The Algebraic Group Model and its Applications

Eike Kiltz and Julian Loss

Ruhr University Bochum
{eike.kiltz,julian.loss}@rub.de

Abstract. One of the most important tools for assessing hardness assumptions in cryptography is the Generic Group Model (GGM). Over the past two decades, several such assumptions 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 but still allowing to derive simple and meaningful security statements. We are able to 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 assumption in the AGM. 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 assumptions in the GGM.

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

1 Introduction

Starting with Nechaev [30] and Shoup [34], much work has been devoted to studying the computational complexity of problems with respect to generic group algorithms over cyclic groups [10,29,28]. At the highest level, generic group algorithms are algorithms that do not exploit any special structure of the representation of the group elements and thus can 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 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 [21] algorithm) as well as Pollard's rho algorithm [32]. 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 of 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 consider extensions of the generic group model [33,25,3,22]. Mostly relevant seems to be [22] which considers assumptions over bilinear groups with a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_3$, 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 and $\mathbb{G}_3$ is a sub-group of $\mathbb{Z}_p^*$, which is not generic.) However, it seems that so far, none of them capture algorithms which 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 has to output 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. Note that the classical "knowledge of exponent assumption" is given by definition in this model.

Such algebraic algorithms were first considered by Boneh and Venkatesan [11] who considered them in the context of straight-line programs computing polynomials over the ring of integers $\mathbb{Z}_n$, where $n = pq$. Later, [31] 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 security definitions and experiments, we distinguish group elements from all other parameters at a *syntactical level* to rule out pathological exploits of the model. It is not hard to see that our definition of algebraic algorithms is equivalent to the one given in [31]; we discuss this in more detail in Section 2. While this class of 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 [31] that the class of algebraic algorithms actually *includes* the class of generic algorithms. Extending [11], algebraic algorithms have been studied intensively [17,31,12,19,2,24] but, surprisingly, with the sole purpose to show negative results. More precisely, these works prove the impossibility of 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) in which all adversaries are modeled as algebraic. In contrast to the generic group model, we are not able to prove 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, $A \Rightarrow_{\mathsf{alg}} B$ implies that an algebraic adversary $\mathsf{B}_{\mathsf{alg}}$ against some primitive B can be transformed into an algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ against some other primitive A with (polynomially) related running times and success probabilities. It follows that if A is secure against algebraic adversaries, so is B. 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. The hope is that in the AGM one can exploit the algebraic nature of the adversary to obtain stronger security implications. Indeed, 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 [18]
- (Interactive) strong Diffie-Hellman assumption [1]
- (Interactive) LRSW assumption [26,16].

The Strong Diffie-Hellman Assumption is important since it is equivalent to the IND-CCA security of Hashed ElGamal encryption (also known as Diffie-Hellman Integrated Encryption Standard) in the random oracle model [1]. The LSRW assumption (named after the authors of [26]) is of great importance since it is equivalent to the (UF-CMA) security of Camenisch-Lysyanskaya (CL) signatures [16]. CL signatures are the main building block for anonymous credentials [16,7,6], group signatures [16,5], Ecash [14], unclonable functions [13], batch verification [15], and RFID encryption [4]. By 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 the above mentioned Diffie-Hellman based problems/schemes, has to 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 seem to help in solving these problems.

Furthermore, we show tight equivalence of IND-CCA1 security of the standard ElGamal Encryption and a decisional variant of the q-strong Diffie-Hellman problem in the algebraic group model.

Relation to the Generic Group model. The algebraic group model is stronger (in the sense that is puts more restrictions on the attackers) than the standard model, but weaker than the generic group model. 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 A and B be two computational problems and

let $\mathsf{S}_{\mathsf{alg}}$ be an algebraic algorithm that solves problem B. If we can convert $\mathsf{S}_{\mathsf{alg}}$ by means of a generic reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ into a solver $\mathsf{T}_{\mathsf{alg}}$ for problem B, then clearly, $\mathsf{T}_{\mathsf{alg}}$ is also an algebraic algorithm. However, we obtain an even stronger statement for free: Namely, if $\mathsf{S}_{\mathsf{gen}}$ is a generic algorithm solving A, then $\mathsf{T}_{\mathsf{gen}}$ is a generic algorithm solving B. This means that many results in the algebraic adversary model directly carry over to the generic group model. 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 for 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. To the best of our knowledge, no rigorous proof for the hardness of the LRSW assumption exists within the GGM; [26] provide only a proof sketch.

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 an easy example, we consider a group $\mathbb{G}$ which is equipped with a symmetric pairing $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$. e can be modeled in the GGM as an oracle which answers decisional Diffie-Hellman queries. However, it is not known 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 can not help in answering this question, because the GGM itself is based on the conjecture that e does not offer any functionality beyond a DDH oracle. In contrast, the AGM implicitly captures any such exploit without the need of having to model it and considers the relation between two problems instead of their *individual hardness*. This means that if one can reduce A to B in the AGM and A is conjectured to remain hard with respect to algebraic algorithms, even when given e, then also B remains hard. A similar statement can not 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 can not meaningfully be modeled as an oracle in the GGM. As an example, consider the Jacobi symbol, which was shown to be generically hard to compute in [23]. Indeed, it was left as an open problem in [3] to reexamine 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. On the other hand, in order to sample a random group element, one can sample instead $r \in \mathbb{Z}_p$ at random and then

compute g^r, thus obtaining in the process a representation. This view coincides with the GGM of Maurer [28] and excludes the possibility of obliviously sampling a random group element. For this reason, our model is strictly weaker than the one from [28]. In contrast, the GGM defined by Shoup [34] *does* allow for such a sampling process. We address this issue by modeling the process of algorithm $\mathsf{A}_{\mathsf{alg}}$ sampling a random group element $\mathbf{X}$ obliviously 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 $\vec{\mathbf{L}}$. In this manner, $\mathsf{A}_{\mathsf{alg}}$ trivially obtains a representation for $\mathbf{X}$, which resolves the problem. 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.

Finally, we remark that all of our results require that the group be of prime order. Generalizing them to composite-order groups would require a more involved analysis. Generic hardness bounds for composite-order groups have been considered in [34,29,28].

Further related work. [20] consider the LRSW assumption over a group of composite order and prove it secure under certain non-interactive hardness assumptions that can be shown to hold in the GGM.

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 (unless stated otherwise) probabilistic and written in uppercase letters A, B. To indicate that algorithm A runs on some inputs $(x_1, x_2, ...)$ and returns y, we write $y \xleftarrow{\$} \mathsf{A}(x_1, x_2, ...)$. If additionally, A has access to an algorithm B (via oracle access) during its execution, we write $y \xleftarrow{\$} \mathsf{A}^{\mathsf{B}}(x_1, x_2, ...)$.

Security games. We use a variant of (code-based) *security games* [9]. 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 $\mathbf{Adv}^{\mathbf{G}}_{par,\mathsf{A}} := \Pr[\mathbf{G}^{\mathsf{A}}_{par} = 1]$ 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 $\mathbf{G}_{par} \overset{(\Delta_\varepsilon, \Delta_t)}{\Longrightarrow} \mathbf{H}_{par}$ if there exists an algorithm R (called $(\Delta_\varepsilon, \Delta_t)$-*reduction*) such that for all algorithms

$\underline{\mathbf{cdh}}^{\mathsf{A}}_{\mathcal{G}}$	$\underline{\mathbf{cdh}}^{\mathsf{A}_{\mathsf{alg}}}_{\mathcal{G}}$
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 $\mathbf{Z} = g^{xy}$

Fig. 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 $\mathbf{Z} = g^a \mathbf{X}^b \mathbf{Y}^c$.

B, algorithm A defined as $\mathsf{A} := \mathsf{R}^{\mathsf{B}}$ satisfies

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

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 with generator 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. 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 come up with a new group element $\mathbf{Z}$ is to derive it via group multiplications from known group elements.

Definition 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* $\mathbf{L}_0, ..., \mathbf{L}_m \in \mathbb{G}$ *that* $\mathsf{A}_{\mathsf{alg}}$ *has received so far (w.l.o.g.* $\mathbf{L}_0 = g$*), then* $\mathsf{A}_{\mathsf{alg}}$ *also has to 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 [11,31] where they are defined using an additional extractor algorithm which computes for an output group element a representation in the basis of $\vec{\mathbf{L}}$. We believe that our definition gives a simpler and cleaner definition of algebraic algorithms. If one assumes the extractor algorithm to require constant running time, then our definition is easily seen to be equivalent to theirs. Indeed, this view makes sense for algorithms placed 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. However, if the cost of the extractor is similar to that of the solver, then reductions in our model that neither call an algebraic solver multiple times or receive from it a non-constant amount of group elements (along with their representations), will remain largely the same in both models.

We also remark that the syntactic distinction of group elements in our games is necessary to rule out pathological examples in which an adversary receives "disguised" group elements and is forced to output an algebraic representation of this element. To see why this is necessary, consider the following algorithm $\mathsf{A}_{\mathsf{alg}}$ that solves the discrete logarithm problem: $\mathsf{A}_{\mathsf{alg}}$ disguises its problem instance $\mathbf{X} = g^x$ as $\mathbf{X}' := \mathbf{X} \| \perp$. This way $\mathbf{X}'$ is formally not a group element but from $\mathbf{X}'$ one can efficiently reconstruct $\mathbf{X}$ (and vice-versa). Consider the trivial algebraic algorithm $\mathsf{B}_{\mathsf{alg}}$, which on input $(g, \mathbf{X}')$ extracts the disguised group element $\mathbf{X}$ and outputs it in a canonical form. Since $\mathsf{B}_{\mathsf{alg}}$ is algebraic, it must output a representation of $\mathbf{X}$ in g, which can only be the discrete logarithm, x. Clearly, our syntactical distinction rules out such an exploit, since now the game recognizes $\mathbf{X}'$ as a group element (at a syntactical level).

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 algebraic algorithm $\mathsf{A}_{\mathsf{alg}}$ 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 [8]. 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*[1] *in the generic group model* if for every generic algorithm $\mathsf{A}_{\mathsf{gen}}$ it holds that

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

[1] 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 $\mathbf{dlog}_{\mathcal{G}}$ (Figure 2) which is $\mathbf{dlog}_{\mathcal{G}}$ is $\big(t^2/p, t\big)$-hard in the generic group model [34,28].

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 $\mathsf{A}_{\mathsf{gen}}$ is a generic algorithm and $\mathsf{B}_{\mathsf{alg}}$ is an algebraic algorithm, then $\mathsf{A}_{\mathsf{alg}} := \mathsf{A}_{\mathsf{gen}}^{\mathsf{B}_{\mathsf{alg}}}$ is also is an algebraic algorithm. We refer to [28] for more information of generic algorithms.

2.3 Generic security reductions between algebraic security games

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

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

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

Composing information theoretic lower bounds with reductions in the AGM.

Lemma 1. *Let $\mathbf{G}_{\mathcal{G}}$ and $\mathbf{H}_{\mathcal{G}}$ be algebraic security games such that $\mathbf{G}_{\mathcal{G}} \overset{(\Delta_\varepsilon,\Delta_t)}{\Longrightarrow}_{\mathsf{alg}} \mathbf{H}_{\mathcal{G}}$ and winning $\mathbf{G}_{\mathcal{G}}$ is (ε, t)-hard in the GGM. Then, $\mathbf{H}_{\mathcal{G}}$ is $(\varepsilon \cdot \Delta_\varepsilon, t/\Delta_t)$-hard in the GGM.*

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

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

If $\mathbf{Time}^{\mathbf{H}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}} \leq t/\Delta_t$, then

$$\mathbf{Time}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}} \leq \Delta_t \cdot \mathbf{Time}^{\mathbf{H}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}} \leq t.$$

Since winning $\mathbf{G}_{\mathcal{G}}$ is (ε, t)-hard in the GGM, it follows that

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

and thus $\varepsilon\Delta_\varepsilon \geq \mathbf{Adv}^{\mathbf{H}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}}$, which is equivalent to $\mathbf{H}_{\mathcal{G}}$ being $(\varepsilon\Delta_\varepsilon, t/\Delta_t)$-hard in the GGM.

3 The Diffie-Hellman Assumption and Variants

In this section we consider some variants of the standard Diffie-Hellman assumption [18] 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 Root Diffie-Hellman Assumption [27] 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).

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.

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

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

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

Proof. Let $\mathsf{B}_{\mathsf{alg}}$ be an algebraic adversary executed in game $\mathbf{sq\text{-}dh}_{\mathcal{G}}$, cf. Figure 3. As $\mathsf{B}_{\mathsf{alg}}$ is an algebraic adversary, it returns a solution $\mathbf{Z}$ together with a representation $a, b \in \mathbb{Z}_p$ such that

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

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

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

$\mathsf{R}_{\mathsf{gen}}$ works as follows. It inputs its discrete logarithm instance $\mathbf{X}$ and runs $\mathsf{B}_{\mathsf{alg}}$ on $\mathbf{X}$. Suppose $\mathsf{B}_{\mathsf{alg}}$ is successful. Equation (1) is equivalent to the quadratic equation $x^2 - ax - b \equiv_p 0$ with at most two solutions in x. Note that in general such equations are not guaranteed to have a solution but since the representation is valid and $\mathsf{B}_{\mathsf{alg}}$ is assumed to be correct, there exists at least one solution for

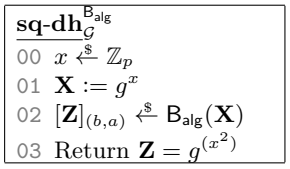

Fig. 3. Algebraic adversary $\mathsf{B}_{\mathsf{alg}}$ playing in $\mathbf{sq\text{-}dh}_{\mathcal{G}}$.

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, $\mathsf{A}_{\mathsf{alg}} := \mathsf{R}_{\mathsf{gen}}^{\mathsf{B}_{\mathsf{alg}}}$ is algebraic, which proves

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

The statement $\mathbf{dlog}_{\mathcal{G}} \overset{(1,1)}{\Longrightarrow}_{\mathsf{alg}} \mathbf{cdh}_{\mathcal{G}}$ easily follows as it is straightforward to construct an adversary against $\mathbf{sq\text{-}dh}_{\mathcal{G}}$ from any adversary against $\mathbf{cdh}_{\mathcal{G}}$ that runs in the same time and has the same probabiltiy of success.

Thus 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 $\mathsf{B}_{\mathsf{alg}}$ against $\mathbf{sq\text{-}dh}_{\mathcal{G}}$ as follows: On input $\mathbf{X} = g^x$, $\mathsf{B}_{\mathsf{alg}}$ samples $r \xleftarrow{\$} \mathbb{Z}_p$ and computes either $(\mathbf{X}, g^r)$, $(g^r, \mathbf{X})$, or $(\mathbf{X}, \mathbf{X}^r)$ with probability $1/3$, respectively. 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 of $\mathbf{X}_1 = \mathbf{X}, \mathbf{X}_2 = g^r$ 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, $\mathsf{B}_{\mathsf{alg}}$ 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 1. $\mathbf{cdh}_{\mathcal{G}}$ *and* $\mathbf{sq\text{-}dh}_{\mathcal{G}}$ *are* $\big(t^2/p, t\big)$*-hard in the generic group model and* $\mathbf{lc\text{-}dh}_{\mathcal{G}}$ *is* $\big(3t^2/p, t\big)$*-hard in the generic group model.*

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

3.2 Strong Diffie-Hellman

Consider the Strong Diffie-Hellman Assumption [1] described via game $\mathbf{sdh}_{\mathcal{G}}$ (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 2 can be found in Appendix A. 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

$\underline{\mathbf{sdh}}^{\mathsf{A}}_{\mathcal{G}}$	$\mathsf{O}(\mathbf{Y}', \mathbf{Z}')$:
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 $\mathbf{Z} = g^{xy}$	

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

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 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 one of the games $\mathbf{cdh}_{\mathcal{G}}$, $\mathbf{sq\text{-}dh}_{\mathcal{G}}$, or $\mathbf{lc\text{-}dh}_{\mathcal{G}}$. All of these games can be efficiently simulated by $\mathsf{B}_{\mathsf{alg}}$, as we have shown in the previous section.

Theorem 2. $\mathbf{dlog}_{\mathcal{G}} \overset{(12q,1)}{\Longrightarrow}_{\mathsf{alg}} \mathbf{sdh}_{\mathcal{G}}$, *where q is the maximal number of queries to oracle $\mathsf{O}(\cdot,\cdot)$ in $\mathbf{sdh}_{\mathcal{G}}$.*

Corollary 2. $\mathbf{sdh}_{\mathcal{G}}$ *is* $\left(t, \frac{t^2}{12pq}\right)$*-hard in the generic group model.*

4 The LRSW Assumption

The interactive LRSW assumption [26,16] is defined via the algebraic security game **lrsw**, see Figure 5.

$\underline{\mathbf{lrsw}}^{\mathsf{A}}_{\mathcal{G}}$	$\mathsf{O}(m_j)$ *//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 $\mathbf{B}_j := g^{yr_j}$
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 $m^* \notin Q \wedge m^* \neq 0$	09 $Q := Q \cup \{m_j\}$
$\wedge\ \mathbf{A}^* \neq 1 \wedge\ \mathbf{B}^* = (\mathbf{A}^*)^y \wedge\ \mathbf{C}^* = (\mathbf{A}^*)^{xm^*y+x}$	10 Return $(\mathbf{A}_j, \mathbf{B}_j, \mathbf{C}_j)$

Fig. 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 and defer the full proof of Theorem 3 to Appendix B. 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. The main idea is to let adversary $\mathsf{B}_{\mathsf{alg}}$ against $\mathbf{dlog}_{\mathcal{G}}$ 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 chooses randomly the i^*th query of $\mathsf{A}_{\mathsf{alg}}$ to the oracle $\mathsf{O}(\cdot)$ in $\mathbf{lrsw}_{\mathcal{G}}$ to embed the value of z. These behaviors correspond in the proof below 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 2). 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 3. $\mathbf{dlog}_{\mathcal{G}} \overset{(6q,1)}{\Longrightarrow}_{\mathsf{alg}} \mathbf{lrsw}_{\mathcal{G}}$, *where* $q \geq 6$ *is the maximal number of queries to* $\mathsf{O}(\cdot)$ *in* $\mathbf{lrsw}_{\mathcal{G}}$.

Corollary 3. $\mathbf{lrsw}_{\mathcal{G}}$ *is* $\left(t, \frac{t^2}{6pq}\right)$*-hard in the generic group model.*

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 the q-Strong Decision Diffie-Hellman Assumption in the Algebraic Group Model.

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) that can be output by $\mathsf{Enc}(pk)$, we have $\mathsf{Dec}(sk, C) = K$. We formalize IND-CCA1 security of a KEM via the games depicted in Figure 6.

In the following, we consider the ElGamal KEM EG depicted in Figure 7. We also consider the following, stronger variant of the well known Decisional Diffie-Hellman Game which we call q-Strong Decisional Diffie-Hellman Game. This game is depicted in Figure 8.

$\underline{\mathbf{ind\text{-}cca1}}^{\mathsf{A}}_{\mathsf{KEM},par,b}$	$\mathsf{Dec}(C)$	$\mathsf{Enc}()$ //*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^*)

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

Advantage for decisional algebraic security games. We parameterize *decisional* algebraic games such as the q-Strong Decisional Diffie-Hellman Game

with a parameter bit b. Let $\mathbf{G}$ be an algebraic decisional security game. We define the advantage of adversary A in $\mathbf{G}$ as

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

Additionally we slightly alter the definition of algebraic security games for this section. Namely, instead of setting $par = \mathcal{G}$, we only require that $\mathcal{G}$ be *included* in par. Let $\mathbf{G}_{par}, \mathbf{H}_{par}$ be decisional algebraic security games. We write $\mathbf{G}_{par} \overset{(\Delta_\varepsilon,\Delta_t)}{\Longrightarrow}_{\mathsf{alg}} \mathbf{H}_{par}$ if there exists a generic algorithm $\mathsf{R}_{\mathsf{gen}}$ (called generic $(\Delta_\varepsilon, \Delta_t)$-*reduction*) such that for algebraic algorithm $\mathsf{A}_{\mathsf{alg}}$ defined as $\mathsf{A}_{\mathsf{alg}} := \mathsf{R}^{\mathsf{B}_{\mathsf{alg}}}_{\mathsf{gen}}$, we have

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

$\mathsf{Gen}(\mathcal{G})$	$\mathsf{Enc}(pk)$:	$\mathsf{Dec}(\mathbf{C}, sk)$:
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 $\tilde{\mathbf{K}}$

Fig. 7. ElGamal KEM $\mathsf{EG} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$

$\underline{\mathbf{q\text{-}sddh}^{\mathsf{A}}_{\mathcal{G},b}}$
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'

Fig. 8. q-Strong Decisional Diffie-Hellman Game **q-sddh** relative to $\mathcal{G}$ and adversary A.

Theorem 4. $\mathbf{ind\text{-}cca1}_{\mathsf{EG},\mathcal{G}} \overset{(1,1)}{\Longleftrightarrow}_{\mathsf{alg}} \mathbf{q\text{-}sddh}_{\mathcal{G}}$, *where* $q-1$ *is the maximal number of queries to* $\mathsf{Dec}(\cdot)$ *in* $\mathbf{ind\text{-}cca1}_{\mathsf{EG},\mathcal{G}}$.

Proof. First note that given an adversary $\mathsf{A}_{\mathsf{alg}}$ against $\mathbf{q\text{-}sddh}_{\mathcal{G}}$ 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

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

$\underline{\textbf{ind-cca1}}^{\mathsf{A}}_{\mathsf{EG},\mathcal{G}}$	$\mathsf{Dec}([\mathbf{C}]_{\vec{a}})$	$\mathsf{Enc}()$ //*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 $\underline{\mathbf{K}^* := \mathbf{X}^r}$
03 Return b'		09 $\boxed{\mathbf{K}^* \xleftarrow{\$} \mathcal{K}}$
		10 Return $(\mathbf{K}^*, \mathbf{C}^*)$

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

For the converse, let $\mathsf{A}_{\mathsf{alg}}$ be an algebraic adversary playing in one of the games $\textbf{ind-cca1}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{EG},\mathcal{G},0}, \textbf{ind-cca1}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{EG},\mathcal{G},1}$. We construct an adversary $\mathsf{B}_{\mathsf{alg}}$ against **q-sddh** that interpolates between $\textbf{ind-cca1}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{EG},\mathcal{G},0}$ and $\textbf{ind-cca1}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{EG},\mathcal{G},1}$ by simulating one of these games to $\mathsf{A}_{\mathsf{alg}}$. $\mathsf{B}_{\mathsf{alg}}$ is depicted in Figure 10.

$\mathsf{B}_{\mathsf{alg}}(\vec{\mathbf{I}})$	$\mathsf{Dec}([\mathbf{C}]_{\vec{a}})$	$\mathsf{Enc}()$ //*One time*
00 $\mathbf{X} := g^x$	//*Before* Enc *is called*	06 $\mathbf{C}^* := g^r$
01 $b' \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}^{\mathsf{Dec}(\cdot),\mathsf{Enc}}(\mathbf{X})$	03 Compute $\vec{a}'$ s.t.	07 $\mathbf{K}^* := g^{xr+zb}$
02 Return b'	$\mathbf{C} = \prod_{j<q} g^{a'_j x^j}$	08 Return $(\mathbf{K}^*, \mathbf{C}^*)$
	04 $\mathbf{K} := \mathbf{C}^x = \prod_{j\leq q} g^{a'_j x^j}$	
	05 Return $\mathbf{K}$	

Fig. 10. Adversary $\mathsf{B}_{\mathsf{alg}}$.

Let $\vec{\mathbf{I}} := (g, g^x, g^{x^2}, ..., g^{x^q}, g^r, g^{xr+zb})$ be the problem instance given to $\mathsf{B}_{\mathsf{alg}}$ in $\textbf{q-sddh}^{\mathsf{B}_{\mathsf{alg}}}_{\mathcal{G},b}$. We now analyze $\mathsf{B}_{\mathsf{alg}}$. As $\mathsf{A}_{\mathsf{alg}}$ is an algebraic adversary, it sends along with its i^{th} query $\mathbf{C}$ to $\mathsf{Dec}(\cdot)$ a vector $\vec{a}$ such that $\mathbf{C} = \prod_i \mathbf{L}_i^{a_i}$ 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 $\mathsf{Dec}(\cdot)$. It is easy to see that given $\vec{a}$, $\mathsf{B}_{\mathsf{alg}}$ can rewrite $\mathbf{C}$ 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 $(g, g^x, g^{x^2}, ..., g^{x^q})$ and raising them to appropriate powers a'_i. When $\mathsf{A}_{\mathsf{alg}}$ queries $\mathsf{Enc}()$, $\mathsf{B}_{\mathsf{alg}}$ returns (g^{xr+zb}, g^r). When $\mathsf{A}_{\mathsf{alg}}$ returns with output b', $\mathsf{B}_{\mathsf{alg}}$ returns b'. Clearly, $\mathsf{B}_{\mathsf{alg}}$ perfectly simulates either $\textbf{ind-cca1}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{EG},\mathcal{G},0}$ or $\textbf{ind-cca1}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{EG},\mathcal{G},1}$ to $\mathsf{A}_{\mathsf{alg}}$. Finally, note that whenever $\mathsf{A}_{\mathsf{alg}}$ wins $\textbf{ind-cca1}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{EG},\mathcal{G},b}$, $\mathsf{B}_{\mathsf{alg}}$ wins $\textbf{q-sddh}^{\mathsf{B}_{\mathsf{alg}}}_{\mathcal{G},b}$. Therefore,

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

References

1. M. Abdalla, M. Bellare, and P. Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In D. Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 143–158. Springer, Heidelberg, Apr. 2001. 3, 10
2. M. Abe, J. Groth, and M. Ohkubo. Separating short structure-preserving signatures from non-interactive assumptions. In D. H. Lee and X. Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 628–646. Springer, Heidelberg, Dec. 2011. 2
3. D. Aggarwal and U. Maurer. Breaking RSA generically is equivalent to factoring. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 36–53. Springer, Heidelberg, Apr. 2009. 2, 4
4. G. Ateniese, J. Camenisch, and B. de Medeiros. Untraceable RFID tags via insubvertible encryption. In V. Atluri, C. Meadows, and A. Juels, editors, *ACM CCS 05*, pages 92–101. ACM Press, Nov. 2005. 3
5. G. Ateniese, J. Camenisch, S. Hohenberger, and B. de Medeiros. Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385, 2005. http://eprint.iacr.org/2005/385. 3
6. M. Backes, J. Camenisch, and D. Sommer. Anonymous yet accountable access control. In *WPES*, pages 40–46, 2005. 3
7. 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. 3
8. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993. 7
9. M. Bellare and P. 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. 5
10. D. Boneh and R. J. Lipton. Algorithms for black-box fields and their application to cryptography (extended abstract). In N. Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 283–297. Springer, Heidelberg, Aug. 1996. 1
11. D. Boneh and R. Venkatesan. Breaking RSA may not be equivalent to factoring. In K. Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 59–71. Springer, Heidelberg, May / June 1998. 2, 6
12. E. Bresson, J. Monnerat, and D. Vergnaud. Separation results on the "one-more" computational problems. In T. Malkin, editor, *CT-RSA 2008*, volume 4964 of *LNCS*, pages 71–87. Springer, Heidelberg, Apr. 2008. 2
13. J. Camenisch, S. Hohenberger, M. Kohlweiss, A. Lysyanskaya, and M. Meyerovich. How to win the clonewars: Efficient periodic n-times anonymous authentication. In A. Juels, R. N. Wright, and S. Vimercati, editors, *ACM CCS 06*, pages 201–210. ACM Press, Oct. / Nov. 2006. 3
14. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-cash. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 302–321. Springer, Heidelberg, May 2005. 3
15. J. Camenisch, S. Hohenberger, and M. Ø. Pedersen. Batch verification of short signatures. In M. Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 246–263. Springer, Heidelberg, May 2007. 3
16. J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, Heidelberg, Aug. 2004. 3, 11

17. J.-S. Coron. Optimal security proofs for PSS and other signature schemes. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 272–287. Springer, Heidelberg, Apr. / May 2002. 2
18. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 3, 9
19. S. Garg, R. Bhaskar, and S. V. Lokam. Improved bounds on security reductions for discrete log based signatures. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 93–107. Springer, Heidelberg, Aug. 2008. 2
20. M. Gerbush, A. B. Lewko, A. O'Neill, and B. Waters. Dual form signatures: An approach for proving security from static assumptions. In X. Wang and K. Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 25–42. Springer, Heidelberg, Dec. 2012. 5
21. 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. 2
22. T. Jager and A. Rupp. The semi-generic group model and applications to pairing-based cryptography. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 539–556. Springer, Heidelberg, Dec. 2010. 2
23. T. Jager and J. Schwenk. On the analysis of cryptographic assumptions in the generic ring model. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 399–416. Springer, Heidelberg, Dec. 2009. 4
24. E. Kiltz, D. Masny, and J. Pan. Optimal security proofs for signatures from identification schemes. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 33–61. Springer, Heidelberg, Aug. 2016. 2
25. G. Leander and A. Rupp. On the equivalence of RSA and factoring regarding generic ring algorithms. In X. Lai and K. Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 241–251. Springer, Heidelberg, Dec. 2006. 2
26. A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In H. M. Heys and C. M. Adams, editors, *SAC 1999*, volume 1758 of *LNCS*, pages 184–199. Springer, Heidelberg, Aug. 1999. 3, 4, 11
27. U. Maurer and S. Wolf. The relationship between breaking the diffie-hellman protocol and computing discrete logarithms. *SIAM Journal on Computing*, 28(5):1689–1721, 1999. 9
28. U. M. Maurer. Abstract models of computation in cryptography (invited paper). In N. P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Heidelberg, Dec. 2005. 1, 5, 8
29. U. M. Maurer and S. Wolf. Lower bounds on generic algorithms in groups. In K. Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 72–84. Springer, Heidelberg, May / June 1998. 1, 5
30. V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994. 1
31. P. Paillier and D. Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In B. K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 1–20. Springer, Heidelberg, Dec. 2005. 2, 6
32. J. M. Pollard. Monte Carlo methods for index computation mod p. *Mathematics of Computation*, 32:918–924, 1978. 2
33. R. L. Rivest. On the notion of pseudo-free groups. In M. Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 505–521. Springer, Heidelberg, Feb. 2004. 2
34. V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997. 1, 5, 8

35. V. 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. 18

A Proof of Theorem 2

Proof.

$$\mathbf{Adv}^{\mathbf{cdh}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}} \geq \mathbf{Adv}^{\mathbf{sdh}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}} - q(\mathbf{Adv}^{\mathbf{cdh}}_{\mathcal{G},\mathsf{C}_{\mathsf{alg}}} + \mathbf{Adv}^{\mathbf{sq\text{-}dh}}_{\mathcal{G},\mathsf{D}_{\mathsf{alg}}} + \mathbf{Adv}^{\mathbf{lc\text{-}dh}}_{\mathcal{G},\mathsf{E}_{\mathsf{alg}}}). \tag{2}$$

Applying Theorem 1 yields the theorem. We now prove (2) 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 12.

$\mathbf{G_1}$: 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 12. 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 11. We now show the existence of adversaries $\mathsf{C}_{\mathsf{alg}}, \mathsf{D}_{\mathsf{alg}}, \mathsf{E}_{\mathsf{alg}}$ such that

$\mathsf{B}_{\mathsf{alg}}(\mathbf{X} = g^x, \mathbf{Y} = g^y)$	$\mathsf{O}([\mathbf{Y}']_{\vec{b}}, [\mathbf{Z}']_{\vec{c}})$:
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 $\mathbf{Z}' = \mathbf{X}^{b_1}$

Fig. 11. Behavior of adversary $\mathsf{B}_{\mathsf{alg}}$.

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

Fig. 12. Games $\mathbf{G_0}$ and $\mathbf{G_1}$. The boxed statements are only executed in $\mathbf{G_1}$.

$$\left|\mathbf{Adv}^{\mathbf{G_0}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}} - \mathbf{Adv}^{\mathbf{G_1}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}\right| \leq q \cdot (\mathbf{Adv}^{\mathbf{cdh}}_{\mathcal{G},\mathsf{C}_{\mathsf{alg}}} + \mathbf{Adv}^{\mathbf{sq\text{-}dh}}_{\mathcal{G},\mathsf{D}_{\mathsf{alg}}} + \mathbf{Adv}^{\mathbf{lc\text{-}dh}}_{\mathcal{G},\mathsf{E}_{\mathsf{alg}}}).$$

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 [35], 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

$$\Pr[F \mid b_2 \neq 0 \wedge b_3 \neq 0] \leq q \cdot \mathbf{Adv}^{\mathbf{lc\text{-}dh}}_{\mathcal{G},\mathsf{E}_{\mathsf{alg}}}.$$

$\mathsf{E}_{\mathsf{alg}}$ is depicted in Figure 13.

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

Fig. 13. Behavior of adversary $\mathsf{E}_{\mathsf{alg}}$.

We now analyze $\mathsf{E}_{\mathsf{alg}}$. Clearly, $\mathsf{E}_{\mathsf{alg}}$ runs in the same time as $\mathsf{A}_{\mathsf{alg}}$. Once $\mathsf{A}_{\mathsf{alg}}$ halts, $\mathsf{E}_{\mathsf{alg}}$ 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 $b_2 \neq 0$, $b_3 \neq 0$, and

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

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 $\mathbf{lc\text{-}dh}^{\mathsf{E}_{\mathsf{alg}}}_{\mathcal{G}}$ as

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

As $\mathsf{E}_{\mathsf{alg}}$ picks $\tilde{\mathbf{Z}}$ at random from at most q elements in Q, it picks a correct solution with probability at least

$$\mathbf{Adv}^{\mathbf{lc\text{-}dh}}_{\mathcal{G},\mathsf{E}_{\mathsf{alg}}} \geq \frac{\Pr[F \mid b_2 \neq 0 \wedge b_3 \neq 0]}{q}.$$

By a similar argument, we obtain

$$\Pr[F \mid b_2 = 0 \wedge b_3 \neq 0] \leq q \cdot \mathbf{Adv}^{\mathbf{cdh}}_{\mathcal{G},\mathsf{C}_{\mathsf{alg}}}$$

and

$$\Pr[F \mid b_2 \neq 0 \wedge b_3 = 0] \leq q \cdot \mathbf{Adv}^{\mathbf{sq\text{-}dh}}_{\mathcal{G},\mathsf{D}_{\mathsf{alg}}}.$$

Combining terms, this yields

$$\begin{aligned}\Pr[F] &\leq \Pr[F \mid b_2 = 0 \wedge b_3 \neq 0] + \Pr[F \mid b_2 \neq 0 \wedge b_3 = 0] + \Pr[F \mid b_2 \neq 0 \wedge b_3 \neq 0] \\ &\leq q \cdot (\mathbf{Adv}^{\mathbf{cdh}}_{\mathcal{G},\mathsf{C}_{\mathsf{alg}}} + \mathbf{Adv}^{\mathbf{sq\text{-}dh}}_{\mathcal{G},\mathsf{D}_{\mathsf{alg}}} + \mathbf{Adv}^{\mathbf{lc\text{-}dh}}_{\mathcal{G},\mathsf{E}_{\mathsf{alg}}}).\end{aligned}$$

Thus, we now have

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

It is straight forward to see that all the steps performed in the above simulations are generic. This proves (2).

B Proof of Theorem 3

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

$\mathbf{G_0}$: Let $\mathsf{A}_{\mathsf{alg}}$ be an algebraic adversary playing in $\mathbf{G_0} := \mathbf{lrsw}^{\mathsf{A}_{\mathsf{alg}}}_{\mathcal{G}}$. Game $\mathbf{G_0}$ is depicted in Figures 14. 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:

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

$$\mathbf{B}^* = \prod_{i=1}^{q} \mathbf{A}_i^{b_i} g^{b_{q+1}} \prod_{i=q+2}^{2q+1} \mathbf{B}_{i-q-1}^{b_i} \prod_{i=2q+2}^{3q+1} \mathbf{C}_{i-2q-1}^{b_i} \mathbf{X}^{b_{3q+2}} \mathbf{Y}^{b_{3q+3}}, \tag{4}$$

$$\mathbf{C}^* = \prod_{i=1}^{q} \mathbf{C}_i^{c_i} \mathbf{X}^{c_{q+1}} \prod_{i=q+2}^{2q+1} \mathbf{A}_{i-q-1}^{c_i} \prod_{i=2q+2}^{3q+1} \mathbf{B}_{i-2q-1}^{c_i} g^{c_{3q+2}} \mathbf{Y}^{c_{3q+3}}. \tag{5}$$

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 14. 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 $s_1, s_2, t_1, t_2, u_1, u_2, v_1, v_2 \in$

$\mathbf{G_0}, \mathbf{G_1}$	$\mathsf{O}(m_j)$ //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 $\mathbf{B}_j := g^{y r_j}$
03 $Q := \emptyset$	11 $\mathbf{C}_j := g^{r_j m_j x y + r_j x}$
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 $m^* \notin Q \wedge m^* \neq 0$	
$\wedge\ \mathbf{A}^* \neq 1 \wedge\ \mathbf{B}^* = (\mathbf{A}^*)^y \wedge\ \mathbf{C}^* = (\mathbf{A}^*)^{x m^* y + x}$	

Fig. 14. Games $\mathbf{G_0}$ and $\mathbf{G_1}$ with algebraic adversary $\mathsf{A}_{\mathsf{alg}}$. The boxed statements are only executed in $\mathbf{G_1}$.

$\mathbb{Z}_p$ as

$$s_1 := a_{3q+2} + \sum_{i=2q+3}^{3q+2} a_i r_{i-2q-2}, \qquad s_2 := c_{q+1} + \sum_{i=1}^{q} c_i r_i,$$

$$t_1 := \sum_{i=2q+3}^{3q+2} a_i m_i r_{i-2q-2}, \qquad t_2 := \sum_{i=1}^{q} c_i m_i r_i,$$

$$u_1 := a_{3q+3} + \sum_{i=q+2}^{2q+1} a_i r_{i-q-1}, \qquad u_2 := c_{3q+3} + \sum_{i=2q+2}^{3q+1} c_i r_{i-q-1},$$

$$v_1 := g^{a_{q+1}} + \sum_{i=1}^{q} a_i r_i, \qquad v_2 := g^{c_{3q+2}} + \sum_{i=q+2}^{2q+1} c_i r_{i-q-1},$$

equations (3) and (5) can be further simplified to

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

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

$$\Delta := m^* t_1 y^2 + t_1 y + s_1 m^* y + s_1, \tag{6}$$
$$\Delta' := u_1 m^* y^2 + m^* y v_1 + u_1 y - t_2 y - s_2 + v_1, \tag{7}$$
$$\Delta'' := u_2 y + v_2, \tag{8}$$

and the boolean variable F^* as

$$F^* = 1 \Leftrightarrow s_1 \equiv_p t_1 \equiv_p u_1 \equiv_p u_2 \equiv_p v_2 \equiv_p 0. \tag{9}$$

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

Lemma 2. *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 2, 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:

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

Note that H^* is only well defined (by Lemma 2) 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 3. *There exist $\mathsf{C_{alg}}, \mathsf{D_{alg}}, \mathsf{E_{alg}}$ playing in the discrete logarithm game such that:*

$$\Pr\left[\mathbf{dlog}^{\mathsf{C_{alg}}} = 1\right] = \Pr[\mathbf{G_1} = 1 \mid F_1] \tag{10}$$

$$\Pr\left[\mathbf{dlog}^{\mathsf{D_{alg}}} = 1\right] \geq (1 - \frac{2}{q}) \Pr[\mathbf{G_1} = 1 \mid F_2] \tag{11}$$

$$\Pr\left[\mathbf{dlog}^{\mathsf{E_{alg}}} = 1\right] \geq \frac{1}{q} \Pr[\mathbf{G_1} = 1 \mid F_3]. \tag{12}$$

Proof. 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 $\mathsf{A}_{\mathsf{alg}}$ 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

$$\mathbf{A}^* = g^{a_{q+1}} \prod_{i=1}^{q} \mathbf{A}_i^{a_i}$$

and

$$\mathbf{C}^* = \mathbf{X}^{c_{q+1}} \prod_{i=1}^{q} \mathbf{C}_i^{c_i}.$$

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

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

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 3 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).

Proof. Let $\mathbf{Z} = g^z$ denote the discrete logarithm instance. $\mathsf{C_{alg}}, \mathsf{D_{alg}}, \mathsf{E_{alg}}$ simulate $\mathbf{G_1}$ to $\mathsf{A_{alg}}$. 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 $\mathsf{C_{alg}}$. Adversary $\mathsf{C_{alg}}$ 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

$\mathsf{C_{alg}}(\mathbf{Z} = g^z)$	$\mathsf{O}(m_j)$: *//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 $(m^*, [\mathbf{A}^*]_{\vec{a}}, [\mathbf{B}^*]_{\vec{b}}, [\mathbf{C}^*]_{\vec{c}}) \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}^{\mathsf{O}(\cdot)}(\mathbf{Z}, g^\alpha)$	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 $(\mathbf{A}_j, \mathbf{B}_j, \mathbf{C}_j)$

Fig. 15. Behavior of adversary $\mathsf{C_{alg}}$.

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

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

$$x^2\Delta + x\Delta' - \Delta'' \equiv_p 0 \tag{13}$$

where $\Delta, \Delta', \Delta''$ are defined in (6)-(8). Quadratic equation (13) in indeterminate x has exactly two (possibly equal) solutions, say x_1 and x_2, that can be computed efficiently by $\mathsf{C_{alg}}$. 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 (10).

Adversary $\mathsf{D_{alg}}$*:* Adversary $\mathsf{D_{alg}}$ 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

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

where F^* is defined in (9).

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

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

$\mathsf{D}_{\mathsf{alg}}(\mathbf{Z}=g^z)$	$\mathsf{O}(m_j)$: //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 $(\mathbf{A}_j,\mathbf{B}_j,\mathbf{C}_j)$

Fig. 16. Behaviour of $\mathsf{D}_{\mathsf{alg}}$.

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

$$\Delta \equiv_p 0,$$
$$\Delta' \equiv_p 0,$$
$$\Delta'' \equiv_p 0.$$

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 2 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

$$\forall j \notin \{\ell^*, k^*\} : r_j \varepsilon_j (1 + m^* y) - r_j \delta_j (1 + y m_j) \equiv_p 0, \tag{14}$$

$\mathsf{D}_{\mathsf{alg}}$ 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 (14). 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 (11).

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 17. 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 2 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

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

$\mathsf{E}_{\mathsf{alg}}(\mathbf{Z} = g^z)$	$\mathsf{O}(m_j)$: //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 $(\mathbf{A}_j, \mathbf{B}_j, \mathbf{C}_j)$

Fig. 17. Behaviour of $\mathsf{E}_{\mathsf{alg}}$.

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 $q \geq 6$.