first apply a certain number a_1 of walks from the source to the group vertex of the first SCC, then a certain number a_2 of walks from the group vertex of the second SCC to the group vertex of third SCC and so on.

We can easily analyze the effects of this strategy: Doing $a_1 > 0$ walks in a row from the source to the group vertex of the first SCC results in a weight in the latter group vertex of at most

$$d_{\max} + a_1(u_{\max} - 1)d_{\max} \leq a_1 u_{\max} d_{\max}.$$

Then doing $a_2 > 0$ walks in a row from the group vertex of the second SCC to the group vertex of the third SCC results in a weight in the latter group vertex of at most

$$\begin{aligned} a_1 u_{\max} d_{\max} + a_2(u_{\max} - 1)(a_1 u_{\max} d_{\max}) &\leq a_2 u_{\max}(a_1 u_{\max} d_{\max}) \\ &= a_1 a_2 d_{\max} u_{\max}^2 \end{aligned}$$

In general, when applying this strategy and doing $a_1 > 0$ walks to the 1st SCC, $a_2 > 0$ walks to the 2nd SCC, ..., and finally $a_t > 0$ walks from the group vertex of the $(t-1)$-th SCC to the group vertex of the t-th SCC, this results in a weight of at most

$$d_{\max} u_{\max}^t \prod_{i=1}^{t} a_i \,. \tag{3}$$

Remark 3. The product in Equation (3) can immediately be upper bounded by observing that each a_i is clearly smaller or equal to q. This leads to the rough upper bound

$$d_{\max} u_{\max}^{n_1} q^{n_1} \,.$$

However, it is possible to derive a slightly tighter bound than this which is done in the following.

Let $m \leq n_1$ be the number of SCCs along this path and q be the total number of walks we perform on this path. We like to upper bound the product (3) under the following constraints on the occurring parameters: $d_{\max} \geq 1$, $u_{\max} \geq 2$, $\sum_{i=1}^{t} a_i = q$, and $t \leq \min(q, m)$. First, we can show that the t-partition of q in summands a_i leads to a maximal product of summands if all a_i are equal. More precisely, for all $t \geq 1$ and all $a_1, \ldots, a_t > 0$ with $\sum_{i=1}^{t} a_i = q$ we have

$$\prod_{i=1}^{t} a_i \leq \left(\frac{q}{t}\right)^t \,.$$

The above inequality can be shown using induction on t: For $t = 1$ the statement is obviously true. For $t > 1$ we have

$$\prod_{i=1}^{t} a_i = a_t \prod_{i=1}^{t-1} a_i \leq a_t \left(\frac{q - a_t}{t-1}\right)^{t-1} \leq \frac{q}{t}\left(\frac{q - \frac{q}{t}}{t-1}\right)^{t-1} = \left(\frac{q}{t}\right)^t \tag{4}$$

The 2nd inequality of Equation (4) follows from the fact that the function $f(x) = x\left(\frac{q-x}{t-1}\right)^{t-1}$ restricted to the domain $(0, q)$ reaches its absolute maximum for $x = \frac{q}{t}$.

Hence, if we have already decided to involve a certain fixed number t of consecutive SCCs of the path in our walks, we obtain a maximal weight by spending (approximately) the same number of walks on each SCC since

$$d_{\max} u_{\max}^t \prod_{i=1}^{t} a_i \leq d_{\max}\left(\frac{u_{\max} q}{t}\right)^t \,.$$

It remains to determine the "best" choice for t.

Considering the function $f(x) = \left(\frac{u_{\max}q}{x}\right)^x$ in the domain $(0,\infty)$, yields that f reaches its absolute maximum $f(x) = \mathsf{e}^{\frac{u_{\max}q}{\mathsf{e}}}$ at $x = \frac{u_{\max}q}{\mathsf{e}}$, where e denotes Euler's number. Furthermore, f has no other local extremum. So it is monotonically increasing in $(0, \frac{uq}{\mathsf{e}}]$ and decreasing in $(\frac{uq}{\mathsf{e}}, \infty)$. From this observation it follows immediately that for all $t \leq n_1$ we have

$$d_{\max}\left(\frac{u_{\max}q}{t}\right)^t \leq \begin{cases} d_{\max}\mathsf{e}^{n_1}, & q < \frac{\mathsf{e}}{u_{\max}}n_1 \\ d_{\max}\left(\frac{u_{\max}q}{n_1}\right)^{n_1}, & q \geq \frac{\mathsf{e}}{u_{\max}}n_1 \end{cases}$$

□

Based on the ideas underlying the reductions in the previous proof, it is possible to devise a graph algorithm (Algorithm 1) that finds individual bounds on the q-weights of the group vertices which are often tighter than the generic bound from Theorem 2. The principle of the algorithm is simple. We consider the directed acyclic graph that is composed of the SCCs of the operation set graph. We move from the sources to the sinks of the DAG and recursively bound the q-weights of the vertices within each SCC. More precisely, we label each SCC with a q-weight (bound) based on the weight of the vertices inside the SCC and the SCCs having direct edges to the considered SCC. As a starting point the algorithm considers the SCCs of indegree zero that due to Condition 1 may only be composed of group vertices. Then all SCCs receiving only edges from these sources are labeled, and so on. In the end when all SCCs are labeled, the q-weight bound of a group vertex is simply set to be the q-weight bound of the (unique) SCC it is contained in.

Algorithm 1 Computation of the q-weigths of group vertices.

```
Input: $q$, reduced operation set graph $\mathcal{G}$ satisfying Condition 1, initial weights for the $k$ group vertices in $G$
Output: $q$-weights $w_1, \ldots, w_k$ of vertices $1, \ldots, k$
 1: Perform a topological sort on the DAG of $\mathcal{G}$, i.e., arrange the SCCs of $\mathcal{G}$ in layers 0 to $\ell$ such that SCCs in layer
    $j$ can only receive edges from SCCs contained in layers $i < j$.
 2: for each layer $j = 0 : \ell$ do
 3:     for each SCC $\mathcal{C}$ in layer $j$ do
 4:         if $\mathcal{C}$ consists only of group vertices then
 5:             set weight of $\mathcal{C}$ to maximum of weights of vertices contained in $\mathcal{C}$ and weights of SCCs in layers $i < j$
                having edges to $\mathcal{C}$
 6:         end if
 7:         if $\mathcal{C}$ consists only of a single product vertex then
 8:             set weight of $\mathcal{C}$ to sum of weights of SCCs in layers $i < j$ having edges to $\mathcal{C}$
 9:         end if
10:         if $\mathcal{C}$ consists of at least one product vertex and one group vertex then
11:             let $w$ be the maximum of the weights of group vertices contained in $\mathcal{C}$ and the weights of SCCs in layers
                $i < j$ having edges to these group vertices
12:             for each product vertex $\Pi$ in $\mathcal{C}$, compute sum of weights of SCCs in layers $i < j$ having edges to $\Pi$, and
                let $v$ be the maximum of these sums
13:             set weight of $\mathcal{C}$ to $w + qv$
14:         end if
15:     end for
16: end for
17: for $i = 1 : k$ do
18:     set $w_i$ to weight of SCC containing the group vertex $i$
19: end for
```

5.3 Practical Conditions: Leak-Resistance

To provide leak-resistance, we ensure that any difference of two distinct polynomials computable by a (Ω, q)-SLP-generator is of low degree. We do so by demanding that the input polynomials **I** of a problem $\mathcal{P}$ have low degrees (Condition 2) and restrict to operation sets Ω only allowing for small increase of degrees (Condition 1).

Condition 2 *There exists* $r_1 \in \mathsf{poly}(x)$ *such that for all* $\kappa \in \mathbb{N}$, $\mathbf{I} \in [\mathsf{SIGen}_{\mathcal{P}}^{\mathbf{I}}(\kappa)]$*:*

$$\max_{1 \le i \le k, P \in \mathbf{I_i}} (\deg(P)) \le r_1(\kappa).$$

If these conditions are satisfied, we can derive a concrete leak-resistance bound ν for any runtime bound q as done in Theorem 3.

Theorem 3. *Let Ω be an operation set such that Condition 1 is satisfied. Furthermore, let $\mathcal{P}$ be a DL-type or DH-type problem satisfying Condition 2. Then for any $q \in \mathsf{poly}(x)$, the problem $\mathcal{P}$ is (Ω, q, ν)-leak-resistant, where*

$$\nu(\kappa) = k(q(\kappa) + z(\kappa))^2 \frac{(\ell - c + 1) D(n_1, n_2, u_{\max}, r_1(\kappa), q(\kappa))}{2^\kappa}$$

is a negligible function.

Example 6 (Leak-resistance for w-BDHIP). The degrees of the input polynomials of the w-BDHI problem are polynomially upper bounded through $w(\kappa)$ by definition. As we have already seen in Example 5, Ω satisfies Condition 1 yielding $D(1, 0, 2, w(\kappa), q(\kappa)) = 2w(\kappa)$. Furthermore, for w-BDHIP we have parameters $k = 3$, $\ell = 1$, and $c = 0$. Thus, by Theorem 3 the problem $\mathcal{P}$ is (Ω, q, ν)-leak-resistant, where

$$\nu(\kappa) = \frac{12(q(\kappa) + w(\kappa) + 1)^2 w(\kappa)}{2^\kappa}.$$

Proof. Let $\mathcal{S}$ be some (Ω, q)-SLP-generator. Furthermore, let $(n, \mathbf{I}, Q) \overset{\text{R}}{\leftarrow} \mathsf{SIGen}_{\mathcal{P}}^{(n,\mathbf{I},Q)}(\kappa)$ and $(L_1, \dots, L_k) \overset{\text{R}}{\leftarrow} \mathcal{S}(\kappa, n, \mathbf{I}, Q)$. Remember that $|L_i|$ is upper bounded by $q(\kappa) + z(\kappa)$. Applying Theorem 2 and Lemma3 yields that for for every $P \in L_i$, $\deg(P)$ is upper bounded by $D(n_1, n_2, u_{\max}, r_1(\kappa), q(\kappa))$, where n_1 denotes the number of product vertices contained in $\mathcal{G}_\Omega$, $u_{\max}$ the maximal indegree of these product vertices, and n_2 the number of SCCs containing at least one product and one group vertex. Moreover, note that every polynomial not contained in the ideal $\mathcal{I}_n$ is certainly not the zero polynomial in $\mathfrak{L}_n^{(\ell,c)}$. Thus, by applying Lemma 2 and using the assumption $2^\kappa < p < 2^{\kappa+1}$ we get

$$\begin{aligned}
&\Pr[\exists i \text{ and } P, P' \in L_i : (P - P')(\mathbf{x}) \equiv 0 \bmod n \ \wedge \ P - P' \notin \mathcal{I}_n] \\
&\le k(q(\kappa) + z(\kappa))^2 \frac{(\ell - c + 1)(D(n_1, n_2, u_{\max}, r_1(\kappa), q(\kappa)))}{p - 1} \\
&\le k(q(\kappa) + z(\kappa))^2 \frac{(\ell - c + 1)(D(n_1, n_2, u_{\max}, r_1(\kappa), q(\kappa)))}{2^\kappa}
\end{aligned}$$

where $\mathbf{x} \overset{\text{R}}{\leftarrow} \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell - c}$. □

5.4 Practical Conditions: SLP-intractability of DL-type Problems

In view of Lemma 2, in order to ensure SLP-intractability for a DL-type problem it suffices to require the challenge polynomial being non-constant (Condition 3) and of low degree (Condition 4).

Condition 3 (Non-Triviality of DL-type Problems) *There exists* $\kappa_0 \in \mathbb{N}$ *such that for all* $\kappa \ge \kappa_0$, $(n, Q) \in [\mathsf{SIGen}_{\mathcal{P}}^{(n,Q)}(\kappa)]$ *the polynomial Q is not a constant in* $\mathfrak{L}_n^{(\ell,c)}$.

In plain language, Condition 3 is the very natural condition that the solution of a DL-type problem should depend on the secret choices. Otherwise, the considered problem would be totally flawed, i.e., solvable with a success probability equal to 1.

Note that Definition 1 allows a problem to have a "completely different" challenge polynomial Q for each possible group order $n = p^e$. However, in practice, the form of the challenge polynomial usually does not depend on the parameter n. More precisely, there exists a Laurent polynomial Q' over $\mathbb{Z}$ such that for any n the corresponding challenge polynomial is simply defined by $Q' \bmod n$ (e.g., $Q' = X_1$ for the DLP). In this case observing that Q' is not a constant (over the integers) immediately implies that for almost all primes p and all integers e the polynomial $Q' \bmod p^e$ is not constant in $\mathfrak{L}_n^{(\ell,c)}$.

Moreover, by Condition 4 we restrict to challenge polynomials with low degrees. Note that we had a similar condition for the input polynomials.

Condition 4 *There exists $r_2 \in \mathsf{poly}(x)$ such that for all $\kappa \in \mathbb{N}$, $Q \in [\mathsf{SIGen}_{\mathcal{P}}^{Q}(\kappa)]$:*

$$\deg(Q) \leq r_2(\kappa).$$

Assuming the above conditions are satisfied for a DL-type problem, Theorem 4 implies that the problem is ν-non-trivial, where ν is a negligible function in the security parameter.

Theorem 4. *Let $\mathcal{P}$ be a DL-type problem satisfying Condition 3 and Condition 4. Then the problem $\mathcal{P}$ is ν-non-trivial, where*

$$\nu(\kappa) = \begin{cases} 1, & \kappa < \kappa_0 \\ \frac{(\ell-c+1)r_2(\kappa)}{2^\kappa}, & \kappa \geq \kappa_0 \end{cases}$$

is a negligible function.

Example 7. Let us consider the DLP. Here, Conditions 3 and 4 are satisfied since we always have $\deg(Q) = 1 =: r_2(\kappa)$. Moreover, remember that $\ell = c = 1$ for the DLP. Thus, by Theorem 4 the problem is ν-SLP-intractable where $\nu(\kappa) = \frac{1}{2^\kappa}$.

Proof. Clearly, the probability considered in Definition 8 is always upper bounded by 1 for $\kappa \leq \kappa_0$, where κ_0 is the bound from Condition 3. Now let $\kappa \geq \kappa_0$. Furthermore, let $(n, Q) \overset{R}{\leftarrow} \mathsf{SIGen}_{\mathcal{P}}^{(n,Q)}(\kappa)$ and an arbitrary value $a \in \mathbb{Z}_n$ be given. Condition 3 yields that $Q - a$ is not zero in $\mathfrak{L}_n^{(\ell,c)}$ and from Condition 2 we know that $\deg(Q - a)$ is upper bounded by $r_2(\kappa)$. Thus, by applying Lemma 2 and using the assumption $2^\kappa < p < 2^{\kappa+1}$ we get

$$\Pr[Q(\mathbf{x}) \equiv a \bmod n] \leq \frac{(\ell - c + 1)r_2(\kappa)}{p-1} \leq \frac{(\ell - c + 1)r_2(\kappa)}{2^\kappa}$$

for $\mathbf{x} \overset{R}{\leftarrow} \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c}$. □

5.5 Practical Conditions: SLP-intractability of DH-type Problems

To ensure SLP-intractability of DH-type problems we formulate similar conditions as in the case of DL-type problems. More precisely, we ensure that the difference polynomials considered in the definition of SLP-intractability (Definition 10) are never zero and of low degree.

The non-triviality condition (Condition 5) states that an efficient SLP-generator can hardly ever compute the challenge polynomial (in L_1), and thus solve the problem with probability 1.

Condition 5 (Non-Triviality of DH-type Problems) *For every $q \in \mathsf{poly}(x)$ there exists $\kappa_0 \in \mathbb{N}$ such that for all $\kappa \geq \kappa_0$, (Ω, q)-SLP-generators $\mathcal{S}$, $(n, \mathbf{I}, Q) \in [\mathsf{SIGen}_{\mathcal{P}}^{(n,\mathbf{I},Q)}(\kappa)]$, and $(L_1, \ldots, L_k) \in [\mathcal{S}(\kappa, n, \mathbf{I}, Q)]$, $P \in L_1$ we have $P \neq Q$ in $\mathfrak{L}_n^{(\ell,c)}$.*

We note that Condition 5 appears to be more complex compared to the practical conditions seen so far and it is not clear to us how to verify it in its full generality. However, to the best of our knowledge it is easy to check for any particular problem of practical relevance. Usually, one of the following properties is satisfied implying the validity of Condition 5:

- The total degree of every $P \in L_1$ is bounded by a value which is smaller than the total degree of Q.
- The positive/negative degree of every $P \in L_1$ is bounded by a value which is smaller than the positive/negative degree of Q.
- The positive/negative degree of some variable X_j in every $P \in L_1$ is bounded by a value which is smaller than the positive/negative degree of that variable in Q.

Remember, that we can make use of the results from Section 5.2 for proving that a problem satisfies one these properties. It should also be pointed out that it is possible to derive easier versions of Condition 5 by restricting the considered class of problems and operation sets. However, we do not go into detail.

Moreover, we have to prevent that an (Ω, q)-SLP-generator outputs a polynomial $P \neq Q$ frequently "colliding" with Q and thus serving as a good interpolation for Q. Assuming P is low degree, which is given by Conditions 1 and 2, it remains to demand that Q is of low degree as well:

Condition 6 *There exists $r_2 \in \mathsf{poly}(x)$ such that for all $\kappa \in \mathbb{N}$, $Q \in [\mathsf{SIGen}_{\mathcal{P}}^{Q}(\kappa)]$:*

$$\deg(Q) \leq r_2(\kappa).$$

We need the practical conditions for leak-resistance in addition to the ones stated in this section for showing that a DH-type problem is (Ω, q, ν)-SLP-intractable, where ν is a negligible function in the security parameter. This is captured by Theorem 5.

Theorem 5. *Let Ω be an operation set such that Condition 1 is satisfied. Furthermore, let $\mathcal{P}$ be DH-type problem satisfying Condition 2, Condition 5, and Condition 6. Then for any $q \in \mathsf{poly}(x)$, the problem $\mathcal{P}$ is (Ω, q, ν)-SLP-intractable, where*

$$\nu(\kappa) = \begin{cases} 1, & \kappa < \kappa_0 \\ \frac{(\ell - c + 1)(r_2(\kappa) + D(n_1, n_2, u_{\max}, r_1(\kappa), q(\kappa)))}{2^\kappa}, & \kappa \geq \kappa_0 \end{cases}$$

is a negligible function.

Example 8 (SLP-intractability of w-BDHIP). Remember, that for this problem the challenge polynomial is fixed to $Q = X_1^{-1}$. Moreover, observe that all variables occur only with positive exponents in the input polynomials. Thus, also every polynomial $P \in L_1$ will exhibit only variables occurring with positive exponents. Hence, Condition 5 is trivially satisfied (independently of the considered operation set Ω).[4] Condition 6 is satisfied since we always have $\deg(Q) = 1 =: r_2(\kappa)$. As we have already seen in the previous section, Conditions 1 and 2 hold yielding the upper bound $D(1, 0, 2, w(\kappa), q(\kappa)) = 2w(\kappa)$ on the degrees of the polynomials $P \in L_1$. Moreover, we have parameters $k = 3$, $\ell = 1$, and $c = 0$. Thus, by Theorem 5 the problem is (Ω, q, ν)-SLP-intractable, where

$$\nu(\kappa) = \frac{2 + 4w(\kappa)}{2^\kappa}.$$

[4] Note that $X_1^{-1} \neq X_1^{\phi(n)-1}$ in $\mathfrak{L}_n^{(1,0)}$ but these polynomials evaluate to the same value for all $x_1 \in \mathbb{Z}_n^*$. However, Cond. 1 and 2 ensure that for any efficient SLP-generator there exists κ_0 s.t. for all $\kappa \geq \kappa_0$ the polynomial $X_1^{\phi(n)-1}$ cannot be computed.

Proof. Clearly, the probability considered in Definition 10 is bounded by 1 for $\kappa \leq \kappa_0$, where κ_0 is the bound from Condition 5. Now let $\kappa \geq \kappa_0$, q be an arbitrary polynomial, and $\mathcal{S}$ be some (Ω, q)-SLP-generator. Furthermore, let $(n, \mathbf{I}, Q) \overset{\mathrm{R}}{\leftarrow} \mathsf{SIGen}_{\mathcal{P}}^{(n,\mathbf{I},Q)}(\kappa)$ and $(L_1, \ldots, L_k) \overset{\mathrm{R}}{\leftarrow} \mathcal{S}(\kappa, n, \mathbf{I}, Q)$. Remember that $|L_1|$ is upper bounded by $q(\kappa) + z(\kappa)$. Condition 5 yields that for all Laurent polynomials $P \in L_1$ the difference $Q - P$ is not zero in $\mathfrak{L}_n^{(\ell,c)}$. From Condition 2 we know that $\deg(Q)$ is polynomially upper bounded by $r_2(\kappa)$ and $\deg(P)$ is upper bounded by $D(n_1, n_2, u_{\max}, r_1(\kappa), q(\kappa))$ which results from Theorem 2 and Lemma3. Thus, we have

$$\deg(Q - P) \leq r_2(\kappa) + D(n_1, n_2, u_{\max}, r_1(\kappa), q(\kappa))$$

for all $P \in L_1$. By applying Lemma 2, this leads to a probability of

$$\begin{aligned}\Pr[(Q-P)(\mathbf{x}) \equiv 0 \bmod n] &\leq \frac{(\ell - c + 1)(r_2(\kappa) + D(n_1, n_2, u_{\max}, r_1(\kappa), q(\kappa)))}{p-1} \\ &\leq \frac{(\ell - c + 1)(r_2(\kappa) + D(n_1, n_2, u_{\max}, r_1(\kappa), q(\kappa)))}{2^\kappa}\end{aligned}$$

for $\mathbf{x} \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell - c}$. □

6 Extensions

6.1 Showing the Non-Existence of Generic Reductions

The basic role of operation sets is to reflect the *natural abilities* (e.g., computing the group law) of a generic algorithm with respect to the algebraic setting but they can also be used to model oracles, additionally given to an algorithm, solving certain computational or decisional problems. In this way, the conditions can also serve as a means to analyze the relationship of cryptographic problems. More precisely, the conditions can help us to prove that there exists no generic (polynomial-time) reduction between certain problems: a problem $\mathcal{P}$ is *not* (Ω, q)-generically-reducible to a problem $\mathcal{P}'$ if $\mathcal{P}$ is (Ω', q, ν)-GGA-intractable, where ν is a negligible function and Ω' is an operation set containing all operations that Ω does plus an operation implementing a $\mathcal{P}'$-oracle.

In the following we analyze which kind of problem solving oracles can be represented by a polynomial function. Besides distinguishing between the well-known classes of computational and decisional oracles, we also distinguish between *fixed-base oracles* solving problem instances only with respect to fixed generators and *regular oracles* that receive the generators for which the problem instance should be solved as an extra input. More precisely, let us assume that an instance of a problem $\mathcal{P}$ should be solved with the help of a fixed-base oracle for a problem $\mathcal{P}'$. Then this oracle only works with respect to the generators given as part of the problem instance of $\mathcal{P}$. Clearly, fixed-base oracles are restricted versions of the corresponding regular oracles. Thus, such an oracle may only be used to prove or disprove the existence of a reduction from $\mathcal{P}$ to a restricted (fixed-base) version of the problem $\mathcal{P}'$.[5]

We can realize most fixed-base computational oracles as an operation of the form given in Section 3.1, i.e., using a fixed polynomial F over $\mathbb{Z}$. However, we cannot implement regular oracles solving computational problems. For instance, consider the algebraic setting of the w-BDHI problem. Let $g_2, g_3, g_1 = e(g_2, g_3)$ be fixed generators of cyclic groups G_2, G_3, G_1 given as part of a problem instance of the w-BDHI problem. It is important to observe that all group elements are represented by the generic oracle with respect to these generators or more precisely with respect to the isomorphic generator 1 in $(\mathbb{Z}_n, +)$. A fixed-base oracle for the bilinear Diffie-Hellman problem over the three groups receives three elements $g_2^{y_1}, g_2^{y_2}, g_2^{y_3} \in G_2$ as input and returns $e(g_2, g_3)^{y_1 y_2 y_3} \in G_1$.

[5] However, it is easy to see that in the case of the Diffie-Hellman problem both types of oracles are equivalent.

The operation $(\mathcal{O}_{\text{FB-BDH}}, 2, 2, 2, 1, Y_1Y_2Y_3)$ realizes such an oracle. It is easy to see that our practical conditions are still satisfied for the w-BDHI problem if we we add this operation to the operation set considered in Example 4. Thus, the w-BDHI problem remains "hard" in this case. A regular oracle for the BDH problem is given generators $g_2^{y_4} \in G_2, g_3^{y_5} \in G_3$ as an additional input and returns

$$e(g_2^{y_4}, g_3^{y_5})^{y_1'y_2'y_3'} = e(g_2, g_3)^{y_1y_2y_3y_4^{-2}y_5},$$

where y_1', y_2', y_3' are the discrete logarithms of $g_2^{y_1}, g_2^{y_2}, g_2^{y_3}$ to the base $g_2^{y_4}$. Note that we cannot realize this oracle as an operation compliant to our definition since a Laurent polynomial $F = Y_1Y_2Y_3Y_4^{-2}Y_5$ would be required for its representation. However, only "regular" polynomials are permitted for representing operations. This restriction is not an arbitrary decision. Without it it is not clear how to ensure the leak-resistance property.

Besides computational oracles, it is also of interest to realize decisional oracles that can be queried in order to solve a DL-/DH-type problem. For instance, this is needed for the analysis of so-called gap problems (e.g., see [OP01]). To include such oracles in our setting, we need to add a new type of operation similar to the one already known: we allow decisional oracles that that can be represented as a mapping of the form

$$f : G_{s_1} \times \ldots \times G_{s_u} \to \{0, 1\}$$
$$f(g_{s_1}^{y_1}, \ldots, g_{s_u}^{y_u}) = 1 \text{ iff } F(y_1, \ldots, y_u) \equiv 0 \bmod n,$$

where $u \geq 1$, $s_1, \ldots, s_u \in \{1, \ldots, k\}$ are some fixed constants, $F \in \mathbb{Z}[Y_1, \ldots, Y_u]$ is some fixed polynomial, and $g_1, \ldots, g_u$ are generators given as part of a problem instance of the respective DL-/DH-type problem. We denote such a *decision operation* by a tuple $(f, s_1, \ldots, s_u, \{0, 1\}, F)$.

The above formulation is particularly useful as it allows to express decision oracles defined by rational functions. In this way, most fixed-base *and* regular decisional oracles can be implemented in our framework. As an example, consider a regular decisional BDH oracle. This oracle takes $e(g_2, g_3)^{y_6} \in G_1$ as additional input compared to the regular computational oracle described above. It outputs 1 iff

$$e(g_2, g_3)^{y_1y_2y_3y_4^{-2}y_5} = e(g_2, g_3)^{y_6} \iff y_1y_2y_3y_4^{-2}y_5 \equiv y_6 \bmod n\,.$$

Clearly, this is a rational equation. However, testing the validity of this equation is certainly equivalent to testing whether

$$y_1y_2y_3y_5 - y_4^2y_6 \equiv 0 \bmod n$$

is satisfied. Thus, we can realize a regular decision BDH oracle in our model by means of the decision operation $(\mathcal{O}_{\text{DBDH}}, 2, 3, 2, 2, 2, 1, \{0, 1\}, Y_1Y_2Y_3Y_5 - Y_4^2Y_6)$.

Clearly, a decision operation constitutes another source of information for a generic algorithm. Thus, we have to adapt the definition of leak-resistance slightly: Let us assume that any (Ω, q)-SLP-generator additionally outputs a list L_0 which is initially empty and to which polynomials $P := F(P_1, \ldots, P_u)$, where $P_1 \in L_{s_1}, \ldots, P_u \in L_{s_u}$, may be added that result from applying any decision operation $(f, s_1, \ldots, s_u, \{0, 1\}, F)$ contained in Ω. Similar to equalities between computed group elements (cf. Section 4), an equality $P(\mathbf{x}) \equiv 0 \bmod n$ verified by a decision operation yields no information about particular choices $\mathbf{x}$ if it holds for *all* elements from $\mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c}$. This observation leads to the following extended version of the leak-resistance property:

Definition 13 (Leak-resistance). *A DL-/DH-type problem $\mathcal{P}$ is called (Ω, q, ν)-leak-resistant if for all (Ω, q)-SLP-generators $\mathcal{S}$ and $\kappa \in \mathbb{N}$ we have*

$$\begin{aligned}
\Pr[&\exists i > 0, P, P' \in L_i \textit{ s.t. } (P - P')(\mathbf{x}) \equiv 0 \bmod n \ \wedge \ P - P' \notin \mathcal{I}_n \textit{ or}\\
&\exists P \in L_0 \textit{ s.t. } P(\mathbf{x}) \equiv 0 \bmod n \ \wedge \ P \notin \mathcal{I}_n :\\
&\quad (n, \mathbf{I}, Q) \xleftarrow{R} \mathsf{SIGen}_{\mathcal{P}}^{(n,\mathbf{I},Q)}(\kappa);\\
&\quad (L_0, L_1, \ldots, L_k) \xleftarrow{R} \mathcal{S}(\kappa, n, \mathbf{I}, Q);\\
&\quad \mathbf{x} \xleftarrow{R} \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell - c}\\
&] \le \nu(\kappa)
\end{aligned}$$

It is not hard to see that no extra (practical) condition (in addition to Conditions 1 and 2) is needed to ensure that computations remain leak-resistant.[6] Just to clarify, the new operation type does not need to be represented in an operation set graph and correspondingly Condition 1 only concerns an operation set excluding decision operations. We can immediately bound the probability of leakage due to decision operations leading to the leak-resistance bound ν given in Theorem 6. Note that compared to the previous bound, only the negligible function $dq(\kappa)$, where d is a constant, has been added.

Theorem 6. *Let Ω be an operation set such that Condition 1 is satisfied and let $d \in \mathbb{N}$ be a bound on the degree of any polynomial describing a decision operation in Ω (we set $d = 0$ if no such operation exists). Furthermore, let $\mathcal{P}$ be a DL-type or DH-type problem satisfying Condition 2. Then for any $q \in \mathsf{poly}(x)$, the problem $\mathcal{P}$ is (Ω, q, ν)-leak-resistant, where*

$$\nu(\kappa) = \frac{1}{2^\kappa}(\ell - c + 1)(k(q(\kappa) + z(\kappa))^2 + dq(\kappa))D(n_1, n_2, u_{\max}, r_1(\kappa), q(\kappa))$$

is a negligible function.

Informally speaking, the above theorem implies that any DL-/DH-type problem that is hard for GGAs (by satisfying our practical conditions) remains being hard even if we additionally admit any decision oracle representable by an operation of the above type. We therefore conclude that a decision oracle is no significant help in solving a computational problem over black-box groups.

Example 9 (Leak-resistance for w-BDHIP with DBDH oracle). Let us assume we extend the operation set given in Example 4 by the operation

$$(\mathcal{O}_{\mathrm{DBDH}}, 2, 3, 2, 2, 2, 1, \{0, 1\}, Y_1 Y_2 Y_3 Y_5 - Y_4^2 Y_6)$$

implementing a decision BDH oracle. The polynomial in the above tuple is of degree 4. Thus, by Theorem 6 the w-BDHI problem is (Ω, q, ν)-leak-resistant, where

$$\nu(\kappa) = \frac{1}{2^\kappa} 2(2w(\kappa))(3(q(\kappa) + w(\kappa) + 1)^2 + 4w(\kappa))\,.$$

6.2 Rational Functions Specifying Problem Challenges

Our framework defined so far only covers problems where the solution of a problem instance can be represented as a (Laurent) polynomial function of the secret choices. This restriction excludes important problems like the w-strong Diffie-Hellman problem and the w-strong Bilinear Diffie-Hellman problem. Informally speaking, the w-SDH problem can be described as follows: Let G

[6] If we allow decision oracles to depend on the security parameter, i.e., number of inputs u and the form of the polynomial F defining the output, then we would need to demand that $\deg(F)$ is polynomial bounded in κ.

be a cyclic group of prime order p and g a corresponding generator. Then given group elements $g, g^{x^1}, g^{x^2}, \ldots, g^{x^w} \in G$, where $x \in \mathbb{Z}_p^*$, the task is to find $v \in \mathbb{Z}_p^*$ and a group element $a \in G$ such that $a = g^{\frac{1}{(x+v)}}$. Observe that here the solution is defined by a *rational function* of the secret choices *and* the value v that can be chosen freely. If $\frac{1}{(x+v)}$ is not defined over $\mathbb{Z}_p$ for particular x and v (this is the case when $v = -x$), then the problem instance shall be deemed to be unsolved.

We can easily extend our framework in this way. To let the class of DL-/DH-type problem (Definition 1) cover this problem type we do the following: We first need to introduce two additional parameters ℓ' and c' defining the range $\mathbb{Z}_n^{c'} \times (\mathbb{Z}_n^*)^{\ell'-c'}$ from which the algorithm is allowed to choose the value $\mathbf{v}$. Furthermore, we consider structure instance generators SIGen that, instead of $Q \in \mathfrak{L}_n^{(\ell,c)}$, output two Laurent polynomials Q_1 and Q_2 over $\mathbb{Z}_n$ in the variables $X_1, \ldots, X_\ell, V_1, \ldots V_\ell$ where only the variables $X_{c+1}, \ldots, X_\ell$ and $V_{c'+1}, \ldots, V_{\ell'}$ may appear with negative exponents. These polynomials represent a rational function

$$R : (\mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c}) \times (\mathbb{Z}_n^{c'} \times (\mathbb{Z}_n^*)^{\ell'-c'}) \to \mathbb{Z}_n$$
$$(\mathbf{x}, \mathbf{v}) \mapsto \frac{Q_1(\mathbf{x}, \mathbf{v})}{Q_2(\mathbf{x}, \mathbf{v})}.$$

Note that by partially evaluating Q_1 and Q_2 with some $\mathbf{v} \in \mathbb{Z}_n^{c'} \times (\mathbb{Z}_n^*)^{\ell'-c'}$, we obtain two Laurent polynomials $Q_1(\mathbf{X}, \mathbf{v})$ and $Q_2(\mathbf{X}, \mathbf{v})$ from $\mathfrak{L}_n^{(\ell,c)}$.

A problem instance of such an extended DL-/DH-type problem is defined as before. Given a problem instance, the challenge is to output some $\mathbf{v} \in \mathbb{Z}_n^{c'} \times (\mathbb{Z}_n^*)^{\ell'-c'}$ such that $Q_2(\mathbf{x}, \mathbf{v}) \in \mathbb{Z}_n^*$ as well as the element

$$\begin{cases} \frac{Q_1(\mathbf{x},\mathbf{v})}{Q_2(\mathbf{x},\mathbf{v})}, & \text{for a DL-type problem} \\ g_1^{\frac{Q_1(\mathbf{x},\mathbf{v})}{Q_2(\mathbf{x},\mathbf{v})}}, & \text{for a DH-type problem} \end{cases}.$$

Note that this modification is indeed a generalization of our problem class definition. All problems covered by the old definition are also covered by the new definition simply by setting $Q_2 = 1$ and $\ell' = 0$.

Adapting the major part of the framework to the new definition is quite straightforward. In fact, the definition of leak-resistance, the corresponding conditions and theorems stay the same since we did not make any change to the functions describing the relation between the private choices and the public group elements, i.e., the input polynomials. In the following, we only sketch important differences in comparison to the previous version of the conditions.

For this purpose, we need to introduce some new notation. We denote by

$$\mathfrak{F}(\mathfrak{L}_n^{(\ell,c)}) = \left\{ \frac{Q_1}{Q_2} \mid Q_1, Q_2 \in \mathfrak{L}_n^{(\ell,c)},\ Q_2 \text{ is not a zero-divisor} \right\}$$

the (total) ring of fractions (aka. ring of rational functions) of $\mathfrak{L}_n^{(\ell,c)}$. An element $\frac{a}{b} \in \mathfrak{F}(\mathfrak{L}_n^{(\ell,c)})$ with $a, b \in \mathbb{Z}_n$ is called a *constant fraction*. The ring $\mathfrak{L}_n^{(\ell,c)}$ can be seen as a subring of this ring by identifying $Q \in \mathfrak{L}_n^{(\ell,c)}$ with $\frac{Q}{1} \in \mathfrak{F}(\mathfrak{L}_n^{(\ell,c)})$.

Note that if we evaluate the fraction $\frac{Q_1}{Q_2}$, describing the challenge of an extended DL-/DH-type problem, with some $\mathbf{v} \in \mathbb{Z}_n^{c'} \times (\mathbb{Z}_n^*)^{\ell'-c'}$ we obtain a fraction $\frac{Q_1(\mathbf{X},\mathbf{v})}{Q_2(\mathbf{X},\mathbf{v})}$ that is not necessarily a well-defined element of $\mathfrak{F}(\mathfrak{L}_n^{(\ell,c)})$. This is because $Q_2(\mathbf{X}, \mathbf{v})$ might be a zero divisor in $\mathfrak{L}_n^{(\ell,c)}$. However, we can exclude this case, because by choosing such a fraction (i.e., by selecting this particular $\mathbf{v}$) an algorithm can never solve a problem instance:

Lemma 4. *If $P \in \mathfrak{L}_n^{(\ell,c)}$ is a zero-divisor then for any $\mathbf{x} \in \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c}$ the element $P(\mathbf{x}) \in \mathbb{Z}_n$ is a zero-divisor.*

Proof. It is easy to see that if P is a zero-divisor then there exists an element $a \neq 0 \in \mathbb{Z}_n$ such that $a \cdot P = 0$. Thus, also for any $P(\mathbf{x})$ we have $a \cdot P(\mathbf{x}) = 0$. $\square$

Thus, our conditions only need to address fractions being elements of $\mathfrak{F}(\mathfrak{L}_n^{(\ell,c)})$.

We stipulate the following extended definitions for the SLP-intractability of a DL-type and a DH-type problem, respectively. Note that the SLP-generators now additionally output $\mathbf{v}$ in order to select a specific fraction.

Definition 14 (SLP-intractability of DL-Type Problems). *A DL-type problem $\mathcal{P}$ is called (Ω, q, ν)-SLP-intractable if for all $\kappa \in \kappa$ we have*

$$\begin{aligned}\Pr[&Q_2(\mathbf{x},\mathbf{v}) \in \mathbb{Z}_n^* \text{ and } R(\mathbf{x},\mathbf{v}) \equiv a \bmod n \ : \\ &(n,\mathbf{I},Q_1,Q_2) \xleftarrow{R} \mathsf{SIGen}_{\mathcal{P}}^{(n,\mathbf{I},Q_1,Q_2)}(\kappa); \\ &(\mathbf{v},a,L_0,\dots,L_k) \xleftarrow{R} \mathcal{S}(\kappa,n,\mathbf{I},Q_1,Q_2); \\ &R \leftarrow \frac{Q_1}{Q_2}; \\ &\mathbf{x} \xleftarrow{R} \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c} \\ ] &\leq \nu(\kappa)\end{aligned}$$

Definition 15 (SLP-intractability of DH-type Problems). *A DH-type problem $\mathcal{P}$ is called (Ω, q, ν)-SLP-intractable if for all (Ω, q)-SLP-generators $\mathcal{S}$ and $\kappa \in \mathbb{N}$ we have*

$$\begin{aligned}\Pr[&(Q_2(\mathbf{x},\mathbf{v}) \in \mathbb{Z}_n^* \text{ and } (P - R(\mathbf{X},\mathbf{v}))(\mathbf{x}) \equiv 0 \bmod n \ : \\ &(n,\mathbf{I},Q_1,Q_2) \xleftarrow{R} \mathsf{SIGen}_{\mathcal{P}}^{(n,\mathbf{I},Q_1,Q_2)}(\kappa); \\ &(\mathbf{v},P,L_0,\dots,L_k) \xleftarrow{R} \mathcal{S}(\kappa,n,\mathbf{I},Q_1,Q_2); \\ &R \leftarrow \frac{Q_1}{Q_2}; \\ &\mathbf{x} \xleftarrow{R} \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{\ell-c} \\ ] &\leq \nu(\kappa)\end{aligned}$$

The GGA-intractability of a DL-/DH-type problem is still related in the same way to the leak-resistance property and the SLP-intractability of the problem. That means, Theorem 1 still holds for our extension and can reproved very easily.

To ensure SLP-intractability, we require Condition 7 and 8 for DL-type problems and Condition 7 and 9 for DH-type problems to be satisfied.

Condition 7 *There exists $r_2 \in \mathsf{poly}(x)$ such that for all $\kappa \in \mathbb{N}$, $(n, Q_1, Q_2) \in [\mathsf{SIGen}_{\mathcal{P}}^{(n,Q_1,Q_2)}(\kappa)]$, and $\mathbf{v} \in \mathbb{Z}_n^{c'} \times (\mathbb{Z}_n^*)^{\ell'-c'}$:*

$$\max\{\deg(Q_1(\mathbf{X},\mathbf{v})), \deg(Q_1(\mathbf{X},\mathbf{v}))\} \leq r_2(\kappa).$$

Condition 8 (Non-Triviality of DL-type Problems) *There exists $\kappa_0 \in \mathbb{N}$ such that for all $\kappa \geq \kappa_0$, $(n, Q_1, Q_2) \in [\mathsf{SIGen}_{\mathcal{P}}^{(n,Q_1,Q_2)}(\kappa)]$, and $\mathbf{v} \in \mathbb{Z}_n^{c'} \times (\mathbb{Z}_n^*)^{\ell'-c'}$ we have that $\frac{Q_1(\mathbf{X},\mathbf{v})}{Q_2(\mathbf{X},\mathbf{v})}$ is not a constant fraction in $\mathfrak{F}(\mathfrak{L}_n^{(\ell,c)})$ (provided that it is a well-defined element, i.e., $Q_2(\mathbf{X},\mathbf{v})$ is not a zero-divisor).*

Condition 9 (Non-Triviality of DH-type Problems) *For every $q \in \mathsf{poly}(x)$ there exists $\kappa_0 \in \mathbb{N}$ such that for all $\kappa \geq \kappa_0$, (Ω, q)-SLP-generators $\mathcal{S}$, $(n, \mathbf{I}, Q_1, Q_2) \in [\mathsf{SIGen}_{\mathcal{P}}^{(n,\mathbf{I},Q_1,Q_2)}(\kappa)]$, $(P, L_0, \ldots, L_k) \in [\mathcal{S}(\kappa, n, \mathbf{I}, Q_1, Q_2)]$, and $\mathbf{v} \in \mathbb{Z}_n^{c'} \times (\mathbb{Z}_n^*)^{\ell'-c'}$ we have that $\frac{Q_1(\mathbf{X},\mathbf{v})}{Q_2(\mathbf{X},\mathbf{v})} \neq P$ in $\mathfrak{F}(\mathfrak{L}_n^{(\ell,c)})$ (provided that it is a well-defined element, i.e., $Q_2(\mathbf{X}, \mathbf{v})$ is not a zero-divisor).*

These conditions imply (Ω, q, ν)-SLP-intractability for the same negligible functions ν as stated in Theorems 4 and 5. To reprove Theorem 4, one needs to observe that the conditions ensure that for any a and non-zero-divisor $Q_2(\mathbf{X}, \mathbf{v})$ the Laurent polynomial

$$Q_1(\mathbf{X}, \mathbf{v}) - aQ_2(\mathbf{X}, \mathbf{v})$$

is non-zero and of total degree bounded by r_2. Similarly, for the proof of Theorem 5 we observe that for any P and non-zero-divisor $Q_2(\mathbf{X}, \mathbf{v})$ the Laurent polynomial

$$Q_1(\mathbf{X}, \mathbf{v}) - PQ_2(\mathbf{X}, \mathbf{v})$$

is non-zero and of total degree bounded by $r_1 + r_2$.

Example 10 (SLP-intractability of w-SDHP). For the w-SDH problem we have parameters

$$Param_{w-\mathrm{SDH}} = (k = 1, \ell = 1, c = 0, \ell' = 1, c' = 0, z = w + 1)$$

and a structure instance generator $\mathsf{SIGen}_{w-\mathrm{SDH}}$ that on input κ returns

$$((\mathbf{G} = G_1, \mathbf{g} = g_1, n = p), (\mathbf{I} = \mathbf{I_1} = \{1, X_1^1, \ldots, X_1^{w(\kappa)}\}, Q_1 = 1, Q_2 = X_1 + V_1))\,.$$

Note that for any $v_1 \in \mathbb{Z}_p^*$, the fraction

$$\frac{Q_1(\mathbf{X}, \mathbf{v})}{Q_2(\mathbf{X}, \mathbf{v})} = \frac{1}{X_1 + v_1}$$

is an element of $\mathfrak{F}(\mathfrak{L}_n^{(\ell,c)})$ but not an element of the subring $\mathfrak{L}_n^{(\ell,c)}$. Hence, Condition 9 is trivially satisfied, since P is always a Laurent polynomial (independently of the considered operation set Ω). Condition 7 is satisfied since we always have

$$\max\{\deg(Q_1(\mathbf{X}, \mathbf{v}), \deg(Q_2(\mathbf{X}, \mathbf{v}))\} = 1 =: r_2(\kappa)\,.$$

As we can easily see, Conditions 1 and 2 hold assuming an operation set containing operations for performing the group law and inversion of elements in G_1, i.e.,

$$\Omega = \{(\circ, 1, 1, 1, Y_1 + Y_2), (inv, 1, 1, -Y_1)\}$$

yielding the upper bound $D(0, 0, 0, w(\kappa), q(\kappa)) = w(\kappa)$ on the degrees of the polynomials $P \in L_1$. Thus, the problem is (Ω, q, ν)-SLP-intractable, where

$$\nu(\kappa) = \frac{w(\kappa) + 1}{2^\kappa}\,.$$

Acknowledgements. The authors would like to thank Daniel Brown for his comments. The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The information in this document reflects only the authors' views, is provided as is and no guarantee or warranty that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

References

[BB04a] D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In *Advances in Cryptology: Proceedings of EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer-Verlag, 2004.

[BB04b] D. Boneh and X. Boyen. Short signatures without random oracles. In *Advances in Cryptology: Proceedings of EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer-Verlag, 2004.

[BBG05a] D. Boneh, X. Boyen, and E. Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology: Proceedings of EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer-Verlag, 2005.

[BBG05b] D. Boneh, X. Boyen, and E. Goh. Hierarchical identity based encryption with constant size ciphertext (full paper). Cryptology ePrint Archive, Report 2005/015, 2005. `http://eprint.iacr.org/2005/015`.

[BDZ03] F. Bao, R. H. Deng, and H. Zhu. Variations of Diffie-Hellman problem. In *ICICS'03*, volume 2971 of *Lecture Notes in Computer Science*, pages 301–312. Springer-Verlag, 2003.

[BF01] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology: Proceedings of CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer-Verlag, 2001.

[BGLS03] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology: Proceedings of EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer-Verlag, 2003.

[BLMW07] E. Bresson, Y. Lakhnech, L. Mazaré, and B. Warinschi. A generalization of DDH with applications to protocol analysis and computational soundness. In A. J. Menezes, editor, *Advances in Cryptology: Proceedings of CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 482–499. Springer-Verlag, August 2007.

[Bon07] D. Boneh. Number-theoretic assumptions. Invited Talk at TCC's Special Session on Assumptions for Cryptography, 2007.

[Bro06] D. L. Brown. Breaking RSA may be as difficult as factoring. Cryptology ePrint Archive, Report 2005/380, 2006. `http://eprint.iacr.org/2005/380`.

[BV98] D. Boneh and R. Venkatesan. Breaking RSA may not be equivalent to factoring. In *Advances in Cryptology: Proceedings of EUROCRYPT 1998*, volume 1403 of *Lecture Notes in Computer Science*, pages 59–71. Springer-Verlag, 1998.

[Che06] J. H. Cheon. Security analysis of the strong Diffie-Hellman problem. In *Advances in Cryptology: Proceedings of EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 1–11. Springer-Verlag, 2006.

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press/McGraw-Hill, 1990.

[Den02] A. W. Dent. Adapting the weaknesses of the random oracle model to the generic group model. In *Advances in Cryptology: Proceedings of ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 100–109. Springer-Verlag, 2002.

[DK02] I. Damgård and M. Koprowski. Generic lower bounds for root extraction and signature schemes in general groups. In *Advances in Cryptology: Proceedings of EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 256–271. Springer-Verlag, 2002.

[Kil01] E. Kiltz. A tool box of cryptographic functions related to the Diffie-Hellman function. In *INDOCRYPT '01: Proceedings of the Second International Conference on Cryptology in India*, volume 2247 of *Lecture Notes in Computer Science*, pages 339–350. Springer-Verlag, 2001.

[Mau05] U. Maurer. Abstract models of computation in cryptography. In Nigel Smart, editor, *Cryptography and Coding 2005*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag, 2005.

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

[OP01] T. Okamoto and D. Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In *PKC '01: Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 104–118. Springer-Verlag, 2001.

[PH78] S. Pohlig and M. Hellman. An improved algorithm for computing discrete logarithms over GF(p) and its cryptographic significance. *IEEE Transactions on Information Theory*, 24:106–110, 1978.

[Pol78] J. M. Pollard. Monte carlo methods for index computation mod p. *Mathematics of Computation*, 32:918–924, 1978.

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the Association for Computing Machinery*, 27(4):701–717, 1980.

[Sho97a] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology: Proceedings of EUROCRYPT 1997*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer-Verlag, 1997.

[Sho97b] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997.

[Sho06] Victor Shoup. Sequences of games: A tool for taming complexity in security proofs. IACR eprint report 2004/332, 2006.

[SS01] A. Sadeghi and M. Steiner. Assumptions related to discrete logarithms: Why subtleties make a real difference. In *Advances in Cryptology: Proceedings of EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 243–260. Springer-Verlag, 2001.

[Yac02] Y. Yacobi. A note on the bilinear Diffie-Hellman assumption. Cryptology ePrint Archive, Report 2002/113, 2002. `http://eprint.iacr.org/2002/113`.