Sufficient Conditions for Computational Intractability Regarding Generic Algorithms

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 group model is a valuable methodology for analyzing the computational hardness of the number-theoretic problems used in cryptography. Although generic hardness proofs exhibit many similarities, still the computational intractability of every newly introduced problem needs to be proven from scratch, a task that can easily become complicated and cumbersome when done rigorously. In this paper we make the first steps towards overcoming this problem by identifying verifiable criteria which if met by a cryptographic problem guarantee its hardness with respect to generic algorithms. As useful means for formalization of definitions and proofs we relate the concepts of generic algorithms and straight-line programs that have only been used independently in cryptography so far.

The class of problems we cover includes a significant number of the cryptographic problems currently known, and is general enough to also include many future problems. Moreover, we strengthen the conventional generic model by incorporating a broader class of possible oracles (operations) since the underlying algebraic groups may possibly be related through mappings such as isomorphisms, homomorphisms or multilinear maps. Our approach could serve as an appropriate basis for tool-aided hardness verification in the generic model.

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

1 Introduction

The *generic group model* was introduced by Nechaev [Nec94] and Shoup [Sho97]. 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.

A variety of 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) [Sho97] 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].

As further application, the generic model has been used to analyze relations between several cryptographic assumptions [MW98,SS01,LR06] as well as for proving security properties of cryptographic schemes [SJ00,AdM04,Bro05]. However, when used beyond the analysis of assumptions, as it is done in the latter case, the generic model is known to have some pitfalls [Fis00,SPMLS02,Den02] similar to the random oracle model.

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,BB04,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 group model and for each novel problem an entire hardness proof needs to be redone 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. However, whether a problem is intractable for generic algorithms is not obvious and the properties that ensure generic hardness have not been well-studied yet.

Our contributions. In a nutshell, we identify the core aspects making cryptographic problems hard in the generic model. We provide a set of conditions, which given the description of a cryptographic problem allow to check whether the problem at hand is intractable with respect to generic algorithms performing certain operations. In this way we aim at (i) providing means 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 introduce quite comprehensive classes of computational problems over *known order* groups, we call DL-type and DH-type problems. We consider a comprehensive list of well-known cryptographic problems falling into these classes in Section 3. Given the broadness of these classes, we expect that similar future computational problems will also comply with them.

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 the operations one decides to make available. We note that currently no concrete security bounds are provided. Deriving such bounds is difficult due to the generality of our approach, however we think it is nevertheless possible and leave it as future work.

In this context, we explore 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. This relation provides useful means for formalizing the definitions and proofs. 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 analyzed before.

As a further interesting result, one of our theorems (cf. Section 6) imply that most decision oracles do not significantly help generic algorithms in solving any hard DL- or DH-type problem.

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 concerning the P-DH problem it is shown there that the computational and decisional variant of this problem class is hard in the 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 the generic model. 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. Maurer also considers the case of bounded storage and the presence of side information in terms of decision oracles. In contrast to our framework this model does not cover the case of multiple groups. [Mau05] proves lower bounds on the computation complexity for several specific examples within his setting, however the paper does not give sufficient nor necessary conditions for the hardness of problem classes in the framework. Though, we believe that our results carry over to this 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 either quite restricted classes of cryptographic problems and/or a fixed algebraic settings. 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.

2 Some Preliminaries

Let $\mathsf{poly}(x)$ denote the class of univariate polynomials in x with non-negative integer coefficients. We call a function $f : \mathbb{N} \to \mathbb{R}^+$ *polynomial bounded* if there exists $poly \in \mathsf{poly}(x)$ s.t. for all sec: $f(sec) \leq poly(sec)$. For convenience, we sometimes treat such an f as polynomial function. 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., $\forall poly \in \mathsf{poly}(x)\, \exists sec_0\, \forall sec \geq sec_0 : f(sec) \leq \frac{1}{poly(sec)}$.

Throughout the paper we are concerned with regular multivariate polynomials and multivariate Laurent polynomials over the ring $\mathbb{Z}$ and $\mathbb{Z}_n$, respectively. For a regular polynomial $F = \sum a_{\alpha_1,\ldots,\alpha_u} Y_1^{\alpha_1} \cdots Y_u^{\alpha_u} \neq 0 \in \mathbb{Z}[Y_1, \ldots, Y_u]$ the *total degree* is defined as $\deg(F) = \max\{\sum_i \alpha_i \mid a_{\alpha_1,\ldots,\alpha_u} \neq 0\}$ (and $\deg(0) = -\infty$) and the *degree in a variable* Y_j as $\deg_{Y_j}(F) = \max\{\alpha_j \mid a_{\alpha_1,\ldots,\alpha_u} \neq 0\}$. Roughly speaking, *Laurent polynomials* are polynomials whose variables may also have negative exponents. More precisely, a Laurent polynomial P over $\mathbb{Z}_n$ in indeterminates $X_1, \ldots, X_l$ is a finite sum $P = \sum a_{\alpha_1,\ldots,\alpha_l} X_1^{\alpha_1} \cdots X_l^{\alpha_l}$ 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 $\deg(P) = \max\{\sum_i |\alpha_i| \mid a_{\alpha_1,\ldots,\alpha_l} \neq 0 \bmod n\}$ (and $\deg(0) = -\infty$) we denote the *(absolute) total degree* and by $\deg_{X_j}(P) = \max\{|\alpha_j| \mid a_{\alpha_1,\ldots,\alpha_l} \neq 0 \bmod n\}$ the *degree in a variable* X_j of a Laurent polynomial $P \neq 0$. Furthermore, we denote by $\mathcal{L}_n^{(l,c)}$ (where $0 \leq c \leq l$) the subring of Laurent polynomials over $\mathbb{Z}_n$ where only the variables $X_{c+1}, \ldots, X_l$ can appear with negative exponents. Note that for any $P \in \mathcal{L}_n^{(l,c)}$ and $priv = (x_1, \dots, x_l) \in \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{l-c}$ the evaluation $P(priv)$, where we set $X_1 = x_1, \dots, X_l = x_l$ in P, is well-defined.

3 Problem Classes: Discrete Log and Diffie-Hellman Type of Problems

In this section we introduce the classes of computational problems for which we prove the existence of hardness conditions in the generic model. A large class of computational problems defined over a single known order group G can generally be described as follows: Let g be a generator of G and $l, z \in \mathbb{N}$. Given a tuple $(g, a_1, \dots, a_z)$ of (public) elements from G which are related to some secret random choices $priv = (x_1, \dots, x_l)$ from $\mathbb{Z}_{|G|}$ or $\mathbb{Z}_{|G|}^*$ 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 *priv*. For instance, consider the description of the CDH problem: Given $(g, a_1 = g^{x_1}, a_2 = g^{x_2}) \in G^3$, where $priv = (x_1, x_2) \in_U \mathbb{Z}_{|G|}^2$ the challenge is to compute the element $a = g^{x_1 x_2}$.

Generally, the relation between the secret choices *priv* and the public elements of a problem can be represented by a set of functions $I_1, \dots, I_z : \mathbb{Z}_{|G|}^c \times (\mathbb{Z}_{|G|}^*)^{l-c} \to \mathbb{Z}_{|G|}$ (where $c = l$ in above example), i.e., $a_j = g^{I_j(priv)}$. In this paper we restrict our focus to Laurent polynomials $I_j \in \mathcal{L}_n^{(l,c)}$. We call them *input polynomials*. Similarly, we represent the relation between the solution $a \in G$ resp. $x \in \mathbb{Z}_{|G|}$ and *priv* by a Laurent polynomial $Q \in \mathcal{L}_n^{(l,c)}$, i.e., $a = g^{Q(priv)}$ resp. $x \equiv Q(priv) \bmod \mathbb{Z}_{|G|}$. We call Q the *challenge polynomial*. We summarize the considered problem classes in Definition 1.

Definition 1 (DL-type and DH-type problem). *A* DL-type *and* DH-type problem*, respectively, is a problem of the following form:*
Given:
- *Constants: $k, l \in \mathbb{N}, c \in \mathbb{N}_0$ where $c \leq l$.*
- *Cyclic groups and generators: $G_1, \dots, G_k$ of known order n with corresponding generators $g_1, \dots, g_k$ that are output by some probabilistic algorithm* $\texttt{GGen}(sec)$ *where $sec \in \mathbb{N}$ is a security parameter.*
- *Laurent polynomials: $I_{11}, \dots, I_{1z}, I_{21}, \dots, I_{2z}, \dots, I_{k1}, \dots, I_{kz} \in \mathcal{L}_n^{(l,c)}$ and $Q \in \mathcal{L}_n^{(l,c)}$ that are output by some deterministic algorithm* $\texttt{PGen}(sec, n)$*, where $z = z(sec)$ is polynomial bounded in sec. We call the I_{ij} the* input polynomials *and Q the* challenge polynomial.
- *Group elements: $(g_1^{I_{11}(priv)}, \dots, g_1^{I_{1z}(priv)}), (g_2^{I_{21}(priv)}, \dots, g_2^{I_{2z}(priv)}), \dots, (g_k^{I_{k1}(priv)}, \dots, g_k^{I_{kz}(priv)})$, where $priv = (x_1, \dots, x_l) \in_U \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{l-c}$ are* secret choices *and not given.*

Find:
- *For a DL-type problem: an integer x such that $x \equiv Q(priv) \bmod n$.*
- *For a DH-type problem: an element $a \in G_1$ such that $a = g_1^{Q(priv)}$.*

A large number of cryptographically relevant problems fall into the classes of DL-type and 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], l-weak and l-strong DHP [Che06], bilinear DHP [BF01], q-bilinear DH inversion problem [BB04], q-bilinear DH exponent problem [BBG05a], co-DHP [BGLS03], co-bilinear DHP [BF01], bilinear pairing inversion problem [Yac02] and many more. As an illustration of the definition, we consider two examples of DL-/DH-type problems in more detail. The values of the parameters k, l, c, $\texttt{PGen}(.,.)$, and $\texttt{GGen}(.)$ found in Definition 1 are implied by the following examples and therefore not explicitly indicated.

Example 1 (Discrete Logarithm Problem).
Given: A cyclic group G_1 of order n, a generator g_1, an input polynomial $I_{11} = X_1$, the challenge polynomial $Q = X_1$, and a group element $g_1^{x_1} = g_1^{I_{11}(priv)}$, where $priv = x_1 \in_U \mathbb{Z}_n$.
Find: An integer x such that $x \equiv x_1 \equiv Q(priv) \bmod n$.

Another more comprehensive example of a DH-type problem (which will be used in the following to illustrate our conditions), is the q-BDHI problem originally defined in [BB04]. We note that in contrast to the standard DL or CDH problem, the number of input polynomials and their degrees are not constant values for the q-BDHIP but a polynomial bounded function in sec.

Example 2 (q-Bilinear Diffie-Hellman Inversion Problem).
Given: Cyclic groups G_1, G_2, G_3 of prime order n where there exists a non-degenerate efficiently computable bilinear mapping $e : G_2 \times G_3 \to G_1$ and isomorphism $\psi : G_2 \to G_3$, generators g_1, g_2, g_3 (such that $e(g_2, g_3) = g_1$ and $\psi(g_2) = g_3$), input polynomials $I_{11} = \ldots = I_{1q} = 1$, $I_{21} = X_1, I_{22} = X_1^2, \ldots, I_{2q} = X_1^q$, $I_{31} = \ldots = I_{3q} = 1$ for some polynomial bounded $q = q(sec)$, the challenge polynomial $Q = X_1^{-1}$, and group elements $g_1 = g_1^{I_{11}(priv)} = \ldots = g_1^{I_{1q}(priv)}$, $g_2^{I_{21}(priv)} = g_2^{x_1}, \ldots, g_2^{I_{2q}(priv)} = g_2^{x_1^q}$, and $g_3 = g_3^{I_{31}(priv)} = \ldots = g_3^{I_{3q}(priv)}$, where $priv = x_1 \in_U \mathbb{Z}_n^*$.
Find: An element $a \in G_1$ such that $a = g_1^{x_1^{-1}} = g_1^{Q(priv)}$.

4 Generic Algorithms

In the generic model an algorithm $\mathcal{A}$ is given access to a *generic group oracle* $\mathcal{O}$ which provides generic group operations to $\mathcal{A}$. Our model is an extension of Maurer's variant of the generic model [Mau00] informally described in the following:[1] Elements of a group which are inputs to $\mathcal{A}$ as part of a problem instance are stored in a list not directly accessible by $\mathcal{A}$. $\mathcal{A}$ is simply given indices into the list as handles to these group elements. It can perform operations on desired elements by issuing queries to $\mathcal{O}$ containing the respective indices. $\mathcal{O}$ then performs the requested operation, appends the resulting element to the list and returns a handle to it. However, in this way a group element is not uniquely represented by its handle, i.e., an algorithm given two indices cannot decide by itself whether the referenced group elements are equal or not. To this end, each time an element is appended to the list, not only the index to this element is given to $\mathcal{A}$ but a set, called *equality set*, containing the indices to all elements in the list which equal the new element.

Note that 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 $\mathcal{O}$ should offer all operations to $\mathcal{A}$ that are known to be efficiently computable in this algebraic setting and do not make intrinsic use of the encoding of group elements. In Section 4.1 we define the class of generic operations supported by our model.

4.1 Generic Operation Sets

Given a problem and its underlying algebraic setting, we call the operations being made available in the generic 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

$$\begin{aligned} f : G_{s_1} \times \ldots \times G_{s_u} &\to G_d \\ (a_1, \ldots, a_u) &\mapsto f(a_1, \ldots, a_u) \end{aligned}$$

[1] Though the variant of the generic model due to Shoup [Sho97] is more popular, we decided to use Maurer's variant because it induces less technical overhead concerning our formalizations. However, we note that our results hold unchanged in Shoup's model (see also Appendix A).

where $u \geq 1$, $s_1, \dots, s_u, d \in \{1, \dots, k\}$ are some fixed constants (that does not depend on *sec*). Furthermore, we demand that the action of f on the group elements can be represented by a fixed regular polynomial. That means, there exists a fixed $F \in \mathbb{Z}[Y_1, \dots, Y_u]$ (also not depending on *sec*) such that for any generators $g_1, \dots, g_u, g_d$ given as part of a problem instance hold

$$f(a_1, \dots, a_u) = g_d^{F(y_1, \dots, y_u)}$$

where $a_1 = g_1^{y_1}, \dots, a_u = g_u^{y_u}$. For instance, the bilinear mapping $e : G_2 \times G_3 \to G_1$ which is part of the algebraic setting of the q-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 $F = Y_1 Y_2$. 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.

We represent an operation of the above form by a tuple $(f, s_1, \dots, s_u, d, F)$, where the first component serves as identifier for 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 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, denoted by OPSet, is a finite set of tuples of the form $(f, s_1, \dots, s_u, d, F)$, where the components of these tuples are defined as above.*

Example 3 (Operation Set for q-BDHIP). The operation set $OPSet = \{(\circ_1, 1, 1, 1, Y_1 + Y_2), (\circ_2, 2, 2, 2, Y_1 + Y_2), (\circ_3, 3, 3, 3, Y_1 + Y_2), (inv_1, 1, 1, -Y_1), (inv_2, 2, 2, -Y_1), (inv_3, 3, 3, -Y_1), (\psi, 2, 3, Y_1), (e, 2, 3, 1, Y_1 \cdot Y_2)\}$ 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 $e : G_2 \times G_3 \to G_1$.

4.2 Generic Group Oracle

We define a generic oracle for DL-/DH-type problems based on Maurer's approach [Mau00]:

Definition 3 (Generic Group Oracle for DL- and DH-type Problems). *Let be given a DL- or DH-type problem $\mathcal{P}$. Let $G_1, \dots, G_k$ be groups of order n which are output of* GGen(.) *on input sec and let OPSet denote a generic operation set for $\mathcal{P}$ on $G_1, \dots, G_k$. Then a* generic group oracle *for $\mathcal{P}$, denoted by $\mathcal{O}_{\mathcal{P},OPSet}$, has an input and output port and performs computations as follows.*
Internal state. *As internal state $\mathcal{O}_{\mathcal{P},OPSet}$ maintains lists $L_1, \dots, L_k$, where a list L_i is used for storing computed elements from $\mathbb{Z}_n \cong G_i$. By L_{ij} we denote the j-th element of L_i.*
Input. *As input $\mathcal{O}_{\mathcal{P},OPSet}$ is given sec, n and secret choices priv belonging to an instance of $\mathcal{P}$.*
Computation of equality sets. *Each time an element a is appended to a list L_i the following computation is triggered: $\mathcal{O}_{\mathcal{P},OPSet}$ computes the* equality set

$$EQS(i, a) = \{j \in \{1, \dots, |L_i|\} \mid a - L_{ij} \equiv 0 \bmod n\},$$

and writes it to its output port.
The computation of $\mathcal{O}_{\mathcal{P},OPSet}$ consists of an initialization phase, which is run once, followed by the execution of the query-handling phase:
Initialization. *Each list L_i is initialized with the elements $1, I_{i1}(priv), \dots, I_{iz}(priv)$ (representing the given group elements) where $(I_{11}, \dots, I_{kz}) \leftarrow$* PGen*$(sec, n)$.*
Query-handling. *Upon receiving a query $(f, j_1, \dots, j_u)$ on its input port, where $(f, s_1, \dots, s_u, d, F) \in OPSet$, $\mathcal{O}_{\mathcal{P},OPSet}$ computes the element $F(L_{s_1 j_1}, \dots, L_{s_u j_u})$ and appends it to the list L_d.*

Now, a *generic algorithm* is an interactive probabilistic Turing machine $\mathcal{A}$ that is given access to $\mathcal{O}_{\mathcal{P},OPSet}$, i.e., which is connected to the input and output port of $\mathcal{O}_{\mathcal{P},OPSet}$. We say that $\mathcal{A}$ has solved a problem instance of a DL-type problem $\mathcal{P}$ if for its final output, denoted by *out*, holds that $Q(priv) - out \equiv 0 \bmod n$. In the case that $\mathcal{P}$ is a DH-type problem, *out* is interpreted as an index into the list L_1 and $\mathcal{A}$ is said to solve a problem instance iff $Q(priv) - L_{1out} \equiv 0 \bmod n$.

5 Hardness Conditions

In this section we present conditions on DL-/DH-type problems and their corresponding operation sets that if met guarantees a problem's intractability regarding generic algorithms. For formalizing these conditions we we make use of the concept of straight-line programs defined in Section 5.1. The first collection of conditions presented in Section 5.2 expresses what is actually needed in a hardness proof. However, since these conditions are of rather abstract nature more restrictive conditions are developed in Section 5.3 which can be verified more easily for a given problem.

It is easy to see that in order to assess the intractability of DL-/DH-type problems we can restrict to consider groups of prime power order (see Appendix B). So let us assume in the following that $\texttt{GGen}(sec)$ generates groups of order $n = p^e$ such that p is a prime that is exponential in the given security parameter *sec* (e.g., $2^{sec-1} < p < 2^{sec}$) and e is constant over all *sec*. Let the finite set of all possible group orders for a fixed value of *sec* be denoted by $\mathcal{N}^{(sec)}$.

5.1 Generic Algorithms and Straight-Line Programs

Roughly speaking, a DL-/DH-type problem is intractable for generic algorithms if two conditions are satisfied: First, in the course of a computation virtually no information about the secret choices *priv* must be gainable by means of equality testing, i.e., equality sets must hardly ever contain useful information for solving a problem instance. Second, in this case the hardness of the problem w.r.t. general generic algorithms can be reduced to the hardness of the problem for generic algorithms that take their actions independently of (in)equalities of computed group elements and thus independent of the specific problem instance.

More precisely, we can restrict our considerations to the subclass of non-probabilistic generic algorithms that for fixed n always apply the *same fixed sequence of operations* from the given *OPSet* and output the *same fixed value* out_n in order to solve an arbitrary problem instance. Thus, for fixed n such an algorithm is fully specified by the constant out_n and the sequence of applied group operations. The sequence of operation applications can be viewed as a so-called *straight-line program (SLP)* and represented as a k-tuple $SLP_n = (L_1, \ldots, L_k)$ of finite lists of certain Laurent polynomials from $\mathcal{L}_n^{(l,c)}$. These polynomials result from applying the respective fixed sequence of F-functions corresponding to the operations (cf. Definition 2) to the given input polynomials. If we evaluate a straight-line program SLP_n, i.e., all polynomials in $L_1, \ldots, L_k$, with the private choices *priv* corresponding to a problem instance, we obtain exactly the group elements that would occur during an interaction of the respective generic algorithm and the oracle (as internal state).

Hence, regarding the applied group operations a generic algorithm from the considered subclass can be seen as an *SLP-generator* of the form depicted in Algorithm 1. The function $m(.)$ is the run-time function of the SLP-generator. Using the deterministic function $\mathsf{select}(.)$ the SLP-generator (somehow) selects in each iteration which operation should be applied to which polynomials.

For formalizing our conditions we need the following set. Given some $m(.)$ the set

$$SLPGen^{(m)} := \left\{\text{SLP-generator}\, \mathcal{S} \mid \forall sec : m'(sec) \leq m(sec),\ \text{where}\, m'(.)\, \text{is the run-time of}\ \mathcal{S}\right\}$$

consists of all SLP-generators that run in time at most $m(sec)$ for all values of *sec*. It is important to observe that all possible sequences of group operations that could ever be applied in run-time at most $m(.)$ are also covered by the SLP-generators in this set.

Algorithm 1 SLP-Generator $\mathcal{S}(sec, n)$

```
Input:   security parameter $sec$, group order $n$
Output: a straight-line program $SLP_n = (L_1, \dots, L_k)$
  $(I_{11}, \dots, I_{kz}, Q) \leftarrow \texttt{PGen}(sec, n)$;
  $L_1 \leftarrow [1, I_{11}, \dots, I_{1z}]; \dots; L_k \leftarrow [1, I_{k1}, \dots, I_{kz}]$;
  for $i \leftarrow 1$ to $m(sec)$ do
     $(f, j_1, \dots, j_u) \leftarrow \mathsf{select}(i, sec, n, L_1, \dots L_k, Q)$;
     if $f \neq \perp$ and $(f, s_1, \dots, s_u, d, F) \in OPSet$ then $\mathsf{append}(L_d, F(L_{s_1 j_1}, \dots, L_{s_u j_u}))$;
  end for
  return $SLP_n = (L_1, \dots, L_k)$;
```

5.2 Abstract Conditions

In this section we formalize our first set of hardness conditions. For short, we demand that not too much information about the secret choices is leaked by computable group elements (*leak-resistance*) and that the considered problem is not solvable in a trivial way (*SLP-intractability*). We can formulate both conditions using SLP-generators.

In the generic model described before, the only way to possibly obtain information about the secrets is due to equalities or inequalities between computed elements, since these relations are the only output of the oracle. But when do group elements actually leak information? Let a generic algorithm $\mathcal{A}$ for some DL-/DH-type problem be given whose run-time is upper bounded by the function $m(.)$. Consider a run of $\mathcal{A}$ for fixed sec, $n \in \mathcal{N}^{(sec)}$ and secret choices $priv$. Then observe that all group elements computed in this run can also be generated by evaluating the straight-line program $SLP_n \leftarrow \mathcal{S}(sec, n)$ with $priv$ which is the output of some $\mathcal{S} \in SLPGen^{(m)}$. So we can analyze the leakage of a run by considering the polynomials of an SLP. Two computed group elements $P(priv)$ and $P'(priv)$ ($P, P' \in L_i$) are equal if $P(priv) - P'(priv) \equiv 0 \bmod n$. However, it is clear that such a relation yields no information about the particular choices $priv$ if it holds for all elements of $\mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{l-c}$. Otherwise, we obtain the non-trivial information that $priv$ is a modular root of the difference polynomial $P - P'$. The smaller the set of roots the more information one gains about $priv$. Similarly, an inequality yields the more information about $priv$ the greater the number of roots is.

Condition 1 ensures that all polynomial-time computations are leak-resistant. Essentially, this is done by requiring that there is only a negligible chance to randomly pick a root of any non-zero difference polynomial. So we only have a negligible chance of obtaining any significant information about a random $priv$ due to equalities between group elements. Inequalities virtually yield no information (since we have a negligible number of roots). Since the set of all polynomials computable by polynomial-time SLP-generators is closed under subtraction, we do not need to explicitly consider the difference of any two polynomials computed by a certain $\mathcal{S} \in SLPGen^{(m)}$ in Condition 1. Note that this condition (and also Condition 3 later on) demands a form of uniform negligibility, i.e., there should be a negligible function that serves as upper bound for all SLP-generators with the same run-time bound $m(.)$.

Condition 1 (Leak-resistance). *Let the ideal consisting of all Laurent polynomials which are zero for all possible choices of priv be denoted by*

$$\mathcal{I}_n = \{P \in \mathcal{L}_n^{(l,c)} \mid \forall (x_1, \dots, x_l) \in \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{l-c} : \; P(x_1, \dots, x_l) \equiv 0 \bmod n\}.$$

Then for each $m \in \mathsf{poly}(x)$ there exists a negligible function f such that for all sec, $n \in \mathcal{N}^{(sec)}$, $\mathcal{S} \in SLPGen^{(m)}$ and $P \in \bigcup_i L_i \setminus \mathcal{I}_n$ where $(L_1, \dots, L_k) \leftarrow \mathcal{S}(sec, n)$ hold

$$\Pr[P(priv) \equiv 0 \bmod n :: priv \in_U \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{l-c}] \leq f(sec).$$

Now that we have ensured that virtually no information about $priv$ can be gained (for sufficiently large sec) we can restrict to consider algorithms applying trivial solution strategies to find $Q(priv)$.

For DL-type problems we stipulate Condition 2 which prevents an algorithm that for fixed n always outputs a constant value (independent of the particular problem instance) from being successful. Such an algorithm can be seen as an extended form of an SLP-generator.

Condition 2 (SLP-intractability of DL-Type Problems). *There exists a negligible function f such that for all sec, $n \in \mathcal{N}^{(sec)}$ and $\alpha \in \mathbb{Z}_n$ it holds that*

$$\Pr[Q(priv) \equiv \alpha \bmod n \ :: \ priv \in_U \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{l-c}] \leq f(sec)\,,$$

where $Q \leftarrow \mathtt{PGen}(sec, n)$ is the challenge polynomial.

For DH-type problems we demand that the SLPs of any efficient SLP-generator solve problem instances with negligible probability, i.e., the probability to obtain $Q(priv)$ when evaluating such an SLP with $priv$ must be negligible.

Condition 3 (SLP-intractability of DH-type Problems). *For each $m \in \mathsf{poly}(x)$ there exists a negligible function f such that for all sec, $n \in \mathcal{N}^{(sec)}$, $\mathcal{S} \in SLPGen^{(m)}$ and $P \in L_1$ holds that*

$$\Pr[(Q-P)(priv) \equiv 0 \bmod n :: priv \in_U \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{l-c}] \leq f(sec)\,,$$

where $(L_1, \dots, L_k) \leftarrow \mathcal{S}(sec, n)$ and $Q \leftarrow \mathtt{PGen}(sec, n)$ is the challenge polynomial.

If the above conditions are satisfied then the success probability of any efficient generic algorithm in solving a DL-type resp. DH-type problem by only using the allowed set of generic operations is negligible. For the proof of Theorem 1 please refer to Appendix C.

Theorem 1 (Intractability of DL/DH-type Problems). *Let $\mathcal{P}$ be DL-type (resp. DH-type problem) and OPSet be an operation set for $\mathcal{P}$ such that Condition 1 and Condition 2 (resp. 3) are satisfied. Then for each generic algorithm $\mathcal{A}$ whose number of oracle queries to $\mathcal{O}_{\mathcal{P},OPSet}$ is polynomial bounded, there exists a negligible function f such that for all sec and $n \in \mathcal{N}^{(sec)}$ the probability (taken over the coin tosses of $\mathcal{A}$ and the secret choices) that $\mathcal{A}$ outputs a solution to a problem instance of $\mathcal{P}$ is upper bounded by $f(sec)$.*

5.3 Practical Conditions

Essentially, the presented abstract conditions demand that for certain multivariate Laurent polynomials there is a negligible probability to find a root by randomly picking elements. This property is satisfied for non-zero polynomials in $\mathcal{L}_n^{(l,c)}$ that have total absolute degrees which are polynomial bounded. Showing this claim is based on Lemma 1 in [Sho97] but requires several extensions. The above observation is the main line along which we developed another set of sufficient hardness conditions that can be be verified more easily. Following the ideas set up for the abstract conditions, we again ensure leak-resistance on the one hand and non-triviality on the other hand. Finally, Theorem 2 shows precisely how the practical conditions are related to the abstract conditions.

Leak-resistance. We specify two conditions ensuring that polynomials computable by a polynomial-time SLP-generator (using the given *OPSet*) are of low degree: first we simply demand low degrees for the input polynomials (see Condition 4) and then we restrict to operation sets that do not allow to increase the degree too much (see Condition 5).

Condition 4. *There exists $r \in \mathsf{poly}(x)$ such that for all sec, $n \in \mathcal{N}^{(sec)}$: $\max_{i,j}(\deg(I_{ij})) \leq r(sec)$ where $(I_{11}, \dots, I_{kz}) \leftarrow \mathtt{PGen}(sec, n)$.*

Example 4 (Cond. 4 satisfied for q-BDHIP). For q-BDHIP we have $\max_{i,j}(\deg(I_{ij})) \leq q$ where $q = q(sec)$ is polynomial bounded by definition.

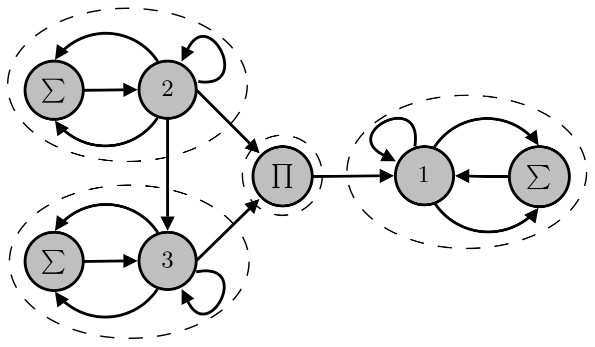

Fig. 1. $\mathcal{G}_{OPSet}$ for $OPSet = \{(\circ_1, 1, 1, 1, Y_1 + Y_2), (\circ_2, 2, 2, 2, Y_1 + Y_2), (\circ_3, 3, 3, 3, Y_1 + Y_2), (inv_1, 1, 1, -Y_1), (inv_2, 2, 2, -Y_1), (inv_3, 3, 3, -Y_1), (\psi, 2, 3, Y_1), (e, 2, 3, 1, Y_1 \cdot Y_2)\}$ of q-BDHIP. Strongly connected components are marked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

Now, we are formalizing operation sets that only allow for a small raise of degrees. It turned out that this can be done easily if operation sets are modeled as a graph according to the Definitions 4 and 5. In regard to the groups $G_1, \ldots, G_k$ (which are represented by group vertices) such a graph reflects how these are related by the given operations. That means, it shows if and how the given group elements can be mapped from one group to another either directly or by means of a composition of operations. Concerning SLP-generators, the graph shows if and how input polynomials from certain lists can be combined to form a new polynomial for a list L_i. Furthermore, it reflects whether the degrees can be increased in this way.

Definition 4 (Operation Set Graph). *Let OPSet be an operation set for a DL-/DH-type problem $\mathcal{P}$. The operation set graph $\mathcal{G}_{OPSet} = (V, E)$ is a directed multi-edge multi-vertex graph. There are four types of vertices, namely group, sum, product and power vertices. Initially, V contains k group vertices labeled with the numbers 1 to k representing the k different groups underlying $\mathcal{P}$. For each operation $(f, s_1, \ldots, s_u, d, F) \in OPSet$, where $F = \sum_i S_i$ is represented as the sum of non-zero monomials, the following additional vertices and edges are added to the graph:*

1. *A sum vertex with label $\sum$ and an edge from this sum vertex to the group vertex with label d.*
2. *For each monomial S_i:*
   (a) *A product vertex with label $\prod$ and an edge from this vertex to the sum vertex (added before).*
   (b) *For each variable Y_j $(1 \leq j \leq u)$ occurring with non-zero exponent ℓ in S_i: A power vertex with label $(.)^\ell$, an edge from this power vertex to the product vertex (added before) and an edge from the group vertex with label $s_j \in \{1, \ldots, k\}$ to this power vertex.*

Definition 5 (Reduced Operation Set Graph). *A reduced operation set graph is an operation set graph where each power vertex with label $(.)^1$ and each product or sum vertex with indegree 1 has been removed and the two edges entering and leaving this vertex have been replaced by one edge directly connecting the predecessor vertex to the successor vertex.*

As an illustrating example, consider the reduced operation set graph depicted in Figure 1, which belongs to the operation set for the q-BDHI problem (cf. Example 3).

Considering the graph belonging to an operation set, it becomes clear how operations can be used by an SLP-generator to increase the maximal absolute total degree of the polynomials computed so far: a product vertex corresponds to an operation that might lead to a polynomial whose degree is the sum of the degrees of (a constant number of) polynomials already computed. A power vertex respectively corresponds to an operation that might lead to a polynomial whose degree is a (constant) multiple of the degree of polynomials already computed. However, we do not need to completely disallow the existence of such operations in an operation set, at least as long as they cannot be used to repeatedly double the maximal degree. Thus, the following condition only rules out operation sets where any kind of repeated squaring - and therefore a way to exponentially increase the degree of polynomials - is possible.

Condition 5. *Let $\mathcal{G}_{OPSet}$ be a reduced operation set graph. Then for every strongly connected component*[2] *$\mathcal{C}$ of $\mathcal{G}_{OPSet}$ it holds that:*

1. *if $\mathcal{C}$ contains a power vertex then it contains no other vertices.*
2. *every product vertex contained in $\mathcal{C}$ has at most one incoming edge from a group vertex that is also contained in $\mathcal{C}$.*

Example 5 (Cond. 5 satisfied for OPSet of q-BDHIP). Condition 5 is satisfied since $\mathcal{G}_{OPSet}$ (cf. Figure 1) contains no power vertex and the strongly connected component containing the product vertex contains no other vertices.

Non-Triviality. To ensure SLP-intractability (cf. Conditions 2 and 3), we first restrict the challenge polynomial to have low degrees, similarly to the input polynomials.

Condition 6. *There exists $r \in \mathsf{poly}(x)$ such that for all sec, $n \in \mathcal{N}^{(sec)}$:* $\deg(Q) \leq r(sec)$ *where $Q \leftarrow \mathtt{PGen}(sec, n)$.*

Example 6 (Cond. 6 satisfied for q-BDHIP). Here we always have $\deg(Q) = 1$.

Now assuming that the practical conditions stated so far are satisfied, we just have to demand that the considered problem is not totally flawed (w.r.t. the given operations), i.e., solvable with probability 1 in a way not depending on the secret choices. The non-triviality condition for DL-type problems is simply the very natural condition that the solutions depend on the secret choices.

Condition 7 (Non-Triviality of DL-type Problems). *There exists sec_0 s.t. for all $sec \geq sec_0$ and $n \in \mathcal{N}^{(sec)}$ the challenge polynomial $Q \leftarrow \mathtt{PGen}(sec, n)$ is not a constant in $\mathcal{L}_n^{(l,c)}$.*

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

In the case of DH-type problems the non-triviality condition states that an efficient SLP-generator can hardly ever compute a polynomial $P \in L_1$ that, when evaluated with the secret choices is *always* a solution of the problem.

Condition 8 (Non-Triviality of DH-type Problems). *For each $m \in \mathsf{poly}(x)$ there exists sec_0 s.t. for all $sec \geq sec_0$, $n \in \mathcal{N}^{(sec)}$, $\mathcal{S} \in SLPGen^{(m)}$ and $P \in L_1$ holds $P \neq Q$ in $\mathcal{L}_n^{(l,c)}$ where $(L_1, \ldots, L_k) \leftarrow \mathcal{S}(sec, n)$ and $Q \leftarrow \mathtt{PGen}(sec, n)$.*

We note that Condition 8 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 generality. However, it is possible to derive easier versions of this condition for several more restricted classes of problems and operation sets. Anyway, above condition is easy to check for every particular problem of practical relevance we are aware of. As an example consider the case of the q-BDHI problem:

[2] A strongly connected component of a directed graph $\mathcal{G}_{OPSet} = (V, E)$ 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]).

Example 7 (Non-Triviality of q-BDHIP). Here the challenge polynomial is fixed to $Q = X_1^{-1}$ for all n. Moreover, observe that all variables occurring in the input polynomials only have positive exponents and thus no operation set can ever lead to polynomials P with negative exponents in any variable. Hence, Condition 8 is trivially satisfied (independently of the given *OPSet*).[3]

Theorem 2 connects the abstract and practical conditions. A proof is sketched in Appendix D.

Theorem 2 (Relation between Abstract and Practical Conditions).
(a) If Cond. 4 and 5 hold for a given DL-/DH-type problem and operation set then also Cond. 1.
(b) If Cond. 6 and 7 hold for a given DL-type problem then also Cond. 2.
(c) If Cond. 4, 5, 6, and 8 hold for a given DH-type problem and operation set then also Cond. 3.

Corollary 1. *q-BDHIP is intractable for generic algorithms allowed to apply the group law and inversion operations, the isomorphism $\psi : G_2 \to G_3$ and the bilinear map $e : G_2 \times G_3 \to G_1$.*

6 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 reduction between certain problems. Note that a problem $\mathcal{P}$ is not generically reducible to a problem $\mathcal{P}'$ if $\mathcal{P}$ is intractable for generic algorithms even if these algorithms can query an oracle that solves $\mathcal{P}'$.

Besides distinguishing between computational and decisional oracles we also distinguish between oracles that solve a problem only with respect to fixed generators (fixed-base oracle) and regular oracles that receive the generators for which the problem should be solved as an extra input. More precisely, let us assume that a problem 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 respective 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 $\mathcal{P}'$.[4]

We can realize most fixed-base computational oracles as an operation of the form given in Section 4.1, i.e., using a fixed polynomial F over $\mathbb{Z}$. However, we cannot represent regular oracles solving computational problems. For instance, consider the algebraic setting of the q-BDHI problem. Let $g_2, g_3, g_1 = e(g_2, g_3)$ be fixed generators of G_2, G_3, G_1 given as part of a problem instance of q-BDHIP. It is important to observe that all group elements are represented by the generic oracle with respect to these generators. A fixed-base oracle for the bilinear Diffie-Hellman problem over the three groups is given three elements $g_2^{y_1}, g_2^{y_2}, g_2^{y_3} \in G_2$ and returns $e(g_2, g_3)^{y_1 y_2 y_3} \in G_1$. The operation $(\text{FB-BDH}, 2, 2, 2, 1, Y_1 Y_2 Y_3)$ realizes such an oracle. It is easy to see that q-BDHIP remains hard if we add this operation to the operation set considered in Example 3. A regular oracle for the BDH problem is given generators $g_2^{y_4}, g_3^{y_5}$ of G_2, G_3 as an additional input and returns

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

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 it would require a Laurent

[3] Note that $X_1^{-1} \neq X_1^{\phi(n)-1}$ in $\mathcal{L}_n^{(1,0)}$ but these polynomials evaluate to the same value for all $x_1 \in \mathbb{Z}_n^*$. However, Cond. 4 and 5 ensure that for any efficient SLP-generator there exists sec_0 s.t. for all $sec \geq sec_0$ and $n \in \mathcal{N}^{(sec)}$ the polynomial $X_1^{\phi(n)-1}$ cannot be computed.

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

polynomial $F = Y_1 Y_2 Y_3 Y_4^{-2} Y_5$. However, only regular polynomials are permitted for representing operations. This restriction was 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 operations of the form $(f, s_1, \ldots, s_u, \{0,1\}, F)$ where $u \geq 1$, $s_1, \ldots, s_u \in \{1, \ldots, k\}$ are some fixed constants and $F \in \mathbb{Z}[Y_1, \ldots, Y_u]$ is some fixed polynomial. Such a tuple is interpreted by the generic oracle as a mapping

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

where $g_1, \ldots, g_u$ are generators given as part of a problem instance of the respective DL-/DH-type problem. Using this new type of operation, most fixed-base *and* regular decisional oracles can be implemented in our framework. As an example, consider the operation $(\mathrm{DBDH}, 2, 3, 2, 2, 2, 1, \{0,1\}, Y_1 Y_2 Y_3 Y_5 - Y_4^2 Y_6)$. It realizes a regular decisional BDH oracle. This oracle takes an additional input $e(g_2, g_3)^{y_6} \in G_1$ compared to the regular computational oracle described above. It outputs 1 iff $e(g_2, g_3)^{y_1 y_2 y_3 y_4^{-2} y_5} = e(g_2, g_3)^{y_6}$ which is equivalent to

$$y_1 y_2 y_3 y_5 - y_4^2 y_6 \equiv 0 \bmod n.$$

Clearly, a decisional oracle is another source of information for a generic algorithm. However, it is not hard to see that no extra condition (in addition to Conditions 4 and 5) is needed to ensure that computations remain leak-resistant. Just to clarify, the new operation type does not need to be represented in the operation set graph. Thus, any DL-/DH-type problem that is intractable (by satisfying our practical conditions) remains intractable even if we additionally admit any decisional oracle representable by an operation of above type.[5] This is captured by Theorem 3. Proving this theorem only requires simple extensions to the proof in Appendix C and D and is therefore omitted. As a simple consequence, the theorem implies that q-BDHIP is intractable for generic algorithms even if these can query a regular DBDH oracle.

Theorem 3. *Let a DL-type (resp. DH-type) problem $\mathcal{P}$ and operation set OPSet (compliant to Definition 2) be given satisfying Conditions 4, 5, 6 and Condition 7 (resp. 8). Then $\mathcal{P}$ remains intractable for generic algorithms even if OPSet is extended by one or more arbitrary decisional operations of the form described above.*

References

[AdM04] G. Ateniese and B. de Medeiros. A provably secure nyberg-rueppel signature variant with applications. Cryptology ePrint Archive, Report 2004/93, 2004. `http://eprint.iacr.org/`.

[BB04] 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.

[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/`.

[5] This proposition can be extended to decisional oracles representable by an operation where the number of inputs u and the polynomial F depends on the security parameter as long as $\deg(F)$ is polynomial bounded in *sec*.

[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.

[BR04] M. Bellare and P. Rogaway. The game-playing technique, 2004. `citeseer.ist.psu.edu/bellare04gameplaying.html`.

[Bro05] D. L. Brown. Generic groups, collision resistance, and ecdsa. *Des. Codes Cryptography*, 35(1):119–152, 2005.

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

[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.

[Fis00] M. Fischlin. A note on security proofs in the generic model. In *Advances in Cryptology: Proceedings of ASIACRYPT 2002*, volume 1976 of *Lecture Notes in Computer Science*, pages 458–469. Springer-Verlag, 2000.

[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.

[LR06] G. Leander and A. Rupp. On the equivalence of RSA and factoring regarding generic ring algorithms. In *Advances in Cryptology: Proceedings of ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 241–251. Springer-Verlag, 2006.

[Mau00] U. Maurer. Index search, discrete logarithms, and Diffie-Hellman. Talk at MRSI Number-theoretic cryptography workshop, Mathematical Sciences Research Institut (MRSI), Berkeley, October 2000.

[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.

[MW98] U. Maurer and S. Wolf. Lower bounds on generic algorithms in groups. In *Advances in Cryptology: Proceedings of EUROCRYPT 1998*, volume 1403 of *Lecture Notes in Computer Science*, pages 72–84. Springer-Verlag, 1998.

[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.

[Sho97] 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.

[SJ00] C. Schnorr and M. Jakobsson. Security of signed elgamal encryption. In *Advances in Cryptology: Proceedings of ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 73–89. Springer-Verlag, 2000.

[SPMLS02] J. Stern, D. Pointcheval, J. Malone-Lee, and N. P. Smart. Flaws in applying proof methodologies to signature schemes. In *Advances in Cryptology: Proceedings of CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 93–110. Springer-Verlag, August 2002.

[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/`.

A Two Variants of the Generic Model - Maurer vs. Shoup

Let us consider Maurer's and Shoup's generic group model in the case of a single cyclic group G of order n. In both models group elements that are input to the generic algorithm or result from applying an operation on the group are stored in a list L which is part of the secret state of the generic oracle $\mathcal{O}$. Algorithms can perform operations on desired group elements of G by issuing queries to the oracle containing the respective indices into L.

The models differ regarding the output of the generic oracle: every time an element $a \in G$ is appended to L, Maurer's oracle returns an equality set

$$EQS(a) = \{j \in \{1, \dots, |L|\} \mid a = L_j\},$$

i.e., the set of indices to all elements in L that are equal to a. In Shoup's model group elements are represented by unique random bit strings (called encodings) from $S_n \subset \{0,1\}^{\lceil \log_2(n) \rceil}$, where $|S_n| = n$. Shoup's oracle maintains an encoding list E, accessible for generic algorithms, containing the encodings of the group elements in L, i.e., for all indices j, E_j is the encoding of the element L_j. Thus, if an element a should appended to L, Shoup's oracle determines whether there exists j such that $L_j = a$. If this is not the case, it appends a random bit string from $S_n \setminus E$ to E which is different from all encodings so far contained in E. Otherwise, the bit string E_j is again appended to E.

The second and last difference between the models concerns the final output of a generic algorithm in the case of DH-type problems: In Maurer's model a generic algorithm can only output an element that has already occurred in the course of the computation, i.e., it outputs an index j into the list L. In Shoup's model a generic algorithm is also allowed to output a new group element that has not occurred before, i.e., it can output an arbitrary encoding from S_n.

For presenting our results, we decided to use Maurer's variant of the generic model because it induces less technical overhead especially regarding the proof of Theorem C. In particular, no random encodings need to be introduced and handled (for every group) which artificially enlarge the probability space compared to the plain model. However, we note that our conditions and theorems hold unchanged in Shoup's model. The proof of Theorem D does not need to be adapted at all. The following arguments show how the proof of Theorem C carries over to Shoup's model.

First, we can assume that also in Shoup's model generic algorithms only output already computed group elements: if an algorithm outputs a new encoding, this encoding is associated with a *random element* from $G \setminus L$. So even assuming that the solution to a problem instance is an element from this set, the algorithm's success probability is $\frac{1}{|G \setminus L|}$ which is negligible if the number of oracle queries is polynomial bounded. Thus, we can restrict to consider the case where an algorithm outputs an encoding from E corresponding to an already computed element L_j. So alternatively it could also have output j. This case is covered by the proof.

Second, observe that determining an encoding for an element a actually means to compute the equality set for this element. More precisely, let us assume that the element a is appended to L at index i. Then alternatively, Shoup's oracle could compute $EQS(a)$ and return a new random encoding from $S_n \setminus E$ if $EQS(a)$ only contains i and otherwise it could return an old encoding E_j where $i \neq j \in EQS(a)$. So encodings can simply be derived from equality sets (and exhibit no extra information). In the course of the proof, we only consider and modify the computation of equality sets. Hence, for the proof we can actually treat a Shoup oracle like a Maurer oracle where the determination of encodings from equality sets is an add-on that has no importance for the arguments in the proof.

B Reduction to Prime Power Groups

Let a DL-/DH-type problem over cyclic groups $G_1, \dots, G_k$ 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 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 clear that the considered DL/DH-type problem is also defined over these product groups and any generic algorithm using only operations of the form as defined in Section 4.1 and Section 6 works as good in this case as for any other groups. Considering these product groups, it is easy to see that solving a DL-/DH-type over groups G_i 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 operations, and a solution over $G_i^{(s)}$ are of no help to find a solution over $G_i^{(p^e)}$. More precisely, we have the following lemma:

Lemma 1. *Let a DL-/DH-type problem $\mathcal{P}$ over groups of order $n = p^e \cdot s$ ($\gcd(p, s) = 1$, $s > 1$) and an operation set OPSet be given. Then for every efficient generic algorithm $\mathcal{A}$ only applying operations from OPSet that solves $\mathcal{P}$ with probability α there is an efficient generic algorithm $\mathcal{A}'$ that solves this problem over groups of order p^e with probability at least α.*

Proof. Given a problem instance over k groups $G_i^{(p^e)}$ of order p^e, the generic algorithm $\mathcal{A}'$ generates an arbitrary instance of the same problem over the k groups $G_i^{(s)} := \mathbb{Z}_s$ of order s. Then it uses the algorithm $\mathcal{A}$ to solve the respective problem instance over the groups $G_i := G_i^{(p^e)} \times \mathbb{Z}_s$ which are of order n.

Thus, computing a solution to a DL-type or DH-type problem over G_i is a least as hard for generic algorithms as computing a solution over $G_i^{(p^e)}$ 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.

C Proof of Theorem 1

In this section we give a proof of Theorem 1. We restrict to the case of DH-type problems. The respective proof for DL-type problems works analogously and only differs in small details.

In the following we assume a DH-type problem $\mathcal{P}$ according to Definition 1 and an operation set *OPSet* according to Definition 2 satisfying Conditions 1 and 3. Unless stated otherwise we consider arbitrary but fixed values sec and $n \in \mathcal{N}^{(sec)}$. Let $(I_{11}, \ldots, I_{kz}, Q) \leftarrow \texttt{PGen}(sec, n)$ be the input polynomials and the challenge polynomial of $\mathcal{P}$ for the fixed values sec and n. Finally, let $\mathcal{A}$ be a generic algorithm whose number of oracle queries is upper bounded by $m(sec)$ for all sec, where $m \in \mathsf{poly}(x)$.

C.1 Outline

We are going to replace the original generic group oracle, in the following denote by $\mathcal{O}_1$, with an oracle $\mathcal{O}_3$ that simulates $\mathcal{O}_1$ without using the knowledge of the secret choices *priv*. Then we show that the behavior of $\mathcal{O}_3$ is perfectly indistinguishable from $\mathcal{O}_1$ unless a certain simulation failure $\mathcal{F}$ occurs. From this, it immediately follows that the success probability of $\mathcal{A}$ when interacting with $\mathcal{O}_1$ is upper bounded by the sum of failure probability and the success probability of $\mathcal{A}$ when interacting with $\mathcal{O}_3$. This part of the proof given in Section C.3 is organized as a short sequence of attack games. Finally, we upper bound these probabilities in Section C.4 using the Conditions 1 and 3.

C.2 Attack Games

We define an *attack game* in the generic group model similar to [BR04] as a tuple $D = (\mathcal{O}, \mathcal{E})$ consisting of an oracle $\mathcal{O}$ and a machine $\mathcal{E}$ we call the *success evaluator*. An algorithm $\mathcal{A}$ playing the game D, which is denoted by $(D, \mathcal{A})$, queries the oracle $\mathcal{O}$ at most $m(sec)$ times before it gives its final output.[6] The success evaluator $\mathcal{E}$ gets the final output *out* of an algorithm and determines whether the algorithm wins the game. To do this we assume that $\mathcal{E}$ has read access to the internal state and input of $\mathcal{O}$. $\mathcal{E}$ outputs 1 if $\mathcal{A}$ wins the game D and 0 if not. Consequently, we call $\Pr[1 \leftarrow (D, \mathcal{A})]$ the success probability of $\mathcal{A}$ playing the game D where this probability is taken over the random choices *priv* and the coin tosses of $\mathcal{A}$.

C.3 Introducing a Simulation Game

Starting from the real game D_1 we first introduce an intermediate game D_2 which is equivalent to D_1 before we specify the simulation game D_3.

Game D_1 (Real Game). The real attack game denote by $D_1 = (\mathcal{O}_1, \mathcal{E}_1)$ consists of the generic group oracle $\mathcal{O}_1 = \mathcal{O}_{\mathcal{P},OPSet}$ specified in Definition 3 and the success evaluator $\mathcal{E}_1$ that simply outputs 1 if $\mathcal{A}$'s final output *out* is a solution to the instance of the DH-type problem, i.e., $Q(priv) - L_{1out} \equiv 0 \bmod n$.

Game D_2. Let us define a new game $D_2 = (\mathcal{O}_2, \mathcal{E}_2)$ by doing a slight change to the original oracle: instead of using the isomorphic groups $\mathbb{Z}_n \cong G_1 \cong \ldots \cong G_k$ for the internal representation of group elements, the new oracle $\mathcal{O}_2$ represents these elements by Laurent polynomials from $\mathcal{L}_n^{(l,c)}$ which are evaluated with *priv* each time an equality set is computed. Definition 6 precisely describes the behavior of $\mathcal{O}_2$. It is easy to see that $\mathcal{O}_1$ and $\mathcal{O}_2$ are equivalent, i.e., given the same inputs $\mathcal{O}_2$ behaves exactly like $\mathcal{O}_1$.

Definition 6 (An Equivalent Generic Group Oracle). *The oracle $\mathcal{O}_2$ has an input and output port and performs computations as follows.*
Internal state. *As internal state $\mathcal{O}_2$ maintains lists $L_1, \ldots, L_k$, where a list L_i is used for storing computed polynomials from $\mathcal{L}_n^{(l,c)}$.*
Input. *As input $\mathcal{O}_2$ is given sec, n and secret choices priv belonging to an instance of $\mathcal{P}$.*

[6] We assume that an algorithm always terminates in any game.

Computation of equality sets. *Each time a polynomial P is appended to a list L_i the following computation is triggered: $\mathcal{O}_2$ computes the equality set*

$$EQS(i,P) = \{j \in \{1,\dots,|L_i|\} \mid (P - L_{ij})(priv) \equiv 0 \bmod n\}\,,$$

and writes it to its output port.

The computation of $\mathcal{O}_2$ consists of an initialization phase, which is run once, followed by the execution of the query-handling phase:

Initialization. *Each list L_i is initialized with the polynomials $1, I_{i1}, \dots, I_{iz}$ where $(I_{11},\dots,I_{kz}) \leftarrow$ $\mathtt{PGen}(sec, n)$.*

Query-handling. *Upon receiving a query $(f, j_1, \dots, j_u)$ on its input port, where $(f, s_1, \dots, s_u, d, F)$ $\in OPSet$, $\mathcal{O}_2$ computes the polynomial $F(L_{s_1 j_1}, \dots, L_{s_u j_u})$ and appends it to the list L_d.*

The definition of success of an algorithm is adapted accordingly: $\mathcal{E}_2$ outputs 1 iff $(Q - L_{1out})(priv) \equiv 0 \bmod n$.

It is easy to see that the games D_1 and D_2 are equivalent and it holds

$$\Pr[1 \leftarrow (D_1, \mathcal{A})] = \Pr[1 \leftarrow (D_2, \mathcal{A})].$$

Game D_3 (Simulation Game). Now, we modify the game D_2 to obtain the final simulation game $D_3 = (\mathcal{O}_3, \mathcal{E}_3)$. Let the oracle $\mathcal{O}_3$ be defined exactly like $\mathcal{O}_2$ with the exception that equality sets are computed differently: instead of checking whether a new polynomial P that is appended to some list L_i becomes equal to a polynomial L_{ij} under evaluation with the *given* choices *priv*, $\mathcal{O}_3$ just verifies if the two polynomial are equal for *all* elements of $\mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{l-c}$. More precisely, $\mathcal{O}_2$ computes equality sets according to the following definition:

$$EQS(i,P) = \{j \in \{1,\dots,|L_i|\} \mid P - L_{ij} \equiv 0 \bmod \mathcal{I}_n\}\,.$$

Note that in this way, the returned equality sets and thus the behavior of $\mathcal{O}_3$ is independent of the particular secret choices *priv* given as input to it.

Now, consider an algorithm $\mathcal{A}$ with an arbitrary but fixed input on its random tape playing the games D_2 and D_3, where we assume that both oracles receive the same input. Then it is clear that if $\mathcal{A}$ receives the same sequence of equality sets from $\mathcal{O}_2$ and $\mathcal{O}_3$ it issues the same sequence of queries in both games and also outputs the same value *out*. Hence, the games D_2 and D_3 proceed identically and the states (especially the lists of Laurent polynomials) maintained by $\mathcal{O}_2$ and $\mathcal{O}_3$ are equal unless the difference between $\mathcal{O}_2$ and $\mathcal{O}_3$ yields to different equality sets when a newly computed polynomial P is appended to some list L_i. This is the case if there exists some polynomial L_{ij} such that $P - L_{ij} \not\equiv 0 \bmod \mathcal{I}_n$ but $P - L_{ij}(priv) \equiv 0 \bmod n$ for the particular choices *priv* given as part of the input.

So we define the following events that can occur in the game D_3 as well as D_2:

- Failure event over L_i, denoted by $\mathcal{F}_i$: There exists $j > j' \in \{1,\dots,|L_i|\}$ such that
$$L_{ij} - L_{ij'} \not\equiv 0 \bmod \mathcal{I}_n \wedge (L_{ij} - L_{ij'})(priv) \equiv 0 \bmod n.$$
- Failure event, denoted by $\mathcal{F}$: $\mathcal{F} := \mathcal{F}_1 \vee \dots \vee \mathcal{F}_k$.

Since the games D_2 and D_3 proceed identically unless the event $\mathcal{F}$ occurs, we have the following relation between the success events in these games and the failure event:

$$(1 \leftarrow (D_2, \mathcal{A})) \wedge \neg\mathcal{F} \Leftrightarrow (1 \leftarrow (D_3, \mathcal{A})) \wedge \neg\mathcal{F}.$$

Using this relation we immediately obtain the desired upper bound on the success probability $\Pr[1 \leftarrow (D_1, \mathcal{A})]$ in the original game in terms of the success probability $\Pr[1 \leftarrow (D_3, \mathcal{A})]$ and the failure probability $\Pr[\mathcal{F}]$ in the simulation game:

$$\begin{aligned}
\Pr[1 \leftarrow (D_1, \mathcal{A})] &= \Pr[1 \leftarrow (D_2, \mathcal{A})] \\
&= \Pr[1 \leftarrow (D_2, \mathcal{A}) \wedge \neg\mathcal{F}] + \Pr[1 \leftarrow (D_2, \mathcal{A}) \wedge \mathcal{F}] \\
&= \Pr[1 \leftarrow (D_3, \mathcal{A}) \wedge \neg\mathcal{F}] + \Pr[1 \leftarrow (D_2, \mathcal{A}) \wedge \mathcal{F}] \\
&\leq \Pr[1 \leftarrow (D_3, \mathcal{A})] + \Pr[\mathcal{F}]
\end{aligned}$$

C.4 Bounding Probabilities in the Simulation Game

In this section we show that $\Pr[1 \leftarrow (D_3, \mathcal{A})]$ and $\Pr[\mathcal{F}]$ are negligible functions in the security parameter.

We consider an adversary $\mathcal{A}$ that issues at most $m(sec)$ queries for any sec and $n \in \mathcal{N}^{(sec)}$ where $m \in \mathsf{poly}(x)$. Let sec and n be arbitrary but fixed values in the following when examining game D_3. Furthermore, let $\bar{\mathcal{A}}$ be the algorithm $\mathcal{A}$ on inputs sec and n with arbitrary but fixed content of the random tape.

Now, it is important to observe that for arbitrary choices $priv \in_U \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{l-c}$ the game D_3 played by $\bar{\mathcal{A}}$ proceeds identically. This is because the behavior of the simulation oracle $\mathcal{O}_3$ does not depend on $priv$. Thus, the lists of polynomials $L_1, \dots, L_k$ maintained by $\mathcal{O}_3$ stay the same for any $priv$. In particular, we have a fixed output polynomial L_{1out}.

Hence, we obtain

$$\Pr[1 \leftarrow (D_3, \bar{\mathcal{A}})] = \Pr[(Q - L_{1out})(priv) \equiv 0 \bmod n :: priv \in_U \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{l-c}],$$

where $Q \leftarrow \mathtt{PGen}(sec, n)$ is the challenge polynomial. To bound this probability, we just have to notice that there is some SLP-generator in $SLPGen^{(m)}$ that on inputs sec and n outputs exactly the lists $L_1, \dots, L_k$ (in particular containing L_{1out}) maintained by $\mathcal{O}_3$ after an interaction with $\bar{\mathcal{A}}$. Hence, Condition 3 yields that there exists some negligible function f such that above probability is upper bounded by $f(sec)$. Furthermore, the upper bound $f(sec)$ is independent of content of the random tape of $\mathcal{A}$ that we have fixed, i.e., we have the *same* upper bound for any $\bar{\mathcal{A}}$. Hence it holds

$$\Pr[1 \leftarrow (D_3, \mathcal{A})] \leq f(sec),$$

where this probability is taken over the choices of $priv$ and the coin tosses of $\mathcal{A}$. Finally, observe that also by Condition 3 we have the *same* negligible function f in above probability for all values of sec and n. Thus, the success probability of $\mathcal{A}$ in the simulation game is negligible.

Let us now consider the probability $\Pr[\mathcal{F}]$ of a failure in game D_3 played by $\bar{\mathcal{A}}$. Clearly, we have $\Pr[\mathcal{F}] \leq \sum_{i=1}^{k} \Pr[\mathcal{F}_i]$. So let us determine a bound on $\Pr[\mathcal{F}_i]$. We can write this probability as

$$\Pr[\mathcal{F}_i] \leq \sum_{1 \leq j' < j \leq |L_i| : L_{ij} - L_{ij'} \not\equiv 0 \bmod \mathcal{I}_n} \Pr[(L_{ij} - L_{ij'})(priv) \equiv 0 \bmod n :: priv \in_U \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{l-c}].$$

Now, observe all the difference polynomials $L_{ij} - L_{ij'}$ are also part of the output of some polynomial-time SLP-generator: we know that there exists an SLP-generator that computes exactly the lists $L_1, \dots, L_k$ of polynomials using at most $m(sec)$ steps. Thus, there also exists an SLP-generator that computes these polynomials and in addition all differences of these polynomials in not more than $m(sec) + (kz(sec) + m(sec)) + (kz(sec) + m(sec))^2$ steps (remember, to compute $P - P'$ we first need to compute $-P'$ and then add this to P), where k is a constant and $z(.)$ is polynomial bounded. Hence, from Condition 1 follows that there exists a negligible function f' such that

$$\Pr[\mathcal{F}_i] \leq |L_i|^2 f'(sec) \leq (z(sec) + m(sec))^2 f'(sec).$$

Moreover, since we have the same bound on $\Pr[\mathcal{F}_i]$ for all i, we obtain the following upper bound on the total failure probability for game D_3 played by $\bar{\mathcal{A}}$:

$$\Pr[\mathcal{F}] \leq k(z(sec) + m(sec))^2 f'(sec).$$

Note that above bound in turn holds for arbitrary coin tosses of $\mathcal{A}$ and we have the *same* negligible function f' for all values of sec and n. Thus, the failure probability of $\mathcal{A}$ (taken over the random choices of $priv$ and coin tosses of $\mathcal{A}$) playing the simulation game is negligible.

D Sketch of the Proof of Theorem 2

In this section we give a sketch of the proof of Theorem 2 connecting the practical conditions with the abstract conditions. Before we come to the proofs of the three different parts of this theorem, we first need to introduce several lemmas. These lemmas provide the desired link between the degrees of a multivariate polynomial Laurent polynomial and the probability of randomly picking one of its roots.

Lemma 2 is an extended version of Lemma 1 in [Sho97] that upper bounds the probability of randomly picking a root over $\mathbb{Z}_{p^e}^c \times (\mathbb{Z}_{p^e}^*)^{l-c}$ of a regular non-zero multivariate polynomial.

Lemma 2. *Let $p \in \mathbb{P}$ be a prime and $e \in \mathbb{N}$. Let $P \in \mathbb{Z}_{p^e}[X_1, \ldots, X_l]$ be a non-zero polynomial of total degree d. Then for random independent choices $x_1, \ldots, x_c \in_U \mathbb{Z}_{p^e}$ and $x_{c+1}, \ldots, x_l \in_U \mathbb{Z}_{p^e}^*$, where $0 \leq c \leq l$ the following holds*

$$\Pr[P(x_1, \ldots, x_l) \equiv 0 \bmod p^e] \leq \frac{d}{p-1}.$$

Proof. We can write the polynomial P as $P = p^s \cdot P'$ for some integer $s \geq 0$ and polynomial $P' \in \mathbb{Z}_{p^e}[X_1, \ldots, X_l]$ s.t. $P' \not\equiv 0 \pmod p$ and $\deg(P') = d$. Since we assume that $P \not\equiv 0 \pmod{p^e}$ it follows that $p^s < p^e$. Let $(x_1, \ldots, x_l) \in \mathbb{Z}_{p^e}^l$ be a root of the polynomial P. Then we have $p^e | P(x_1, \ldots, x_l) \Leftrightarrow p^e | p^s P'(x_1, \ldots, x_l)$. Since $p^s < p^e$ it follows that $p^{e-s} | P'(x_1, \ldots, x_l)$ and thus $P'(x_1, \ldots, x_l) \equiv 0 \pmod p$. That means every root $(x_1, \ldots, x_l) \in \mathbb{Z}_{p^e}^l$ of P over $\mathbb{Z}_{p^e}$ is also a root of P' over the prime field $\mathbb{Z}_p$. Thus, if $(x_1, \ldots, x_l) \in \mathbb{Z}_{p^e}^c \times (\mathbb{Z}_{p^e}^*)^{l-c}$ is a root of P modulo p^e then $(x_1', \ldots, x_l') = (x_1 \bmod p, \ldots, x_l \bmod p) \in \mathbb{Z}_p^c \times (\mathbb{Z}_p^*)^{l-c}$ 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

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

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

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

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

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

□

Lemma 3 states that the probability is negligible to randomly pick a root of a non-zero Laurent polynomials with polynomial-bounded degrees. We make use of Lemma 2 to prove this result. Remember, for the remainder of this section, that we assume that $n \in \mathcal{N}^{(sec)}$ is always of the form $n = p^e$ for all sec, where p is a prime that is exponential in sec and $e \geq 1$ is constant over all sec.

Lemma 3. *Let $(\mathcal{M}_n)_{sec \in \mathbb{N}, n \in \mathcal{N}^{(sec)}}$ be a sequence of sets of Laurent polynomials, where $\mathcal{M}_n \subset \mathcal{L}_n^{(l,c)}$. Let $r \in \mathsf{poly}(x)$ a polynomial such that for all sec and $n = p^e \in \mathcal{N}^{(sec)}$:*

$$\deg(P) \leq r(sec) \text{ for all } P \in \mathcal{M}_n.$$

Then for each non-zero polynomial $P \in \mathcal{M}_n$ holds

$$\Pr[P(priv) \equiv 0 \bmod n \ :: \ priv \in_U \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{l-c}] \leq \frac{(l-c+1)r(sec)}{p-1},$$

where $f(sec) := \frac{(l-c+1)r(sec)}{p-1}$ is a negligible function.

Proof. Let a polynomial non-zero polynomial $P \in \mathcal{M}_n$ be given. We like to upper bound the probability of randomly picking a root of this polynomial by applying Lemma 2. However, P is a Laurent polynomial and may have a negative exponents in some variables which prevents us from using this lemma directly. Therefore, let us consider the polynomial

$$R \cdot P, \text{ where } R = \prod_{c<j\leq l} X_j^{r(sec)}.$$

Since we have $R(priv) \in \mathbb{Z}_n^*$ for all $priv \in \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{l-c}$, 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 $R \cdot P$ is a regular polynomial: by multiplying with R, a term of the form $X_j^{\pm d'}$ occurring in P is reduced to the term $X_j^{r(sec)\pm d'}$, where $d' \leq r(sec)$. The total degree d of this polynomial is upper bounded by

$$r(sec) + (l-c)r(sec) \leq (l-c+1)r(sec).$$

Since $R \cdot P$ is not the zero polynomial we can apply Lemma 2 which yields

$$\begin{aligned}\Pr[P(priv) \equiv 0 \bmod n :: priv \in_U \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{l-c}] &= \Pr[(R\cdot P)(priv) \equiv 0 \bmod n :: priv \in_U \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{l-c}] \\ &\leq \tfrac{d}{p-1} \\ &\leq \tfrac{(l-c+1)r(sec)}{p-1}.\end{aligned}$$

This upper bound holds for all values of sec, $n \in \mathcal{N}^{(sec)}$ and non-zero polynomials $P \in \mathcal{M}_n$. Since l and c are constants and r is a polynomial the function $f(sec) := \frac{(l-c+1)r(sec)}{p-1}$ is negligible.

□

Corollary 2. *The claim of Lemma 3 especially holds for all $P \in \mathcal{M}_n$ that are not in $\mathcal{I}_n$.*

Proof. A Laurent polynomial $P \in \mathcal{L}_n^{(l,c)}$ that is not in $\mathcal{I}_n$, i.e., that does not represent the zero function, is certainly not the zero polynomial.

D.1 Theorem 2(a): Cond. 4, 5 imply Cond. 1

Let us consider the outputs $SLP_n = (L_1, \ldots, L_k)$ of an SLP-generator $\mathcal{S} \in SLPGen^{(m)}$ on inputs sec and $n \in \mathcal{N}^{(sec)}$, where $m \in \mathsf{poly}(x)$. We denote the sequence of sets of output polynomials by $(\mathcal{M}_n)_{sec,n\in\mathcal{N}^{(sec)}}$, i.e.,

$$\mathcal{M}_n = \bigcup_{1\leq i\leq k} L_i.$$

Observe that any polynomial $P \in \mathcal{M}_n$ is of the form

$$P = \sum_{d=(d_{11},\ldots,d_{kz})\in\mathbb{N}_0^{kz}} a_d I_{11}^{d_{11}} \cdot \ldots \cdot I_{kz}^{d_{kz}}$$

where $I_{11}, \ldots, I_{kz} \leftarrow \texttt{PGen}(sec, n)$ are the input polynomials. Then Condition 1 follows from the following two claims:

- **Claim 1:** If *OPSet* satisfies Condition 5 then there exists a polynomial $r' \in \mathsf{poly}(x)$ such that for all sec, $n \in \mathcal{N}^{(sec)}$ and $P \in \mathcal{M}_n$ holds

$$\max_{i,j}(\deg_{I_{ij}}(P)) \leq r'(sec). \tag{2}$$

- **Claim 2:** If Condition 4 is satisfied and Equation (2) holds then Condition 1 is satisfied.

Let us first show Claim 2. To this end, consider the degree $\deg(S)$ with respect to the variables $X_1, \ldots X_l$ of a monomial

$$S = a_d I_{11}^{d_{11}} \cdot \ldots \cdot I_{kz}^{d_{kz}}$$

of $P \in \mathcal{M}_n$. According to Condition 4 there exists a polynomial $r \in \mathsf{poly}(x)$ such that $r(sec)$ is an upper bound on $\deg(I_{ij})$. Assuming Equation (2) is satisfied it holds that $\deg(I_{ij}^{d_{ij}}) \leq r'(sec)r(sec)$ and thus

$$\deg(S) \leq kzr'(sec)r(sec)$$

for an arbitrary monomial of P. Hence, we have

$$\deg(P) \leq kzr'(sec)r(sec), \tag{3}$$

where k is a constant and $z = z(sec)$ is polynomial bounded per definition. So there exists a polynomial that bounds the degrees of all $P \in \mathcal{M}_n$. Condition 1 immediately follows from applying Corollary 2.

Now, we justify Claim 1. We aim to show that the maximum total degree of any polynomial P that can be constructed by an SLP in the variables I_{ij} is bounded by a polynomial $r'(sec)$. To do this we use the $\mathcal{G}_{OPSet}$ graph and the graph $\bar{\mathcal{G}}_{OPSet}$ of strongly connected components of $\mathcal{G}_{OPSet}$. Note that $\bar{\mathcal{G}}_{OPSet}$ is a finite cycle-free directed graph. We are going to label the nodes of $\bar{\mathcal{G}}_{OPSet}$ with a bound for the maximum possible total degree for a polynomial P that might be constructed by an attacker in that component.

Since $\bar{\mathcal{G}}_{OPSet}$ is finite and cycle-free, there exists a node with no directed edges into that component. This corresponds to a strongly connected component in $\mathcal{G}_{OPSet}$ which contains at least one group node and for which all operations that output elements in groups contained in this component, these operations only take inputs from groups within this component. Furthermore, by Condition 5, we can deduce that this component contains no power nodes or product nodes. Hence, if this component contains group G_i, the maximum total degree of any polynomial $P \in L_i$ is 1 (in the variables I_{ij}).

We label this component with the label 1 (and repeat this for all strongly connected components of $\mathcal{G}_{OPSet}$ that contain a group node and have all operations which output elements in that component taking inputs in that component).

Next, we label the components of $\mathcal{G}_{OPSet}$ that do not contain a group node and which only receive edges from labelled components. We label each of the nodes inside the components individually:

- If the node is a $(\cdot)^d$ power node, and the input is from a node/component with label r, then the power node is labelled rd.
- If the node is a product node, taking u inputs from nodes/components with labels $r_1, \ldots, r_u$, then the node is labelled $r_1 + r_2 + \ldots + r_u$.
- If the node if a sum node, taking inputs from nodes/components with labels $r_i, \ldots, r_u$, then the nodes is labelled $\max\{r_1, \ldots, r_u\}$.

The label of the component is maximum value of any node within the component. Note that since $OPSet$ is fixed and independent of both the group order n and the security parameter sec, the labels are polynomially-bounded.

We now consider the more complex case of components that receive inputs from other components and contain group nodes. We note that these components cannot contain power nodes (by Condition 5) but may contain product nodes and sum nodes, as well as at least one group node. If we remove the labelled nodes of $\bar{\mathcal{G}}_{OPSet}$, then the graph that remains is still a finite cycle-free directed graph; hence, it must contain a component which receives no edges. This corresponds in $\mathcal{G}_{OPSet}$ to a component that contains a group node and have all operations that output elements in the groups in that component taking inputs from that component or from already labelled components. We select one such component and attempt to bound the total degree of the polynomial that can be created in that component.

The situation is complicated because the sum and product nodes may take inputs from both inside the component and outside the component. We begin by considering product nodes:

- If the product node receives no edges from inside the component, then we may bound the degree of any polynomial produced by that node. It must receive u edges from components labelled with $r_1, \ldots, r_u$. The maximum degree of any polynomial that can be created using this node is $r_1 + r_2 + \ldots + r_u$.
- If the produce node receives some edges from inside the component, then by Condition 5, there exists a single incoming edge from a group node. An example of a $\mathcal{G}_{OPSet}$ for which this is true can be seen in Fig. 2. If a polynomial of degree d can be computed in this input group in the component, then after applying this operation, the output will be a polynomial of degree greater than d. In fact, if the product nodes receives u edges from other components with labels $r_1, \ldots, r_u$ then the total degree of the output polynomial will be bounded by $d+r_1+r_2+\ldots+r_u$.

  We note that, since the SLP generates at most $m(sec)$ equations, this degree increasing operation can be executed at most $m(sec)$ times. Naively, we can see that if we start with a polynomial I_{ij} in the input group, then the highest possible total degree that could be obtained by repeatedly applying this operation has total degree $1 + m(sec)(r_1 + r_2 + \ldots + r_u)$. Note that this value is polynomially bounded, as $m(sec)$ is polynomially bounded and each of the values $r_1, \ldots, r_u$ is polynomially bounded.

This current analysis of the product nodes is some sense incomplete, because it may be possible to use the product node to create polynomials of degree larger than $1+m(sec)(r_1+r_2+\ldots+r_u)$ if we start with a polynomial of total degree $d > 1$. We therefore have to consider the largest possible value d that can be computed without the use of product nodes. Since sum nodes cannot increase the total degree of a polynomial, the greatest possible total degree that can be obtained without using a product node bounded by is $\max\{1, r\}$ where r is the value of the largest labelled node with an edge that enters the component that we are currently labelling. Therefore, the total degree of any polynomial that can be constructed in this component is bounded by $\max\{1, r\} + m(sec)ur$, where u is the largest number of inputs that any product node in the component takes. This value is polynomially bounded as u is constant and r is polynomially bounded. We label this component with the label $\max\{1, r\} + m(sec)ur$.

We may now repeat the process of labelling components that do not contain a group node, and the labelling components that do contain groups nodes and only take inputs from other labelled components, until the entire graph $\bar{\mathcal{G}}_{OPSet}$ is labelled. In each case, we can show that the value of the label represents a bound on the total degree of the polynomial that can be constructed in a group in that component, and that the label is polynomially bounded. Since the number components is fixed and finite, this implies that the total degree of any polynomial that can be constructed is polynomially bounded, which proves the claim.

D.2 Theorem 2(b): Cond. 6, 7 imply Cond. 2

Let the negligible function demanded in Condition 2 be denoted by f. We consider the sequence of sets $\mathcal{M}_n = \{Q - \alpha \mid \alpha \in \mathbb{Z}_{p^e}\}$ of Laurent polynomials for all $sec, n \in \mathcal{N}^{(sec)}$, where $Q \leftarrow \texttt{PGen}(sec, n)$.

From Condition 7 follows that there exists sec_0 such that for all $sec \geq sec_0$, $n \in \mathcal{N}^{(sec)}$ and $P \in \mathcal{M}_n$ it holds that P is not zero in $\mathcal{L}_n^{(l,c)}$. For all $sec < sec_0$ we set $f(sec) = 1$.

From Condition 6 follows that there exists a polynomial r such that for all $sec, n \in \mathcal{N}^{(sec)}$ and $P \in \mathcal{M}_n$ holds $\deg(P) \leq r(sec)$. Thus, Lemma 3 can be applied. From this follows that for all $sec \geq sec_0$, $n \in \mathcal{N}^{(sec)}$ and $P \in \mathcal{M}_n$ holds that

$$\Pr[P(priv) \equiv 0 \bmod n \;::\; priv \in_U \mathbb{Z}_n^c \times (\mathbb{Z}_n^*)^{l-c}] \leq \frac{(l-c+1)r(sec)}{p-1}.$$

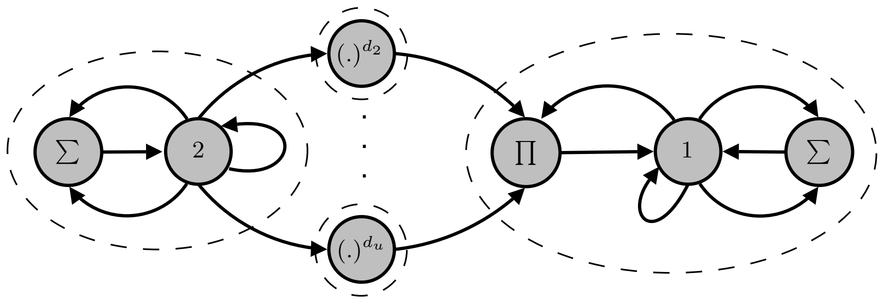

Fig. 2. Worst-case graph $\mathcal{G}_{OPSet}$ and its SCCs (marked by dashed borders)

Thus, for all $sec \geq sec_0$ we set $f(sec) = \frac{(l-c+1)r(sec)}{p-1}$.

D.3 Theorem 2(c): Cond. 4, 5, 6, 8 imply Cond. 3

Let the negligible function demanded in Condition 3 be denoted by f. Let $\mathcal{S} \in SLPGen^{(m)}$ be an arbitrary SLP-generator, where $m \in \mathsf{poly}(x)$. We consider the sequence of sets $\mathcal{M}_n = \{Q - P \mid P \in L_1\}$ of Laurent polynomials for all $sec, n \in \mathcal{N}^{(sec)}$, where $Q \leftarrow \mathtt{PGen}(sec, n)$ and $(L_1, \ldots, L_k) \leftarrow \mathcal{S}(sec, n)$.

From Condition 8 follows that there exists sec_0 such that for all $sec \geq sec_0$, $n \in \mathcal{N}^{(sec)}$ and $R \in \mathcal{M}_n$ it holds that R is not zero in $\mathcal{L}_n^{(l,c)}$. For all $sec < sec_0$ we set $f(sec) = 1$.

From Condition 6 follows that there exists a polynomial $r_1 \in \mathsf{poly}(x)$ such that for all sec and $n \in \mathcal{N}^{(sec)}$ holds $\deg(Q) \leq r_1(sec)$ where $Q \leftarrow \mathtt{PGen}(sec, n)$. Assuming Conditions 4 and 5 are satisfied, we know from Section D.1 (Eq. (3)) that there exists a polynomial $r_2 \in \mathsf{poly}(x)$ such that for all sec, $n \in \mathcal{N}^{(sec)}$ and $P \in L_1$ holds $\deg(P) \leq r_2(sec)$ where $(L_1, \ldots, L_k) \leftarrow \mathcal{S}(sec, n)$. Thus, for all sec, $n \in \mathcal{N}^{(sec)}$ and $P \in \mathcal{M}_n$ holds $\deg(P) \leq r_1(sec) + r_2(sec)$. Hence, similar to Section D.2 we can apply Lemma 3 and set $f(sec) = \frac{(l-c+1)(r_1(sec)+r_2(sec))}{p-1}$ for all $sec \geq sec_0$.