The Algebraic Group Model and its Applications

Georg Fuchsbauer[1] Eike Kiltz[2] Julian Loss[2]

May 16, 2018

[1] Inria, ENS, CNRS, PSL, France
georg.fuchsbauer@ens.fr

[2] Ruhr University Bochum, Germany
{eike.kiltz,julian.loss}@rub.de

Abstract

One of the most important and successful tools for assessing hardness assumptions in cryptography is the Generic Group Model (GGM). Over the past two decades, numerous assumptions and protocols have been analyzed within this model. While a proof in the GGM can certainly provide some measure of confidence in an assumption, its scope is rather limited since it does not capture group-specific algorithms that make use of the representation of the group.

To overcome this limitation, we propose the Algebraic Group Model (AGM), a model that lies in between the Standard Model and the GGM. It is the first restricted model of computation covering group-specific algorithms yet allowing to derive simple and meaningful security statements. To prove its usefulness, we show that several important assumptions, among them the Computational Diffie-Hellman, the Strong Diffie-Hellman, and the interactive LRSW assumptions, are equivalent to the Discrete Logarithm (DLog) assumption in the AGM. On the more practical side, we prove tight security reductions for two important schemes in the AGM to DLog or a variant thereof: the BLS signature scheme and Groth's zero-knowledge SNARK (EUROCRYPT 2016), which is the most efficient SNARK for which only a proof in the GGM was known. Our proofs are quite simple and therefore less prone to subtle errors than those in the GGM.

Moreover, in combination with known lower bounds on the Discrete Logarithm assumption in the GGM, our results can be used to derive lower bounds for all the above-mentioned results in the GGM.

Keywords: Algebraic algorithms, generic group model, security reductions, cryptographic assumptions.

1 Introduction

Starting with Nechaev [Nec94] and Shoup [Sho97], much work has been devoted to studying the computational complexity of problems with respect to generic group algorithms over cyclic groups [BL96, MW98, Mau05]. At the highest level, generic group algorithms are algorithms that do not exploit any special structure of the representation of the group elements and can thus be applied in any cyclic group. More concretely, a generic algorithm may use only the abstract group operation and test whether two group elements are equal. This property makes it possible to prove information-theoretic lower bounds on the running time for generic algorithms.

Such lower bounds are of great interest since for many important groups, in particular for elliptic curves, no helpful exploitation of the representation is currently known.

The class of generic algorithms encompasses many important algorithms such as the baby-step giant-step algorithm and its generalization for composite-order groups (also known as Pohlig-Hellman algorithm [HP78]) as well as Pollard's rho algorithm [Pol78]. However, part of the common criticism against the generic group model is that many algorithms of practical interest are in fact not generic. Perhaps most notably, index-calculus and some factoring attacks fall outside the family of generic algorithms, as they are applicable only over groups in which the elements are represented as *integers*. Another example is the "trivial" discrete logarithm algorithm over the additive group $\mathbb{Z}_p$, which is the identity function.

With this motivation in mind, a number of previous works considered extensions of the generic group model [Riv04, LR06, AM09, JR10]. Jager and Rupp [JR10] considered assumptions over groups equipped with a bilinear map $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_3$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are modeled as generic groups, and $\mathbb{G}_3$ is modeled in the Standard Model. (This is motivated by the fact that in all practical bilinear groups, $\mathbb{G}_1$ and $\mathbb{G}_2$ are elliptic curves whereas $\mathbb{G}_3$ is a sub-group of a finite field). However, none of these models so far capture algorithms that can freely exploit the representation of the group. In this work, we propose a restricted model of computation which does exactly this.

1.1 Algebraic Algorithms

Let $\mathbb{G}$ be a cyclic group of prime order p. Informally, we call an algorithm $\mathsf{A}_{\mathsf{alg}}$ *algebraic* if it fulfills the following requirement: whenever $\mathsf{A}_{\mathsf{alg}}$ outputs a group element $\mathbf{Z} \in \mathbb{G}$, it also outputs a "representation" $\vec{z} = (z_1, \ldots, z_t) \in \mathbb{Z}_p^t$ such that $\mathbf{Z} = \prod_i \mathbf{L}_i^{z_i}$, where $\vec{\mathbf{L}} = (\mathbf{L}_1, \ldots, \mathbf{L}_t)$ is the list of all group elements that were given to $\mathsf{A}_{\mathsf{alg}}$ during its execution so far.

Such algebraic algorithms were first considered by Boneh and Venkatesan [BV98] in the context of straight-line programs computing polynomials over the ring of integers $\mathbb{Z}_n$, where $n = pq$. Later, Paillier and Vergnaud [PV05] gave a more formal and general definition of algebraic algorithms using the notion of an *extractor algorithm* which efficiently computes the representation $\vec{z}$.

In our formalization of algebraic algorithms, we distinguish group elements from all other parameters at a *syntactical level*, that is, other parameters must not depend on any group elements. This is to rule out pathological exploits of the model, see below. While this class of algebraic algorithms certainly captures a much broader class of algorithms than the class of generic algorithms (e.g., index-calculus algorithms), it was first noted in [PV05] that the class of algebraic algorithms actually *includes* the class of generic algorithms.

Surprisingly, algebraic algorithms have only been studied so far in the context of proving *impossibility results* [BV98, Cor02, PV05, BMV08, GBL08, AGO11, KMP16], i.e., to disprove the existence of an algebraic security reduction between two cryptographic primitives (with certain good parameters).

1.2 Algebraic Group Model

We propose the *algebraic group model* (AGM) — a computational model in which all adversaries are modeled as algebraic. In contrast to the GGM, the AGM does not allow for proving information-theoretic lower bounds on the complexity of an algebraic adversary. Similar to the Standard Model, in the AGM one proves security implications via reductions. Specifically, $H \Rightarrow_{\mathsf{alg}} G$ for two primitives H and G means that every algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ against G can be transformed into an algebraic adversary $\mathsf{B}_{\mathsf{alg}}$ against H with (polynomially) related running times and success probabilities. It follows that if H is secure against algebraic adversaries, so is

G. To the best of our knowledge, our work is the first to consider algebraic algorithms in the role of an active adversary within a security game.

Concrete Security Implications in the AGM. Indeed, one can exploit the algebraic nature of an adversary in the AGM to obtain stronger security implications than in the Standard Model. The first trivial observation is that the classical *knowledge of exponent assumption*[1] [Dam92] holds by definition in the AGM.

We are able to show that several important computational assumptions are in fact equivalent to the Discrete Logarithm assumption over prime-order groups in the AGM, including the following:

- Diffie-Hellman assumption [DH76]
- (Interactive) strong Diffie-Hellman assumption [ABR01]
- (Interactive) LRSW assumption [LRSW99, CL04].

The significance of the Strong Diffie-Hellman Assumption comes from its equivalence to the IND-CCA security of Hashed ElGamal encryption (also known as Diffie-Hellman Integrated Encryption Standard) in the random oracle model [ABR01]. The LSRW assumption (named its authors [LRSW99]) is of importance since it is equivalent to the (UF-CMA) security of Camenisch-Lysyanskaya (CL) signatures [CL04]. CL signatures are a central building block for anonymous credentials [CL04, BCL04, BCS05], group signatures [CL04, ACHdM05], e-cash [CHL05], unclonable functions [CHK+06], batch verification [CHP07], and RFID encryption [ACdM05]. Via our results, the security of all these schemes is implied by the discrete logarithm assumption in the AGM.

Our result can be interpreted as follows. Every algorithm attacking one of the above-mentioned problems and schemes must solve the standard discrete logarithm problem directly, unless the algorithm relies on inherently non-algebraic operations. In particular, powerful techniques such as the index-calculus algorithms do not help in solving these problems any better then they do for solving the discrete logarithm problem directly.

Moreover, we show the *tight* equivalence of the security of the following schemes to the underlying hardness assumptions in the AGM:

- IND-CCA1 (aka lunchtime) security of the standard ElGamal Encryption to a parametrized variant of Decisional Diffie-Hellman assumption where in addition to g^x, g^y the adversary receives $g^{x^2}, \ldots, g^{x^q}$, where q is the maximal number of decryption queries.
- The UF-CMA security of the BLS signature scheme [BLS04] to the discrete logarithm problem in the random oracle model. Previous reductions non-tightly reduced from the CDH problem, with a tightness loss linear in the number of signing queries. This loss is known to be inherent [Cor02, KK12], even in the random oracle model.
- The security of the so far most efficient zero-knowledge SNARK scheme by Groth [Gro16] to a parametrized variant of the discrete logarithm problem, where in addition to g^x the adversary receives $g^{x^2}, \ldots, g^{x^{2n-1}}$, where n is the degree of the quadratic arithmetic programs. The only previous proof of the security of this scheme is in the generic group model.

[1] The knowledge of exponent assumption states that for every algorithm A that, given g and $\mathbf{X} = g^x$, outputs $(\mathbf{A}, \mathbf{B})$ with $\mathbf{B} = \mathbf{A}^x$, there exists an extractor algorithms that, given the same input, outputs a satisfying $(\mathbf{A}, \mathbf{B}) = (g^a, \mathbf{X}^a)$.

Relation to the Generic Group Model. The AGM is stronger (in the sense that it puts more restrictions on the attackers) than the Standard Model, but weaker than the GGMl. In spite of this, all of our reductions are purely generic algorithms. As mentioned above, any generic algorithm can be modeled within the AGM. In particular, combining arbitrary generic operations with algebraic ones will yield an algebraic algorithm. This suggests the following idea. Let H and G be two computational problems and let $\mathsf{A}_{\mathsf{alg}}$ be an algebraic algorithm that solves problem G. If we can convert $\mathsf{A}_{\mathsf{alg}}$ by means of a generic reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ into an algorithm $\mathsf{B}_{\mathsf{alg}}$ for problem H, then clearly, $\mathsf{B}_{\mathsf{alg}}$ is also an algebraic algorithm. However, we obtain an even stronger statement for free: Namely, if $\mathsf{A}_{\mathsf{gen}}$ is a generic algorithm solving G, then $\mathsf{B}_{\mathsf{gen}}$ is a generic algorithm solving H. This means that results in the AGM directly carry over to the GGM.

For this reason, we believe that our model offers an alternative, perhaps simpler method of proving the hardness of computational problems within the GGM. This applies in particular to interactive assumptions, which can be rather difficult to analyze in the GGM. For example, we prove that the discrete logarithm assumption implies the LRSW assumption in the AGM. As the discrete logarithm assumption holds in the GGM, we instantly obtain that the LRSW assumption holds in the GGM. The first (rigorous) proof of the LRSW assumption within the GGM was presented in the work of [BFF+14] (the original work [LRSW99] provided only a proof sketch), but was derived from a more general theorem and proven using an automated proof verification tool. We hope that our proof can offer some additional insight over the proof of [BFF+14]. Another example is our tight equivalence of the IND-CCA1 security of ElGamal and our parametrized variant of the Decisional Diffie-Hellman (DDH) assumption in the algebraic group model. Together with the known generic $\sqrt{p/q}$ attack on ElGamal [BG04] for certain primes p (see also [Che06]), our result proves the tight generic bound $\tilde{\Theta}(\sqrt{p/q})$ on the complexity of breaking IND-CCA1 security of ElGamal in the GGM.

We also remark that proofs in the AGM have an inherently different interpretation than proofs in the GGM. To analyze the hardness of an assumption in the GGM, one must explicitly augment the model by any functionality that is offered by the structure of the group. As a simple example, let us consider a group $\mathbb{G}$ which is equipped with a symmetric bilinear map $e\colon \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$. The bilinear map can be modeled in the GGM via an oracle. However, it is not clear whether e can be used to gather even further information about the elements of $\mathbb{G}$. Though it is widely believed that this is not the case, a proof in the GGM provides no answer to this question, because the GGM itself is based on the conjecture that e does not offer any functionality beyond a bilinear map. In contrast, the AGM captures any such exploit without the need of having to model it explicitly and considers the relation between two problems instead of their *individual hardness*. This means that if one can reduce H to G in the AGM and H is conjectured to remain hard with respect to algebraic algorithms, even when given e, then also G remains hard. No similar statement can be inferred in the GGM. Thus, the AGM allows for a more fine grained assessment of the hardness of computational problems than the GGM.

The gap between the two models becomes even more apparent if one considers structural properties of $\mathbb{G}$ which cannot be meaningfully modeled as an oracle in the GGM. As an example, consider the Jacobi symbol, which was shown to be generically hard to compute in [JS09]. Indeed, it was left as an open problem in [AM09] to re-examine the equivalence of factoring and breaking the RSA assumption if an additional oracle for the Jacobi symbol were given. Though their results are stated in the *generic ring model* rather than the GGM, it seems they are similarly confronted with the issue of explicitly modeling such an oracle.

Limitations of the AGM. As already noted, one of the main benefits of our model over the GGM is the ability to reason about algorithms that arbitrarily exploit the structure of

the group. So which algorithms are not covered in this manner? Obviously, outputting an obliviously sampled group element (with unknown representation) is forbidden. This coincides with the GGM of Maurer [Mau05] and which also excludes the possibility of obliviously sampling a random group element. For this reason, our model is strictly weaker than the one from [Mau05] in the sense that any security reduction derived in Maurer's GGM also holds in the AGM. In contrast, the GGM defined by Shoup [Sho97] *does* allow for such a sampling process. Similar to Maurer's GGM, we can allow obliviously sampling a random group element $\mathbf{X}$ through an additional oracle $\mathsf{O}()$ that can be called during the execution of $\mathsf{A}_{\mathsf{alg}}$. By definition, the outputs of $\mathsf{O}()$ are added to the list $\vec{\mathbf{L}}$. We have thus argued that both versions of the GGM (i.e., the ones by Maurer and Shoup) are strictly stronger than the AGM. Also note that simulating $\mathsf{O}()$ to $\mathsf{A}_{\mathsf{alg}}$ as part of a reduction is straight-forward and always possible; the reduction simply samples r and returns g^r to the adversary. As the reduction knows r, adding $\mathsf{O}()$ to an experiment does not change it and is completely without loss of generality. From a practical point of view, it seems that generating and outputting a random group element without knowing a representation is generally not of much help. We therefore believe that the AGM captures most algorithms of practical interest.

1.3 Related Work and Open Questions

We have already mentioned the semi generic group model (SGGM) [JR10] as related work, but we discuss here some key differences of their model to ours in more detail. First, the SGGM is a very restrictive model in the sense that the class of problems it captures is limited. The main theorem of [JR10] (Theorem 3) holds only for pairing-based computational problems in which the output consists of a single element in either one of the base groups. In contrast, the AGM does not require a pairing group setting and thus applies to a much broader class of computational problems. Second, by extending the AGM to pairing groups, we are able to model all three groups as algebraic and reason again about a broader class of problems, in which the output can also consist of elements in the target group. To extend the AGM to the pairing setting, we allow the algebraic adversary to compute any element in the target group by applying the pairing to elements in the respective base groups.

Dent [Den02] shows that the generic group model as proposed Shoup [Sho97] inherits the known weaknesses in the random oracle model [CGH98]. Thus, there exist schemes which can be proven secure in Shoup's GGM, but are pathologically insecure when viewed in the standard model. An interesting open question is whether the AGM bears similar weaknesses. A promising line of research related to this question has recently been initiated by Bitansky et al. [BCPR16]. Namely, they show that indistinguishability obfuscation (iO) implies the existence of non-extractable one-way functions. If these non-extractable one-way functions were furthermore algebraic (such as the knowledge of exponent assumption [Dam92]), then this would invalidate the AGM (under the assumption that iO exists).

Another promising direction for future research is to prove further reductions between common computational assumptions in the AGM. In particular, it would be interesting to classify different such assumptions within the AGM, for example along the lines of work [SS01, Kil01, Boy08, JR15, CM14, MRV16, GG17].

We leave it as an open problem to come up with a meaningful formalization of the AGM for *decisional assumptions*. At a technical level, the main difficulty in this task arises from the fact that an algorithm, i.e., distinguisher, in a decisional problem is asked to output a *bit* rather than a group element. Therefore, such an algorithm is trivially considered algebraic in our framework. It would therefore be interesting to develop a model which captures the algebraic properties of such algorithms in more detail.

A further potential for follow-up work would be to investigate whether it is possible to automate proofs in the AGM. Indeed, for the case of the GGM this has been considered in [BFF+14, ABS16] and it would be interesting to see if similar automated tools can be derived for the AGM.

Finally, we remark that all of our results require prime-order groups and do not yet extend to the setting of pairing groups. When generalizing our results to composite-order groups, we expect to encounter the following technical difficulty: Given, e.g., an equation of the form $ax \equiv_n b$, where n is composite, there might be (exponentially) many solutions for the unknown x in case $\mathsf{gcd}(a, n) > 1$. This interferes with the proof strategies presented in this work and requires a more involved analysis. In fact, proving a reduction from the discrete logarithm problem to the CDH problem in the AGM for group orders containing multiple prime factors (eg, $n = p^2$) is excluded by [MW98]. Hardness bounds in the GGM for composite-order groups have been considered in [Sho97, MW98, Mau05]. Generalizing the GGM to pairing groups has been the subject, e.g., of the works of [Boy08, KSW08, RLB+08]. Extending the AGM to either one of these regimes is an interesting line of research for future work.

2 Algebraic Algorithms

Algorithms. We denote by $s \xleftarrow{\$} S$ the uniform sampling of the variable s from the (finite) set S. All our algorithms are probabilistic (unless stated otherwise) and written in uppercase letters A, B. To indicate that algorithm A runs on some inputs $(x_1, \ldots, x_n)$ and returns y, we write $y \xleftarrow{\$} \mathsf{A}(x_1, \ldots, x_n)$. If A has access to an algorithm B (via oracle access) during its execution, we write $y \xleftarrow{\$} \mathsf{A}^{\mathsf{B}}(x_1, \ldots, x_n)$.

Security games. We use a variant of (code-based) *security games* [BR04]. In game $\mathbf{G}_{par}$ (defined relative to a set of parameters par), an adversary A interacts with a challenger that answers oracle queries issued by A. It has a main procedure and (possibly zero) oracle procedures which describe how oracle queries are answered. We denote the output of a game $\mathbf{G}_{par}$ between a challenger and an adversary A via $\mathbf{G}^{\mathsf{A}}_{par}$. A is said to *win* if $\mathbf{G}^{\mathsf{A}}_{par} = 1$. We define the *advantage* of A in $\mathbf{G}_{par}$ as $\mathbf{Adv}^{\mathbf{G}}_{par,\mathsf{A}} := \Pr\left[\mathbf{G}^{\mathsf{A}}_{par} = 1\right]$ and the running time of $\mathbf{G}^{\mathsf{A}}_{par}$ as $\mathbf{Time}^{\mathbf{G}}_{par,\mathsf{A}}$.

Security Reductions. Let $\mathbf{G}, \mathbf{H}$ be security games. We write $\mathbf{H}_{par} \overset{(\Delta_\varepsilon, \Delta_t)}{\Longrightarrow} \mathbf{G}_{par}$ if there exists an algorithm R (called $(\Delta_\varepsilon, \Delta_t)$-*reduction*) such that for all algorithms A, algorithm B defined as $\mathsf{B} := \mathsf{R}^{\mathsf{A}}$ satisfies

$$\mathbf{Adv}^{\mathbf{H}}_{par,\mathsf{B}} \geq \frac{1}{\Delta_\varepsilon} \cdot \mathbf{Adv}^{\mathbf{G}}_{par,\mathsf{A}}, \quad \mathbf{Time}^{\mathbf{H}}_{par,\mathsf{B}} \leq \Delta_t \cdot \mathbf{Time}^{\mathbf{G}}_{par,\mathsf{A}}.$$

2.1 Algebraic Security Games and Algorithms

We consider *algebraic security games* $\mathbf{G}_{\mathcal{G}}$ for which we set par to a fixed group description $\mathcal{G} = (\mathbb{G}, g, p)$, where $\mathbb{G}$ is a cyclic group of prime order p generated by g. In algebraic security games, we syntactically distinguish between elements of group $\mathbb{G}$ (written in bold, uppercase letters, e.g., $\mathbf{A}$) and all other elements, which must not depend on any group elements. As an example of an algebraic security game, consider the Computational Diffie-Hellman game $\mathbf{cdh}^{\mathsf{A}}_{\mathcal{G}}$, depicted in Figure 1 (left).

We now define algebraic algorithms. Intuitively, the only way for an algebraic algorithm to output a new group element $\mathbf{Z}$ is to derive it via group multiplications from known group elements.

$\underline{\mathbf{cdh}}^{\mathsf{A}}_{\mathcal{G}}$	$\underline{\mathbf{cdh}}^{\mathsf{A}_{\mathsf{alg}}}_{\mathcal{G}}$
00 $x, y \xleftarrow{\$} \mathbb{Z}_p$	00 $x, y \xleftarrow{\$} \mathbb{Z}_p$
01 $(\mathbf{X}, \mathbf{Y}) := (g^x, g^y)$	01 $(\mathbf{X}, \mathbf{Y}) := (g^x, g^y)$
02 $\mathbf{Z} \xleftarrow{\$} \mathsf{A}(\mathbf{X}, \mathbf{Y})$	02 $[\mathbf{Z}]_{\vec{z}} \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}(\mathbf{X}, \mathbf{Y})$
03 Return $(\mathbf{Z} = g^{xy})$	03 Return $(\mathbf{Z} = g^{xy})$

Figure 1: **Left:** Algebraic game **cdh** relative to group description $\mathcal{G} = (\mathbb{G}, g, p)$ and adversary A. All group elements are written in bold, uppercase letters. **Right:** Algebraic game **cdh** relative to group description $\mathcal{G} = (\mathbb{G}, g, p)$ and algebraic adversary $\mathsf{A}_{\mathsf{alg}}$. The algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ additionally returns a representation $\vec{z} = (a, b, c)$ of $\mathbf{Z}$ such that $\mathbf{Z} = g^a \mathbf{X}^b \mathbf{Y}^c$.

Definition 2.1 (Algebraic algorithm) An algorithm $\mathsf{A}_{\mathsf{alg}}$ executed in an algebraic game $\mathbf{G}_{\mathcal{G}}$ is called *algebraic* if for all group elements $\mathbf{Z}$ that $\mathsf{A}_{\mathsf{alg}}$ outputs (i.e., the elements in bold uppercase letters), it additionally provides the representation of $\mathbf{Z}$ relative to all previously received group elements. That is, if $\vec{\mathbf{L}}$ is the list of group elements $\mathbf{L}_0, \ldots, \mathbf{L}_m \in \mathbb{G}$ that $\mathsf{A}_{\mathsf{alg}}$ has received so far (w.l.o.g. $\mathbf{L}_0 = g$), then $\mathsf{A}_{\mathsf{alg}}$ must also provide a vector $\vec{z}$ such that $\mathbf{Z} = \prod_i \mathbf{L}_i^{z_i}$. We denote such an output as $[\mathbf{Z}]_{\vec{z}}$.

Remarks on Our Model. Algebraic algorithms were first considered in [BV98, PV05], where they are defined using an additional extractor algorithm which computes for an output group element a representation in basis $\vec{\mathbf{L}}$. We believe that our definition gives a simpler and cleaner definition of algebraic algorithms. If one assumes that the extractor algorithm has constant running time, then our definition is easily seen to be equivalent to theirs. Indeed, this view makes sense for algorithms in the GGM since the representation $\vec{z}$ trivially follows from the description of the algorithm. However, if running the extractor algorithm imposes some additional cost, then this will clearly affect the running times of our reductions. If the cost of the extractor is similar to that of the solver adversary, then reductions in our model that neither call an algebraic solver multiple times nor receive from it a non-constant amount of group elements (along with their representations) will remain largely the same in both models.

For the inputs to algebraic adversaries we syntactically distinguish group elements from other inputs and require that the latter not depend on any group elements. This is necessary to rule out pathological cases in which an algorithm receives "disguised" group elements and is forced to output an algebraic representation of them (which it might not know). To illustrate the issue, consider an efficient algorithm A, which on input $X' := \mathbf{X} \| \bot$ returns $\mathbf{X}$, where $\mathbf{X}$ is a group element, but X' is not. If A is algebraic then it must return a representation of $\mathbf{X}$ in g (the only group element previously seen), which would be the discrete logarithm of $\mathbf{X}$.

Allowing inputs of form X' while requiring algorithms to be algebraic leads to contradictions. (E.g., one could use $\mathsf{A}_{\mathsf{alg}}$ to compute discrete logarithms: given a challenge $\mathbf{X} = g^x$, run $[\mathbf{X}]_x \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}(\mathbf{X} \| \bot)$ and return x.) We therefore demand that non-group-element inputs must not depend on group elements. (Note that if $\mathsf{A}_{\mathsf{alg}}$'s input contains $\mathbf{X}$ explicitly then it can output $[\mathbf{X}]_{(0,1)}$ with a valid representation of $\mathbf{X}$ relative to $\vec{\mathbf{L}} = (g, \mathbf{X})$.)

Finally, we slightly abuse notation and let an algebraic algorithm also represent output group elements as combinations of previous *outputs*. This makes some of our proofs easier and is justified since all previous outputs must themselves have been given along with an according representation. Therefore, one can always recompute a representation that depends only on the initial inputs to the algebraic algorithm.

Integrating with Random Oracles in the AGM. As mentioned above, an algorithm A that samples (and outputs) a group element $\mathbf{X}$ obliviously, i.e., without knowing its representation,

is not algebraic. This appears to be problematic if one wishes to combine the AGM with the Random Oracle Model [BR93]. However, group elements output by the random oracle are included by definition in the list $\vec{\mathbf{L}}$. This means that for any such element, a representation is trivially available to $\mathsf{A}_{\mathsf{alg}}$.

2.2 Generic Security Games and Algorithms

Generic algorithms $\mathsf{A}_{\mathsf{gen}}$ are only allowed to use generic properties of group $\mathcal{G}$. Informally, an algorithm is generic if it works regardless of what group it is run in. This is usually modeled by giving an algorithm indirect access to group elements via abstract handles. It is straight-forward to translate all of our algebraic games into games that are syntactically compatible with generic algorithms accessing group elements only via abstract handles.

We say that winning algebraic game $\mathbf{G}_{\mathcal{G}}$ is (ε, t)-*hard in the generic group model* if for every generic algorithm $\mathsf{A}_{\mathsf{gen}}$ it holds that

$$\mathbf{Time}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}_{\mathsf{gen}}} \leq t \implies \mathbf{Adv}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}_{\mathsf{gen}}} \leq \varepsilon.$$

We remark that usually in the generic group model one considers group operations (i.e., oracle calls) instead of the running time. In our context it is more convenient to measure the running time instead, assuming every oracle call takes one unit time.

As an important example, consider the algebraic Discrete Logarithm Game $\mathbf{dlog}_{\mathcal{G}}$ in Figure 2 which is $(t^2/p, t)$-hard in the generic group model [Sho97, Mau05].

We assume that a generic algorithm $\mathsf{A}_{\mathsf{gen}}$ additionally provides the representation of $\mathbf{Z}$ relative to all previously received group elements, for all group elements $\mathbf{Z}$ that it outputs. This assumption is w.l.o.g. since a generic algorithm can only obtain new group elements by multiplying two known group elements; hence it always knows a valid representation. This way, every generic algorithm is also an algebraic algorithm.

Furthermore, if $\mathsf{B}_{\mathsf{gen}}$ is a generic algorithm and $\mathsf{A}_{\mathsf{alg}}$ is an algebraic algorithm, then $\mathsf{B}_{\mathsf{alg}} := \mathsf{B}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{gen}}$ is also is an algebraic algorithm. We refer to [Mau05] for more on generic algorithms.

2.3 Generic Reductions Between Algebraic Security Games

Let $\mathbf{G}_{\mathcal{G}}$ and $\mathbf{H}_{\mathcal{G}}$ be two algebraic security games. We write $\mathbf{H}_{\mathcal{G}} \overset{(\Delta_\varepsilon,\Delta_t)}{\Longrightarrow}_{\mathsf{alg}} \mathbf{G}_{\mathcal{G}}$ if there exists a generic algorithm $\mathsf{R}_{\mathsf{gen}}$ (called generic $(\Delta_\varepsilon, \Delta_t)$-*reduction*) such that for every algebraic algorithm $\mathsf{A}_{\mathsf{alg}}$, algorithm $\mathsf{B}_{\mathsf{alg}}$ defined as $\mathsf{B}_{\mathsf{alg}} := \mathsf{R}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{gen}}$ satisfies

$$\mathbf{Adv}^{\mathbf{H}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}} \geq \frac{1}{\Delta_\varepsilon} \cdot \mathbf{Adv}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}, \quad \mathbf{Time}^{\mathbf{H}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}} \leq \Delta_t \cdot \mathbf{Time}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}.$$

Note that we deliberately require reduction $\mathsf{R}_{\mathsf{gen}}$ to be generic. Hence, if $\mathsf{A}_{\mathsf{alg}}$ is algebraic, then $\mathsf{B}_{\mathsf{alg}} := \mathsf{R}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{gen}}$ is algebraic; if $\mathsf{A}_{\mathsf{alg}}$ is generic, then $\mathsf{B}_{\mathsf{alg}} := \mathsf{R}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{gen}}$ is generic. If one is only interested in algebraic adversaries, then it suffices to require reduction $\mathsf{R}_{\mathsf{gen}}$ to be algebraic. But in that case one can no longer infer that $\mathsf{B}_{\mathsf{alg}} := \mathsf{R}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{gen}}$ is generic in case $\mathsf{A}_{\mathsf{alg}}$ is generic.

Composing information-theoretic lower bounds with reductions in the AGM. The following lemma explains how statements in the AGM carry over to the GGM.

Lemma 2.2 *Let $\mathbf{G}_{\mathcal{G}}$ and $\mathbf{H}_{\mathcal{G}}$ be algebraic security games such that $\mathbf{H}_{\mathcal{G}} \overset{(\Delta_\varepsilon,\Delta_t)}{\Longrightarrow}_{\mathsf{alg}} \mathbf{G}_{\mathcal{G}}$ and winning $\mathbf{H}_{\mathcal{G}}$ is (ε, t)-hard in the GGM. Then, $\mathbf{G}_{\mathcal{G}}$ is $(\varepsilon \cdot \Delta_\varepsilon, t/\Delta_t)$-hard in the GGM.*

$\underline{\mathbf{dlog}}^{\mathsf{A}}_{\mathcal{G}}$	$\underline{\mathbf{lc\text{-}dh}}^{\mathsf{A}}_{\mathcal{G}}$	$\underline{\mathbf{sq\text{-}dh}}^{\mathsf{A}}_{\mathcal{G}}$
00 $x \xleftarrow{\$} \mathbb{Z}_p$	00 $x, y \xleftarrow{\$} \mathbb{Z}_p$	00 $x \xleftarrow{\$} \mathbb{Z}_p$
01 $\mathbf{X} := g^x$	01 $(\mathbf{X}, \mathbf{Y}) := (g^x, g^y)$	01 $\mathbf{X} := g^x$
02 $z \xleftarrow{\$} \mathsf{A}(\mathbf{X})$	02 $(\mathbf{Z}, u, v, w) \xleftarrow{\$} \mathsf{A}(\mathbf{X}, \mathbf{Y})$	02 $\mathbf{Z} \xleftarrow{\$} \mathsf{A}(\mathbf{X})$
03 Return $(z = x)$	03 Return $(\mathbf{Z} = g^{ux^2+vxy+wy^2} \wedge (u \neq 0 \vee v \neq 0 \vee w \neq 0))$	03 Return $(\mathbf{Z} = g^{x^2})$

Figure 2: Discrete Logarithm Game **dlog**, Square Diffie-Hellman Game **sq-dh**, and Linear Combination Diffie-Hellman Game **lc-dh** relative to group $\mathcal{G}$ and adversary A.

Proof. Let $\mathsf{A}_{\mathsf{gen}}$ be a generic algorithm playing in game $\mathbf{G}_{\mathcal{G}}$. Then by our premise there exists a generic algorithm $\mathsf{B}_{\mathsf{alg}} = \mathsf{R}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{gen}}$ such that

$$\mathbf{Adv}^{\mathbf{H}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}} \geq \tfrac{1}{\Delta_\varepsilon} \cdot \mathbf{Adv}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}, \quad \mathbf{Time}^{\mathbf{H}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}} \leq \Delta_t \cdot \mathbf{Time}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}.$$

Assume $\mathbf{Time}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}} \leq t/\Delta_t$; then $\mathbf{Time}^{\mathbf{H}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}} \leq \Delta_t \cdot \mathbf{Time}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}} \leq t$. Since winning $\mathbf{H}_{\mathcal{G}}$ is (ε, t)-hard in the GGM, it follows that

$$\varepsilon \geq \mathbf{Adv}^{\mathbf{H}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}} \geq \tfrac{1}{\Delta_\varepsilon} \cdot \mathbf{Adv}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$$

and thus $\varepsilon \cdot \Delta_\varepsilon \geq \mathbf{Adv}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$, which proves that $\mathbf{G}_{\mathcal{G}}$ is $(\varepsilon\Delta_\varepsilon, t/\Delta_t)$-hard in the GGM. ∎

3 The Diffie-Hellman Assumption and Variants

In this section we consider some variants of the standard Diffie-Hellman assumption [DH76] and prove them to be equivalent to the discrete logarithm assumption (defined via algebraic game $\mathbf{dlog}_{\mathcal{G}}$ of Figure 2) in the Algebraic Group Model.

3.1 Computational Diffie-Hellman

Consider the Square Diffie-Hellman Assumption [MW99] described in algebraic game $\mathbf{sq\text{-}dh}_{\mathcal{G}}$ and the Linear Combination Diffie-Hellman Assumption described in algebraic game $\mathbf{lc\text{-}dh}_{\mathcal{G}}$ (both in Figure 2), which will be convenient for the proof of Theorem 3.3.

As a warm-up we now prove that the Discrete Logarithm assumption is tightly equivalent to the Diffie-Hellman, the Square Diffie-Hellman, and the Linear Combination Diffie-Hellman Assumption in the Algebraic Group Model. The equivalence of the Square Diffie-Hellman and Diffie-Hellman problems was previously proven in [MW99, BDZ03].

Theorem 3.1 $\mathbf{dlog}_{\mathcal{G}} \overset{(1,1)}{\Longrightarrow}_{\mathsf{alg}} \{\mathbf{cdh}_{\mathcal{G}}, \mathbf{sq\text{-}dh}_{\mathcal{G}}\}$ *and* $\mathbf{dlog}_{\mathcal{G}} \overset{(3,1)}{\Longrightarrow}_{\mathsf{alg}} \mathbf{lc\text{-}dh}_{\mathcal{G}}$.

Proof. Let $\mathsf{A}_{\mathsf{alg}}$ be an algebraic adversary executed in game $\mathbf{sq\text{-}dh}_{\mathcal{G}}$; cf. Figure 3.

As $\mathsf{A}_{\mathsf{alg}}$ is an algebraic adversary, it returns a solution $\mathbf{Z}$ together with a representation $(a, b) \in \mathbb{Z}_p^2$ such that

$$\mathbf{Z} = g^{x^2} = g^a (g^x)^b. \tag{1}$$

We now show how to construct a generic reduction $\mathsf{R}_{\mathsf{gen}}$ that calls $\mathsf{A}_{\mathsf{alg}}$ exactly once such that for $\mathsf{B}_{\mathsf{alg}} := \mathsf{R}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{gen}}$ we have

$$\mathbf{Adv}^{\mathbf{dlog}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}} = \mathbf{Adv}^{\mathbf{sq\text{-}dh}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}.$$

$\underline{\mathbf{sq\text{-}dh}^{\mathsf{A}_{\mathsf{alg}}}_{\mathcal{G}}}$
00 $x \xleftarrow{\$} \mathbb{Z}_p$
01 $\mathbf{X} := g^x$
02 $[\mathbf{Z}]_{(a,b)} \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}(\mathbf{X})$
03 Return $(\mathbf{Z} = g^{x^2})$

Figure 3: Algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ playing in $\mathbf{sq\text{-}dh}_{\mathcal{G}}$.

$\mathsf{R}_{\mathsf{gen}}$ works as follows. On input a discrete logarithm instance $\mathbf{X}$, it runs $\mathsf{A}_{\mathsf{alg}}$ on $\mathbf{X}$. Suppose $\mathsf{A}_{\mathsf{alg}}$ is successful. Equation (1) is equivalent to the quadratic equation $x^2 - bx - a \equiv_p 0$ with at most two solutions in x. (In general such equations are not guaranteed to have a solution but since the representation is valid and $\mathsf{A}_{\mathsf{alg}}$ is assumed to be correct, there exists at least one solution for x.) $\mathsf{R}_{\mathsf{gen}}$ can test which one (out of the two) is the correct solution x by testing against $\mathbf{X} = g^x$. Moreover, it is easy to see that $\mathsf{R}_{\mathsf{gen}}$ only performs generic group operations and is therefore generic. Hence, $\mathsf{B}_{\mathsf{alg}} := \mathsf{R}_{\mathsf{gen}}^{\mathsf{A}_{\mathsf{alg}}}$ is algebraic, which proves

$$\mathbf{dlog}_{\mathcal{G}} \overset{(1,1)}{\Longrightarrow}_{\mathsf{alg}} \mathbf{sq\text{-}dh}_{\mathcal{G}}.$$

The statement $\mathbf{dlog}_{\mathcal{G}} \overset{(1,1)}{\Longrightarrow}_{\mathsf{alg}} \mathbf{cdh}_{\mathcal{G}}$ follows, since given an adversary against $\mathbf{cdh}_{\mathcal{G}}$ (see Figure 1), we can easily construct an adversary against $\mathbf{sq\text{-}dh}_{\mathcal{G}}$ that runs in the same time and has the same probability of success (given $\mathbf{X} = g^x$, sample $r \xleftarrow{\$} \mathbb{Z}_p$, run the **cdh** adversary on $(\mathbf{X}, \mathbf{X}^r)$, obtain $\mathbf{Z}$ and return $\mathbf{Z}^{\frac{1}{r}}$).

It remains to show that $\mathbf{sq\text{-}dh}_{\mathcal{G}} \overset{(3,1)}{\Longrightarrow}_{\mathsf{alg}} \mathbf{lc\text{-}dh}_{\mathcal{G}}$. Given an algebraic solver $\mathsf{C}_{\mathsf{alg}}$ executed in game $\mathbf{lc\text{-}dh}_{\mathcal{G}}$, we construct an adversary $\mathsf{A}_{\mathsf{alg}}$ against $\mathbf{sq\text{-}dh}_{\mathcal{G}}$ as follows: On input $\mathbf{X} = g^x$, $\mathsf{A}_{\mathsf{alg}}$ samples $r \xleftarrow{\$} \mathbb{Z}_p$ and computes either $(\mathbf{X}, g^r)$, $(g^r, \mathbf{X})$, or $(\mathbf{X}, \mathbf{X}^r)$ each with probability $1/3$. Note that this instance is correctly distributed. It then runs $\mathsf{C}_{\mathsf{alg}}$ on the resulting tuple $(\mathbf{X}_1, \mathbf{X}_2)$ and receives $(\mathbf{Z}, u, v, w)$ together with (a, b, c) s.t. $\mathbf{Z} = g^a \mathbf{X}_1^b \mathbf{X}_2^c$. If $u \neq 0$, then the choice $\mathbf{X}_1 = \mathbf{X}$, $\mathbf{X}_2 = g^r$ yields $\mathbf{Z} = g^{ux^2 + vxr + wr^2}$, from which g^{x^2} can be computed as $g^{x^2} = (\mathbf{Z}\mathbf{X}^{-vr} g^{-wr^2})^{\frac{1}{u}}$. Clearly, $\mathsf{A}_{\mathsf{alg}}$ is able to compute an algebraic representation of g^{x^2} from the values (a, b, c) and thus is algebraic itself. The cases $v \neq 0, w \neq 0$ follow in a similar fashion. ■

Corollary 3.2 $\mathbf{cdh}_{\mathcal{G}}$ *and* $\mathbf{sq\text{-}dh}_{\mathcal{G}}$ *are* $(t^2/p, t)$*-hard in the generic group model and* $\mathbf{lc\text{-}dh}_{\mathcal{G}}$ *is* $(3t^2/p, t)$*-hard in the generic group model.*

For the subsequent sections and proofs, we will not make explicit the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ every time (as done above).

3.2 Strong Diffie-Hellman

Consider the Strong Diffie-Hellman Assumption [ABR01] described via game $\mathbf{sdh}_{\mathcal{G}}$ in Figure 4. We now prove that the Discrete Logarithm Assumption (non-tightly) implies the Strong Diffie-Hellman Assumption in the Algebraic Group Model. We briefly present the main ideas of the proof. The full proof of Theorem 3.3 can be found in Supplementary Material A.1. Let $\mathsf{A}_{\mathsf{alg}}$ be an algebraic adversary playing in $\mathbf{sdh}_{\mathcal{G}}$ and let $\mathbf{Z} = g^z$ denote the Discrete Logarithm challenge. We show an adversary $\mathsf{B}_{\mathsf{alg}}$ against $\mathbf{dlog}_{\mathcal{G}}$ that simulates $\mathbf{sdh}_{\mathcal{G}}$ to $\mathsf{A}_{\mathsf{alg}}$. $\mathsf{B}_{\mathsf{alg}}$ appropriately answers $\mathsf{A}_{\mathsf{alg}}$'s queries to the oracle $\mathsf{O}(\cdot, \cdot)$ by using the algebraic representation of the queried elements provided by $\mathsf{A}_{\mathsf{alg}}$. Namely, when $(\mathbf{Y}', \mathbf{Z}')$ is asked to the oracle, $\mathsf{B}_{\mathsf{alg}}$ obtains vectors $\vec{b}, \vec{c}$ such that $\mathbf{Y}' = g^{b_1} \mathbf{X}^{b_2} \mathbf{Y}^{b_3}$ and $\mathbf{Z}' = g^{c_1} \mathbf{X}^{c_2} \mathbf{Y}^{c_3}$. As long as $b_2 = b_3 = 0$, $\mathsf{B}_{\mathsf{alg}}$ can answer all of $\mathsf{A}_{\mathsf{alg}}$'s

$\underline{\mathbf{sdh}}^{\mathsf{A}}_{\mathcal{G}}$	$\underline{\mathsf{O}}(\mathbf{Y}', \mathbf{Z}')$
00 $x, y \xleftarrow{\$} \mathbb{Z}_p$	04 Return $(\mathbf{Z}' = (\mathbf{Y}')^x)$
01 $(\mathbf{X}, \mathbf{Y}) := (g^x, g^y)$	
02 $\mathbf{Z} \xleftarrow{\$} \mathsf{A}^{\mathsf{O}(\cdot,\cdot)}(\mathbf{X}, \mathbf{Y})$	
03 Return $(\mathbf{Z} = g^{xy})$	

Figure 4: Strong Diffie-Hellman Game **sdh** relative to $\mathcal{G}$ and adversary A.

queries by checking whether $\mathbf{X}^{b_1} = \mathbf{Z}'$. On the other hand, if $b_2 \neq 0$ or $b_3 \neq 0$, then $\mathsf{B}_{\mathsf{alg}}$ simply returns 0. Informally, the simulation will be perfect unless $\mathsf{A}_{\mathsf{alg}}$ manages to compute a valid solution to $\mathbf{lc\text{-}dh}_{\mathcal{G}}$. All of these games can be efficiently simulated by $\mathsf{B}_{\mathsf{alg}}$, as we have shown in the previous section.

Theorem 3.3 $\mathbf{dlog}_{\mathcal{G}} \overset{(4q,1)}{\Longrightarrow}_{\mathsf{alg}} \mathbf{sdh}_{\mathcal{G}}$, *where q is the maximum number of queries to oracle $\mathsf{O}(\cdot,\cdot)$ in* $\mathbf{sdh}_{\mathcal{G}}$.

Corollary 3.4 $\mathbf{sdh}_{\mathcal{G}}$ *is $(\frac{t^2}{4pq}, t)$-hard in the generic group model.*

4 The LRSW Assumption

The interactive LRSW assumption [LRSW99, CL04] is defined via the algebraic security game **lrsw** in Figure 5.

$\underline{\mathbf{lrsw}}^{\mathsf{A}}_{\mathcal{G}}$	$\underline{\mathsf{O}}(m_j)$ *//For query j*
00 $Q := \emptyset$	05 $r_j \xleftarrow{\$} \mathbb{Z}_p$;
01 $x, y \xleftarrow{\$} \mathbb{Z}_p$	06 $\mathbf{A}_j := g^{r_j}$
02 $\mathbf{X} := g^x, \mathbf{Y} := g^y$	07 $\mathbf{B}_j := g^{r_j y}$
03 $(m^*, \mathbf{A}^*, \mathbf{B}^*, \mathbf{C}^*) \xleftarrow{\$} \mathsf{A}^{\mathsf{O}(\cdot)}(\mathbf{X}, \mathbf{Y})$	08 $\mathbf{C}_j := g^{r_j m_j xy + r_j x}$
04 Return $(m^* \notin Q \wedge m^* \neq 0 \wedge \mathbf{A}^* \neq 1$	09 $Q := Q \cup \{m_j\}$
$\wedge\ \mathbf{B}^* = (\mathbf{A}^*)^y \wedge\ \mathbf{C}^* = (\mathbf{A}^*)^{m^* xy + x})$	10 Return $(\mathbf{A}_j, \mathbf{B}_j, \mathbf{C}_j)$

Figure 5: Game **lrsw** relative to $\mathcal{G}$ and adversary A.

We now prove that the LRSW assumption is (non-tightly) implied by the Discrete Logarithm Assumption in the Algebraic Group Model. We give a high-level sketch of the main ideas here and defer the full proof of Theorem 4.1 to Supplementary Material A.2. Let $\mathsf{A}_{\mathsf{alg}}$ be an algebraic adversary playing in $\mathbf{lrsw}_{\mathcal{G}}$ and let $\mathbf{Z} = g^z$ denote the Discrete Logarithm challenge. We construct an adversary $\mathsf{B}_{\mathsf{alg}}$ against $\mathbf{dlog}_{\mathcal{G}}$, which simulate $\mathbf{lrsw}_{\mathcal{G}}$ to $\mathsf{A}_{\mathsf{alg}}$ by embedding the value of z in one of three possible ways. Namely, it either sets $\mathbf{X} := \mathbf{Z}$ or $\mathbf{Y} := \mathbf{Z}$, or it chooses a random the query by $\mathsf{A}_{\mathsf{alg}}$ to the oracle $\mathsf{O}(\cdot)$ in $\mathbf{lrsw}_{\mathcal{G}}$ to embed the value of z. These behaviours correspond in our proof to the adversaries $\mathsf{C}_{\mathsf{alg}}, \mathsf{D}_{\mathsf{alg}}$, and $\mathsf{E}_{\mathsf{alg}}$, respectively. After obtaining a solution $(m^*, [\mathbf{A}^*]_{\vec{a}}, [\mathbf{B}^*]_{\vec{b}}, [\mathbf{C}^*]_{\vec{c}})$ on a fresh value $m^* \neq 0$ from $\mathsf{A}_{\mathsf{alg}}$, the adversaries use the algebraic representations $\vec{a}, \vec{b}, \vec{c}$ obtained from $\mathsf{A}_{\mathsf{alg}}$ to suitably rewrite the values of $\mathbf{A}^*, \mathbf{C}^*$ (Lemma A.1). They then make use of the relation $(\mathbf{A}^*)^{(xm^*y+x)} = \mathbf{C}^*$ to obtain an equation mod p, which in turn gives z.

Theorem 4.1 $\mathbf{dlog}_{\mathcal{G}} \overset{(6q,1)}{\Longrightarrow}_{\mathsf{alg}} \mathbf{lrsw}_{\mathcal{G}}$, *where $q \geq 6$ is the maximum number of queries to $\mathsf{O}(\cdot)$ in* $\mathbf{lrsw}_{\mathcal{G}}$.

Corollary 4.2 $\mathbf{lrsw}_{\mathcal{G}}$ *is $(t, \frac{t^2}{6pq})$-hard in the generic group model.*

$\underline{\textbf{ind-cca1}}^{\mathsf{A}}_{\mathsf{KEM},par,b}$	$\underline{\mathsf{Dec}}(C)$	$\underline{\mathsf{Enc}}()$ //*One time*
00 $(pk, sk) \xleftarrow{\$} \mathsf{Gen}(par)$	//*Before* Enc *is called*	05 $(K_0^*, C^*) \xleftarrow{\$} \mathsf{Enc}(pk)$
01 $b' \xleftarrow{\$} \mathsf{A}^{\mathsf{Dec},\mathsf{Enc}}(pk)$	03 $K \xleftarrow{\$} \mathsf{Dec}(C, sk)$	06 $K_1^* \xleftarrow{\$} \mathcal{K}$
02 Return b'	04 Return K	07 Return (K_b^*, C^*)

Figure 6: IND-CCA1 Game **ind-cca1** relative to KEM $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, parameters par, and adversary A.

5 ElGamal Encryption

In this section we prove that the IND-CCA1 (aka. lunchtime security) of the ElGamal encryption scheme (in its abstraction as a KEM) is implied by a parametrized ("q-type") variant of the Decision Diffie-Hellman Assumption in the Algebraic Group Model.

Advantage for decisional algebraic security games. We parameterize a *decisional* algebraic game $\mathbf{G}$ (such as the game in Figure 7) with a parameter bit b. We define the advantage of adversary A in $\mathbf{G}$ as

$$\mathbf{Adv}^{\mathbf{G}}_{par,\mathsf{A}} := \left| \Pr\left[\mathbf{G}^{\mathsf{A}}_{par,0} = 1\right] - \Pr\left[\mathbf{G}^{\mathsf{A}}_{par,1} = 1\right] \right|.$$

We define $\mathbf{Time}^{\mathbf{G}}_{par,\mathsf{A}_{\mathsf{alg}}}$ independently of the parameter bit b, i.e., we consider only adversaries that have the same running time in both games $\mathbf{G}_{par,0}, \mathbf{G}_{par,1}$. In order to cover games that define the security of schemes (rather than assumptions), instead of $par = \mathcal{G}$, we only require that $\mathcal{G}$ be *included* in par. Let $\mathbf{G}_{par}, \mathbf{H}_{par}$ be decisional algebraic security games. As before, we write $\mathbf{H}_{par} \overset{(\Delta_\varepsilon, \Delta_t)}{\Longrightarrow}_{\mathsf{alg}} \mathbf{G}_{par}$ if there exists a generic algorithm $\mathsf{R}_{\mathsf{gen}}$ (called generic $(\Delta_\varepsilon, \Delta_t)$-*reduction*) such that for algebraic algorithm $\mathsf{B}_{\mathsf{alg}}$ defined as $\mathsf{B}_{\mathsf{alg}} := \mathsf{R}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{gen}}$, we have

$$\mathbf{Adv}^{\mathbf{H}}_{par,\mathsf{B}_{\mathsf{alg}}} \geq \frac{1}{\Delta_\varepsilon} \cdot \mathbf{Adv}^{\mathbf{G}}_{par,\mathsf{A}_{\mathsf{alg}}}, \quad \mathbf{Time}^{\mathbf{H}}_{par,\mathsf{B}_{\mathsf{alg}}} \leq \Delta_t \cdot \mathbf{Time}^{\mathbf{G}}_{par,\mathsf{A}_{\mathsf{alg}}}.$$

Key Encapsulation Mechanisms. A *key encapsulation mechanism* (KEM for short) $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a triple of algorithms together with a symmetric-key space $\mathcal{K}$. The randomized *key generation algorithm* Gen takes as input a set of parameters, par, and outputs a public/secret key pair (pk, sk). The *encapsulation algorithm* Enc takes as input a public key pk and outputs a key/ciphertext pair (K, C) such that $K \xleftarrow{\$} \mathcal{K}$. The deterministic *decapsulation algorithm* Dec takes as input a secret key sk and a ciphertext C and outputs a key $K \in \mathcal{K}$ or a special symbol $\perp$ if C is invalid. We require that KEM be *correct*: For all possible pairs (K, C) output by $\mathsf{Enc}(pk)$, we have $\mathsf{Dec}(sk, C) = K$. We formalize IND-CCA1 security of a KEM via the games (for $b = 0, 1$) depicted in Figure 6.

In the following, we consider the ElGamal KEM EG defined in Figure 8. We also consider a stronger variant of the well-known Decisional Diffie-Hellman (DDH) assumption that is parametrized by an integer q. In the q-DDH game, defined in Figure 7, the adversary receives, in addition to (g^x, g^y), the values $g^{x^2}, \ldots, g^{x^q}$.

$\underline{\textbf{q-ddh}}^{\mathsf{A}}_{\mathcal{G},b}$
00 $x, r, z \xleftarrow{\$} \mathbb{Z}_p$
01 $b' \xleftarrow{\$} \mathsf{A}(g^x, g^{x^2}, ..., g^{x^q}, g^r, g^{xr+zb})$
02 Return b'

Figure 7: q-Decisional Diffie-Hellman Game **q-ddh** relative to $\mathcal{G}$ and adversary A.

$\underline{\mathsf{Gen}}(\mathcal{G})$	$\underline{\mathsf{Enc}}(pk)$:	$\underline{\mathsf{Dec}}(\mathbf{C}, sk)$:
00 $x \xleftarrow{\$} \mathbb{Z}_p$	03 $r \xleftarrow{\$} \mathbb{Z}_p$	07 If $\mathbf{C} \notin \mathbb{G}$
01 $\mathbf{X} := g^x$	04 $\mathbf{C} := g^r$	08 Return $\perp$
02 Return $(pk, sk) := (\mathbf{X}, x)$	05 $\mathbf{K} := \mathbf{X}^r$	09 $\tilde{\mathbf{K}} := \mathbf{C}^x$
	06 Return $(\mathbf{K}, \mathbf{C})$	10 Return $\tilde{\mathbf{K}}$

Figure 8: ElGamal KEM $\mathsf{EG} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$

Lemma 5.1 *[Che06] For $q < p^{1/3}$, $\mathbf{q\text{-}ddh}_{\mathcal{G}}$ is $(\frac{t^2 q}{p \log p}, t)$-hard in the generic group model.*

The proof of the following theorem can be found in Supplementary Material A.3.

Theorem 5.2 $\mathbf{ind\text{-}cca1}_{\mathsf{EG},\mathcal{G}} \overset{(1,1)}{\Longleftrightarrow}_{\mathsf{alg}} \mathbf{q\text{-}ddh}_{\mathcal{G}}$, *where $q-1$ is the maximal number of queries to $\mathsf{Dec}(\cdot)$ in $\mathbf{ind\text{-}cca1}_{\mathsf{EG},\mathcal{G}}$.*

$\underline{\mathbf{ind\text{-}cca1}^{\mathsf{A}}_{\mathsf{EG},\mathcal{G},0}}$ $\boxed{\mathbf{ind\text{-}cca1}^{\mathsf{A}}_{\mathsf{EG},\mathcal{G},1}}$	$\underline{\mathsf{Dec}}([\mathbf{C}]_{\vec{a}})$	$\underline{\mathsf{Enc}}()$ *//One time*
00 $x \xleftarrow{\$} \mathbb{Z}_p$	*//Before* Enc *is called*	06 $r \xleftarrow{\$} \mathbb{Z}_p$
01 $\mathbf{X} := g^x$	04 $\mathbf{K} := \mathbf{C}^x$	07 $\mathbf{C}^* := g^r$
02 $b' \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}^{\mathsf{Dec},\mathsf{Enc}}(\mathbf{X})$	05 Return $\mathbf{K}$	08 $\mathbf{K}^* := \mathbf{X}^r$
03 Return b'		09 $\boxed{\mathbf{K}^* \xleftarrow{\$} \mathcal{K}}$
		10 Return $(\mathbf{K}^*, \mathbf{C}^*)$

Figure 9: Games $\mathbf{ind\text{-}cca1}^{\mathsf{A}}_{\mathsf{EG},\mathcal{G},0}$ and $\mathbf{ind\text{-}cca1}^{\mathsf{A}}_{\mathsf{EG},\mathcal{G},1}$ with algebraic adversary $\mathsf{A}_{\mathsf{alg}}$. The boxed statement is only executed in $\mathbf{ind\text{-}cca1}^{\mathsf{A}}_{\mathsf{EG},\mathcal{G},1}$.

Corollary 5.3 *For $q < p^{1/3}$, $\mathbf{ind\text{-}cca1}_{\mathsf{EG},\mathcal{G}}$ is $(\frac{t^2 q}{p \log p}, t)$-hard in the generic group model, where $q-1$ is the maximal number of queries to $\mathsf{Dec}(\cdot)$ in $\mathbf{ind\text{-}cca1}_{\mathsf{EG},\mathcal{G}}$.*

6 Tight Reduction for the BLS Scheme

For this section, we introduce the notion of groups $\mathbb{G}$ equipped with a symmetric, (non-degenerate) bilinear map $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G}_T$ denotes the so-called *target group*. We now set $\mathcal{G} = (p, \mathbb{G}, \mathbb{G}_T, g, e)$.

Signature Schemes. A *signature scheme* $\mathsf{SIG} = (\mathsf{SIGGen}, \mathsf{SIGSig}, \mathsf{SIGVer})$ is a triple of algorithms. The randomized *key generation algorithm* SIGGen takes as input a set of parameters, *par*, and outputs a public/secret key pair (pk, sk). The randomized *signing algorithm* SIGSig takes as input a secret key sk and a message m in the message space $\mathcal{M}$ and outputs a signature σ in the signature space $\mathcal{S}$. The deterministic *signature verification algorithm* SIGVer takes as input a public key pk, a message m, and a signature σ and outputs $b \in \{0, 1\}$. We require that SIG be *correct*: For all possible pairs (pk, sk) output by SIGGen, and all messages $m \in \mathcal{M}$, we have $\Pr[\mathsf{SIGVer}(pk, m, \mathsf{SIGSig}(m, sk)) = 1] = 1$. We formalize *unforgeability under chosen message attacks* for SIG via game $\mathbf{uf\text{-}cma}_{\mathsf{SIG},par}$ depicted in Figure 10.

In the following, we show how in the AGM with a random oracle, the security of the BLS signature scheme [BLS04], depicted in Figure 11, can be tightly reduced to the discrete logarithm problem. Boneh, Lynn and Shacham [BLS04] only prove a loose reduction to the CDH problem. In the AGM we manage to improve the quality of the reduction, leveraging the fact that a

$\underline{\textbf{uf-cma}^{\mathsf{A}}_{\mathsf{SIG},par}}$	$\underline{\mathsf{O}}(m)$
00 $(pk, sk) \xleftarrow{\$} \mathsf{SIGGen}$	04 $Q := Q \cup \{m\}$
01 $Q := \emptyset$	05 $\sigma \xleftarrow{\$} \mathsf{SIGSig}(m, sk)$
02 $(m^*, \sigma^*) \xleftarrow{\$} \mathsf{A}^{\mathsf{O}(\cdot)}$	06 Return σ
03 Return $(m^* \notin Q \wedge \mathsf{SIGVer}(m^*, \sigma^*))$	

Figure 10: Game **uf-cma** defining (existential) unforgeability under chosen-message attacks for signature scheme SIG, parameters par and adversary A.

forgery comes with a representation in the basis of all previously answered random oracle and signature queries. We embed a discrete logarithm challenge in either the secret key or inside the random oracle queries–a choice that remains hidden from the adversary. Depending on the adversary's behavior we always solve the discrete logarithm challenge in one of the cases.

$\underline{\mathsf{BLSGen}}(\mathcal{G})$	$\underline{\mathsf{BLSSig}}(m)$	$\underline{\mathsf{BLSVer}}(m, \mathbf{\Sigma})$
00 $x \xleftarrow{\$} \mathbb{Z}_p$	05 $\mathbf{\Sigma} := H(m)^x$	07 Return $(e(H(m), \mathbf{X}) = e(\mathbf{\Sigma}, g))$
01 $\mathbf{X} := g^x$	06 Return $\mathbf{\Sigma}$	
02 $sk := x$		
03 $pk := \mathbf{X}$		
04 Return (pk, sk)		

Figure 11: Boneh, Lynn and Shacham's signature scheme $\mathsf{BLS}_{\mathcal{G}}$. Here, H is a hash function that is modeled as a random oracle.

Theorem 6.1 $\mathbf{dlog}_{\mathcal{G}} \overset{(4,1)}{\Longrightarrow}_{\mathsf{alg}} \mathbf{uf\text{-}cma}_{\mathsf{BLS},\mathcal{G}}$ *in the random oracle model.*

Proof. Let $\mathsf{A}_{\mathsf{alg}}$ be an algebraic adversary playing in $\mathbf{G} := \mathbf{uf\text{-}cma}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{BLS},\mathcal{G}}$, depicted in Figure 12.

As $\mathsf{A}_{\mathsf{alg}}$ is an algebraic adversary, at the end of $\mathbf{G}$, it returns a forgery $\mathbf{\Sigma}^*$ on a message $m^* \notin Q$ together with a representation $\vec{a} = (\hat{a}, a', \bar{a}_1, ..., \bar{a}_q, \tilde{a}_1, ..., \tilde{a}_q)$ s.t.

$$\mathbf{\Sigma}^* = H(m^*)^x = g^{\hat{a}} \mathbf{X}^{a'} \prod_{i=1}^{q} \mathbf{H}_i^{\bar{a}_i} \prod_{i=1}^{q} \mathbf{\Sigma}_i^{\tilde{a}_i}. \tag{2}$$

Here, the representation is split (from left to right) into powers of the generator g, the public key $\mathbf{X}$, all of the answers to hash queries $\mathbf{H}_i, i \in [q]$, and the signatures $\mathbf{\Sigma}_i, i \in [q]$, returned by the signing oracle. In the following, we denote the discrete logarithm of $H(m^*)$ w.r.t. basis g as r^* and for all $i \in [q]$, we denote the discrete logarithm of $H(m_i)$ as r_i. Equation (2) is thus equivalent to

$$xr^* \equiv_p x(a' + \Sigma_i r_i \tilde{a}_i) + (\hat{a} + \Sigma_i r_i \bar{a}_i). \tag{3}$$

$\underline{\mathbf{G}^{\mathsf{A}_{\mathsf{alg}}}}$	$\underline{\mathsf{O}}(m_i)$	$\underline{\mathsf{H}}(m_i)$
00 $x \xleftarrow{\$} \mathbb{Z}_p$	05 $Q := Q \cup \{m_i\}$	08 $\mathbf{H}_i \leftarrow H(m_i)$
01 $\mathbf{X} := g^x$	06 $\mathbf{\Sigma}_i \leftarrow \mathsf{H}(m_i)^x$	09 Return $\mathbf{H}_i$
02 $Q := \emptyset$	07 Return $\mathbf{\Sigma}_i$	
03 $(m^*, [\mathbf{\Sigma}^*]_{\vec{a}}) \xleftarrow{\$} \mathsf{A}^{\mathsf{O}(\cdot),\mathsf{H}(\cdot)}_{\mathsf{alg}}(\mathbf{X})$		
04 Return $(m^* \notin Q \wedge \mathbf{\Sigma}^* = \mathsf{H}(m^*)^x)$		

Figure 12: Game $\mathbf{G} = \mathbf{uf\text{-}cma}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{BLS},\mathcal{G}}$ relative to adversary $\mathsf{A}_{\mathsf{alg}}$.

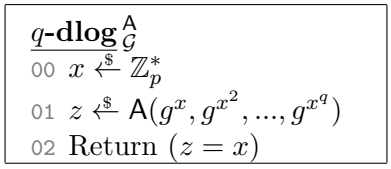

Figure 13: q-Discrete Logarithm Game q-**dlog** relative to $\mathcal{G}$ and adversary A.

We remark that since $\mathsf{A}_{\mathsf{alg}}$ wins $\mathbf{G}$, the sum $\Sigma_i r_i \tilde{a}_i$ may not include a term of the form $r^* \tilde{a}^*$ (since $\mathsf{A}_{\mathsf{alg}}$ may not query the signing oracle on m^*). We will now describe the behavior of adversaries $\mathsf{C}_{\mathsf{alg}}, \mathsf{D}_{\mathsf{alg}}$ (depicted in Figures 27 and 28, respectively) playing in the discrete logarithm game. Each of these adversaries simulates $\mathbf{G}$ to $\mathsf{A}_{\mathsf{alg}}$ in a different way. Concretely, we prove the following lemma in.

Lemma 6.2 *There exist* $\mathsf{C}_{\mathsf{alg}}, \mathsf{D}_{\mathsf{alg}}$ *playing in the discrete logarithm game such that:*

$$\Pr[\mathbf{dlog}^{\mathsf{C}_{\mathsf{alg}}} = 1] = \Pr[\mathbf{G} = 1 \mid F] \tag{4}$$

$$\Pr[\mathbf{dlog}^{\mathsf{D}_{\mathsf{alg}}} = 1] \geq \tfrac{p-1}{p} \Pr[\mathbf{G} = 1 \mid \neg F] \tag{5}$$

Now, we can simply let an adversary $\mathsf{B}_{\mathsf{alg}}$ choose to emulate one of the described adversaries $\mathsf{C}_{\mathsf{alg}}$ or $\mathsf{D}_{\mathsf{alg}}$ with probability $\frac{1}{2}$ each. All in all, $\mathbf{Adv}^{\mathsf{B}_{\mathsf{alg}},\mathcal{G}}_{\mathbf{dlog}} \geq \frac{p-1}{2p} \mathbf{Adv}^{\mathsf{A}_{\mathsf{alg}},\mathcal{G}}_{\mathbf{uf\text{-}cma}_{\mathsf{BLS}}} \geq \frac{1}{4} \mathbf{Adv}^{\mathsf{A}_{\mathsf{alg}},\mathcal{G}}_{\mathbf{uf\text{-}cma}_{\mathsf{BLS}}}$. The proof of Lemma 6.2 can be found in Supplementary Material A.4. ∎

Corollary 6.3 $\mathbf{uf\text{-}cma}_{\mathsf{BLS},\mathcal{G}}$ *is* $(t, \frac{t^2}{4p})$*-hard in the generic group model with a random oracle.*

7 Groth's Near-Optimal zk-SNARK

In order to cover notions such as knowledge soundness, which are defined via games for two algorithms, we generalize the notion of algebraic games and reductions between them. We write $\mathbf{G}^{\mathsf{A},\mathsf{X}}_{par}$ to denote that A and X play in $\mathbf{G}_{par}$ and define the advantage $\mathbf{Adv}^{\mathbf{G}}_{par,\mathsf{A},\mathsf{X}} := \Pr[\mathbf{G}^{\mathsf{A},\mathsf{X}}_{par} = 1]$ and the running time $\mathbf{Time}^{\mathbf{G}}_{\mathbf{par},\mathsf{A},\mathsf{X}}$ as before. To capture definitions that require that for every A there exists some X (which has black-box access to A) such that $\mathbf{Adv}^{\mathbf{G}}_{par,\mathsf{A},\mathsf{X}}$ is small, we define algebraic reductions for games $\mathbf{G}_{par}$ of this type as follows.

We write $\mathbf{H}_{par} \overset{(\Delta_\varepsilon, \Delta_t)}{\Longrightarrow}_{\mathsf{alg}} \mathbf{G}_{par}$ if there exist generic algorithms $\mathsf{R}_{\mathsf{gen}}$ and $\mathsf{S}_{\mathsf{gen}}$ such that for all algebraic algorithms $\mathsf{A}_{\mathsf{alg}}$ we have

$$\mathbf{Adv}^{\mathbf{H}}_{par,\mathsf{B}_{\mathsf{alg}}} \geq \frac{1}{\Delta_\varepsilon} \cdot \mathbf{Adv}^{\mathbf{G}}_{par,\mathsf{A}_{\mathsf{alg}},\mathsf{X}_{\mathsf{alg}}}, \quad \mathbf{Time}^{\mathbf{H}}_{par,\mathsf{B}_{\mathsf{alg}}} \leq \Delta_t \cdot \mathbf{Time}^{\mathbf{G}}_{par,\mathsf{A}_{\mathsf{alg}},\mathsf{X}_{\mathsf{alg}}}.$$

with $\mathsf{B}_{\mathsf{alg}}$ defined as $\mathsf{B}_{\mathsf{alg}} := \mathsf{R}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{gen}}$ and $\mathsf{X}_{\mathsf{alg}}$ defined as $\mathsf{X}_{\mathsf{alg}} := \mathsf{S}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{gen}}$.

The q-discrete logarithm assumption. We define a parametrized ("q-type") variant of the DLog assumption via the algebraic security game q-**dlog** in Figure 13. We will show that Groth's [Gro16] scheme, which is the most efficient SNARK system to date, is secure under q-DLog in the algebraic group model.

Non-interactive zero-knowledge arguments of knowledge. Groth [Gro16] considers proof systems for satisfiability of arithmetic circuits, which consist of addition and multiplication gates over a finite field $\mathbb{F}$. As a tool, Gennaro et al. [GGPR13] show how to efficiently convert any arithmetic circuit into a quadratic arithmetic program (QAP) R, which is described by

$\underline{\textbf{knw-snd}}^{\mathsf{A},\mathsf{X_A}}_{\mathsf{SNK},R}$	$\underline{\textbf{k-snd-aff}}^{\mathsf{X},\mathsf{A}}_{\mathsf{NILP},R}$
00 $crs \xleftarrow{\$} \mathsf{Setup}(R)$	03 $\vec{\sigma} \xleftarrow{\$} \mathsf{LinSetup}(R)$
01 $((\phi,\pi);w) \xleftarrow{\$} (\mathsf{A} \,\|\|\, \mathsf{X_A})(R, crs)$	04 $(\phi, P) \xleftarrow{\$} \mathsf{A}(R)$
02 Return $((\phi,w) \notin R$	05 $w \xleftarrow{\$} \mathsf{X}(R,\phi,P)$
$\wedge\ \mathsf{Vfy}(R, crs, \phi, \pi) = 1)$	06 Return $(P \in \mathbb{F}^{\mu\times\kappa} \ \wedge\ (\phi,w) \notin R$
	$\wedge\ \mathsf{LinVfy}(R, \vec{\sigma}, \phi, P\vec{\sigma}) = 1)$

Figure 14: **Left:** Knowledge soundness game **knw-snd** relative to $\mathsf{SNK} = (\mathsf{Setup}, \mathsf{Prv}, \mathsf{Vfy})$, adversary A and extractor $\mathsf{X_A}$. **Right:** Knowledge soundness game **k-snd-aff** relative to $\mathsf{NILP} = (\mathsf{LinSetup}, \mathsf{PrfMtrx}, \mathsf{Test})$, extractor X and *affine* adversary A (right).

$\mathbb{F}$, integers $\ell \le m$ and polynomials $u_i, v_i, w_i \in \mathbb{F}[X]$, for $0 \le i \le m$, and $t \in \mathbb{F}[X]$, where the degrees of u_i, v_i, w_i are less than the degree n of t. (The relation R can also contain additional information *aux*.)

A QAP R defines the following binary relation of statements ϕ and witnesses w, where we set $a_0 := 1$:

$$R = \left\{ (\phi, w) \;\middle|\; \begin{array}{l} \phi = (a_1, \ldots, a_\ell) \in \mathbb{F}^\ell,\ w = (a_{\ell+1}, \ldots, a_m) \in \mathbb{F}^{m-\ell} \\ \big(\sum_{i=0}^m a_i u_i(X)\big) \cdot \big(\sum_{i=0}^m a_i v_i(X)\big) \equiv \sum_{i=0}^m a_i w_i(X) \pmod{t(X)} \end{array} \right\}$$

A non-interactive argument system for a class of relations $\mathcal{R}$ is a 3-tuple $\mathsf{SNK} = (\mathsf{Setup}, \mathsf{Prv}, \mathsf{Vfy})$ of algorithms. Setup on input a relation $R \in \mathcal{R}$ outputs a common reference string *crs*; prover algorithm Prv on input *crs* and a statement/witness pair $(\phi, w) \in R$ returns an argument π; Verification Vfy on input *crs*, ϕ and π returns either 0 (reject) or 1 (accept). We require SNK to be *complete*, i.e., for all *crs* output by Setup, all arguments for true statements produced by Prv are accepted by Vfy.

Knowledge soundness requires that for every adversary A there exists an extractor $\mathsf{X_A}$ that extracts a witness from any valid argument output by A. We write $(y; z) \xleftarrow{\$} (\mathsf{A} \,\|\, \mathsf{X_A})(x)$ when A on input x outputs y and $\mathsf{X_A}$ on the same input (including A's coins) returns z. Knowledge soundness is defined via game $\textbf{knw-snd}^{\mathsf{A},\mathsf{X_A}}_{\mathsf{SNK},R}$ in Figure 14.

Zero-knowledge for SNK requires that arguments do not leak any information besides the truth of the statement. It is formalized by demanding the existence of a simulator which on input a trapdoor (which is an additional output of Setup) and a true statement ϕ returns an argument which is indistinguishable from an argument for ϕ output by Prv (see [Gro16] for a formal definition).

A (preprocessing) *succinct argument of knowledge (SNARK)* is a knowledge-sound non-interactive argument system whose arguments are of size polynomial in the security parameter and can be verified in polynomial time in the security parameter and the length of the statement.

Non-interactive linear proofs of degree 2. NILPs (in Groth's [Gro16] terminology) are an abstraction of many SNARK constructions introduced by Bitansky et al. [BCI+13]. We only consider NILPs of degree 2 here. Such a system NILP is defined by three algorithms as follows. On input a quadratic arithmetic program R, $\mathsf{LinSetup}$ returns $\vec{\sigma} \in \mathbb{F}^\mu$ for some μ. On input R, ϕ and w, algorithm $\mathsf{PrfMtrx}$ generates a matrix $P \in \mathbb{F}^{\kappa\times\mu}$ (where κ is the proof length). And on input R and ϕ, Test returns matrices $T_1, \ldots, T_\eta \in \mathbb{F}^{\mu+\kappa}$. The last two algorithms implicitly define a prover and a verification algorithm as follows:

- $\vec{\pi} \xleftarrow{\$} \mathsf{LinPrv}(R, \vec{\sigma}, \phi, w)$: run $P \xleftarrow{\$} \mathsf{PrfMtrx}(R, \phi, w)$; return $\vec{\pi} := P\vec{\sigma}$.
- $b \xleftarrow{\$} \mathsf{LinVfy}(R, \vec{\sigma}, \phi, \vec{\pi})$: $(T_1, \ldots, T_\eta) \xleftarrow{\$} \mathsf{Test}(R, \phi)$; return 1 iff for all $1 \le k \le \eta$:

$$(\vec{\sigma}^\top \mid \vec{\pi}^\top)\, T_k\, (\vec{\sigma}^\top \mid \vec{\pi}^\top)^\top = 0. \tag{6}$$

$\mathsf{Setup}(R)$	$\mathsf{Prv}(R, \vec{\mathbf{\Sigma}}, \phi, w)$	$\mathsf{Vfy}(R, \vec{\mathbf{\Sigma}}, \phi, \vec{\mathbf{\Pi}})$
00 $g \xleftarrow{\$} \mathbb{G}$	03 $P \xleftarrow{\$} \mathsf{PrfMtrx}(R, \phi, w)$	09 $T_1, \ldots, T_\eta \xleftarrow{\$} \mathsf{Test}(R, \phi)$
01 $\vec{\sigma} \xleftarrow{\$} \mathsf{LinSetup}(R)$	04 Parse $P = (p_{i,j})_{i,j}$	10 Return 1 iff for all $1 \le \ell \le \eta$:
02 Return $\vec{\mathbf{\Sigma}} := \langle \vec{\sigma} \rangle$	05 For $i = 1 \ldots \kappa$:	$0 = \prod_{i=1}^{m} \prod_{j=1}^{m} e(\mathbf{\Sigma}_i, \mathbf{\Sigma}_j)^{t_{\ell,i,j}}$
	06 $\mathbf{\Pi}_i := \prod_{j=1}^{m} \mathbf{\Sigma}_j^{p_{i,j}}$	$\cdot \prod_{i=1}^{m} \prod_{j=1}^{\kappa} e(\mathbf{\Sigma}_i, \mathbf{\Pi}_j)^{t_{\ell,i,m+j}}$
	07 $\pi := (\mathbf{\Pi}_1, \ldots, \mathbf{\Pi}_\kappa)$	$\cdot \prod_{i=1}^{\kappa} \prod_{j=1}^{\kappa} e(\mathbf{\Pi}_i, \mathbf{\Pi}_j)^{t_{\ell,m+i,m+j}}$
	08 Return π	// This evaluates (6) in the exponent
	// Note that $\pi := \langle P\vec{\sigma} \rangle$	

Figure 15: Argument system $(\mathsf{Setup}, \mathsf{Prv}, \mathsf{Vfy})$ from a NILP $(\mathsf{LinSetup}, \mathsf{PrfMtrx}, \mathsf{Test})$.

Let $T_k =: (t_{k,i,j})_{1 \le i,j, \le \mu+\kappa}$; w.l.o.g. we assume that $t_{k,i,j} = 0$ for all k and $i \in \{\mu+1, \ldots, \mu+\kappa\}$ and $j \in \{1, \ldots, \mu\}$.

We require a NILP to satisfy *statistical knowledge soundness against affine prover strategies*, which requires the existence of an (efficient) extractor X that works for all (unbounded) adversaries A. Whenever A returns a proof matrix P which leads to a valid proof $P\vec{\sigma}$ for a *freshly sampled* $\vec{\sigma}$, X can extract a witness from P. The notion is defined via game $\textbf{k-snd-aff}^{\mathsf{X},\mathsf{A}}_{\mathsf{NILP},R}$ in Figure 14.

Non-interactive arguments from NILPs. From a NILP for a quadratic arithmetic program over a finite field $\mathbb{F} = \mathbb{Z}_p$ for some prime p, one can construct an argument system over a bilinear group $\mathcal{G} = (p, \mathbb{G}, g, e)$. We thus consider QAP relations R of the form

$$R = (\mathcal{G}, \mathbb{F} = \mathbb{Z}_p, \ell, \{u_i(X), v_i(X), w_i(X)\}_{i=0}^{m}, t(X)), \tag{7}$$

and define the *degree* of R as the degree of n of $t(X)$.

The construction of $\mathsf{SNK} = (\mathsf{Setup}, \mathsf{Prv}, \mathsf{Vfy})$ from $\mathsf{NILP} = (\mathsf{LinSetup}, \mathsf{PrfMtrx}, \mathsf{Test})$ is given in Figure 15, where we write $\langle \vec{x} \rangle$ for $(g^{x_1}, \ldots, g^{x_{|\vec{x}|}})$. Setup first samples a random group generator g and then embeds the NILP CRS "in the exponent". Using group operations, Prv computes LinPrv in the exponent, and using the pairing, Vfy verifies LinVfy in the exponent.

Groth's near-optimal SNARK for QAPs. Groth [Gro16] obtains his SNARK system by constructing a NILP for QAPs and then applying the conversion in Figure 15. Recall that R, as in (7), defines a language of statements $\phi = (a_1, \ldots, a_\ell) \in \mathbb{F}^\ell$ with witnesses of the form $w = (a_{\ell+1}, \ldots, a_m) \in \mathbb{F}^{m-\ell}$ such that (with $a_0 := 1$):

$$\left(\sum_{i=0}^{m} a_i u_i(X)\right) \cdot \left(\sum_{i=0}^{m} a_i v_i(X)\right) = \sum_{i=0}^{m} a_i w_i(X) + h(X)t(X) \tag{8}$$

for some $h(X) \in \mathbb{F}[X]$ of degree at most $n-2$. Groth's NILP is given in Figure 16.

Theorem 7.1 ([Gro16, Theorem 1]). *The construction in Figure 16 in a NILP with perfect completeness, perfect zero-knowledge and statistical knowledge soundness against affine prover strategies.*

Groth embeds his NILP in *asymmetric* bilinear groups, which yields a more efficient SNARK. He then shows that the scheme is knowledge-sound in the generic group model for *symmetric* bilinear groups (which yields a stronger result, as the adversary is more powerful than in asymmetric groups). Since we aim at strengthening the security statement, we also consider the symmetric-group variant (which is obtained by applying the transformation in Figure 15). We now show how from an algebraic adversary breaking knowledge soundness one can construct an adversary against the q-DLog assumption.

Theorem 7.2 *Let* SNK *denote Groth's [Gro16] SNARK for degree-n QAPs defined over (symmetric) bilinear group $\mathcal{G}$ with $n^2 \leq (p-1)/8$. Then we have* q**-dlog** $\overset{(3,1)}{\Longrightarrow}_{\mathsf{alg}}$ **knw-snd**$_{\mathsf{SNK}}$ *with* $q := 2n-1$.

Let us start with a proof overview. Consider an algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ against knowledge soundness that returns $(\phi, [\vec{\mathbf{\Pi}}]_P)$ on input $(R, \langle\vec{\sigma}\rangle)$. Since $\mathsf{A}_{\mathsf{alg}}$ is algebraic and its group-element inputs are $\langle\vec{\sigma}\rangle$, we have $\vec{\mathbf{\Pi}} = \langle P\vec{\sigma}\rangle$ with $P \in \mathbb{F}^{3\times\mu}$. By the definition of Vfy, the proof $\vec{\mathbf{\Pi}}$ is valid iff $P\vec{\sigma}$ satisfies LinVfy. By Groth's theorem (Theorem 7.1) there exists an extractor X, which on input P such that $P\vec{\sigma}$ satisfies LinVfy extracts a witness (see game **k-snd-aff**$_{\mathsf{NILP},R}$ in Figure 14).

So it seems this extractor X should also work for $\mathsf{A}_{\mathsf{alg}}$ (which returns P as required). However, X is only guaranteed to succeed if $P\vec{\sigma}$ verifies for a *randomly* sampled $\vec{\sigma}$, whereas for $\mathsf{A}_{\mathsf{alg}}$ in **knw-snd**$_{\mathsf{SNK},R}$ it suffices to return P so that $P\vec{\sigma}$ verifies for the specific $\vec{\sigma}$ for which it received $\langle\vec{\sigma}\rangle$. To prove knowledge soundness, it now suffices to show that an adversary can only output P which works for *all* choices of $\vec{\sigma}$ (from which X will then extract a witness).

In the generic group model this follows rather straight-forwardly, since the adversary has no information about the concrete $\vec{\sigma}$. In the AGM however, A is given $\langle\vec{\sigma}\rangle$. Examining the structure of a NILP CRS $\vec{\sigma}$ (Figure 16), we see that its components are defined as multivariate (Laurent) polynomials evaluated at a random point $\vec{x} = (\alpha, \beta, \gamma, \delta, \tau)$.

Now what does it mean for $\mathsf{A}_{\mathsf{alg}}$ to output a valid P? By the definition of LinVfy via Test (cf. Equation (6) with $\vec{\pi} := P\vec{\sigma}$), it means that $\mathsf{A}_{\mathsf{alg}}$ found P such that

$$(\vec{\sigma}^\top \,|\, (P\vec{\sigma}))^\top)\, T\, (\vec{\sigma}^\top \,|\, (P\vec{\sigma})^\top)^\top = 0. \tag{9}$$

If we interpret the components of $\vec{\sigma}$ as polynomials over $X_1, \ldots, X_5$ (corresponding to $\vec{x} = (\alpha, \beta, \gamma, \delta, \tau)$) then the left-hand side of (9) defines a polynomial $Q_P(\vec{X})$.

On the other hand, what does it mean that $P\vec{\sigma}$ verifies only for the specific $\vec{\sigma}$ from $\mathsf{A}_{\mathsf{alg}}$'s input? It means that $Q_P \not\equiv 0$, that is, Q_P is not the zero polynomial, since otherwise (9) would hold for *any* choice of $\vec{x}$, that is $P\vec{\sigma}'$ would verify for any $\vec{\sigma}'$.

$\mathsf{LinSetup}(R)$
00 $\alpha, \beta, \gamma, \delta, \tau \xleftarrow{\$} \mathbb{F}^*$
01 $\vec{\sigma} := \big(1, \alpha, \beta, \gamma, \delta, \{\tau^i\}_{i=0}^{n-1}, \{\frac{1}{\gamma}(\beta u_i(\tau) + \alpha v_i(\tau) + w_i(\tau))\}_{i=0}^{\ell},$
$\{\frac{1}{\delta}(\beta u_i(\tau) + \alpha v_i(\tau) + w_i(\tau))\}_{i=\ell+1}^{m}, \{\frac{1}{\delta}(\tau^i t(\tau))\}_{i=0}^{n-2}\big)$
02 Return $\vec{\sigma}$

$\mathsf{PrfMtrx}(R, \phi, w)$
03 Let $h(X)$ be as in (8)
04 $r, s \xleftarrow{\$} \mathbb{F}$
05 Return $P \in \mathbb{F}^{3\times(m+2n+4)}$ s.t. $P\vec{\sigma} = (A, B, C)$ with
06 $A := \alpha + \sum_{i=0}^{m} a_i u_i(\tau) + r\delta$
07 $B := \beta + \sum_{i=0}^{m} a_i v_i(\tau) + s\delta$
08 $C := \frac{1}{\delta}\big(\sum_{i=\ell+1}^{m} a_i(\beta u_i(\tau) + \alpha v_i(\tau) + w_i(\tau)) + h(\tau)t(\tau)\big) + As + rB - rs\delta$

$\mathsf{Test}(R, \phi)$
09 Return $T \in \mathbb{F}^{(m+2n+7)\times(m+2n+7)}$ corresponding to the test
$A \cdot B = \alpha \cdot \beta + \sum_{i=0}^{\ell} a_i \frac{1}{\gamma}(\beta u_i(\tau) + \alpha v_i(\tau) + w_i(\tau)) \cdot \gamma + C \cdot \delta$

Figure 16: Groth's NILP ($\mathsf{LinSetup}, \mathsf{PrfMtrx}, \mathsf{Test}$).

$\underline{\mathsf{X}_\mathsf{A}(R, \langle\vec{\sigma}\rangle)}$	$\underline{\textbf{knw-snd}}^{\mathsf{A}_\mathsf{alg},\mathsf{X}_\mathsf{A}}_{\mathsf{SNK},R}$
00 $(\phi, [\vec{\mathbf{\Pi}}]_P) \xleftarrow{\$} \mathsf{A}_\mathsf{alg}(R, \langle\vec{\sigma}\rangle)$	03 $\vec{\sigma} \xleftarrow{\$} \mathsf{LinSetup}(R)$
01 $w \xleftarrow{\$} \mathsf{X}(R, \phi, P)$	04 $(\phi, [\vec{\mathbf{\Pi}}]_P) \xleftarrow{\$} \mathsf{A}_\mathsf{alg}(R, \langle\vec{\sigma}\rangle)$
02 Return w.	05 $w \xleftarrow{\$} \mathsf{X}(R, \phi, P)$
	06 Return $((\phi, w) \notin R \wedge \mathsf{LinVfy}(R, \vec{\sigma}, \phi, P\vec{\sigma}) = 1)$

Figure 17: Extractor X_A defined from X and A_alg (left) and knowledge soundness game **knw-snd** for a SNARK built from $\mathsf{NILP} = (\mathsf{LinSetup}, \mathsf{PrfMtrx}, \mathsf{Test})$, algebraic adversary A_alg and X_A (right).

We now bound the probability that A_alg behaves "badly", that is, it returns a proof that only holds with respect to its specific CRS. To do so, we bound the probability that given $\langle\vec{\sigma}\rangle$, A_alg returns a nonzero polynomial Q_P which vanishes at $\vec{x}$, that is, the point that defines $\vec{\sigma}$. By factoring Q_P, we can then extract information about $\vec{x}$, which was only given as group elements $\langle\vec{\sigma}\rangle$. In particular, we embed a q-DLog instance into the CRS $\langle\vec{\sigma}\rangle$, for which we need q to be at least the maximum of the total degrees of the polynomials defining σ, which for Groth's NILP is $2n - 1$. We then factor the polynomial to obtain its roots, which yields the challenge discrete logarithm

Proof of Theorem 7.2. Let R be a QAP of degree n (cf. (7)). Let $\mathsf{NILP} = (\mathsf{LinSetup}, \mathsf{PrfMtrx}, \mathsf{Test})$ denote Groth's NILP (Figure 16). By Theorem 7.1 there exists an extractor X, which on input R, statement $\phi \in L_R$, and $P \in \mathbb{F}^{\mu\times\kappa}$ such that $\mathsf{LinVfy}(R, \vec{\sigma}, \phi, P\vec{\sigma}) = 1$ for $\vec{\sigma} \xleftarrow{\$} \mathsf{LinSetup}(R)$ returns a witness w with probability $\mathbf{Adv}^{\textbf{k-snd-aff}}_{\mathsf{NILP},R,\mathsf{X},\mathsf{F}}$ for any affine F.

Let SNK denote Groth's SNARK obtained from NILP via the transformation in Figure 15 and let A_alg be an algebraic adversary in the game $\textbf{knw-snd}_{\mathsf{SNK},R}$. From X we construct an extractor X_A for A_alg in Figure 17. Note that since A_alg is algebraic, we have $\vec{\mathbf{\Pi}} = \langle P\vec{\sigma}\rangle$. We thus have

$$\mathsf{Vfy}(R, \vec{\mathbf{\Sigma}}, \phi, \vec{\mathbf{\Pi}}) = \mathsf{Vfy}(R, \vec{\mathbf{\Sigma}}, \phi, \langle P\vec{\sigma}\rangle) = \mathsf{LinVfy}(R, \vec{\sigma}, \phi, P\vec{\sigma}), \tag{10}$$

where the last equality follows from the definition of Vfy (Figure 15). Game $\textbf{knw-snd}^{\mathsf{A}_\mathsf{alg},\mathsf{X}_\mathsf{A}}_{\mathsf{SNK},R}$ is written out in Figure 17 and our goal is to upperbound $\mathbf{Adv}^{\textbf{knw-snd}}_{\mathsf{SNK},R,\mathsf{A}_\mathsf{alg},\mathsf{X}_\mathsf{A}}$.

Consider the affine prover A' in Figure 18 and $\textbf{k-snd-aff}^{\mathsf{X},\mathsf{A}'}_{\mathsf{NILP}}$, with the code of A' written out, also in Figure 18.

$\underline{\mathsf{A}'(R)}$	$\underline{\textbf{k-snd-aff}}^{\mathsf{X},\mathsf{A}'}_{\mathsf{NILP},R}$
00 $\vec{\sigma} \xleftarrow{\$} \mathsf{LinSetup}(R)$	03 $\vec{\rho} \xleftarrow{\$} \mathsf{LinSetup}(R)$;
01 $(\phi, [\vec{\mathbf{\Pi}}]_P) \xleftarrow{\$} \mathsf{A}_\mathsf{alg}(R, \langle\vec{\sigma}\rangle)$	04 $\vec{\sigma} \xleftarrow{\$} \mathsf{LinSetup}(R)$
02 Return (ϕ, P).	05 $(\phi, [\vec{\mathbf{\Pi}}]_P) \xleftarrow{\$} \mathsf{A}_\mathsf{alg}(R, \langle\vec{\sigma}\rangle)$
	06 $w \xleftarrow{\$} \mathsf{X}(R, \phi, P)$
	07 If $((\phi, w) \notin R \wedge \mathsf{LinVfy}(R, \vec{\sigma}, \phi, P\vec{\sigma}) = 1$
	$\wedge\ \mathsf{LinVfy}(R, \vec{\rho}, \phi, P\vec{\rho}) = 0)$
	08 Then $\mathbf{bad} := 1$
	09 Return $((\phi, w) \notin R \wedge \mathsf{LinVfy}(R, \vec{\rho}, \phi, P\vec{\rho}) = 1)$

Figure 18: Affine prover A' defined from A_alg (left) and game **k-snd-aff** for NILP, extractor X and A' (right).

Comparing the right-hand sides of Figures 17 and 18, we see that **knw-snd** returns 1 whereas **k-snd-aff** returns 0 if LinVfy returns 0 for $P\vec{\rho}$ w.r.t. $\vec{\rho}$, but it returns 1 for $P\vec{\sigma}$ w.r.t. $\vec{\sigma}$. Let **bad**

denote the event when this happens; formally defined as a flag in game **k-snd-aff** in Figure 18. By definition, we have

$$\mathbf{Adv}^{\textbf{knw-snd}}_{\mathsf{SNK},R,\mathsf{A}_{\mathsf{alg}},\mathsf{X}_\mathsf{A}} \leq \mathbf{Adv}^{\textbf{k-snd-aff}}_{\mathsf{NILP},R,\mathsf{X},\mathsf{A}'} + \Pr\left[\mathbf{bad} = 1\right]. \tag{11}$$

In order to simplify our analysis, we first make a syntactical change to NILP by multiplying out all denominators, that is, we let LinSetup (cf. Figure 16) return

$$\vec{\sigma} := \Big(\delta\gamma, \alpha\delta\gamma, \beta\delta\gamma, \delta\gamma^2, \delta^2\gamma, \{\delta\gamma\tau^i\}_{i=0}^{n-1}, \{\delta(\beta u_i(\tau) + \alpha v_i(\tau) + w_i(\tau))\}_{i=0}^{\ell}, \\ \{\gamma(\beta u_i(\tau) + \alpha v_i(\tau) + w_i(\tau))\}_{i=\ell+1}^{m}, \{\gamma\tau^i t(\tau)\}_{i=0}^{n-2}\Big). \tag{12}$$

Note that this does not affect the distribution of the SNARK CRS as running the modified LinSetup amounts to the same as choosing $g' \xleftarrow{\$} \mathbb{G}$ and running the original setup with $g := (g')^{\delta\gamma}$, which again is a uniformly random generator.

Observe that the components of LinSetup defined in (12) can be described via multivariate polynomials $S_i(\vec{x})$ of total degree at most $2n-1$ with $\vec{x} := (\alpha, \beta, \gamma, \delta, \tau)$, and LinSetup can be defined as picking a random point $\vec{x} \xleftarrow{\$} (\mathbb{F}^*)^5$ and returning the evaluations $\sigma_i := S_i(\vec{x})$ of these polynomials.

Let T be as defined by Test in Figure 16. By (6) we have

$$\mathsf{LinVfy}(R, \vec{\sigma}, \phi, P\vec{\sigma}) = 1 \iff \vec{\sigma}^\top\left((Id \,|\, P^\top) \cdot T \cdot (Id \,|\, P^\top)^\top\right)\vec{\sigma} = 0.$$

Let $\vec{S}$ be the vector of polynomials defined by LinSetup. For a matrix P define the following multivariate polynomial

$$Q_P(\vec{X}) := (\vec{S}(\vec{X}))^\top\left((Id \,|\, P^\top) \cdot T \cdot (Id \,|\, P^\top)^\top\right)\vec{S}(\vec{X}) \tag{13}$$

of degree at most $(2n-1)^2$. Then for any $\vec{x} \in (\mathbb{F}^*)^5$ and $\vec{\sigma} := \vec{S}(\vec{x})$ we have

$$\mathsf{LinVfy}(R, \vec{\sigma}, \phi, P\vec{\sigma})) = 1 \iff Q_P(\vec{x}) = 0. \tag{14}$$

Groth [Gro16] proves Theorem 7.1 by arguing that an affine prover without knowledge of $\vec{\sigma}$ can only succeed in the game $\mathbf{k\text{-}snd\text{-}aff}_{\mathsf{NILP},R}$ by making LinVfy return 1 on *every* $\vec{\sigma}$, or stated differently using (14), by returning P with $Q_P \equiv 0$. He then shows that from such P one can efficiently extract a witness.

The adversary's probability of succeeding despite $Q_P \not\equiv 0$ is bounded via the Schwartz-Zippel lemma: the total degree of Q_P is at most $d = (2n-1)^2$ (using the modified $\vec{\sigma}$ from (12)). The probability that $Q_P(\vec{x}) = 0$ for a random $\vec{x} \xleftarrow{\$} (F^*)^5$ is thus bounded by $\frac{d}{p-1}$. This yields

$$\mathbf{Adv}^{\textbf{k-snd-aff}}_{\mathsf{NILP},R,\mathsf{X},\mathsf{A}'} \leq \tfrac{(2n-1)^2}{p-1}. \tag{15}$$

In order to bound $\mathbf{Adv}^{\textbf{knw-snd}}_{\mathsf{SNK},R,\mathsf{A}_{\mathsf{alg}},\mathsf{X}_\mathsf{A}}$ in (11) we will construct an adversary $\mathsf{B}_{\mathsf{alg}}$ such that

$$\Pr\left[\mathbf{bad} = 1\right] \leq (1 - \tfrac{(2n-1)^2}{p-1}) \cdot \mathbf{Adv}^{q\textbf{-dlog}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}} \quad \text{with} \quad q = 2n-1. \tag{16}$$

For **bad** to be set to 1, $\mathsf{A}_{\mathsf{alg}}$'s output P must be such that $Q_P \not\equiv 0$: otherwise, LinVfy returns 1 for *any* $\vec{x}$ and in particular $\mathsf{LinVfy}(R, \vec{\rho}, \phi, P\vec{\rho}) = 1$.

bad $= 1$ implies thus that $\mathsf{A}_{\mathsf{alg}}$ on input $\langle\vec{\sigma}\rangle = \langle\vec{S}(\vec{x})\rangle$ returns P such that

$$Q_P \not\equiv 0 \quad \text{and} \quad Q_P(\vec{x}) = 0. \tag{17}$$

We now use such $\mathsf{A}_{\mathsf{alg}}$ to construct an adversary $\mathsf{B}_{\mathsf{alg}}$ against q-DLog with $q := 2n-1$.

Adversary $\mathsf{B}_{\mathsf{alg}}(\langle z\rangle, \langle z^2\rangle, ..., \langle z^q\rangle)$*:* On input a q-DLog instance, $\mathsf{B}_{\mathsf{alg}}$ simulates $\mathbf{k\text{-}snd\text{-}aff}^{\mathsf{X},\mathsf{A}'}_{\mathsf{NILP},R}$ for $\mathsf{A}_{\mathsf{alg}}$. It first picks a random value $\vec{y} \leftarrow (\mathbb{F}^*)^5$, (implicitly) sets $x_i := y_i z$, that is,

$$\alpha := y_1 z \qquad \beta := y_2 z \qquad \gamma := y_3 z \qquad \delta := y_4 z \qquad \tau := y_5 z$$

and generates a CRS $\langle\vec{\sigma}\rangle := \langle\vec{S}(\vec{x})\rangle = \langle\vec{S}(\alpha,\beta,\gamma,\delta,\tau)\rangle$ as defined in (12). Since the total degree of the polynomials S_i defining $\vec{\sigma}$ is bounded by $2n-1$ (the degree of the last component of $\vec{\sigma}$), $\mathsf{B}_{\mathsf{alg}}$ can compute $\vec{\sigma}$ from its q-DLog instance.

Next, $\mathsf{B}_{\mathsf{alg}}$ runs $(\phi, [\vec{\mathbf{\Pi}}]_P) \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}(R, \langle\vec{\sigma}\rangle)$ and from P computes the multivariate polynomial $Q_P(\vec{X})$ as defined in (13). If $Q_P \equiv 0$ or $Q_P(\vec{x}) \neq 0$ (by (10) and (14) the latter is equivalent to $\mathsf{Vfy}(R, \vec{\mathbf{\Sigma}}, \phi, \vec{\mathbf{\Pi}}) = 0$), then $\mathsf{B}_{\mathsf{alg}}$ aborts. $(*)$

Otherwise $\mathsf{B}_{\mathsf{alg}}$ defines the univariate polynomial

$$Q'_P(X) := Q_P(y_1 X, \ldots, y_5 X).$$

If $Q'_P \equiv 0$ then $\mathsf{B}_{\mathsf{alg}}$ aborts. $(**)$

Otherwise $\mathsf{B}_{\mathsf{alg}}$ factors Q'_P to obtain its roots (of which by (13) there are at most $(2n-1)^2$), checks them against its DLog instance to determine whether z is among them, and if so, returns z.

First note that $\mathsf{B}_{\mathsf{alg}}$ perfectly simulates $\mathbf{k\text{-}snd\text{-}aff}^{\mathsf{X},\mathsf{A}'}_{\mathsf{NILP},R}$ for $\mathsf{A}_{\mathsf{alg}}$. Let us analyze the probability that $\mathsf{B}_{\mathsf{alg}}$ finds the target z provided that $\mathbf{bad} = 1$. In this case $\mathsf{B}_{\mathsf{alg}}$ will not abort at $(*)$.

Since $Q'_P(z) = Q_P(y_1 z, \ldots, y_5 z) = Q_P(\vec{x})$, by (17) we have $Q'_P(z) = 0$. Thus if $Q'_P \not\equiv 0$ then $\mathsf{B}_{\mathsf{alg}}$ finds z by factoring Q'_P. It remains to argue that $Q'_P \not\equiv 0$. By (17) we have $Q_P \not\equiv 0$. By the Schwartz-Zippel lemma, the probability that for a random $\vec{y} \xleftarrow{\$} (\mathbb{F}^*)^5$, we have $Q_P(\vec{y}) = 0$ is bounded by $\frac{d}{p-1}$ where d is the total degree of Q_P, which is bounded by $(2n-1)^2$. Since $Q_P(\vec{y}) = Q'_P(1)$, we have $Q'_P \not\equiv 0$ with probability at least $1 - \frac{(2n-1)^2}{p-1}$. Since the choice of $\vec{y}$ is perfectly hidden from the adversary's view this shows

$$\mathbf{Adv}^{q\text{-}\mathbf{dlog}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}} \geq (1 - \tfrac{(2n-1)^2}{p-1}) \cdot \Pr\left[\mathbf{bad} = 1\right] \geq \tfrac{1}{2} \cdot \Pr\left[\mathbf{bad} = 1\right],$$

where the last inequality comes from $n^2 \leq (p-1)/8$. Putting this together with (15), we have shown that

$$\mathbf{Adv}^{\mathbf{knw\text{-}snd}}_{\mathsf{SNK},R,\mathsf{A}_{\mathsf{alg}},\mathsf{X}_{\mathsf{A}}} \leq \tfrac{q^2}{p-1} + 2 \cdot \mathbf{Adv}^{q\text{-}\mathbf{dlog}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}}.$$

Following the generic bound for Boneh and Boyen's SDH assumption [BB08], we may assume that $\mathbf{Adv}^{q\text{-}\mathbf{dlog}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}} \geq \frac{q^2}{p-1}$. The above equation thus implies

$$\mathbf{Adv}^{\mathbf{knw\text{-}snd}}_{\mathsf{SNK},R,\mathsf{A}_{\mathsf{alg}},\mathsf{X}_{\mathsf{A}}} \leq 3 \cdot \mathbf{Adv}^{q\text{-}\mathbf{dlog}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}},$$

which concludes the proof. ∎

Corollary 7.3 *It is* $(\frac{3t^2q+3q^3}{p}, t)$*-hard to break knowledge soundness of Groth's SNARK [Gro16] in the generic group model.*

The corollary follows from the generic $(\frac{t^2q+q^3}{p}, t)$-hardness of q**-dlog**, which is derived analogously to the bound for Boneh and Boyen's SDH assumption [BB08].

We remark that the above result is not specific to Groth's SNARK; it applies to any SNARK built from a NILP whose setup evaluates multivariate polynomials on a random position. The maximal total degree of these polynomials determines the parameter q in the q-DLog instance.

Acknowledgments

We thank Dan Brown for valuable comments and Pooya Farshim for discussions on polynomials. We also thank Helger Lipmaa for sharing with us his independent security proof for Groth's SNARK. The first author is supported by the French ANR EfTrEC project (ANR-16-CE39-0002). The second author was supported in part by ERC Project ERCC (FP7/615074) and by DFG SPP 1736 Big Data. The third author was supported by ERC Project ERCC (FP7/615074).

References

[ABR01] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In David Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 143–158. Springer, Heidelberg, April 2001. (Cited on page 3, 10.)

[ABS16] Miguel Ambrona, Gilles Barthe, and Benedikt Schmidt. Automated unbounded analysis of cryptographic constructions in the generic group model. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 822–851. Springer, Heidelberg, May 2016. (Cited on page 6.)

[ACdM05] Giuseppe Ateniese, Jan Camenisch, and Breno de Medeiros. Untraceable RFID tags via insubvertible encryption. In Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels, editors, *ACM CCS 05*, pages 92–101. ACM Press, November 2005. (Cited on page 3.)

[ACHdM05] Giuseppe Ateniese, Jan Camenisch, Susan Hohenberger, and Breno de Medeiros. Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385, 2005. http://eprint.iacr.org/2005/385. (Cited on page 3.)

[AGO11] Masayuki Abe, Jens Groth, and Miyako Ohkubo. Separating short structure-preserving signatures from non-interactive assumptions. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 628–646. Springer, Heidelberg, December 2011. (Cited on page 2.)

[AM09] Divesh Aggarwal and Ueli Maurer. Breaking RSA generically is equivalent to factoring. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 36–53. Springer, Heidelberg, April 2009. (Cited on page 2, 4.)

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008. (Cited on page 21.)

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 315–333. Springer, Heidelberg, March 2013. (Cited on page 16.)

[BCL04] E. Bangerter, J. Camenisch, and A. Lysyanskaya. A cryptographic framework for the controlled release of certified data. In *Security Protocols Workshop*, pages 20–24, 2004. (Cited on page 3.)

[BCPR16] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. *SIAM Journal on Computing*, 2016. (Cited on page 5.)

[BCS05] M. Backes, J. Camenisch, and D. Sommer. Anonymous yet accountable access control. In *WPES*, pages 40–46, 2005. (Cited on page 3.)

[BDZ03] Feng Bao, Robert H. Deng, and Huafei Zhu. Variations of Diffie-Hellman problem. In Sihan Qing, Dieter Gollmann, and Jianying Zhou, editors, *ICICS 03*, volume 2836 of *LNCS*, pages 301–312. Springer, Heidelberg, October 2003. (Cited on page 9.)

[BFF+14] Gilles Barthe, Edvard Fagerholm, Dario Fiore, John C. Mitchell, Andre Scedrov, and Benedikt Schmidt. Automated analysis of cryptographic assumptions in generic group models. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 95–112. Springer, Heidelberg, August 2014. (Cited on page 4, 6.)

[BG04] Daniel R. L. Brown and Robert P. Gallant. The static diffie-hellman problem. Cryptology ePrint Archive, Report 2004/306, 2004. http://eprint.iacr.org/2004/306. (Cited on page 4.)

[BL96] Dan Boneh and Richard J. Lipton. Algorithms for black-box fields and their application to cryptography (extended abstract). In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 283–297. Springer, Heidelberg, August 1996. (Cited on page 1.)

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004. (Cited on page 3, 13.)

[BMV08] Emmanuel Bresson, Jean Monnerat, and Damien Vergnaud. Separation results on the "one-more" computational problems. In Tal Malkin, editor, *CT-RSA 2008*, volume 4964 of *LNCS*, pages 71–87. Springer, Heidelberg, April 2008. (Cited on page 2.)

[Boy08] Xavier Boyen. The uber-assumption family (invited talk). In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 39–56. Springer, Heidelberg, September 2008. (Cited on page 5, 6.)

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, November 1993. (Cited on page 8.)

[BR04] Mihir Bellare and Phillip Rogaway. Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Report 2004/331, 2004. http://eprint.iacr.org/2004/331. (Cited on page 6.)

[BV98] Dan Boneh and Ramarathnam Venkatesan. Breaking RSA may not be equivalent to factoring. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 59–71. Springer, Heidelberg, May / June 1998. (Cited on page 2, 7.)

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998. (Cited on page 5.)

[Che06] Jung Hee Cheon. Security analysis of the strong Diffie-Hellman problem. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 1–11. Springer, Heidelberg, May / June 2006. (Cited on page 4, 13.)

[CHK+06] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: Efficient periodic n-times anonymous authentication. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06*, pages 201–210. ACM Press, October / November 2006. (Cited on page 3.)

[CHL05] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 302–321. Springer, Heidelberg, May 2005. (Cited on page 3.)

[CHP07] Jan Camenisch, Susan Hohenberger, and Michael Østergaard Pedersen. Batch verification of short signatures. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 246–263. Springer, Heidelberg, May 2007. (Cited on page 3.)

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, Heidelberg, August 2004. (Cited on page 3, 11.)

[CM14] Melissa Chase and Sarah Meiklejohn. Déjà Q: Using dual systems to revisit q-type assumptions. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 622–639. Springer, Heidelberg, May 2014. (Cited on page 5.)

[Cor02] Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 272–287. Springer, Heidelberg, April / May 2002. (Cited on page 2, 3.)

[Dam92] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 445–456. Springer, Heidelberg, August 1992. (Cited on page 3, 5.)

[Den02] Alexander W. Dent. Adapting the weaknesses of the random oracle model to the generic group model. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 100–109. Springer, Heidelberg, December 2002. (Cited on page 5.)

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. (Cited on page 3, 9.)

[GBL08] Sanjam Garg, Raghav Bhaskar, and Satyanarayana V. Lokam. Improved bounds on security reductions for discrete log based signatures. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 93–107. Springer, Heidelberg, August 2008. (Cited on page 2.)

[GG17] Jens Groth and Essam Ghadafi. Towards a classification of non-interactive computational assumptions in cyclic groups. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017*, volume 10625 of *LNCS*, pages 66–96, Hong Kong, December 2017. Springer. (Cited on page 5.)

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013. (Cited on page 15.)

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016. (Cited on page 3, 15, 16, 17, 18, 20, 21.)

[HP78] M. E. Hellman and S. C. Pohlig. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. *IEEE Transactions on Information Theory*, 24(1):106–110, 1978. (Cited on page 2.)

[JR10] Tibor Jager and Andy Rupp. The semi-generic group model and applications to pairing-based cryptography. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 539–556. Springer, Heidelberg, December 2010. (Cited on page 2, 5.)

[JR15] Antoine Joux and Antoine Rojat. Security ranking among assumptions within the *Uber Assumption* framework. In Yvo Desmedt, editor, *ISC 2013*, volume 7807 of *LNCS*, pages 391–406. Springer, Heidelberg, November 2015. (Cited on page 5.)

[JS09] Tibor Jager and Jörg Schwenk. On the analysis of cryptographic assumptions in the generic ring model. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 399–416. Springer, Heidelberg, December 2009. (Cited on page 4.)

[Kil01] Eike Kiltz. A tool box of cryptographic functions related to the Diffie-Hellman function. In C. Pandu Rangan and Cunsheng Ding, editors, *INDOCRYPT 2001*, volume 2247 of *LNCS*, pages 339–350. Springer, Heidelberg, December 2001. (Cited on page 5.)

[KK12] Saqib A. Kakvi and Eike Kiltz. Optimal security proofs for full domain hash, revisited. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 537–553. Springer, Heidelberg, April 2012. (Cited on page 3.)

[KMP16] Eike Kiltz, Daniel Masny, and Jiaxin Pan. Optimal security proofs for signatures from identification schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 33–61. Springer, Heidelberg, August 2016. (Cited on page 2.)

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, Heidelberg, April 2008. (Cited on page 6.)

[LR06] Gregor Leander and Andy Rupp. On the equivalence of RSA and factoring regarding generic ring algorithms. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 241–251. Springer, Heidelberg, December 2006. (Cited on page 2.)

[LRSW99] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard M. Heys and Carlisle M. Adams, editors, *SAC 1999*, volume 1758 of *LNCS*, pages 184–199. Springer, Heidelberg, August 1999. (Cited on page 3, 4, 11.)

[Mau05] Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Heidelberg, December 2005. (Cited on page 1, 5, 6, 8.)

[MRV16] Paz Morillo, Carla Ràfols, and Jorge Luis Villar. The kernel matrix Diffie-Hellman assumption. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 729–758. Springer, Heidelberg, December 2016. (Cited on page 5.)

[MW98] Ueli M. Maurer and Stefan Wolf. Lower bounds on generic algorithms in groups. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 72–84. Springer, Heidelberg, May / June 1998. (Cited on page 1, 6.)

[MW99] Ueli Maurer and Stefan Wolf. The relationship between breaking the diffie-hellman protocol and computing discrete logarithms. *SIAM Journal on Computing*, 28(5):1689–1721, 1999. (Cited on page 9.)

[Nec94] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994. (Cited on page 1.)

[Pol78] J. M. Pollard. Monte Carlo methods for index computation mod p. *Mathematics of Computation*, 32:918–924, 1978. (Cited on page 2.)

[PV05] Pascal Paillier and Damien Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In Bimal K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2005. (Cited on page 2, 7.)

[Riv04] Ronald L. Rivest. On the notion of pseudo-free groups. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 505–521. Springer, Heidelberg, February 2004. (Cited on page 2.)

[RLB+08] Andy Rupp, Gregor Leander, Endre Bangerter, Alexander W. Dent, and Ahmad-Reza Sadeghi. Sufficient conditions for intractability over black-box groups: Generic lower bounds for generalized DL and DH problems. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 489–505. Springer, Heidelberg, December 2008. (Cited on page 6.)

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997. (Cited on page 1, 5, 6, 8.)

[Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. http://eprint.iacr.org/2004/332. (Cited on page 27.)

[SS01] Ahmad-Reza Sadeghi and Michael Steiner. Assumptions related to discrete logarithms: Why subtleties make a real difference. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 244–261. Springer, Heidelberg, May 2001. (Cited on page 5.)

A Supplementary Material

A.1 Proof of Theorem 3.3

Proof. We prove that

$$\mathbf{Adv}^{\mathbf{cdh}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}} \geq \mathbf{Adv}^{\mathbf{sdh}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}} - q \cdot \mathbf{Adv}^{\mathbf{lc\text{-}dh}}_{\mathcal{G},\mathsf{C}_{\mathsf{alg}}}. \tag{18}$$

Applying Theorem 3.1 yields the theorem. We now prove (18) via a sequence of games.

$\mathbf{G_0}$: Let $\mathsf{A}_{\mathsf{alg}}$ be an algebraic adversary playing in $\mathbf{G_0} := \mathbf{sdh}^{\mathsf{A}_{\mathsf{alg}}}_{\mathcal{G}}$. As $\mathsf{A}_{\mathsf{alg}}$ is an algebraic adversary, it returns a vector $\vec{a}$ along with $\mathbf{Z}$ at the end of the game such that $\mathbf{Z} = g^{a_1}\mathbf{X}^{a_2}\mathbf{Y}^{a_3}$. Furthermore, for any query asked to $\mathsf{O}(\cdot,\cdot)$, it includes vectors $\vec{b}, \vec{c}$ such that $\mathbf{Y}' = g^{b_1}\mathbf{X}^{b_2}\mathbf{Y}^{b_3}$ and $\mathbf{Z}' = g^{c_1}\mathbf{X}^{c_2}\mathbf{Y}^{c_3}$. Game $\mathbf{G_0}$ is depicted in Figure 19.

$\underline{\mathbf{G_0}}$, $\boxed{\underline{\mathbf{G_1}}}$	$\underline{\mathsf{O}}([\mathbf{Y}']_{\vec{b}}, [\mathbf{Z}']_{\vec{c}})$:
00 $x, y \xleftarrow{\$} \mathbb{Z}_p$	04 $\boxed{\text{If } b_2 \neq 0 \vee b_3 \neq 0}$
01 $(\mathbf{X}, \mathbf{Y}) := (g^x, g^y)$	05 $\boxed{\text{Return } 0}$
02 $[\mathbf{Z}]_{\vec{a}} \xleftarrow{\$} \mathsf{A}^{\mathsf{O}(\cdot,\cdot)}_{\mathsf{alg}}(\mathbf{X}, \mathbf{Y})$	06 Return $(\mathbf{Z}' = (\mathbf{Y}')^x)$
03 Return $\mathbf{Z} = g^{xy}$	

Figure 19: Games $\mathbf{G_0}$ and $\mathbf{G_1}$. The boxed statements are only executed in $\mathbf{G_1}$.

$\mathbf{G_1}$: For game $\mathbf{G_1}$ we alter the way that the oracle $\mathsf{O}(\cdot,\cdot)$ answers queries. Namely, if $b_2 \neq 0 \vee b_3 \neq 0$, it always returns 0. Game $\mathbf{G_1}$ is depicted in Figure 19. The check performed by the oracle in $\mathbf{G_1}$ amounts to checking whether $\mathbf{Z}' = \mathbf{X}^{b_1}$, since if $b_2 = b_3 = 0$ then $\mathbf{Y}' = g^{b_1}$. Using this property of $\mathbf{G_1}$, we show an adversary $\mathsf{B}_{\mathsf{alg}}$ against $\mathbf{cdh}_{\mathcal{G}}$ such that $\mathbf{Adv}^{\mathbf{cdh}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}} = \Pr[\mathbf{G_1} = 1]$. $\mathsf{B}_{\mathsf{alg}}$ is depicted in Figure 20.

$\underline{\mathsf{B}_{\mathsf{alg}}}(\mathbf{X} = g^x, \mathbf{Y} = g^y)$	$\underline{\mathsf{O}}([\mathbf{Y}']_{\vec{b}}, [\mathbf{Z}']_{\vec{c}})$:
00 $[\mathbf{Z}]_{\vec{a}} \xleftarrow{\$} \mathsf{A}^{\mathsf{O}(\cdot,\cdot)}_{\mathsf{alg}}(\mathbf{X}, \mathbf{Y})$	02 If $b_2 \neq 0 \vee b_3 \neq 0$
01 Return $\mathbf{Z}$	03 Return 0
	04 Return $\mathbf{Z}' = \mathbf{X}^{b_1}$

Figure 20: Behavior of adversary $\mathsf{B}_{\mathsf{alg}}$.

We now show the existence of adversary $\mathsf{C}_{\mathsf{alg}}$ such that

$$\left|\mathbf{Adv}^{\mathbf{G_0}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}} - \mathbf{Adv}^{\mathbf{G_1}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}\right| \leq q \cdot \mathbf{Adv}^{\mathbf{lcdh}}_{\mathcal{G},\mathsf{C}_{\mathsf{alg}}}.$$

Let F denote the event that $\mathbf{Z}' = (\mathbf{Y}')^x \wedge (b_2 \neq 0 \vee b_3 \neq 0)$ in at least one call to the oracle. Clearly, as long as F does not occur, the games behave identically. By the difference lemma [Sho04], we obtain

$$|\Pr[\mathbf{G_0} = 1] - \Pr[\mathbf{G_1} = 1]| \leq \Pr[F].$$

We show the existence of $\mathsf{E}_{\mathsf{alg}}$ such that

$$\Pr[F] \leq q \cdot \mathbf{Adv}^{\mathbf{lc\text{-}dh}}_{\mathcal{G},\mathsf{C}_{\mathsf{alg}}}.$$

$\underline{\mathsf{C}_{\mathsf{alg}}}(\mathbf{X}=g^x, \mathbf{Y}=g^y)$	$\underline{\mathsf{O}}([\mathbf{Y}']_{\vec{b}}, [\mathbf{Z}']_{\vec{c}})$:
00 $Q := \emptyset$	05 If $b_2 \neq 0 \vee b_3 \neq 0$
01 $[\mathbf{Z}]_{\vec{a}} \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}^{\mathsf{O}(\cdot,\cdot)}(\mathbf{X}, \mathbf{Y})$	06 If $b_2 \neq 0 \wedge b_3 \neq 0$
02 $\tilde{\mathbf{Z}} \xleftarrow{\$} Q$	07 $Q := Q \cup \{\mathbf{Z}'\}$
03 Obtain b_1, b_2, b_3 as described	08 Return 0
04 Return $(\tilde{\mathbf{Z}}X^{-b_1}, b_2, b_3, 0)$	09 Return $(\mathbf{Z}' = \mathbf{X}^{b_1})$

Figure 21: Behavior of adversary $\mathsf{C}_{\mathsf{alg}}$.

$\mathsf{C}_{\mathsf{alg}}$ is depicted in Figure 21.

We now analyze $\mathsf{C}_{\mathsf{alg}}$. Clearly, $\mathsf{E}_{\mathsf{alg}}$ runs in the same time as $\mathsf{A}_{\mathsf{alg}}$. Once $\mathsf{A}_{\mathsf{alg}}$ halts, $\mathsf{C}_{\mathsf{alg}}$ picks at random $\tilde{\mathbf{Z}}$ that was input by $\mathsf{A}_{\mathsf{alg}}$ as one of at most q queries to $\mathsf{O}(\cdot,\cdot)$ along with $\tilde{\mathbf{Y}}$ and $\vec{b}, \vec{c}$ such that $b_2 \neq 0 \vee b_3 \neq 0$, and

$$\begin{aligned} \tilde{\mathbf{Y}} &= g^{b_1}\mathbf{X}^{b_2}\mathbf{Y}^{b_3}, \\ \tilde{\mathbf{Z}} &= g^{c_1}\mathbf{X}^{c_2}\mathbf{Y}^{c_3}. \end{aligned}$$

Clearly, if $(\tilde{\mathbf{Y}})^x = \tilde{\mathbf{Z}}$ then $(\tilde{\mathbf{Z}}\mathbf{X}^{-b_1}, b_2, b_3, 0)$ yields a winning solution for $\mathbf{lc\text{-}dh}_{\mathcal{G}}^{\mathsf{C}_{\mathsf{alg}}}$ as

$$\tilde{\mathbf{Z}}\mathbf{X}^{-b_1} = (\tilde{\mathbf{Y}})^x\mathbf{X}^{-b_1} = g^{b_2x^2+b_3xy}.$$

As $\mathsf{C}_{\mathsf{alg}}$ picks $\tilde{\mathbf{Z}}$ at random from at most q elements in Q, it picks a correct solution with probability at least $\mathbf{Adv}^{\mathbf{lc\text{-}dh}}_{\mathcal{G},\mathsf{C}_{\mathsf{alg}}} \geq \frac{\Pr[F]}{q}$. This yields $\Pr[F] \leq q \cdot \mathbf{Adv}^{\mathbf{lc\text{-}dh}}_{\mathcal{G},\mathsf{C}_{\mathsf{alg}}}$). Thus, we now have

$$\begin{aligned} \mathbf{Adv}^{\mathbf{cdh}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}} &= \Pr[\mathbf{G_1} = 1] \\ &\geq \Pr[\mathbf{G_0} = 1] - |\mathbf{Adv}^{\mathbf{G_0}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}} - \mathbf{Adv}^{\mathbf{G_1}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}| \\ &\geq \mathbf{Adv}^{\mathbf{sdh}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}} - q \cdot \mathbf{Adv}^{\mathbf{lc\text{-}dh}}_{\mathcal{G},\mathsf{C}_{\mathsf{alg}}}. \end{aligned}$$

It is straight forward to see that all the steps performed in the above simulations are generic. This proves (18). ∎

A.2 Proof of Theorem 4.1

Proof. We prove the statement via a sequence of games.

$\mathbf{G_0}$: Let $\mathsf{A}_{\mathsf{alg}}$ be an algebraic adversary playing in $\mathbf{G_0} := \mathbf{lrsw}_{\mathcal{G}}^{\mathsf{A}_{\mathsf{alg}}}$. Game $\mathbf{G_0}$ is depicted in Figures 22. As $\mathsf{A}_{\mathsf{alg}}$ is an algebraic adversary, at the end of the game, it outputs a winning tuple $(m^*, \mathbf{A}^*, \mathbf{B}^*, \mathbf{C}^*)$ along with vectors $\vec{a}, \vec{b}, \vec{c}$ that provide the representation of $\mathbf{A}^*, \mathbf{B}^*, \mathbf{C}^*$ relative to $g, \mathbf{X}, \mathbf{Y}$ and the answers $\mathbf{A}_1, ..., \mathbf{A}_q, \mathbf{B}_1, ..., \mathbf{B}_q, \mathbf{C}_1, ..., \mathbf{C}_q$ from previous oracle queries, where $\mathbf{A}_i = g^{r_i}$, $\mathbf{B}_i = g^{r_i y}$, and $\mathbf{C}_i = g^{r_i(yxm_i+x)}$.

Concretely, the representations of $\mathbf{A}^*$, $\mathbf{B}^*$, and $\mathbf{C}^*$ are as follows:

$$\mathbf{A}^* = \prod_{i=1}^{q} \mathbf{A}_i^{a_i} g^{a_{q+1}} \prod_{i=q+2}^{2q+1} \mathbf{B}_{i-q-1}^{a_i} \prod_{i=2q+2}^{3q+1} \mathbf{C}_{i-2q-1}^{a_i} \mathbf{X}^{a_{3q+2}} \mathbf{Y}^{a_{3q+3}}, \tag{19}$$

$$\mathbf{B}^* = \prod_{i=1}^{q} \mathbf{A}_i^{b_i} g^{b_{q+1}} \prod_{i=q+2}^{2q+1} \mathbf{B}_{i-q-1}^{b_i} \prod_{i=2q+2}^{3q+1} \mathbf{C}_{i-2q-1}^{b_i} \mathbf{X}^{b_{3q+2}} \mathbf{Y}^{b_{3q+3}}, \tag{20}$$

$$\mathbf{C}^* = \prod_{i=1}^{q} \mathbf{C}_i^{c_i} \mathbf{X}^{c_{q+1}} \prod_{i=q+2}^{2q+1} \mathbf{A}_{i-q-1}^{c_i} \prod_{i=2q+2}^{3q+1} \mathbf{B}_{i-2q-1}^{c_i} g^{c_{3q+2}} \mathbf{Y}^{c_{3q+3}}. \tag{21}$$

$\underline{\mathbf{G_0}}$, $\boxed{\underline{\mathbf{G_1}}}$	$\underline{\mathsf{O}}(m_j)$ //For query j
00 $\boxed{\ell^*, k^*, i^* \xleftarrow{\$} \{1, ..., q\}}$	08 $r_j \xleftarrow{\$} \mathbb{Z}_p$;
01 $\boxed{\text{If } k^* = \ell^* \vee k^* = i^* \vee \ell^* = i^*}$	09 $\mathbf{A}_j := g^{r_j}$
02 $\boxed{\text{Abort}}$	10 $\mathbf{B}_j := g^{yr_j}$
03 $Q := \emptyset$	11 $\mathbf{C}_j := g^{r_j m_j xy + r_j x}$
04 $x, y \xleftarrow{\$} \mathbb{Z}_p$	12 $Q := Q \cup \{m_j\}$
05 $\mathbf{X} := g^x, \mathbf{Y} := g^y$	13 Return $(\mathbf{A}_j, \mathbf{B}_j, \mathbf{C}_j)$
06 $(m^*, [\mathbf{A}^*]_{\vec{a}}, [\mathbf{B}^*]_{\vec{b}}, [\mathbf{C}^*]_{\vec{c}}) \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}^{\mathsf{O}(\cdot)}(\mathbf{X}, \mathbf{Y})$	
07 Return $m^* \notin Q \wedge m^* \neq 0$ $\wedge\ \mathbf{A}^* \neq 1 \wedge\ \mathbf{B}^* = (\mathbf{A}^*)^y \wedge\ \mathbf{C}^* = (\mathbf{A}^*)^{xm^*y+x}$	

Figure 22: Games $\mathbf{G_0}$ and $\mathbf{G_1}$ with algebraic adversary $\mathsf{A}_{\mathsf{alg}}$. The boxed statements are only executed in $\mathbf{G_1}$.

We assume that $\mathsf{A}_{\mathsf{alg}}$ never queries the oracle on the same message m_i more than once. (Multiple queries can be simulated by rerandomization.)

$\mathbf{G_1}$: In $\mathbf{G_1}$ we consider a slightly altered game that is defined as follows. Before the first query is asked, the challenger in $\mathbf{G_1}$ also chooses values $k^*, \ell^*, i^* \xleftarrow{\$} \{1, ..., q\}$. If $k^* = \ell^* \vee k^* = i^* \vee \ell^* = i^*$, it aborts the game. $\mathbf{G_1}$ is depicted in Figure 22. Clearly, $\left(1 - \frac{3}{q}\right) \mathbf{Adv}^{\mathbf{G_0}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}} = \mathbf{Adv}^{\mathbf{G_1}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$. By defining $s_1, s_2, t_1, t_2, u_1, u_2, v_1, v_2 \in \mathbb{Z}_p$ as

$$
\begin{aligned}
s_1 &:= a_{3q+2} + \sum_{i=2q+3}^{3q+2} a_i r_{i-2q-2}, & s_2 &:= c_{q+1} + \sum_{i=1}^{q} c_i r_i, \\
t_1 &:= \sum_{i=2q+3}^{3q+2} a_i m_i r_{i-2q-2}, & t_2 &:= \sum_{i=1}^{q} c_i m_i r_i, \\
u_1 &:= a_{3q+3} + \sum_{i=q+2}^{2q+1} a_i r_{i-q-1}, & u_2 &:= c_{3q+3} + \sum_{i=2q+2}^{3q+1} c_i r_{i-q-1}, \\
v_1 &:= g^{a_{q+1}} + \sum_{i=1}^{q} a_i r_i, & v_2 &:= g^{c_{3q+2}} + \sum_{i=q+2}^{2q+1} c_i r_{i-q-1},
\end{aligned}
$$

equations (19) and (21) can be further simplified to

$$
\begin{aligned}
\mathbf{A}^* &= g^{s_1 x + t_1 xy + u_1 y + v_1}, \\
\mathbf{C}^* &= g^{s_2 x + t_2 xy + u_2 y + v_2}.
\end{aligned}
$$

We also define the parameters $\Delta, \Delta', \Delta''$ as

$$
\begin{aligned}
\Delta &:= m^* t_1 y^2 + t_1 y + s_1 m^* y + s_1, && (22) \\
\Delta' &:= u_1 m^* y^2 + m^* y v_1 + u_1 y - t_2 y - s_2 + v_1, && (23) \\
\Delta'' &:= u_2 y + v_2, && (24)
\end{aligned}
$$

and the boolean variable F^* as

$$
F^* = 1 \Leftrightarrow s_1 \equiv_p t_1 \equiv_p u_1 \equiv_p u_2 \equiv_p v_2 \equiv_p 0. \tag{25}
$$

We prove the following lemma that allows us to rewrite $\mathbf{A}^*$ and $\mathbf{C}^*$ in a more convenient form.

Lemma A.1 *If $F^* = 1$, then*

$$\mathbf{A}^* = \prod_{i=1}^{q} \mathbf{A}_i^{\varepsilon_i}, \quad \mathbf{C}^* = \prod_{i=1}^{q} \mathbf{C}_i^{\delta_i}$$

holds for

$$\delta_i := \begin{cases} c_i & i \notin \{k^*, \ell^*\} \\ c_{\ell^*} - \frac{r_{k^*} m_{k^*} c_{q+1}}{(r_{\ell^*} m_{\ell^*})(r_{k^*} - r_{k^*} \frac{m_{k^*}}{m_{\ell^*}})} & i = \ell^* \\ c_{k^*} + \frac{c_{q+1}}{r_{k^*} - r_{k^*} \frac{m_{k^*}}{m_{\ell^*}}} & i = k^* \end{cases}$$

and

$$\varepsilon_i := \begin{cases} a_i & i \neq k^* \\ a_{k^*} + \frac{a_{q+1}}{r_{k^*}} & i = k^* \end{cases}.$$

Using Lemma A.1, we can now formulate the following conditions whenever $\mathbf{G_1}$ does not abort. To further simplify the notation, we define the following Boolean variables:

$$\begin{aligned} G^* = 1 \quad &\Leftrightarrow \quad \Delta \not\equiv_p 0 \vee \Delta' \not\equiv_p 0 \vee \Delta'' \not\equiv_p 0 \\ H^* := 1 \quad &\Leftrightarrow \quad \forall j \notin \{\ell^*, k^*\} : r_j \varepsilon_j (1 + m^* y) - r_j \delta_j (1 + y m_j) \equiv_p 0. \end{aligned}$$

Note that H^* is only well defined (by Lemma A.1) if $F^* = 1$.

- Condition F_1 : This condition holds iff G^*.
- Condition F_2 : This condition holds iff $(\neg G^* \wedge \neg F^*) \vee (F^* \wedge H^*)$.
- Condition F_3 : This condition holds iff $F^* \wedge \neg H^*$.

It is easy to see that $F_1 \vee F_2 \vee F_3 = 1$. We will now describe the behavior of adversaries $\mathsf{C_{alg}}, \mathsf{D_{alg}}, \mathsf{E_{alg}}$ playing in the discrete logarithm game. Each of these adversaries simulates $\mathbf{G_1}$ to $\mathsf{A_{alg}}$ in a different way. Concretely, we prove the following Lemma.

Lemma A.2 *There exist $\mathsf{C_{alg}}, \mathsf{D_{alg}}, \mathsf{E_{alg}}$ playing in the discrete logarithm game such that:*

$$\Pr\left[\mathbf{dlog}^{\mathsf{C_{alg}}} = 1\right] = \Pr[\mathbf{G_1} = 1 \mid F_1] \tag{26}$$

$$\Pr\left[\mathbf{dlog}^{\mathsf{D_{alg}}} = 1\right] \geq (1 - \frac{2}{q}) \Pr[\mathbf{G_1} = 1 \mid F_2] \tag{27}$$

$$\Pr\left[\mathbf{dlog}^{\mathsf{E_{alg}}} = 1\right] \geq \frac{1}{q} \Pr[\mathbf{G_1} = 1 \mid F_3]. \tag{28}$$

of Lemma A.1. We first note that $\delta_{l^*}, \delta_{k^*}$ are well defined, because $\ell^* \neq k^*$ and thus $m_{\ell^*} \neq m_{k^*}$. Otherwise $\mathbf{G_1}$ aborts and there is nothing to prove (since nothing is returned by $\mathsf{A_{alg}}$ in this case). Observe that since $F^* = 1 \Leftrightarrow s_1 \equiv_p t_1 \equiv_p u_1 \equiv_p u_2 \equiv_p v_2 \equiv_p 0$, we have

$$\mathbf{A}^* = g^{a_{q+1}} \prod_{i=1}^{q} \mathbf{A}_i^{a_i} \quad \text{and} \quad \mathbf{C}^* = \mathbf{X}^{c_{q+1}} \prod_{i=1}^{q} \mathbf{C}_i^{c_i}.$$

Now, the choices of $\delta_1, ..., \delta_q, \epsilon_1, ..., \epsilon_q$ satisfy

$$\mathbf{A}^* = \prod_{i=1}^{q} \mathbf{A}_i^{\epsilon_i} \quad \text{and} \quad \mathbf{C}^* = \prod_{i=1}^{q} \mathbf{C}_i^{\delta_i}.$$

To see this, first observe that $\mathbf{X}$ can be written as

$$\begin{aligned}\mathbf{X} &= \left(\mathbf{X}^{r_{k^*}(1-m_{k^*}/m_{\ell^*})}\right)^{\frac{1}{r_{k^*}(1-m_{k^*}/m_{\ell^*})}} \\ &= \left(g^{r_{k^*}(x+yxm_{k^*})} g^{-r_{\ell^*}(x+yxm_{\ell^*})(r_{k^*}m_{k^*})/(r_{\ell^*}m_{\ell^*})}\right)^{\frac{1}{r_{k^*}(1-m_{k^*}/m_{\ell^*})}} \\ &= \left(\mathbf{C}_{k^*}\mathbf{C}_{\ell^*}^{-(r_{k^*}m_{k^*})/(r_{\ell^*}m_{\ell^*})}\right)^{\frac{1}{r_{k^*}(1-m_{k^*}/m_{\ell^*})}}.\end{aligned}$$

Because of this, setting

$$\begin{aligned}\delta_{k^*} &:= c_{k^*} + \frac{c_{q+1}}{r_{k^*} - r_{k^*}\frac{m_{k^*}}{m_{\ell^*}}}, \\ \delta_{\ell^*} &:= c_{\ell^*} - \frac{r_{k^*}m_{k^*}c_{q+1}}{(r_{\ell^*}m_{\ell^*})(r_{k^*} - r_{k^*}\frac{m_{k^*}}{m_{\ell^*}})},\end{aligned}$$

and $\delta_i := c_i$ for $i \notin \{\ell^*, k^*\}$ we obtain

$$\mathbf{X}^{c_{q+1}}\mathbf{C}_{k^*}^{c_{k^*}}\mathbf{C}_{\ell^*}^{c_{\ell^*}} = \mathbf{C}_{k^*}^{c_{k^*}}\mathbf{C}_{\ell^*}^{c_{\ell^*}}\left(\mathbf{C}_{k^*}\mathbf{C}_{\ell^*}^{-(r_{k^*}m_{k^*})/(r_{\ell^*}m_{\ell^*})}\right)^{\frac{c_{q+1}}{r_{k^*}(1-m_{k^*}/m_{\ell^*})}} = \mathbf{C}_{k^*}^{\delta_{k^*}}\mathbf{C}_{\ell^*}^{\delta_{\ell^*}}.$$

This means that

$$\mathbf{X}^{c_{q+1}}\prod_i \mathbf{C}_i^{c_i} = (\mathbf{X}^{c_{q+1}}\mathbf{C}_{k^*}^{c_{k^*}}\mathbf{C}_{\ell^*}^{c_{\ell^*}})\prod_{i\neq k^*,\ell^*}\mathbf{C}_i^{c_i} = \mathbf{C}_{k^*}^{\delta_{k^*}}\mathbf{C}_{\ell^*}^{\delta_{\ell^*}}\prod_{i\neq k^*,\ell^*}\mathbf{C}_i^{\delta_i} = \prod_i \mathbf{C}_i^{\delta_i}.$$

Also observe that $\mathbf{A}_{k^*}^{\epsilon_{k^*}} = \mathbf{A}_{k^*}^{a_{k^*}} g^{a_{q+1}}$ and thus

$$\mathbf{A}^* = g^{a_{q+1}}\prod_i \mathbf{A}_i^{a_i} = \prod_i \mathbf{A}_i^{\epsilon_i}.$$

■

Using Lemma A.2 and the fact that $F_1 \vee F_2 \vee F_3 = 1$, it is now straightforward to construct an adversary $\mathsf{B}_{\mathsf{alg}}$ such that

$$\Pr\left[\mathbf{dlog}^{\mathsf{B}_{\mathsf{alg}}} = 1\right] \geq \frac{1}{3q}\Pr[\mathbf{G_1} = 1]$$

by letting $\mathsf{B}_{\mathsf{alg}}$ emulate one of the adversaries $\mathsf{C}_{\mathsf{alg}}, \mathsf{D}_{\mathsf{alg}}, \mathsf{E}_{\mathsf{alg}}$ (chosen uniformly at random).

of Lemma A.2. Let $\mathbf{Z} = g^z$ denote the discrete logarithm instance. $\mathsf{C}_{\mathsf{alg}}, \mathsf{D}_{\mathsf{alg}}, \mathsf{E}_{\mathsf{alg}}$ simulate $\mathbf{G_1}$ to $\mathsf{A}_{\mathsf{alg}}$. They begin by sampling $k^*, \ell^*, i^* \xleftarrow{\$} \{1, ..., q\}$. If $k^* = \ell^* \vee k^* = i^* \vee \ell^* = i^*$, they abort the simulation. Thus, assume throughout the proof that $k^* \neq \ell^*, k^* \neq i^*, \ell^* \neq i^*$

Adversary $\mathsf{C}_{\mathsf{alg}}$. Adversary $\mathsf{C}_{\mathsf{alg}}$ samples $\alpha \xleftarrow{\$} \mathbb{Z}_p$ and computes $(\mathbf{X}, \mathbf{Y}) = (\mathbf{Z}, g^\alpha)$. This implicitly sets $x = z$ and $y = \alpha$. Recall that

$$F_1 = 1 \Leftrightarrow \Delta \not\equiv_p 0 \vee \Delta' \not\equiv_p 0 \vee \Delta'' \not\equiv_p 0.$$

We now analyze $\mathsf{C}_{\mathsf{alg}}$. Suppose $\mathsf{A}_{\mathsf{alg}}$ wins $\mathbf{G_1}$ given that $F_1 = 1$. Then $\mathbf{C}^* = (\mathbf{A}^*)^{x+m^*xy}$ which is equivalent to

$$x^2\Delta + x\Delta' - \Delta'' \equiv_p 0 \tag{29}$$

where $\Delta, \Delta', \Delta''$ are defined in (22)-(24). Quadratic equation (29) in indeterminate x has exactly two (possibly equal) solutions, say x_1 and x_2, that can be computed efficiently by $\mathsf{C}_{\mathsf{alg}}$. One of them has to be equal to $z = x$, which one can be tested by comparing g^{x_i} to $\mathbf{Z}$. This proves equation (26).

$\underline{\mathsf{C}_{\mathsf{alg}}}(\mathbf{Z} = g^z)$	$\underline{\mathsf{O}}(m_j)$: //For query j
00 $Q := \emptyset$	05 $r_j \xleftarrow{\$} \mathbb{Z}_p$;
01 $\alpha \xleftarrow{\$} \mathbb{Z}_p$	06 $\mathbf{A}_j := g^{r_j}$
02 $(m^*, [\mathbf{A}^*]_{\vec{a}}, [\mathbf{B}^*]_{\vec{b}}, [\mathbf{C}^*]_{\vec{c}}) \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}^{\mathsf{O}(\cdot)}(\mathbf{Z}, g^\alpha)$	07 $\mathbf{B}_j := g^{\alpha r_j}$
03 Solve for $x : x^2\Delta + x\Delta' - \Delta'' \equiv_p 0$	08 $\mathbf{C}_j := \mathbf{Z}^{r_j m_j \alpha + r_j}$
04 Return x	09 $Q := Q \cup \{m_j\}$
	10 Return $(\mathbf{A}_j, \mathbf{B}_j, \mathbf{C}_j)$

Figure 23: Behavior of adversary $\mathsf{C}_{\mathsf{alg}}$.

$\mathsf{D}_{\mathsf{alg}}(\mathbf{Z} = g^z)$	$\mathsf{O}(m_j)$: //For query j
00 $Q := \emptyset$	05 $r_j \xleftarrow{\$} \mathbb{Z}_p$;
01 $\alpha \xleftarrow{\$} \mathbb{Z}_p$	06 $\mathbf{A}_j := g^{r_j}$
02 $(m^*, [\mathbf{A}^*]_{\vec{a}}, [\mathbf{B}^*]_{\vec{b}}, [\mathbf{C}^*]_{\vec{c}}) \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}^{\mathsf{O}(\cdot)}(g^\alpha, \mathbf{Z})$	07 $\mathbf{B}_j := \mathbf{Z}^{r_j}$
03 Compute y as described below	08 $\mathbf{C}_j := \mathbf{Z}^{r_j m_j \alpha} g^{\alpha r_j}$
04 Return y	09 $Q := Q \cup \{m_j\}$
	10 Return $(\mathbf{A}_j, \mathbf{B}_j, \mathbf{C}_j)$

Figure 24: Behaviour of $\mathsf{D}_{\mathsf{alg}}$.

Adversary $\mathsf{D}_{\mathsf{alg}}$: Adversary $\mathsf{D}_{\mathsf{alg}}$ does the following. It samples $\alpha \xleftarrow{\$} \mathbb{Z}_p$ and computes $(\mathbf{X}, \mathbf{Y}) = (g^\alpha, \mathbf{Z})$. This implicitly sets $x = \alpha$ and $y = z$. Recall that $F_2 = 1$ iff

$$\begin{aligned}&\neg F^* \wedge \neg(\Delta \not\equiv_p 0 \vee \Delta' \not\equiv_p 0 \vee \Delta'' \not\equiv_p 0) \vee \\ &F^* \wedge \forall j \notin \{\ell^*, k^*\} : r_j \varepsilon_j (1 + m^* y) - r_j \delta_j (1 + y m_j) \equiv_p 0,\end{aligned}$$

where F^* is defined in (25).

We analyze and describe $\mathsf{D}_{\mathsf{alg}}$. Suppose that $\mathsf{A}_{\mathsf{alg}}$ wins $\mathbf{G_1}$ given that $F_2 = 1$. As before, we have

$$\mathbf{C}^* = (\mathbf{A}^*)^{x + m^* x y} \Leftrightarrow x^2\Delta + x\Delta' - \Delta'' \equiv_p 0.$$

If $\Delta \equiv_p \Delta' \equiv_p \Delta'' \equiv_p 0 \wedge \neg F^*$ then $\mathsf{D}_{\mathsf{alg}}$ can efficiently solve one of the equations

$$\begin{aligned}\Delta &\equiv_p 0, \\ \Delta' &\equiv_p 0, \\ \Delta'' &\equiv_p 0.\end{aligned}$$

in indeterminate $y = z$. This can be seen as follows.

- If $s_1 \not\equiv_p 0 \vee t_1 \not\equiv_p 0$, it can solve the quadratic equation

  $$\Delta \equiv_p m^* t_1 y^2 + t_1 y + s_1 m^* y + s_1 \equiv_p 0,$$

  because $m^* \not\equiv_p 0$ by assumption.

- If $u_1 \not\equiv_p 0$, it can solve the quadratic equation

  $$\Delta' \equiv_p u_1 m^* y^2 + m^* y v_1 - t_2 y + u_1 y + v_1 - s_2 \equiv_p 0,$$

  where again we use the fact that $m^* \not\equiv_p 0$.

- If $v_2 \not\equiv_p 0$, then since
$$\Delta'' \equiv_p v_2 + u_2 y \equiv_p 0,$$
also $u_2 \not\equiv_p 0$ and so $\mathsf{D}_{\mathsf{alg}}$ can solve for y the equation
$$v_2 + u_2 y \equiv_p 0$$
whenever $v_2 \not\equiv_p 0 \vee u_2 \not\equiv_p 0$.

Given two possible solutions y_1, y_2 for a quadratic equation, $\mathsf{D}_{\mathsf{alg}}$ can determine the correct one by comparing g^{y_i} to $\mathbf{Z}$.

If $F^* = 1$, Lemma A.1 guarantees that $\mathsf{D}_{\mathsf{alg}}$ can efficiently compute parameters $\delta_1, ..., \delta_q, \varepsilon_1, ..., \varepsilon_q$ such that $\mathbf{A}^* = \prod_i \mathbf{A}_i^{\varepsilon_i}, \mathbf{C}^* = \prod_i \mathbf{C}_i^{\delta_i}$.We distinguish two cases.

- Case 1: $\exists j \notin \{\ell^*, k^*\} : \varepsilon_j \not\equiv_p 0 \vee \delta_j \not\equiv_p 0$. Without loss of generality, assume that $\varepsilon_j \not\equiv_p 0$. Since
$$\forall j \notin \{\ell^*, k^*\} : r_j \varepsilon_j (1 + m^* y) - r_j \delta_j (1 + y m_j) \equiv_p 0, \tag{30}$$
$\mathsf{D}_{\mathsf{alg}}$ solves the equation
$$\frac{\delta_j}{\varepsilon_j} - 1 \equiv_p y \left(m^* - m_j \frac{\delta_j}{\varepsilon_j} \right)$$
for y, which is obtained from rearranging terms in (30). This equation has a unique solution for y, and its coefficient can not become zero; this would imply that $m^* \equiv_p m_j$, a contradiction.

- Case 2: $\forall j \notin \{\ell^*, k^*\} : \varepsilon_j \equiv_p \delta_j \equiv_p 0$. This means that
$$\mathbf{A}^* = \mathbf{A}_{\ell^*}^{\varepsilon_{\ell^*}} \mathbf{A}_{k^*}^{\varepsilon_{k^*}} = g^{a_{q+1}} \mathbf{A}_{k^*}^{a_{k^*}} \mathbf{A}_{\ell^*}^{a_{\ell^*}},$$
$$\mathbf{C}^* = \mathbf{C}_{k^*}^{\delta_{k^*}} \mathbf{C}_{\ell^*}^{\delta_{\ell^*}} = \mathbf{X}^{c_{q+1}} \mathbf{C}_{k^*}^{c_{k^*}} \mathbf{C}_{\ell^*}^{c_{\ell^*}}.$$
If $a_{\ell^*} \equiv_p a_{k^*} \equiv_p c_{k^*} \equiv_p c_{\ell^*} \equiv_p 0$, then $\mathbf{A}^* = g^{a_{q+1}}, \mathbf{C}^* = \mathbf{X}^{c_{q+1}}$ and therefore
$$c_{q+1} - a_{q+1} \equiv_p y m^* a_{q+1}.$$
Again, this equation has a unique solution for y and its coefficient can not become zero, because $a_{q+1} \not\equiv_p 0$ (recall that $\mathbf{A}^* \not\equiv_p 1$) and $m^* \not\equiv_p 0$.

Finally, we note that with probability at most $\frac{2}{q-1}$, $\mathsf{A}_{\mathsf{alg}}$ succeeds in setting
$$(a_{\ell^*} \not\equiv_p 0 \vee a_{k^*} \not\equiv_p 0 \vee c_{k^*} \not\equiv_p 0 \vee c_{\ell^*} \not\equiv_p 0) \wedge (\forall j \notin \{\ell^*, k^*\} : \varepsilon_j \equiv_p a_j \equiv_p c_j \equiv_p \delta_j \equiv_p 0).$$
This argument is true, because the indices ℓ^*, k^* are information theoretically hidden from $\mathsf{A}'_{\mathsf{alg}}s$ view and so it guesses either of them with probability at most $\frac{2}{q-1}$. All in all, $\mathsf{D}_{\mathsf{alg}}$ succeeds in computing y with probability at least $1 - \frac{2}{q-1}$. This proves equation (27).

$\underline{\mathsf{E}_{\mathsf{alg}}}(\mathbf{Z} = g^z)$	$\underline{\mathsf{O}}(m_j)$: //For query j
00 $Q := \emptyset$	05 $b := (j = i^*)$
01 $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p$	06 $r'_j \xleftarrow{\$} \mathbb{Z}_p$;
02 $(m^*, [\mathbf{A}^*]_{\vec{a}}, [\mathbf{B}^*]_{\vec{b}}, [\mathbf{C}^*]_{\vec{c}}) \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}^{\mathsf{O}(\cdot)}(g^\alpha, g^\beta)$	07 $\mathbf{A}_j := g^{z^b r'_j}$
03 Compute z as described below	08 $\mathbf{B}_j := g^{z^b \beta r'_j}$
04 Return z	09 $\mathbf{C}_j := g^{z^b r'_j m_j \beta \alpha} g^{z^b \alpha r'_j}$
	10 $Q := Q \cup \{m_j\}$
	11 Return $(\mathbf{A}_j, \mathbf{B}_j, \mathbf{C}_j)$

Figure 25: Behaviour of $\mathsf{E}_{\mathsf{alg}}$.

Adversary $\mathsf{E}_{\mathsf{alg}}$: To simulate $\mathbf{G_1}$ to $\mathsf{A}_{\mathsf{alg}}$, the adversary $\mathsf{E}_{\mathsf{alg}}$ does the following. It samples $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p$ and computes $(\mathbf{X}, \mathbf{Y}) = (g^\alpha, g^\beta)$. This implicitly sets $x = \alpha$ and $y = \beta$. It embeds z into the answer to the i^*th oracle query as shown in Figure 25. We now analyze $\mathsf{E}_{\mathsf{alg}}$. If $F_3 = 1$, then

$$F^* \wedge \exists j \notin \{\ell^*, k^*\} : r_j \varepsilon_j (1 + m^* y) - r_j \delta_j (1 + y m_j) \not\equiv_p 0.$$

Lemma A.1 guarantees that $\mathsf{E}_{\mathsf{alg}}$ can efficiently compute the parameters

$$\delta_1, ..., \delta_q, \varepsilon_1, ..., \varepsilon_q$$

such that $\mathbf{A}^* = \prod_i \mathbf{A}_i^{\varepsilon_i}, \mathbf{C}^* = \prod_i \mathbf{C}_i^{\delta_i}$. By assumption

$$\exists j \notin \{\ell^*, k^*\} : r_j \varepsilon_j (1 + m^* y) - r_j \delta_j (1 + y m_j) \not\equiv_p 0.$$

With probability $\frac{1}{q}, j = i^*$, because i^* is information theoretically hidden from $\mathsf{A}_{\mathsf{alg}}$ and thus independent of its computation. This yields the equation

$$\begin{aligned} (\prod_i g^{r_i \varepsilon_i})^{x + m^* xy} &= (\prod_i \mathbf{A}_i^{\varepsilon_i})^{x + m^* xy} = (\mathbf{A}^*)^{(x + m^* xy)} \\ &= \mathbf{C}^* = \prod_i \mathbf{C}_i^{\delta_i} = \prod_i g^{\delta_i r_i (x + m_i yx)}, \end{aligned}$$

which is equivalent to

$$(\sum_i r_i \varepsilon_i)(x + m^* xy) - \sum_i r_i \delta_i (x + m_i yx) \equiv_p 0.$$

Rearranging terms yields

$$z[r'_{i^*} \varepsilon_{i^*} (1 + m^* y) - r'_{i^*} \delta_{i^*} (1 + m_{i^*} y)] \equiv_p \sum_{i \neq i^*} r_i \delta_i (1 + y m_i) - (\sum_{i \neq i^*} r_i \varepsilon_i)(1 + m^* y).$$

■

By assumption, the coefficient of z in this expression is not zero. Therefore, $\mathsf{E}_{\mathsf{alg}}$ can efficiently solve the modular equation to obtain z. Putting things together, we obtain for the adversary $\mathsf{B}_{\mathsf{alg}}$ emulating one of $\mathsf{C}_{\mathsf{alg}}, \mathsf{D}_{\mathsf{alg}}, \mathsf{E}_{\mathsf{alg}}$ the following bound on the advantage $\mathbf{Adv}^{\mathbf{dlog}}_{\mathsf{B}_{\mathsf{alg}}, \mathcal{G}}$:

$$\mathbf{Adv}^{\mathbf{dlog}}_{\mathsf{B}_{\mathsf{alg}}, \mathcal{G}} \geq \frac{1}{3q} \mathbf{Adv}^{\mathbf{G_1}}_{\mathsf{A}_{\mathsf{alg}}, \mathcal{G}} = \frac{q - 3}{3q^2} \mathbf{Adv}^{\mathbf{G_0}}_{\mathsf{A}_{\mathsf{alg}}, \mathcal{G}} \geq \frac{1}{6q} \mathbf{Adv}^{\mathbf{G_0}}_{\mathsf{A}_{\mathsf{alg}}, \mathcal{G}},$$

where the last inequality holds for $q \geq 6$. ■

A.3 Proof of Theroem 5.2

Proof. First note that given an adversary $\mathsf{A}_{\mathsf{alg}}$ against $\mathbf{q\text{-}ddh}_{\mathcal{G}}$ one can easily construct an adversary $\mathsf{B}_{\mathsf{alg}}$ against $\mathbf{ind\text{-}cca1}_{\mathsf{EG},\mathcal{G}}$. $\mathsf{B}_{\mathsf{alg}}$ first calls $\mathsf{Dec}(\cdot)$ to compute the elements $(g^x, ..., g^{x^q})$. When it is presented with a challenge $(\mathbf{K}^*, \mathbf{C}^*)$, it calls $\mathsf{A}_{\mathsf{alg}}$ on input $(g^x, ..., g^{x^q}, \mathbf{C}^*, \mathbf{K}^*)$ and then outputs $\mathsf{A}_{\mathsf{alg}}$'s output bit b'. Clearly, $(g^x, ..., g^{x^q}, \mathbf{C}^*, \mathbf{K}^*)$ is correctly distributed and therefore

$$\mathbf{Adv}^{\mathbf{q\text{-}ddh}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}} = \mathbf{Adv}^{\mathbf{ind\text{-}cca1}}_{\mathsf{EG},\mathcal{G},\mathsf{B}_{\mathsf{alg}}}, \quad \mathbf{Time}^{\mathbf{q\text{-}ddh}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}} = \mathbf{Time}^{\mathbf{ind\text{-}cca1}}_{\mathsf{EG},\mathcal{G},\mathsf{B}_{\mathsf{alg}}}.$$

For the converse, let $\mathsf{A}_{\mathsf{alg}}$ be an algebraic adversary playing in one of the games $\mathbf{ind\text{-}cca1}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{EG},\mathcal{G},0}$, $\mathbf{ind\text{-}cca1}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{EG},\mathcal{G},1}$. We construct an adversary $\mathsf{B}_{\mathsf{alg}}$ against **q-ddh** that interpolates between $\mathbf{ind\text{-}cca1}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{EG},\mathcal{G},0}$ and $\mathbf{ind\text{-}cca1}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{EG},\mathcal{G},1}$ by simulating one of these games to $\mathsf{A}_{\mathsf{alg}}$. $\mathsf{B}_{\mathsf{alg}}$ is depicted in Figure 26.

$\underline{\mathsf{B}_{\mathsf{alg}}}(g, (\mathbf{X}_i)_{i=1}^q, \mathbf{R}, \mathbf{Z})$	$\underline{\mathsf{Dec}}([\mathbf{C}]_{\vec{c}})$	$\underline{\mathsf{Enc}}()$ *//One time*
00 $\mathbf{X} := g^x$	*//Let this be the i-th query*	06 $\mathbf{C}^* := \mathbf{R}$
01 $b' \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}^{\mathsf{Dec}(\cdot),\mathsf{Enc}}(\mathbf{X})$	03 Compute $\vec{a}$ s.t. $\mathbf{C} = \prod_{j=1}^{i} g^{a_j x^j}$	07 $\mathbf{K}^* := \mathbf{Z}$
02 Return b'	04 $\mathbf{K} := \mathbf{C}^x = \prod_{j=1}^{i} \mathbf{X}_{j+1}^{a_j}$	08 Return $(\mathbf{K}^*, \mathbf{C}^*)$
	05 Return $\mathbf{K}$	

Figure 26: Adversary $\mathsf{B}_{\mathsf{alg}}$ against **q-ddh**.

Let $(g, \mathbf{X}_1 = g^x, \mathbf{X}_2 = g^{x^2}, \ldots, \mathbf{X}_q = g^{x^q}, \mathbf{R} = g^r, \mathbf{Z} = g^{xr+zb})$ be the problem instance given to $\mathsf{B}_{\mathsf{alg}}$ in $\mathbf{q\text{-}ddh}^{\mathsf{B}_{\mathsf{alg}}}_{\mathcal{G},b}$. As $\mathsf{A}_{\mathsf{alg}}$ is algebraic, along with its i-th query $\mathbf{C}$ to $\mathsf{Dec}(\cdot)$ it sends a vector $\vec{c}$ such that $\mathbf{C} = \prod_i \mathbf{L}_i^{c_i}$ where $\vec{\mathbf{L}}$ consists of group elements $g, \mathbf{X}, \mathbf{K}_1, ..., \mathbf{K}_{i-1}$. Here, $\mathbf{K}_1, ..., \mathbf{K}_{i-1}$ denote the answers to the first $i-1$ queries asked to $\mathsf{Dec}(\cdot)$.

Observe that given $\vec{c}$, $\mathsf{B}_{\mathsf{alg}}$ can express $\mathbf{C}$ as $\mathbf{C} = \prod_{i \geq j \geq 0} g^{a_j x^j}$, for some known constants a_j. As $\mathsf{A}_{\mathsf{alg}}$ asks at most $q-1$ such queries, $\mathsf{B}_{\mathsf{alg}}$ can answer them using the group elements $(g, g^x, g^{x^2}, ..., g^{x^q})$ from its instance. When $\mathsf{A}_{\mathsf{alg}}$ queries $\mathsf{Enc}()$, $\mathsf{B}_{\mathsf{alg}}$ returns (g^{xr+zb}, g^r). When $\mathsf{A}_{\mathsf{alg}}$ halts with output b', $\mathsf{B}_{\mathsf{alg}}$ returns b'. Clearly, $\mathsf{B}_{\mathsf{alg}}$ perfectly simulates either $\mathbf{ind\text{-}cca1}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{EG},\mathcal{G},0}$ or $\mathbf{ind\text{-}cca1}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{EG},\mathcal{G},1}$ to $\mathsf{A}_{\mathsf{alg}}$. Finally, $\mathbf{ind\text{-}cca1}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{EG},\mathcal{G},b}$ returns 1 if and only if $\mathbf{q\text{-}ddh}^{\mathsf{B}_{\mathsf{alg}}}_{\mathcal{G},b}$ returns 1. Therefore,

$$\mathbf{Adv}^{\mathbf{q\text{-}ddh}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}} = \mathbf{Adv}^{\mathbf{ind\text{-}cca1}}_{\mathsf{EG},\mathcal{G},\mathsf{A}_{\mathsf{alg}}}, \quad \mathbf{Time}^{\mathbf{q\text{-}ddh}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}} = \mathbf{Time}^{\mathbf{ind\text{-}cca1}}_{\mathsf{EG},\mathcal{G},\mathsf{A}_{\mathsf{alg}}}. \qquad \blacksquare$$

A.4 Proof of Lemma 6.2

Proof. Let $\mathbf{Z}$ denote the discrete logarithm instance. $\mathsf{C}_{\mathsf{alg}}$ and $\mathsf{D}_{\mathsf{alg}}$ simulate $\mathbf{G}$ to $\mathsf{A}_{\mathsf{alg}}$.

Adversary $\mathsf{C}_{\mathsf{alg}}(\mathbf{Z} = g^x)$: Adversary $\mathsf{C}_{\mathsf{alg}}$ is depicted in Figure 27 and works as follows. It sets $\mathbf{X} := \mathbf{Z}$, which implicitly sets $x := z$. To answer a query $\mathsf{H}(m_i)$, $\mathsf{C}_{\mathsf{alg}}$ first checks whether $H(m_i) = \perp$. If so, it samples $r_i \xleftarrow{\$} \mathbb{Z}_i$ and sets $H(m_i) := g^{r_i}$. It then returns $H(m_i)$. To answer a query $\mathsf{O}(m_i)$, it first checks whether $\mathsf{H}(m_i)$ has previously been queried. If not, it first calls $\mathsf{H}(m_i)$ which defines $H(m_i)$ (and thus r_i). It then computes and returns $\mathbf{\Sigma}_i := \mathbf{X}^{r_i}$. It is easy to see that $\mathsf{C}_{\mathsf{alg}}$'s simulation is perfect. Suppose that $\mathsf{A}_{\mathsf{alg}}$ wins $\mathbf{G}$ given that $F = 1 \Leftrightarrow a' + \Sigma_i r_i \tilde{a}_i - r^* \not\equiv_p 0$. If r^* has not been defined at this point, $\mathsf{C}_{\mathsf{alg}}$ makes an additional query $\mathsf{H}(m^*)$ which defines r^*. Now,

$$zr^* \equiv_p z(a' + \Sigma_i r_i \tilde{a}_i) + (\hat{a} + \Sigma_i r_i \bar{a}_i). \tag{31}$$

$\mathsf{C}_{\mathsf{alg}}$ efficiently computes z from (31) as $z = (\hat{a} + \Sigma_i r_i a_i^h)(r^* - a' - \Sigma_i r_i \tilde{a}_i)^{-1} \bmod p$.

$\underline{\mathsf{C}_{\mathsf{alg}}}(\mathbf{Z} = g^z)$	$\underline{\mathsf{O}}(m_i)$	$\underline{\mathsf{H}}(m_i)$
00 $\mathbf{X} := \mathbf{Z}$	04 If $H(m_i) = \bot$	09 If $H(m_i) \neq \bot$
01 $(m^*, [\mathbf{\Sigma}^*]_{\vec{a}}) \xleftarrow{\$} \mathsf{A}^{\mathsf{O}(\cdot),\mathsf{H}(\cdot)}(\mathbf{X})$	05 $r_i \xleftarrow{\$} \mathbb{Z}_p$	10 Return $H(m_i)$
02 Compute z (see description)	06 $H(m_i) \leftarrow g^{r_i}$	11 $r_i \xleftarrow{\$} \mathbb{Z}_p$
03 Return z	07 $\mathbf{\Sigma}_i \leftarrow \mathbf{X}^{r_i}$	12 $H(m_i) \leftarrow g^{r_i}$
	08 Return $\mathbf{\Sigma}_i$	13 Return $H(m_i)$

Figure 27: Adversary $\mathsf{C}_{\mathsf{alg}}$ against $\mathbf{dlog}_{\mathcal{G}}$ in the proof of Theorem 6.1.

$\underline{\mathsf{D}_{\mathsf{alg}}}(\mathbf{Z} = g^z)$	$\underline{\mathsf{O}}(m_i)$	$\underline{\mathsf{H}}(m_i)$
00 $x \xleftarrow{\$} \mathbb{Z}_p$	05 $\mathbf{\Sigma}_i := \mathsf{H}(m_i)^x$	07 $b_i, \hat{r}_i \xleftarrow{\$} \mathbb{Z}_p$
01 $\mathbf{X} := g^x$	06 Return $\mathbf{\Sigma}_i$	08 Return $g^{zb_i + \hat{r}_i}$
02 $(m^*, [\mathbf{\Sigma}^*]_{\vec{a}}) \xleftarrow{\$} \mathsf{A}^{\mathsf{O}(\cdot),\mathsf{H}}(\mathbf{X})$		
03 Compute z (see above)		
04 Return z		

Figure 28: Adversary $\mathsf{D}_{\mathsf{alg}}$ against $\mathbf{dlog}_{\mathcal{G}}$ in the proof of Theorem 6.1.

Adversary $\mathsf{D}_{\mathsf{alg}}(\mathbf{Z} = g^z)$: Adversary $\mathsf{D}_{\mathsf{alg}}$ is depicted in Figure 28 and works as follows. It samples its own secret key $x \xleftarrow{\$} \mathbb{Z}_p$ and sets $\mathbf{X} := g^x$. To answer a query $\mathsf{H}(m_i)$, $\mathsf{D}_{\mathsf{alg}}$ first checks whether $H(m_i) = \bot$. If so, it samples $\hat{r}_i \xleftarrow{\$} \mathbb{Z}_p$ and $b_i \xleftarrow{\$} \mathbb{Z}_p$ and sets $H(m_i) = g^{r_i} := \mathbf{Z}^{b_i} g^{\hat{r}_i}$, which implicitly sets $r_i := \hat{r}_i + zb_i$. It then returns $H(m_i)$. To answer a query $\mathsf{O}(m_i)$, it first checks whether $\mathsf{H}(m_i)$ has previously been queried. If not, it first queries $\mathsf{H}(m_i)$, which defines $H(m_i)$ and the values $\hat{r}_i \in \mathbb{Z}_p, b_i \in \mathbb{Z}_p$, and $r_i = \hat{r}_i + zb_i$. It then computes and returns $\mathbf{\Sigma}_i := H(m_i)^x$. Again, it is straight-forward to verify that $\mathsf{D}_{\mathsf{alg}}$'s simulation is perfect. Suppose that $\mathsf{A}_{\mathsf{alg}}$ wins $\mathbf{G}$ given that $F = 0 \Leftrightarrow a' + \Sigma_i r_i \tilde{a}_i - r^* \equiv_p 0$. Now,

$$zb^* + \hat{r}^* \equiv_p r^* \equiv_p a' + \Sigma_i r_i \tilde{a}_i \equiv_p (a' + \Sigma_i \hat{r}_i \tilde{a}_i) + z(\Sigma_i b_i \tilde{a}_i) \equiv_p Az + B, \tag{32}$$

where $A := \Sigma_i b_i \tilde{a}_i$ and $B := (a' + \Sigma_i \hat{r}_i \tilde{a}_i)$. Note that the value of b^* is information-theoretically hidden from $\mathsf{A}_{\mathsf{alg}}$ and thus independent from the value of A. As we have argued, the sum $\Sigma_i r_i \tilde{a}_i$ may not contain a term of the form $a^* r^*$. This means in particular that $a' + \Sigma_i r_i \tilde{a}_i$ is not composed of the singleton term $r^* \equiv_p zb^* + \hat{r}^*$. Therefore, with probability $1 - \frac{1}{p}$, we have $b^* - A \not\equiv_p 0$ and thus $\mathsf{D}_{\mathsf{alg}}$ can compute z as $z := (B - \hat{r}^*)(b^* - A)^{-1} \bmod p$. ∎