On the CCA1-Security of Elgamal and Damgård's Elgamal

Helger Lipmaa

[1] Cybernetica AS, Estonia
[2] Tallinn University, Estonia

Abstract. It is known that there exists a reduction from the CCA1-security of Damgård's Elgamal (DEG) cryptosystem to what we call the ddh^{dsdh} assumption. We show that ddh^{dsdh} is unnecessary for DEG-CCA1, while DDH is insufficient for DEG-CCA1. We also show that CCA1-security of the Elgamal cryptosystem is equivalent to another assumption ddh^{csdh}, while we show that ddh^{dsdh} is insufficient for Elgamal's CCA1-security. Finally, we prove a generic-group model lower bound $\Omega(\sqrt[3]{q})$ for the hardest considered assumption ddh^{csdh}, where q is the largest prime factor of the group order.

Keywords. CCA1-security, DEG cryptosystem, Elgamal cryptosystem, generic group model, irreduction.

1 Introduction

Of the common security notions of public-key cryptosystems, CPA-security (security against chosen plaintext attacks) is not sufficient in many real-life applications. On the other hand, CCA2-security (security against adaptive chosen ciphertext attacks) is often too strong since it does forbid homomorphic properties that are necessary to efficiently implement many cryptographic protocols. CCA2-secure cryptosystems are also typically less efficient than CPA-secure cryptosystems. CCA1-security (security against nonadaptive chosen ciphertext attacks), a notion that is strictly stronger than CPA-security but does not yet forbid the cryptosystem to be homomorphic, seems to be a reasonable compromise.

In particular, CCA1-secure cryptosystems can be used instead of CPA-secure cryptosystems in many cryptographic protocols (say, e-voting) to achieve better security without any loss in efficiency. For example, one might be able to design an e-voting protocol where a vote cannot be decrypted even by an adversary who can *non-adaptively* (say, before the e-voting period starts) decrypt any ciphertexts of her choosing. We emphasize that while designing such cryptographic protocols, one should still recall that CCA1-security is a strictly weaker assumption than CCA2-security.

Unfortunately, CCA1-security itself has received very little study, and in particular not much is known about CCA1-security of most of the commonly used cryptosystems. As a concrete (and important) example, while the Elgamal cryptosystem [7] is one of the best-known and most efficient (number-theory based) public-key cryptosystems, results on its security have been slow to come. Only in 1998, it was proven that Elgamal is CPA-secure [14]. On the other hand, the Elgamal cryptosystem is clearly not CCA2-secure, because it is homomorphic. However, Elgamal's CCA1-security is a well-known open problem.

In 1991, Damgård proposed what we will call the DEG (Damgård's Elgamal) cryptosystem [4]. DEG is a relatively straightforward modification of Elgamal that employs an additional exponentiation to reject "incorrect" ciphertexts. Damgård proved DEG to be CCA1-secure under a nonfalsifiable [11] knowledge-of-the-exponent assumption. Only in 2006, Gjøsteen [8] proved that DEG is CCA1-secure under a more standard assumption that we will call $\mathrm{ddh}^{\mathrm{dsdh}}$: it basically states that DDH remains secure when the adversary is given a nonadaptive access to the Decisional Static Diffie-Hellman (DSDH) oracle [2]. Gjøsteen's security reduction consisted of a relatively long chain of games. Recently, in an unpublished preprint, Wu and Stinson [15] presented two alternative proofs of the CCA1-security of the DEG cryptosystem. First, they showed that DEG is CCA1-secure if both the DDH assumption and a weaker version of the knowledge-of-exponent assumption (see [15] for precise statement) hold. Second, they presented an alternative proof that it is CCA1-secure under the $\mathrm{ddh}^{\mathrm{dsdh}}$ assumption, which is simpler than Gjøsteen's original proof.

Our contributions. In this paper, we establish the complete complexity landscape of CCA1-security of the Elgamal and the DEG cryptosystems. We establish precise security assumptions under which these cryptosystems are CCA1-secure. To be able to do so, we need to introduce several assumptions where the adversary has a nonadaptive oracle access to an oracle solving a more primitive assumption. Denote by X^Y the assumption that no adversary, given a *nonadaptive* oracle access to the Y oracle, can break the assumption X. Here, since Y is usually a static security assumption [2], it will be assumed that the fixed parameters of Y will be the same as the corresponding parameters in X. As an example, in the $\mathrm{ddh}^{\mathrm{dsdh}}$ assumption, the adversary for the ddh problem has four inputs: a generator g and three group elements h_1, h_2, h_3. The DSDH problem is defined with respect to two fixed group elements g' and h'_1, and the adversary obtains two random group elements h'_2 and h'_3. We will assume that in the $\mathrm{ddh}^{\mathrm{dsdh}}$ assumption, $g' = g$ and $h'_1 = h_1$. For the sake of clarity, we will give full definitions of all three used X^Y-type assumptions later.

All our reductions can be seen as simple hybrid arguments following the general guideline "if $X \Rightarrow X'$ and $Y' \Rightarrow Y$, then $X^Y \Rightarrow (X')^{Y'}$".(Here and in what follows, $X \Rightarrow Y$ means that the assumption Y can be reduced to the assumption X.) Thus all our reductions consist of at most two game hops. Our proof technique, albeit simple, may be a contribution by itself.

Regarding DEG, we first give a simple proof that DEG is CCA1-secure if and only if the $\mathrm{ddeg}^{\mathrm{csdeg}}$ assumption holds, where both csdeg and ddeg are new but standard-looking (falsifiable) assumptions; we will give the precise definition of $\mathrm{ddeg}^{\mathrm{csdeg}}$ in Sect. 3. This result is a tautology which is mainly useful to simplify further results. As for Elgamal, we show that Elgamal is CCA1-secure iff the

$\mathrm{ddh}^{\mathrm{csdh}}$ assumption holds, that is, if ddh is secure given nonadaptive access to a Computational Static Diffie-Hellman (csdh, [2]) oracle. This result is also a tautology. While $\mathrm{ddh}^{\mathrm{csdh}}$ is a new assumption, it is again standard-looking (and falsifiable). We emphasize once more that it is the first known positive result about the CCA1-security of Elgamal at all.

We then concentrate on showing that the used assumptions are all (potentially) different. For this we construct several irreductions [3, 1]. However, due to the nature of the studied problems, our irreductions are not ideally strong, and thus only of (somewhat) indicative nature. Briefly, the problem is that the CCA1-security is a static assumption, where the decryption oracle queries are limited to use the same secret key that is later used for encryption. For this reason, not only the underlying assumptions (like $\mathrm{ddh}^{\mathrm{dsdh}}$) inherit the same property, but also reductions and irreductions. On the one hand, for the underlying assumptions and reductions, this is good: for assumptions, since such static assumptions are weaker than non-static assumptions; for reductions, since static reductions are weaker than non-static reductions. On the other hand, for the irreductions this is bad, since static irreductions are weaker than non-static irreductions (i.e., they only show the nonexistence of static reductions and not all possible reductions). A possible solution here is to strengthen the CCA1-security assumption by allowing the decryption oracle to decrypt with a secret key that corresponds to any public key. This would solve the mentioned problem. However, since such a strengthened version of CCA1-security is nonstandard, we leave its study to a followup work.

We present (static) irreductions showing that ddh cannot be reduced to $\mathrm{ddeg}^{\mathrm{csdeg}}$ (unless ddh is easy), $\mathrm{ddeg}^{\mathrm{csdeg}}$ cannot be reduced to $\mathrm{ddh}^{\mathrm{dsdh}}$ (unless $\mathrm{ddeg}^{\mathrm{csdeg}}$ is easy) and $\mathrm{ddh}^{\mathrm{dsdh}}$ cannot be reduced to the $\mathrm{ddh}^{\mathrm{csdh}}$ (unless $\mathrm{ddh}^{\mathrm{dsdh}}$ is easy). All those irreductions are optimal in the sense that they show that if assumption X can be reduced to Y in polynomial time, then X has to be solvable in polynomial time itself and thus *both* assumptions are broken.

Intuitively, the new irreductions show that DEG is CCA1-secure under an assumption that is strictly stronger than DDH (and thus there is no hope to prove that it is CCA1-secure just under the DDH assumption) and strictly weaker than $\mathrm{ddh}^{\mathrm{dsdh}}$, the assumption under which its CCA1-security was known before. Thus means that the CCA1-security of DEG can be rightfully seen as an independent (and plausible) security assumption, which is a new and possibly surprising result. Moreover, the CCA1-security of Elgamal is based on an assumption that is strictly stronger than the assumption that underlies the CCA1-security of DEG. In a nutshell, this means that while being somewhat less efficient than Elgamal, DEG is "more CCA1-secure" in a well-defined sense.

Finally, we show in the generic group model that the hardest considered assumption, $\mathrm{ddh}^{\mathrm{csdh}}$ (that is, the CCA1-security of Elgamal), is secure in the generic group model [13]. More precisely, we show that any generic group algorithm that breaks $\mathrm{ddh}^{\mathrm{csdh}}$ must take $\Omega(\sqrt[3]{q})$ steps, where q is the largest prime factor of the group order. We prove this lower bound in the generic group model by using the formalization of Maurer [10], but due to the use of *nonadaptive* oracle in our assumption, the proof of lower bound is more involved than any of the proofs in [10]. This can be compared to the known exact lower bound $\Omega(\sqrt{q})$ for ddh (that is, the CPA-security of Elgamal) [13], and shows that $\mathrm{ddh}^{\mathrm{csdh}}$ is likely to be secure (in generic group model) while the defined irreductions are likely to be meaningful due to the different lower bound.

To summarize, we prove that:

$$\boxed{\text{Elgamal-CCA1}} \begin{smallmatrix}\Leftarrow\\ \Rightarrow\end{smallmatrix} \boxed{\mathrm{ddh}^{\mathrm{csdh}}} \begin{smallmatrix}\Leftarrow\\ \not\Rightarrow\,\star\end{smallmatrix} \boxed{\mathrm{ddh}^{\mathrm{dsdh}}} \begin{smallmatrix}\Leftarrow\\ \not\Rightarrow\,\star\end{smallmatrix} \boxed{\mathrm{ddeg}^{\mathrm{csdeg}}} \begin{smallmatrix}\Leftarrow\\ \Rightarrow\end{smallmatrix} \boxed{\text{DEG-CCA1}} \begin{smallmatrix}\Leftarrow\\ \not\Rightarrow\,\star\end{smallmatrix} \boxed{\mathrm{ddh}}$$

Here, we have denoted with a star ($\star$) the new (ir)reductions that are most important in our opinion. We use theorems to prove the starred (ir)reductions, and lemmas to prove other reductions.

Therefore, we give a complete map of the related security reductions and irreductions between these security assumptions. We stress that irreductions are not yet commonly used, and we hope that the current paper provides an insight to their significance. (And shows, that they are often *not* difficult to construct.)

Recent Related Work. First, a number of recent papers [6, 5, 9] have studied the CCA1/CCA2-security of *hybrid* versions of the DEG cryptosystem. Such versions use additional cryptographic primitives like symmetric encryption and MAC. Compared to them, nonhybrid versions studied in this paper are both better known and simpler. Moreover, the study of nonhybrid versions is important because they are homomorphic and thus widely usable in cryptographic protocols. Second, in an unpublished preprint [15], Wu and Stinson also show that the Elgamal cryptosystem is one-way (under nonadaptive chosen ciphertext attacks) under two different conditions. They did *not* study the CCA1-security of Elgamal.

2 Preliminaries

2.1 Assumptions

Let the value of the predicate $[a \stackrel{?}{=} b]$ be 1, if $a = b$, and 0 otherwise. In the case of any security assumption X, we let the public variables $(X_1, \dots, X_m)$ be all variables seen by the adversary (in a fixed order implicit in the definition). In the cases that we study in this paper, the first public variables are system parameters (like a generator of the underlying group), then the public key and finally the variables sent to the adversary during the security game.

Denote

$$\mathrm{cdh}(g, g^x, g^y) := g^{xy} \quad , \quad \mathrm{ddh}(g, g^x, g^y, g^z) := [g^z \stackrel{?}{=} \mathrm{cdh}(g, g^x, g^y)] \enspace .$$

Based on these standard cdh and ddh oracles, we also define the Computational and Decisional Static Diffie-Hellman oracles [2]:

$$\begin{aligned} \mathrm{csdh}_{(g,g^x)}(g^y) &:= \mathrm{cdh}(g, g^x, g^y) = g^{xy} \enspace , \\ \mathrm{dsdh}_{(g,g^x)}(g^y, g^z) &:= \mathrm{ddh}(g, g^x, g^y, g^z) = [g^z \stackrel{?}{=} \mathrm{cdh}(g, g^x, g^y)] \enspace . \end{aligned}$$

Note that cdh and csdh are essentially the same functions, but as oracles they behave differently since (g, g^x) have been hardcoded in csdh and thus cannot be chosen by the adversary. The same comment is true for ddh and dsdh.

Fix a group $\mathbb{G} = \langle g \rangle$ of order q. The *ddh game* is defined as follows:

Setup phase. Challenger sets $\mathrm{sk} \leftarrow \mathbb{Z}_q$, $\mathrm{pk} \leftarrow g^{\mathrm{sk}}$. He sends pk to adversary $\mathcal{A}$.
Challenge phase. Challenger sets $b_{\mathcal{A}} \leftarrow \{0,1\}$, $y^* \leftarrow \mathbb{Z}_q$, $z^* \leftarrow \mathbb{Z}_q$, $h_1^* \leftarrow g^{y^*}$. He sets $h_2^* \leftarrow g^{z^*}$ if $b_{\mathcal{A}} = 0$ and $h_2^* = \mathrm{pk}^{y^*}$ if $b_{\mathcal{A}} = 1$. Challenger sends (h_1^*, h_2^*) to $\mathcal{A}$.
Guess phase. $\mathcal{A}$ returns a bit $b'_{\mathcal{A}} \in \{0,1\}$. $\mathcal{A}$ wins if $b'_{\mathcal{A}} = b_{\mathcal{A}}$, that is, if $b'_{\mathcal{A}} = \mathrm{ddh}(g, \mathrm{pk}, h_1^*, h_2^*)$.

Group $\mathbb{G}$ is a (τ, ε)-*ddh group* if for any adversary $\mathcal{A}$ working in time τ, $\Pr[\mathcal{A}$ wins in the ddh game$] \leq \frac{1}{2} + \varepsilon$. Note that the public variables are $\mathrm{ddh}_1 = g$, $\mathrm{ddh}_2 = \mathrm{pk}$, $\mathrm{ddh}_3 = h_1^*$, $\mathrm{ddh}_4 = h_2^*$.

For comparison, we now give a complete description of a static game, csdh. Fix a group $\mathbb{G} = \langle g \rangle$ of order q, a generator g of group $\mathbb{G}$, and a random $\mathrm{pk} \leftarrow \mathbb{G}$. The *csdh*$_{(g,\mathrm{pk})}$ *game* is defined as follows:

Challenge phase. Challenger sets $x_{\mathcal{A}} \leftarrow \mathbb{Z}_q$. He sets $h \leftarrow g^{x_{\mathcal{A}}}$. Challenger sends h to $\mathcal{A}$.
Guess phase. $\mathcal{A}$ returns a group element $h'_{\mathcal{A}} \in \mathbb{G}$. $\mathcal{A}$ wins if $h'_{\mathcal{A}} = \mathrm{csdh}_{g,\mathrm{pk}}(h) = \mathrm{pk}^{x_{\mathcal{A}}}$.

Group $\mathbb{G}$ is a (τ, ε)-*csdh group* if for any g, pk and any adversary $\mathcal{A}$ working in time τ, $\Pr[\mathcal{A}$ wins in the csdh game$] \leq \frac{1}{q} + \varepsilon$.

Based on arbitrary assumptions X and Y we define a new assumption X^Y. In the X^Y game, an adversary has *nonadaptive* oracle access to an oracle solving assumption Y, and she has to break a random instance of the X assumption. In our case, the Y assumption is always a static assumption, that is, it is defined with respect to some public parameters that come from the instance that the adversary for X^Y has to solve. Note that if Y is static, then we have to always fix the public parameters in the definition of X and Y. Clearly, $X^Y \Rightarrow (X')^{Y'}$ when $X \Rightarrow X'$ and $Y' \Rightarrow Y$. This can be proven by using a hybrid argument, showing say that $X^Y \Rightarrow X^{Y'}$, that $X^{Y'} \Rightarrow (X')^{Y'}$, etc. A group is (τ, ε)-X^Y *group* if for any adversary $\mathcal{A}$ working in time τ, $\Pr[\mathcal{A}$ wins in the X^Y game$] \leq \delta + \varepsilon$, where $\delta = \frac{1}{2}$ in a decisional assumption, and $\delta = \frac{1}{q}$ in a computational assumption.

For the sake of clarity, we now give a precise definition of the $\mathrm{ddh}^{\mathrm{dsdh}}$ game, and we state its relation to some of the existing assumptions. Similarly, we will later define all other used assumptions. Fix a group $\mathbb{G} = \langle g \rangle$ of order q. The *ddh*^{dsdh} *game* is defined as follows:

Setup phase. Challenger sets $\mathrm{sk} \leftarrow \mathbb{Z}_q$, $\mathrm{pk} \leftarrow g^{\mathrm{sk}}$. He sends pk to adversary $\mathcal{A}$.
Query phase. $\mathcal{A}$ has a (nonadaptive) access to oracle $\mathrm{dsdh}_{(g,\mathrm{pk})}(\cdot,\cdot)$.
Challenge phase. Challenger sets $b_{\mathcal{A}} \leftarrow \{0,1\}$, $y^*, z^* \leftarrow \mathbb{Z}_q$, $h_1^* \leftarrow g^{y^*}$. He sets $h_2^* \leftarrow g^{z^*}$ if $b_{\mathcal{A}} = 0$ and $h_2^* \leftarrow \mathrm{pk}^{y^*} = \mathrm{csdh}_{(g,\mathrm{pk})}(h_1^*)$ if $b_{\mathcal{A}} = 1$. Challenger sends (h_1^*, h_2^*) to $\mathcal{A}$.

Guess phase. $\mathcal{A}$ returns a bit $b'_{\mathcal{A}} \in \{0,1\}$. $\mathcal{A}$ wins if $b'_{\mathcal{A}} = b_{\mathcal{A}}$, that is, if $b'_{\mathcal{A}} = \mathrm{ddh}(g, \mathrm{pk}, h^*_1, h^*_2)$.

Group $\mathbb{G}$ is a (τ, ε)-*ddh*^{dsdh} *group* if for any adversary $\mathcal{A}$ working in time τ, $\Pr[\mathcal{A}$ wins in the $\mathrm{ddh}^{\mathrm{dsdh}}$ game$] \leq \frac{1}{2} + \varepsilon$. Here, the 2 variables are g and pk are shared by the ddh oracle invoked in the query phase and by the instance the adversary is trying to solve.

Several versions of the X^Y game for different values of X and Y, have been used before. $\mathrm{ddh}^{\mathrm{dsdh}}$ assumption has been used before say in [8]. The gap DH assumption of [12] is similar to $\mathrm{cdh}^{\mathrm{dsdh}}$ (defined later), except that there the adversary gets access to the oracle also after seeing the challenge. Some other papers deal with the so called *one-more DDH* assumption, where $\mathcal{A}$ has to answer correctly to $t+1$ DDH challenges after making only t DDH queries. See, for example, [3].

2.2 Reductions And Irreductions

We say that security assumption Y can be reduced to assumption X, $X \Rightarrow Y$, if there exists a *reduction* $\mathcal{R}$, such that: for every adversary $\mathcal{A}$ that breaks assumption X, $\mathcal{R}$ can break assumption Y by using $\mathcal{A}$ as an oracle. More precisely, in an $X \Rightarrow Y$ reduction game, the challenger $\mathcal{C}$ generates for $\mathcal{R}$ the public parameters of an Y instance. Then, the challenger sends to $\mathcal{R}$ a challenge of the game Y that $\mathcal{R}$ has to solve. $\mathcal{R}$ can use $\mathcal{A}$ as an oracle.

Following [3], we call an algorithm $\mathcal{I}$ an *irreduction* $Z \not\Rightarrow_Y X$, if it can, given as an oracle an arbitrary reduction algorithm $Z \Rightarrow X$, solve problem Y. If $Y = X$, then we say that $\mathcal{I}$ is an *optimal irreduction* algorithm and write $Z \not\Rightarrow_! X$. More precisely, in a $Z \not\Rightarrow_Y X$ irreduction game, the challenger $\mathcal{C}$ generates for $\mathcal{I}$ the public parameters of an Y instance. Then, the challenger sends to $\mathcal{I}$ a challenge of the game Y that $\mathcal{I}$ has to solve. $\mathcal{I}$ can use a reduction $\mathcal{R}$ of the game $Z \Rightarrow X$ as an oracle.

Now, in our case, most of the assumptions are static in nature. That is, we either have an assumption X_α with some externally given variables α, or an assumption $X^{Y_{\alpha,\beta}}_{\alpha,\beta'}$, where variables α of the X's instance are fixed in the invocation of the oracle for Y. (To simplify the notation, we will usually not write down α, β and β', but define them while defining the static assumption X^Y.) Analogously, in a *static* reduction $X^Y \Rightarrow (X')^{Y'}$ game, the adversary in the $X^Y \Rightarrow (X')^{Y'}$ game (1) has to know how to answer the Y queries only when some variables are fixed, and (2) can only query the oracle that solves X^Y or Y' under some fixed variables. Analogously, an adversary in the irreduction $X^Y \not\Rightarrow_Z (X')^{Y'}$ game has similar restrictions.

Briefly, the problem is that the CCA1-security is a static assumption, where the decryption oracle queries are limited to use the same secret key that is later used for encryption. For this reason, not only the underlying assumptions (like $\mathrm{ddh}^{\mathrm{dsdh}}$) inherit the same property, but also reductions and irreductions. This is since in the reduction and irreduction games, some of the oracles are equal to the decryption oracle (or to some other static oracle). On the one hand, for the underlying assumptions and reductions, this is good: for assumptions, since such static assumptions are weaker than non-static assumptions; for reductions, since static reductions are weaker than non-static reductions. On the other hand, for the irreductions this is bad, since static irreductions are weaker than non-static irreductions (i.e., they only show the nonexistence of static reductions and not all possible reductions).

Thus, In our (ir)reductions, it is important to see which variables are fixed in all X, Y, and Z. For example, in an reduction $X \Rightarrow Z$, both instances X and Z may depend on some public generator g and public key pk. In all our reductions, the reduction algorithm only uses the oracle $\mathcal{A}$ with all public parameters and public keys being fixed. We say that such a reduction is *static*. Analogously, we say that an irreduction $Z \not\Rightarrow_Y X$ is *static*, if its oracle reduction algorithm is static. To make this completely clear, we state the names of fixed parameters in all of our results. We refer to the beginning of Sec. 5 for further discussion.

Finally, when we show the existence of a reduction (resp., irreduction), we construct a reduction $\mathcal{R}$ (resp., irreduction $\mathcal{I}$) that simulates the challenger $\mathcal{C}$ to adversary $\mathcal{A}$ (resp., reduction $\mathcal{R}$). In the case of an irreduction, $\mathcal{I}$ also simulates $\mathcal{A}$ to $\mathcal{R}$. If a party $\mathcal{X}$ simulates party $\mathcal{Y}$, then we denote $\mathcal{X}$ as $\mathcal{X}[\mathcal{Y}]$ for the sake of clarity.

2.3 Cryptosystems

A public-key cryptosystem Π is a triple of efficient algorithms (G, E, D), where $G(1^k)$ outputs a key pair $(\mathrm{sk}, \mathrm{pk})$, $E_{\mathrm{pk}}(m; r)$ returns a ciphertext and $D_{\mathrm{sk}}(c)$ returns a plaintext, so that $D_{\mathrm{sk}}(E_{\mathrm{pk}}(m; r)) = m$ for any $(\mathrm{sk}, \mathrm{pk}) \in G(1^k)$. Here, k is a security parameter that we will just handle as a constant.

Fix a cyclic group $\mathbb{G} = \langle g \rangle$ of order q. The *Elgamal cryptosystem* [7] in group $\mathbb{G}$ is defined as follows:

Key generation $G(1^k)$**.** Select a random $\mathrm{sk} \leftarrow \mathbb{Z}_q$, set $\mathrm{pk} \leftarrow g^{\mathrm{sk}}$. Publish pk.

Encryption $E_{\mathrm{pk}}(m; \cdot)$**.** Return $\perp$ if $m \notin \mathbb{G}$. Otherwise, select a random $r \leftarrow \mathbb{Z}_q$, set $E_{\mathrm{pk}}(m; r) \leftarrow (g^r, m \cdot \mathrm{pk}^r)$.

Decryption $D_{\mathrm{sk}}(c)$**.** Parse $c = (c_1, c_2)$, return $\perp$ if $c_i \notin \mathbb{G}$ for some i. Otherwise, return $D_{\mathrm{sk}}(c) \leftarrow c_2 / c_1^{\mathrm{sk}}$.

Fix a group $\mathbb{G} = \langle g \rangle$ of order q. The *Damgård's Elgamal (DEG) cryptosystem [4]* in group $\mathbb{G}$ is defined as follows:

Key generation $G(1^k)$**.** Select random $\mathrm{sk}_1, \mathrm{sk}_2 \leftarrow \mathbb{Z}_q$, set $\mathrm{pk}_1 \leftarrow g^{\mathrm{sk}_1}, \mathrm{pk}_2 \leftarrow g^{\mathrm{sk}_2}$. Publish $\mathrm{pk} \leftarrow (\mathrm{pk}_1, \mathrm{pk}_2)$, set $\mathrm{sk} \leftarrow (\mathrm{sk}_1, \mathrm{sk}_2)$.

Encryption $E_{\mathrm{pk}}(m; \cdot)$**.** Return $\perp$ if $m \notin \mathbb{G}$. Otherwise, select a random $r \leftarrow \mathbb{Z}_q$, set $E_{\mathrm{pk}}(m; r) \leftarrow (g^r, \mathrm{pk}_1^r, m \cdot \mathrm{pk}_2^r)$.

Decryption $D_{\mathrm{sk}}(c)$**.** Parse $c = (c_1, c_2, c_3)$, return $\perp$ if $c_i \notin \mathbb{G}$ for some i. Return $\perp$ if $c_2 \neq c_1^{\mathrm{sk}_1}$. Otherwise, return $D_{sk}(c) \leftarrow c_3 / c_1^{\mathrm{sk}_2}$.

Let $\Pi = (G, E, D)$ be a public-key cryptosystem. The *CCA1-game* for Π is defined as follows:

Setup phase. Challenger chooses $(\mathrm{sk}, \mathrm{pk}) \leftarrow G(1^k)$ and sends pk to adversary $\mathcal{A}$.
Query phase. $\mathcal{A}$ has access to an oracle $D_{\mathrm{sk}}(\cdot)$.
Challenge phase. $\mathcal{A}$ submits (m_0, m_1) to the challenger, who picks a random bit $b_{\mathcal{A}} \leftarrow \{0, 1\}$ and a random $r \leftarrow \mathbb{Z}_q$, and returns $E_{\mathrm{pk}}(m_{b_{\mathcal{A}}}; r)$.
Guess phase. $\mathcal{A}$ returns a bit $b'_{\mathcal{A}} \in \{0, 1\}$. $\mathcal{A}$ wins if $b'_{\mathcal{A}} = b_{\mathcal{A}}$.

A public-key cryptosystem is $(\tau, \gamma, \varepsilon)$-*CCA1-secure* if for any adversary $\mathcal{A}$ working in time τ and making γ queries, $\Pr[\mathcal{A}$ wins in the CCA1-game$] \leq \frac{1}{2} + \varepsilon$. A $(\tau, 0, \varepsilon)$-CCA1-secure cryptosystem is also said to be (τ, ε)-*CPA-secure*. Note that CCA1-security is an explicitly static assumption, since the adversary can only access the decryption oracle with respect to a fixed secret key.

The DEG cryptosystem was proven to be CCA1-secure under the $\mathrm{ddh}^{\mathrm{dsdh}}$ assumption in [8]. More precisely, Gjøsten proved the CCA1-security of a (hash-proof based) generalization of the DEG cryptosystem under a generalization of the $\mathrm{ddh}^{\mathrm{dsdh}}$ assumption. Elgamal's cryptosystem is known to be CPA-secure [14] but not known to be CCA1-secure for $\gamma = \mathrm{poly}(k)$.

3 CCA1-Security of DEG

In this section we investigate the CCA1-security of DEG.

3.1 DEG Is CCA1-Secure ⇔ $\mathrm{ddeg}^{\mathrm{csdeg}}$

First, we prove that the security of DEG is equivalent to a new but standard-looking assumption $\mathrm{ddeg}^{\mathrm{csdeg}}$. This result itself is not so interesting, but combined with the result from the next subsection it will provide a reduction of the CCA1-security of DEG to the more standard (but as we will also see later, a likely stronger) $\mathrm{ddh}^{\mathrm{dsdh}}$ assumption.

The $\mathrm{ddeg}^{\mathrm{csdeg}}$ Assumption. We first define the new assumption. For implicitly defined $g, \mathrm{pk}_1, \mathrm{pk}_2$, let $\mathcal{DEG}_0 := \{(g^y, \mathrm{pk}_1^y, \mathrm{pk}_2^z) : y, z \leftarrow \mathbb{Z}_q\}$ and $\mathcal{DEG}_1 := \{(g^y, \mathrm{pk}_1^y, \mathrm{pk}_2^y) : y \leftarrow \mathbb{Z}_q\}$. Define the next oracles $\mathrm{csdeg}_{(\cdot,\cdot,\cdot)}$ and ddeg:

– $\mathrm{csdeg}_{(g,\mathrm{pk}_1,\mathrm{pk}_2)}(h_1, h_2)$ first checks if $\mathrm{ddh}(g, \mathrm{pk}_1, h_1, h_2) = 1$. If this is not true, it returns $\perp$. Otherwise, it returns $h_3 \leftarrow \mathrm{cdh}(g, \mathrm{pk}_2, h_1)$.
– $\mathrm{ddeg}(g, \mathrm{pk}_1, \mathrm{pk}_2, h_1, h_2, h_3)$ has to distinguish between $\mathcal{DEG}_0$ and $\mathcal{DEG}_1$. That is, on the promise that $\mathrm{ddh}(g, \mathrm{pk}_1, h_1, h_2) = 1$, $\mathrm{ddeg}(g, \mathrm{pk}_1, \mathrm{pk}_2, h_1, h_2, h_3) \leftarrow [\mathrm{ddh}(g, \mathrm{pk}_2, h_1, h_3) \overset{?}{=} 1]$. The oracle is not required to output anything if $\mathrm{ddh}(g, \mathrm{pk}_1, h_1, h_2) = 0$.

Fix a group $\mathbb{G} = \langle g \rangle$ of order q. The *$\mathrm{ddeg}^{\mathrm{csdeg}}$ game* in group $\mathbb{G}$ is defined as follows:

Setup phase. Challenger sets $\mathrm{sk}_1, \mathrm{sk}_2 \leftarrow \mathbb{Z}_q$, $\mathrm{pk}_1 \leftarrow g^{\mathrm{sk}_1}$, $\mathrm{pk}_2 \leftarrow g^{\mathrm{sk}_2}$. He sends $\mathrm{pk} \leftarrow (\mathrm{pk}_1, \mathrm{pk}_2)$ to adversary $\mathcal{A}$, and sets $\mathrm{sk} \leftarrow (\mathrm{sk}_1, \mathrm{sk}_2)$.

Query phase. $\mathcal{A}$ has access to the oracle $\mathrm{csdeg}_{(g,\mathrm{pk}_1,\mathrm{pk}_2)}(\cdot,\cdot)$.
Challenge phase. Challenger sets $b_{\mathcal{A}} \leftarrow \{0,1\}$, $y^*, z^* \leftarrow \mathbb{Z}_q$, $h_1^* \leftarrow g^{y^*}$, and $h_2^* \leftarrow \mathrm{pk}_1^{y^*}$. If $b_{\mathcal{A}} = 0$, then $h_3^* \leftarrow \mathbb{G}$. If $b_{\mathcal{A}} = 1$, then $h_3^* \leftarrow \mathrm{pk}_2^{y^*}$. Challenger sends (h_1^*, h_2^*, h_3^*) to $\mathcal{A}$.
Guess phase. $\mathcal{A}$ returns a bit $b'_{\mathcal{A}} \in \{0,1\}$. $\mathcal{A}$ wins if $b'_{\mathcal{A}} = b_{\mathcal{A}}$.

Group $\mathbb{G}$ is a $(\tau, \gamma, \varepsilon)$-*ddeg*^{csdeg} *group* if for any adversary $\mathcal{A}$ working in time τ and making γ queries, $\Pr[\mathcal{A} \text{ wins}] \leq \frac{1}{2} + \varepsilon$. Note that this definition does directly follow from the definition of the $\mathrm{csdeg}_{(\cdot,\cdot,\cdot)}$ and ddeg oracles.

Security Results. In all next results, small denotes some unspecified small value (usually $O(1)$ group operations) that is dominated by some other addend in the same formula. The next lemma is basically a tautology, and useful mostly to simplify further proofs.

Lemma 1 (DEG-CCA1 $\Leftrightarrow$ ddeg$^{\mathbf{csdeg}}$). *Fix a group $\mathbb{G} = \langle g \rangle$ of order q.*
(1) Assume that $\mathbb{G}$ is a $(\tau, \gamma, \varepsilon)$-ddeg^{csdeg} group. Then DEG is $(\tau - \gamma \cdot (\tau_{\mathrm{csdeg}} + \mathsf{small}) - \mathsf{small}, \gamma, 2\varepsilon)$-CCA1-secure where τ_{csdeg} is the working time of the $\mathrm{csdeg}_{(\cdot,\cdot,\cdot)}$ *oracle.*
(2) Assume that DEG is $(\tau, \gamma, \varepsilon)$-CCA1-secure. Then $\mathbb{G}$ is a $(\tau - \gamma \cdot (\tau_D + \mathsf{small}) - \mathsf{small}, \gamma, \varepsilon)$-ddeg^{csdeg} group, where τ_D is the working time of the decryption oracle D.

Proof. 1) **First direction (DEG-CCA1 $\Rightarrow$ ddeg$^{\mathbf{csdeg}}$ with fixed $(g, \mathrm{pk}_1, \mathrm{pk}_2)$):** Assume $\mathcal{A}$ is an adversary who can $(\tau', \gamma', \varepsilon')$-break the CCA1-security of DEG with probability ε' and in time τ', making γ' queries. Construct the next reduction $\mathcal{R}$ that aims to break ddeg$^{\mathrm{csdeg}}$:

- Challenger generates new $\mathrm{sk} \leftarrow (\mathrm{sk}_1, \mathrm{sk}_2) \leftarrow \mathbb{Z}_q^2$, $\mathrm{pk}_1 \leftarrow g^{\mathrm{sk}_1}$, $\mathrm{pk}_2 \leftarrow g^{\mathrm{sk}_2}$ and sends $\mathrm{pk} \leftarrow (\mathrm{pk}_1, \mathrm{pk}_2)$ to $\mathcal{R}$. $\mathcal{R}$ forwards pk to $\mathcal{A}$.
- In the query phase, whenever $\mathcal{A}$ asks a decryption D_{sk} query (c_1, c_2, c_3) from $D_{\mathrm{sk}}(\cdot,\cdot,\cdot)$, $\mathcal{R}$ rejects if either c_1, c_2 or c_3 is not a valid group element. Otherwise $\mathcal{R}$ makes a $\mathrm{csdeg}_{(g,\mathrm{pk}_1,\mathrm{pk}_2)}(c_1, c_2)$ query. $\mathcal{R}$ receives a c' such that $c' \leftarrow \perp$, if $c_2 \neq c_1^{\mathrm{sk}_1}$, and $c' \leftarrow c_1^{\mathrm{sk}_2}$ otherwise. $\mathcal{R}$ returns $\perp$ in the first case, and c_3/c' in the second case.
- In the challenge phase, whenever $\mathcal{A}$ submits her challenge (m_0^*, m_1^*), $\mathcal{R}$ asks the challenger for his own challenge. The challenger sets $b_{\mathcal{R}} \leftarrow \{0,1\}$, $y^* \leftarrow \mathbb{Z}_q$, $h_1^* \leftarrow g^{y^*}$, $h_2^* \leftarrow \mathrm{pk}_1^{y^*}$. If $b_{\mathcal{R}} = 0$, then he sets $h_3^* \leftarrow \mathbb{G}$, otherwise $h_3^* \leftarrow \mathrm{pk}_2^{y^*}$. $\mathcal{R}$ picks a random bit $b_{\mathcal{A}} \leftarrow \{0,1\}$, and sends $(h_1^*, h_2^*, m_{b_{\mathcal{A}}}^* \cdot h_3^*)$ to $\mathcal{A}$. $\mathcal{A}$ returns a bit $b'_{\mathcal{A}}$.
- In the guess phase, if $b'_{\mathcal{A}} = b_{\mathcal{A}}$, then $\mathcal{R}$ returns $b'_{\mathcal{R}} \leftarrow 1$, otherwise $\mathcal{R}$ returns $b'_{\mathcal{R}} \leftarrow 0$.

Now,$\Pr[\mathcal{R} \text{ wins}] = \Pr[b'_{\mathcal{R}} = b_{\mathcal{R}}] = \Pr[\mathcal{A} \text{ wins} | b_{\mathcal{R}} = 1] \cdot \Pr[b_{\mathcal{R}} = 1] + \Pr[\mathcal{A} \text{ wins} | b_{\mathcal{R}} = 0] \cdot \Pr[b_{\mathcal{R}} = 0] = \left(\frac{1}{2} + \varepsilon'\right) \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} + \frac{\varepsilon'}{2}$. Clearly $\mathcal{R}$ works in time $\tau = \tau' + \gamma \cdot (\tau_{\mathrm{csdeg}} + \mathsf{small}) + \mathsf{small}$. □

2) **Second direction ($\mathbf{ddeg}^{\mathbf{csdeg}} \Rightarrow$ DEG-CCA1 with fixed $(g, \mathrm{pk}_1, \mathrm{pk}_2)$):** Assume $\mathcal{A}$ is an adversary who can $(\tau', \gamma', \varepsilon')$-break the $\mathrm{ddeg}^{\mathrm{csdeg}}$ assumption. Construct the next reduction $\mathcal{R}$ that aims to break the CCA1-security of the DEG cryptosystem:

- Challenger generates new $\mathrm{sk} \leftarrow (\mathrm{sk}_1, \mathrm{sk}_2) \leftarrow \mathbb{Z}_q^2$, $\mathrm{pk}_1 \leftarrow g^{\mathrm{sk}_1}$, $\mathrm{pk}_2 \leftarrow g^{\mathrm{sk}_2}$, and sends $\mathrm{pk} = (\mathrm{pk}_1, \mathrm{pk}_2)$ to $\mathcal{R}$. $\mathcal{R}$ forwards pk to $\mathcal{A}$.
- In the query phase, whenever $\mathcal{A}$ asks a query $\mathrm{csdeg}_{(g, \mathrm{pk}_1, \mathrm{pk}_2)}(h_1, h_2)$, $\mathcal{R}$ makes a decryption D_{sk} query $(h_1, h_2, 1)$, and receives back either $\perp$ or $k \leftarrow h_1^{-\mathrm{sk}_2}$. $\mathcal{R}$ returns $h_3 \leftarrow \perp$ in the first case, and $h_3 \leftarrow k^{-1}$ in the second case.
- In the challenge phase, whenever $\mathcal{A}$ asks for a challenge, $\mathcal{R}$ sends his challenge pair $(m_0^*, m_1^*) \leftarrow (g^{r_1^*}, 1)$, for $r_1^* \leftarrow \mathbb{Z}_q$, to the challenger. Challenger picks a random bit $b_{\mathcal{R}} \leftarrow \{0, 1\}$ and a random $r_2^* \leftarrow \mathbb{Z}_q$, and sends $(c_1^*, c_2^*, c_3^*) \leftarrow (g^{r_2^*}, \mathrm{pk}_1^{r_2^*}, g^{r_1^*(1-b_{\mathcal{R}})} \cdot \mathrm{pk}_2^{r_2^*})$ to $\mathcal{R}$. $\mathcal{R}$ forwards (c_1^*, c_2^*, c_3^*) to $\mathcal{A}$, who returns a guess $b'_{\mathcal{A}}$.
- In the guess phase, $\mathcal{R}$ returns $b'_{\mathcal{R}} \leftarrow b'_{\mathcal{A}}$ to challenger.

Now, $\Pr[\mathcal{R} \text{ wins}] = \Pr[b'_{\mathcal{R}} = b_{\mathcal{R}}] = \Pr[\mathcal{A} \text{ wins}] = \varepsilon'$. Clearly $\mathcal{R}$ works in time $\tau' + \gamma \cdot (\tau_D + \mathsf{small}) + \mathsf{small}$. □

Lemma 2 (DEG-CCA1 $\Rightarrow$ $\mathbf{ddh}^{\mathbf{dsdh}}$ with fixed $(g, \mathrm{pk} = \mathrm{pk}_1)$).
(1) Assume that $\mathbb{G} = \langle g \rangle$ is a $(\tau, \gamma, \varepsilon)$-$ddh^{dsdh}$ group. Then the DEG cryptosystem is CCA1-secure in group $\mathbb{G}$.
(2) Any ddh^{dsdh} group $\mathbb{G} = \langle g \rangle$ is also a $ddeg^{csdeg}$ group.

Proof. Proof of the first claim is given in [8, 15]. The second claim follows from the first claim and Lem. 1. □

By following a very similar proof, a variant of the DEG cryptosystem where the decryption, given an invalid ciphertext, returns a random plaintext instead of $\perp$, is CCA1-secure under the ddh assumption.

Relation with ddh. It is obviously important to establish the relationships of the new assumptions with the well-known assumptions like ddh. Here we construct a static reduction ddh $\Rightarrow$ $\mathrm{ddeg}^{\mathrm{csdeg}}$ and in Thm. 1, we construct a static irreduction $\mathrm{ddeg}^{\mathrm{csdeg}} \not\Rightarrow_! \mathrm{ddh}$. As a careful reader will observe, in fact both the reduction and the irreduction will be to the static version of ddh, where the first three inputs $(g, \mathrm{pk}_1, \mathrm{pk}_2)$ are fixed, and the adversary can only choose the four inputs. This static version of ddh is clearly at least as strong as ddh since anybody who can break the static version can also break the ddh.

Lemma 3 (ddh $\Rightarrow$ $\mathbf{ddeg}^{\mathbf{csdeg}}$ with fixed $(g, \mathrm{pk}_1, \mathrm{pk}_2)$). *Any $(\tau, \gamma, \varepsilon)$-$ddeg^{csdeg}$ group $\mathbb{G} = \langle g \rangle$ is also a $(\tau - \mathsf{small}, \varepsilon)$-ddh group.*

Proof. Fix a group $\mathbb{G} = \langle g \rangle$ of order q. Assume $\mathcal{A}$ is an adversary who can $(\tau', \gamma', \varepsilon')$-break the ddh assumption. Construct the next reduction $\mathcal{R}$ that aims to break $\mathrm{ddeg}^{\mathrm{csdeg}}$ in the same group:

– Challenger generates new $(\mathrm{sk}_1 \leftarrow \mathbb{Z}_q, \mathrm{sk}_2 \leftarrow \mathbb{Z}_q, \mathrm{pk}_1 \leftarrow g^{\mathrm{sk}_1}, \mathrm{pk}_2 \leftarrow g^{\mathrm{sk}_2})$ and sends $\mathrm{pk} = (\mathrm{pk}_1, \mathrm{pk}_2)$ to $\mathcal{R}$. $\mathcal{R}$ forwards (g, pk_2) to $\mathcal{A}$ as her system parameters.
– In the challenge phase, if $\mathcal{A}$ asks for a challenge, then $\mathcal{R}$ asks for a challenge. Challenger sets $b_{\mathcal{R}} \leftarrow \{0,1\}$, $y^* \leftarrow \mathbb{Z}_q$, $h_1^* \leftarrow g^{y^*}$, $h_2^* \leftarrow \mathrm{pk}_1^{y^*}$. If $b_{\mathcal{R}} = 0$, then he sets $h_3^* \leftarrow \mathbb{G}$, otherwise $h_3^* \leftarrow \mathrm{pk}_2^{y^*}$. He sends (h_1^*, h_2^*, h_3^*) to $\mathcal{R}$. $\mathcal{R}$ sends (h_1^*, h_3^*) to $\mathcal{A}$. $\mathcal{A}$ returns a bit $b'_{\mathcal{A}}$. $\mathcal{R}$ returns $b'_{\mathcal{R}} \leftarrow b'_{\mathcal{A}}$ to the challenger.

Clearly, $\mathcal{R}$ wins if and only if $\mathcal{A}$ wins. □

4 CCA1-Security of ElGamal

To prove the security of ElGamal we need the next assumption. As we will see from the security proofs, this assumption basically just asserts that Elgamal is CCA1-secure.

Fix a group $\mathbb{G} = \langle g \rangle$ of order q. The *ddh^{csdh} game* is defined as follows:

Setup phase. Challenger sets $\mathrm{sk} \leftarrow \mathbb{Z}_q$, $\mathrm{pk} \leftarrow g^{\mathrm{sk}}$. He sends pk to adversary $\mathcal{A}$.
Query phase. $\mathcal{A}$ has access to oracle $\mathrm{csdh}_{(g,\mathrm{pk})}(\cdot)$, that is, $\mathrm{csdh}_{(g,\mathrm{pk})}(h) := h^{\mathrm{sk}}$.
Challenge phase. Challenger sets $b_{\mathcal{A}} \leftarrow \{0,1\}$, $y^* \leftarrow \mathbb{Z}_q$, $h_1^* \leftarrow g^{y^*}$. He sets $h_2^* \leftarrow \mathbb{G}$ if $b_{\mathcal{A}} = 0$ and $h_2^* \leftarrow \mathrm{pk}^{y^*} = \mathrm{csdh}_{(g,\mathrm{pk})}(h_1^*)$ if $b_{\mathcal{A}} = 1$. Challenger sends (h_1^*, h_2^*) to $\mathcal{A}$.
Guess phase. $\mathcal{A}$ returns a bit $b'_{\mathcal{A}} \in \{0,1\}$. $\mathcal{A}$ wins if $b'_{\mathcal{A}} = b_{\mathcal{A}}$, that is, if $b_{\mathcal{A}} = \mathrm{ddh}(g, \mathrm{pk}, h_1^*, h_2^*)$.

Group $\mathbb{G}$ is a $(\tau, \gamma, \varepsilon)$-*ddh^{csdh} group* if for any adversary $\mathcal{A}$ working in time τ and making γ queries, $\Pr[\mathcal{A}$ wins in the $\mathrm{ddh}^{\mathrm{csdh}}$ game$] \leq \frac{1}{2} + \varepsilon$.

Lemma 4 (Elgamal-CCA1 ⇔ ddh$^{\mathbf{csdh}}$ with fixed (g, pk)). *Fix a group $\mathbb{G} = \langle g \rangle$ of order q.*
(1) Assume that $\mathbb{G}$ is a $(\tau, \gamma, \varepsilon)$-ddh^{csdh} group. Then ElGamal is $(\tau - \gamma \cdot (\tau_{\mathrm{csdh}} + \mathsf{small}) - \mathsf{small}, \gamma, 2\varepsilon)$-CCA1-secure, where τ_{csdh} is the working time of the $\mathrm{csdh}_{(g,\mathrm{pk})}(\cdot)$ oracle.
(2) Assume that ElGamal is $(\tau, \gamma, \varepsilon)$-CCA1-secure. Then $\mathbb{G}$ is a $(\tau - \gamma \cdot (\tau_D + \mathsf{small}) - \mathsf{small}, \gamma, \varepsilon)$-ddh^{csdh} group, where τ_D is the working time of the D oracle.

Proof. 1) **First direction (Elgamal-CCA1 ⇒ ddh$^{\mathbf{csdh}}$ with fixed (g, pk)):** Assume $\mathcal{A}$ is an adversary who can $(\tau', \gamma', \varepsilon')$-break the CCA1-security of Elgamal in group $\mathbb{G}$ with probability ε' and in time τ', making γ' queries. Construct the next reduction $\mathcal{R}$ that aims to break $\mathrm{ddh}^{\mathrm{csdh}}$ in group $\mathbb{G}$:

- Challenger generates a new keypair ($\mathrm{sk} \leftarrow \mathbb{Z}_q, \mathrm{pk} \leftarrow g^{\mathrm{sk}}$) and sends pk to $\mathcal{R}$. $\mathcal{R}$ forwards pk to $\mathcal{A}$.
- In the query phase, whenever $\mathcal{A}$ asks a decryption D_{sk} query (c_1, c_2), $\mathcal{R}$ rejects if either c_1 or c_2 is not a valid group element. Otherwise $\mathcal{R}$ asks a CSDH query $c_3 \leftarrow \mathrm{csdh}_{(g,\mathrm{pk})}(c_1)$. $\mathcal{R}$ returns c_2/c_3.
- In the challenge phase, whenever $\mathcal{A}$ gives a pair (m_0^*, m_1^*) of messages, $\mathcal{R}$ asks his challenge from the challenger. The challenger sets $b_{\mathcal{R}} \leftarrow \{0,1\}$, $y^* \leftarrow \mathbb{Z}_q$, $h_1^* \leftarrow g^{y^*}$. If $b_{\mathcal{R}} = 0$, then he sets $h_2^* \leftarrow \mathbb{G}$, otherwise $h_2^* \leftarrow \mathrm{pk}^{y^*}$. $\mathcal{R}$ picks a random bit $b_{\mathcal{A}} \leftarrow \{0,1\}$ and sends $(h_1, m_{b_{\mathcal{A}}} \cdot h_2)$ to $\mathcal{A}$. $\mathcal{A}$ returns a bit $b'_{\mathcal{A}}$.
- In the guess phase, if $b'_{\mathcal{A}} = b_{\mathcal{A}}$, then $\mathcal{R}$ returns $b'_{\mathcal{R}} = 1$, otherwise $\mathcal{R}$ returns $b'_{\mathcal{R}} = 0$.

Now, $\Pr[\mathcal{R}$ wins in the $\mathrm{ddh}^{\mathrm{csdh}}$ game$] = \Pr[b'_{\mathcal{R}} = b_{\mathcal{R}}] = \Pr[\mathcal{A}\text{ wins}|b_{\mathcal{R}} = 1] \cdot \Pr[b_{\mathcal{R}} = 1] + \Pr[\mathcal{A}\text{ wins}|b_{\mathcal{R}} = 0] \cdot \Pr[b_{\mathcal{R}} = 0] = (\frac{1}{2} + \varepsilon') \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} + \frac{\varepsilon'}{2}$. Clearly $\mathcal{R}$ works in time $\tau' + \gamma \cdot (\tau_{\mathrm{csdh}} + \mathsf{small}) + \mathsf{small}$. □

2) **Second direction ($\mathbf{ddh}^{\mathbf{csdh}} \Rightarrow$ Elgamal-CCA1 with fixed (g, pk)):** Assume $\mathcal{A}$ is an adversary who can $(\tau', \gamma', \varepsilon')$-break the $\mathrm{ddh}^{\mathrm{csdh}}$ assumption in group $\mathbb{G}$. Construct the next reduction $\mathcal{R}$ that aims to break the CCA1-security of Elgamal:

- Challenger generates a new keypair ($\mathrm{sk} \leftarrow \mathbb{Z}_q, \mathrm{pk} \leftarrow g^{\mathrm{sk}}$) and sends pk to $\mathcal{R}$. $\mathcal{R}$ forwards pk to $\mathcal{A}$.
- In the query phase, whenever $\mathcal{A}$ asks a CSDH query $\mathrm{csdh}_{(g,\mathrm{pk})}(h)$, $\mathcal{R}$ asks a decryption D_{sk} query $(h, 1)$, and receives back $c \leftarrow h^{-\mathrm{sk}}$. $\mathcal{R}$ returns c^{-1} to $\mathcal{A}$.
- In the challenge phase, whenever $\mathcal{A}$ asks for a challenge, $\mathcal{R}$ sends his message pair $(m_0^*, m_1^*) \leftarrow (g^r, 1)$ to challenger, where $r \leftarrow \mathbb{Z}_q$. Challenger picks a random bit $b_{\mathcal{R}} \leftarrow \{0,1\}$ and a random $r^* \leftarrow \mathbb{Z}_q$, and sends $(c_1^*, c_2^*) \leftarrow (g^{r^*}, g^{r(1-b_{\mathcal{R}})} \cdot \mathrm{pk}^{r^*})$ to $\mathcal{R}$. $\mathcal{R}$ forwards (c_1^*, c_2^*) to $\mathcal{A}$, who returns a guess $b'_{\mathcal{A}}$. $\mathcal{R}$ returns $b'_{\mathcal{R}} \leftarrow b'_{\mathcal{A}}$ to challenger.

Now, $\Pr[\mathcal{R}\text{ wins}] = \Pr[b'_{\mathcal{R}} = b_{\mathcal{R}}] = \Pr[\mathcal{A}\text{ wins}] = \varepsilon$. Clearly $\mathcal{R}$ works in time $\tau' + \gamma \cdot (\tau_D + \mathsf{small}) + \mathsf{small}$. □

It is straightforward to prove the next lemma.

Lemma 5 ($\mathbf{ddh}^{\mathbf{dsdh}} \Rightarrow \mathbf{ddh}^{\mathbf{csdh}}$). *If ddh^{csdh} holds, then ddh^{dsdh} holds.*

Proof (Sketch). Build a wrapper that uses the oracle that solves the CDH problem to solve the DDH problem. □

5 Irreductions

We will now show irreductions between the main security assumptions of this paper. We emphasize, see Sect. 2.2, that the irreductions will be somewhat limited by fixing some of the parameters. For example, Lem.3 stated that there does exist a reduction that solves $\mathrm{ddeg}^{\mathrm{csdeg}}$ on some input $(g, \mathrm{pk}_1, \mathrm{pk}_2, h_1^*, h_2^*, h_3^*)$ (with $\mathrm{pk}_1, \mathrm{pk}_2$) being randomly generated) given an access to a $\mathrm{dsdh}_{(g,\mathrm{pk}_2)}$ oracle with inputs $(\cdot, \cdot)$, i.e., with (g, pk_2) being fixed. As an example, the next Thm. 1 shows that there does not exist a reduction that solves dsdh on some input $(g, \mathrm{pk}, h_1^*, h_2^*)$ (with pk being randomly generated) given an access to a $\mathrm{ddeg}^{\mathrm{csdeg}}$ oracle with inputs $(g, \mathrm{pk}, \cdot, \cdot, \cdot)$, i.e., again with two inputs being fixed. Other irreductions are similar. Thus, the next irreductions are somewhat limited, since they do exclude the existence of reductions with arbitrary oracle queries. Nevertheless, they are still important, since all (known to us) reductions between similar problems in fact have limited oracle access.

In what follows, we will not state the concrete security parameters in the theorems, however, they are easy to calculate and one can verify that all following theorems provide exact (ir)reductions.

Theorem 1 ($\mathbf{ddeg}^{\mathbf{csdeg}} \not\Rightarrow_! \mathbf{ddh}$). *If there exists a static reduction $\mathcal{R}$ that reduces ddh to $\mathrm{ddeg}^{\mathrm{csdeg}}$, then there exists an efficient static irreduction $\mathcal{I}$ that, given $\mathcal{R}$ as an oracle and with fixed (g, pk), solves ddh.*

Proof. Fix a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q. Assume that $\mathcal{R} = \mathcal{R}^{\mathcal{A}}$ is an arbitrary reduction that uses $\mathcal{A}$ as an oracle to solve ddh. Here, $\mathcal{A}$ is an arbitrary algorithm that solves $\mathrm{ddeg}^{\mathrm{csdeg}}$. Equivalently, $\mathcal{A} = \mathcal{A}^{\mathrm{csdeg}_{(g,\mathrm{pk}_1,\mathrm{pk}_2)}(\cdot,\cdot)}$ solves ddeg. In particular, $\mathcal{A}$ can have an access to a $\mathrm{csdeg}_{(g,\mathrm{pk}_1,\mathrm{pk}_2)}$ oracle provided to her by $\mathcal{R}$. We now construct the next oracle machine $\mathcal{I} = \mathcal{I}^{\mathcal{R}}$ to solve ddh in time and with success probability comparable with those of $\mathcal{R}$. Note that $\mathcal{I}$ simulates the oracle $\mathcal{A}$ to $\mathcal{R}$, and therefore has access to the oracle $\mathrm{csdeg}_{(g,\mathrm{pk}_1,\mathrm{pk}_2)}(\cdot, \cdot)$. Moreover, $\mathcal{I}$ simulates the challenger $\mathcal{C}_2$ of the internal $\mathrm{ddeg}^{\mathrm{csdeg}} \Rightarrow \mathrm{ddh}$ game to $\mathcal{R}$.

- The challenger $\mathcal{C}_1$ of the $\mathrm{ddeg}^{\mathrm{csdeg}} \not\Rightarrow_! \mathrm{ddh}$ game sets $\mathrm{sk} \leftarrow \mathbb{Z}_q$ and $\mathrm{pk} \leftarrow g^{\mathrm{sk}}$. He sends pk to $\mathcal{I}$ as the public key in the $\mathrm{ddeg}^{\mathrm{csdeg}} \not\Rightarrow_! \mathrm{ddh}$ game.
- $\mathcal{I}$ simulates the challenger $\mathcal{C}_2$ to $\mathcal{R}$ in the $\mathrm{ddeg}^{\mathrm{csdeg}} \Rightarrow \mathrm{ddh}$ game as follows:
  - **Setup phase:** $\mathcal{I}[\mathcal{C}_2]$ forwards $\mathrm{pk}_1 \leftarrow \mathrm{pk}$, as the public parameter of the $\mathrm{ddeg}^{\mathrm{csdeg}} \Rightarrow \mathrm{ddh}$ game, to $\mathcal{R}$. $\mathcal{R}$ generates $\mathrm{pk}_2 \leftarrow \mathbb{G}$.
  - **Query phase:** If $\mathcal{R}$ asks a $\mathrm{ddeg}^{\mathrm{csdeg}_{(g,\mathrm{pk}_1,\mathrm{pk}_2)}}$ query (h_1, h_2, h_3) from $\mathcal{A}$, then $\mathcal{I}[\mathcal{A}]$ simulates $\mathcal{A}$ as follows:
    1. $\mathcal{I}$ sends the query (h_1, h_2) to her $\mathrm{csdeg}_{(g,\mathrm{pk}_1,\mathrm{pk}_2)}(\cdot, \cdot)$ oracle in the $\mathrm{ddeg}^{\mathrm{csdeg}}$ game. The oracle replies with some h_3'.
    2. If $h_3' = h_3$, then $\mathcal{I}$ sets $b_{\mathcal{A}}' \leftarrow 1$, otherwise $\mathcal{I}$ sets $b_{\mathcal{A}}' \leftarrow 0$.
    3. $\mathcal{I}$ replies with $b_{\mathcal{A}}'$ as her answer to the $\mathrm{ddeg}^{\mathrm{csdeg}}$ challenge.
  - **Challenge phase:** If $\mathcal{R}$ asks $\mathcal{C}_2$ for his challenge in the $\mathrm{ddeg}^{\mathrm{csdeg}} \Rightarrow$ ddh game, then $\mathcal{I}[\mathcal{C}_2]$ simulates $\mathcal{C}_2$ as follows. She asks the challenger $\mathcal{C}_1$ for her challenge (h_1^*, h_2^*) in the irreduction game. $\mathcal{I}[\mathcal{A}]$ forwards (h_1^*, h_2^*) to $\mathcal{R}$ as his challenge.
  - **Guess phase:** $\mathcal{R}$ outputs a bit $b_{\mathcal{R}}'$.
- **Guess phase (of irreduction game):** $\mathcal{I}$ returns $b_{\mathcal{R}}'$.

Clearly, $\mathcal{I}$ emulates $\mathcal{A}$ correctly. Thus, $\Pr[\mathcal{I} \text{ wins}] = \Pr[\mathcal{R} \text{ wins}]$, and $\mathcal{I}$ spends marginally more time than $\mathcal{R}$. □

Theorem 2 ($\mathbf{ddh}^{\mathbf{dsdh}} \not\Rightarrow_! \mathbf{ddeg}^{\mathbf{csdeg}}$ for static reductions with fixed $(g, \mathrm{pk}_1, \mathrm{pk}_2)$). *If there is a static reduction $\mathcal{R}$ that reduces ddeg^{csdeg} to ddh^{dsdh}, then there is an efficient static irreduction $\mathcal{I}$ that, given $\mathcal{R}$ as an oracle and with fixed $(g, \mathrm{pk}_1, \mathrm{pk}_2)$, solves ddeg^{csdeg}.*

Proof. Fix a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q. Let $\mathcal{A}$ be an arbitrary algorithm that solves $\mathrm{ddh}^{\mathrm{dsdh}}$. Equivalently, $\mathcal{A}^{\mathrm{ddh}(g,\mathrm{pk},\cdot,\cdot)}$ solves ddh. Assume that $\mathcal{R} = \mathcal{R}^{\mathcal{A}}$ is an efficient algorithm that uses $\mathcal{A}$ as an oracle to solve $\mathrm{ddeg}^{\mathrm{csdeg}}$. Equivalently, $\mathcal{R} = \mathcal{R}^{\mathcal{A},\mathrm{csdeg}_{(g,\mathrm{pk}_1,\mathrm{pk}_2)}(\cdot,\cdot)}$ is an efficient algorithm that solves ddeg. Construct now the next oracle machine $\mathcal{I} = \mathcal{I}^{\mathcal{R},\mathrm{csdeg}(\cdot,\cdot,\cdot)}$ to solve ddeg with the help of $\mathcal{R}$ and $\mathrm{csdeg}_{(g,\mathrm{pk}_1,\mathrm{pk}_2)}(\cdot,\cdot)$ as oracles, in time and with success probability comparable with those of $\mathcal{R}$.

- **Setup phase:** The challenger $\mathcal{C}_1$ of the $\mathrm{ddh}^{\mathrm{dsdh}} \not\Rightarrow_! \mathrm{ddeg}^{\mathrm{csdeg}}$ game sets $\mathrm{sk}_1, \mathrm{sk}_2 \in \mathbb{Z}_q$ and $\mathrm{pk} \leftarrow (\mathrm{pk}_1 \leftarrow g^{\mathrm{sk}_1}, \mathrm{pk}_2 \leftarrow g^{\mathrm{sk}_2})$. He sends pk to $\mathcal{I}$ as the public key in the $\mathrm{ddh}^{\mathrm{dsdh}} \not\Rightarrow_! \mathrm{ddeg}^{\mathrm{csdeg}}$ game.
- $\mathcal{I}$ simulates both the challenger $\mathcal{C}_2$ and $\mathcal{A}$ to $\mathcal{R}$ in the $\mathrm{ddh}^{\mathrm{dsdh}} \Rightarrow \mathrm{ddeg}^{\mathrm{csdeg}}$ game as follows:
  - **Setup phase**: $\mathcal{I}[\mathcal{C}_2]$ forwards pk to $\mathcal{R}$ as $\mathcal{R}$'s public key in the $\mathrm{ddh}^{\mathrm{dsdh}} \Rightarrow \mathrm{ddeg}^{\mathrm{csdeg}}$ game.
  - **Query phase**:
    - If $\mathcal{R}$ asks a $\mathrm{csdeg}_{(g,\mathrm{pk}_1,\mathrm{pk}_2)}(\cdot,\cdot)$ query (h_1, h_2) from $\mathcal{A}$, then $\mathcal{I}[\mathcal{A}]$ forwards it to her own $\mathrm{csdeg}_{(g,\mathrm{pk}_1,\mathrm{pk}_2)}$ oracle.
    - If $\mathcal{R}$ asks a $\mathrm{ddh}^{\mathrm{dsdh}}$ query (h_1, h_2) from $\mathcal{A}$, then $\mathcal{I}[\mathcal{A}]$ forwards it to her $\mathrm{csdeg}_{(g,\mathrm{pk}_1,\mathrm{pk}_2)}(\cdot,\cdot)$ oracle. If the oracle returns $\perp$, then $\mathcal{I}$ returns 0. Otherwise, $\mathcal{I}$ returns 1. (Note that $\mathcal{I}$ does not need to use a ddh oracle of the $\mathrm{ddh}^{\mathrm{dsdh}}$ game here.)
  - **Challenge phase:** If $\mathcal{R}$ asks for a ddeg challenge from $\mathcal{C}_2$, then $\mathcal{I}[\mathcal{C}_2]$ simulates $\mathcal{C}_2$ as follows:
    1. $\mathcal{I}$ asks challenger $\mathcal{C}_1$ for her challenge in the ddh game. $\mathcal{C}_1$ sets $b_{\mathcal{I}} \leftarrow \{0,1\}$ and $y^* \leftarrow \mathbb{Z}_q$. He sets $h_1^* \leftarrow g^{y^*}$ and $h_2^* \leftarrow \mathrm{pk}_1^{y^*}$. If $b_{\mathcal{I}} = 0$ then he sets $h_3^* \leftarrow \mathbb{G}$, otherwise he sets $h_3^* \leftarrow \mathrm{pk}_2^{y^*}$. $\mathcal{C}_1$ sends (h_1^*, h_2^*, h_3^*) to $\mathcal{I}$ as a challenge.
    2. $\mathcal{I}[\mathcal{C}_2]$ forwards (h_1^*, h_2^*, h_3^*) to $\mathcal{R}$ as his challenge.
  - **Guess phase:** $\mathcal{R}$ outputs a bit $b'_{\mathcal{R}}$.
- **Guess phase (of the irreduction game):** $\mathcal{I}$ returns $b'_{\mathcal{I}} \leftarrow b'_{\mathcal{R}}$.

First, $\mathcal{I}$ emulates the queries correctly. Thus if $\mathcal{R}$ responds with a correct answer to the ddh query, then $\mathcal{I}$ responds with a correct answer to the $\mathrm{ddh}^{\mathrm{dsdh}}$ query. Thus $\Pr[\mathcal{I} \text{ wins}] = \Pr[\mathcal{R} \text{ wins}]$, and $\mathcal{I}$ works in time $\tau + \gamma_{\mathrm{ddeg}} \cdot (\tau_{\mathrm{ddeg}} + \mathsf{small}) + \gamma_{\mathcal{A}} \cdot \mathsf{small} + \mathsf{small}$, where τ is the working time of $\mathcal{R}$, τ_{ddeg} is the working time of the ddeg oracle, γ_{ddeg} is the number of queries to the ddeg oracle, $\gamma_{\mathcal{A}}$ is the number of queries to $\mathcal{A}$. □

Theorem 3 ($\text{ddh}^{\text{csdh}} \not\Rightarrow_! \text{ddh}^{\text{dsdh}}$ for static reductions with fixed (g, pk)). *If there is a static reduction $\mathcal{R}$ that reduces ddh^{dsdh} to ddh^{csdh}, then there is an efficient static irreduction $\mathcal{I}$ that, given $\mathcal{R}$ as an oracle and with fixed (g, pk), solves ddh^{dsdh}.*

Proof. Fix a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q. Let $\mathcal{A}$ be an arbitrary algorithm that solves ddh^{csdh}. Equivalently, $\mathcal{A} = \mathcal{A}^{\text{csdh}_{(g,\text{pk}_1)}(\cdot)}$ solves ddh. Assume that $\mathcal{R} = \mathcal{R}^{\mathcal{A}}$ is an efficient algorithm that uses $\mathcal{A}$ as an oracle to solve ddh^{dsdh}. Equivalently, $\mathcal{R} = \mathcal{R}^{\mathcal{A}, \text{dsdh}_{(g,\text{pk})}(\cdot)}$ is an efficient algorithm that solves ddh. Construct now the next oracle machine $\mathcal{I} = \mathcal{I}^{\mathcal{R}, \text{ddh}}$ to solve ddh with the help of $\mathcal{R}$ and $\text{dsdh}_{(g,\text{pk})}(\cdot)$ as oracles, in time and with success probability comparable with those of $\mathcal{R}$.

- Challenger $\mathcal{C}_1$ of the $\text{ddh}^{\text{csdh}} \not\Rightarrow_! \text{ddh}^{\text{dsdh}}$ game sets $\text{sk} \in \mathbb{Z}_q$ and $\text{pk} \leftarrow g^{\text{sk}}$. He sends pk to $\mathcal{I}$ as the public key.
- $\mathcal{I}$ simulates the challenger $\mathcal{C}_2$ of the $\text{ddh}^{\text{csdh}} \Rightarrow \text{ddh}^{\text{dsdh}}$ game to $\mathcal{R}$:
  - **Setup phase:** $\mathcal{I}$ forwards pk to $\mathcal{R}$ as his public key in the $\text{ddh}^{\text{csdh}} \Rightarrow \text{ddh}^{\text{dsdh}}$ game. $\mathcal{R}$ forwards pk as the public key to $\mathcal{A}$ in the ddh^{csdh} game.
  - **Query phase:**
    - * If $\mathcal{R}$ asks a $\text{dsdh}_{(g,\text{pk})}(\cdot, \cdot)$ query (h_1, h_2) from $\mathcal{A}$, then $\mathcal{I}[\mathcal{A}]$ forwards it to her dsdh oracle.
    - * If $\mathcal{R}$ asks a ddh^{csdh} query (h_1, h_2) from $\mathcal{A}$, then $\mathcal{I}[\mathcal{A}]$ forwards h_1 to her $\text{csdh}_{(g,\text{pk})}(\cdot)$ oracle, getting back some value h'. If $h' = h_2$ then $\mathcal{I}$ returns 1, otherwise $\mathcal{I}$ returns 0.
  - **Challenge phase:** When $\mathcal{R}$ asks his challenge from $\mathcal{I}[\mathcal{C}_2]$, then $\mathcal{I}[\mathcal{C}_2]$ asks her challenge from $\mathcal{C}_1$. $\mathcal{C}_1$ sets $b_{\mathcal{I}} \leftarrow \{0, 1\}$ and $y^* \leftarrow \mathbb{Z}_q$. He sets $h_1^* \leftarrow g^{y^*}$. If $b_{\mathcal{I}} = 0$ then he sets $h_2^* \leftarrow \mathbb{G}$, otherwise he sets $h_2^* \leftarrow \text{pk}^{y^*} = \text{cdh}(g, \text{pk}, h_1^*)$. $\mathcal{C}_1$ sends (h_1^*, h_2^*) to $\mathcal{I}$ as a challenge. $\mathcal{I}$ forwards (h_1^*, h_2^*) to $\mathcal{R}$ as his challenge.
  - **Guess phase:** $\mathcal{R}$ outputs a bit $b'_{\mathcal{R}}$.
- **Guess phase (of the irreduction game)**: $\mathcal{I}$ returns $b'_{\mathcal{I}} \leftarrow b'_{\mathcal{R}}$.

First, $\mathcal{I}$ emulates the queries correctly. Thus if $\mathcal{R}$ responds with a correct answer to the cdh query, then $\mathcal{I}$ responds with a correct answer to the ddh^{csdh} query. Thus $\Pr[\mathcal{I} \text{ wins}] = \Pr[\mathcal{R} \text{ wins}]$, and $\mathcal{I}$ works in time $\tau + \gamma_{\text{ddh}} \cdot (\tau_{\text{ddh}} + \mathsf{small}) + \gamma_{\mathcal{A}} \cdot \mathsf{small} + \mathsf{small}$, where τ is the working time of $\mathcal{R}$, τ_{ddh} is the working time of the ddh oracle, γ_{ddh} is the number of queries to the ddh oracle, $\gamma_{\mathcal{A}}$ is the number of queries to $\mathcal{A}$. □

6 Hardness in Generic Group Model

Maurer's Formalization of Generic Group Model. In this section, we show that ddh^{csdh} is hard in the generic group model [13]. To do this, we use the abstraction of generic group model from [10]. Namely, we assume that $\mathbf{B}$ is a black-box that can store values from a certain ring $\mathbb{R}$ in internal state variables V_1, V_2, …, V_m. The storage capacity m is in our case unbounded. The initial state consists of the values of $V^d := [V_1, \ldots, V_d]$ for some $d < m$, which are set according to some probability distribution P_{V^d}. The black-box $\mathbf{B}$ allows two types of operations, computation operations on internal state variables, and queries about the internal state. No other interaction with $\mathbf{B}$ is possible. Formally, for a set Π of operations on $\mathbb{R}$, a computation operation consists of selecting a (say) t-ary operation $f \in \Pi$ and indices $i_1 \ldots, i_{t+1} \leq m$. Then $\mathbf{B}$ computes $f(V_{i_1}, \ldots, V_{i_t})$ and stores the result in $V_{i_{t+1}}$. Since m is unbounded, we can always assume that i_{t+1} is a unique index. We also assume that no computation operation $(f, V_{i_1}, \ldots, V_{i_t})$ is repeated. As for queries, for a set Σ of relations on $\mathbb{R}$, a query consits of selecting a (say) t-ary relation $\sigma \in \Sigma$ and indices $i_1 \ldots, i_t \leq m$. The query is replied by $\sigma(V_{i_1}, \ldots, V_{i_t})$.

In the case of proving lower bounds for a decisional problem, the task is to distinguish between two black boxes $\mathbf{B}$ and $\mathbf{B}'$ of the same type with different distributions of the initial state V^d. The success probability of an algorithm is taken over the choice of the initial state V^d, and of the randomness of the algorithm.

Let $\mathcal{C}$ denote the set of constant operations on $\mathbb{R}$. Let $\mathcal{L}$ denote the set of linear functions (of the form of $a_1 V_1 + \cdots + a_d V_d$) on the initial state V^d. For a given set Π of operations, let $\overline{\Pi}$ be the set of functions on the initial state that can be computed using operations in Π. See [10] for more details.

We also use the following lemma from [13].

Lemma 6 (Shoup [13]). *The fraction of solutions $(x_1, \ldots, x_k) \in \mathbb{Z}_n$ of the multivariate polynomial equation $p(x_1, \ldots, x_k) \equiv 0 \pmod{n}$ of degree d is at most d/q, where q is the largest prime factor of n.*

Hardness of $\mathbf{ddh^{csdh}}$ in Generic Group Model. Recall that the hardest assumption of this paper is $\mathrm{ddh}^{\mathrm{csdh}}$, which is equivalent to the assumption that Elgamal is CCA1-secure. To motivate that $\mathrm{ddh}^{\mathrm{csdh}}$ is a reasonable assumption, we now prove its security in the generic group model. We note here that differently from [10], the adversary here has a *nonadaptive* access to a multiplication operator in $\mathbb{R}$. This will add another level of complication to the proof.

Theorem 4. *Let $\mathbb{R} = \mathbb{Z}_n$, where p is the smallest prime factor of n and q is the largest prime factor of n. For $\Pi = \mathcal{C} \cup \{+\}$ and $\Sigma = \{=\}$, the advantage of every k-step adversary, $k \geq 1$, that has access to a nonadaptive oracle for multiplication with x, for distinguishing a random triple (x, y, z) from a triple $(x, y, x \cdot y)$ is upper bounded by $(4k^3 - (3 + \sqrt{3})k^2 - k + 2)/(54q)$.*

Proof. As in [10], the basic strategy of the proof is to consider two black boxes, one of which has initial state (x, y, z), and another one has initial state $(x, y, x \cdot y)$. For either of the black boxes, we assume that the adversary has been successful if it has found a collision between two different elements V_i and V_j. The distinguishing probability is upperbounded by the maximum of those two collision-finding probabilities.

We only analyze the case where the initial state is $(x, y, x \cdot y)$. Let Q be the number of queries made by the adversary to the nonadaptive oracle (thus the adversary obtains values $x, \ldots, x^Q$), P the number of degree $\leq Q$ polynomials computed by the adversary before the challenge phase starts, and R be the number of polynomials computed after the challenge phase. For simplicity, we assume that when x^i are already given, any degree ℓ polynomial can be computed in 1 step. (The precise bound depends crucially on this. For example, if it took ℓ steps to compute a single polynomial, we would get upper bound $(k^2 - k)/(2q)$.) Due to this,

$$Q + P + R \leq k \ . \tag{1}$$

Due to the presence of the nonadaptive oracle, the adversary first asks the black-box to compute P different polynomials

$$f_i(x, y) := \sum_{j=0}^{Q} f_{ij} x^j + c_i y + d_i xy \tag{2}$$

for $i \in \{0, \ldots, P-1\}$. Since neither y or $x \cdot y$ is available yet, $c_i = d_i = 0$. Thus, after the query phase, **B**'s state is equal to $(x, y, x \cdot y, f_0(x, y), \ldots, f_{P-1}(x, y))$.

After the challenge phase, the adversary can ask the black-box to compute R functions $f_i(x, y) := \sum_{j=0}^{Q} a_{ij} f_j(x) + b_i x + c_i y + d_i xy + e_i = \sum_{t=0}^{Q} \left(\sum_{j=0}^{P-1} a_{ij} f_{jt}\right) x^t + b_i x + c_i y + d_i xy + e_i$ for $i \in \{P, \ldots, P+R-1\}$. Clearly, each $f_i(x, y)$ for $i \geq P$ can be also written in form Eq. (2) though not with c_i and d_i necessarily being equal to 0. Here we assume that $f_i \neq f_j$ as a polynomial.

Now, any $f_i(x, y)$ is a degree $\leq Q$ polynomial. According to Lem. 6, the probability that any two of the $P + R$ functions $f_i \neq f_j$ have a common root is Q/q, and thus the *total* probability of finding a collision is bounded by

$$\frac{Q \cdot \binom{P+R}{2}}{q} \ . \tag{3}$$

Let $k' = \sqrt{k^2 - k + 1}$. Note that $\frac{\sqrt{3}}{2} \cdot k \leq k' \leq k$ for $k \geq 1$. Observe that, under the inequality Eq. (1), Eq. (3) is largest if $Q = (2k - k' - 1)/3$ and $P + R = (k + k' + 1)/3$. Then for $k \geq 1$,

$$\begin{aligned} Q \cdot \binom{P+R}{2} &= \frac{2k^3 + 2k^2k' - 3k^2 - 2kk' - 3k + 2k' + 2}{54} \\ &\leq \frac{4k^3 - (3 + \sqrt{3})k^2 - k + 2}{54} \ . \end{aligned}$$

In particular, constant success probability requires $k = \Omega(\sqrt[3]{q})$ steps. □

Acknowledgments. The author was supported by Estonian Science Foundation, grant #8058, and European Union through the European Regional Development Fund. We thank Daniel Brown for discussions.

References

1. Bresson, E., Monnerat, J., Vergnaud, D.: Separation Results on the "One-More" Computational Problems. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 71–87. Springer-Verlag, San Francisco, CA, USA (Apr 8–11, 2008)
2. Brown, D., Gallant, R.: The Static Diffie-Hellman Problem. Tech. Rep. 2004/306, International Association for Cryptologic Research (2004), http://eprint.iacr.org/2004/306, available at `http://eprint.iacr.org/2004/306`
3. Brown, D.R.L.: Irreducibility to the One-More Evaluation Problems: More May Be Less. Tech. Rep. 2008/435, International Association for Cryptologic Research (2007), available at http://eprint.iacr.org/2007/435
4. Damgård, I.: Towards Practical Public Key Systems Secure against Chosen Ciphertext Attacks. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 445–456. Springer-Verlag, 1992, Santa Barbara, California, USA (Aug 11–15, 1991)
5. Desmedt, Y., Lipmaa, H., Phan, D.H.: Hybrid Damgård Is CCA1-Secure under the DDH Assumption. In: Franklin, M.K., Hui, L.C.K., Wong, D.S. (eds.) CANS 2008. LNCS, vol. 5339, pp. 18–30. Springer-Verlag, Hong Kong, China (Dec 2–4, 2008)
6. Desmedt, Y., Phan, D.H.: A CCA Secure Hybrid Damgård's ElGamal Encryption. In: Bao, F., Chen, K. (eds.) ProvSec 2008. LNCS, vol. 5324, pp. 68–82. Springer-Verlag, Shanghai, China (October 30 – November 1, 2008)
7. Elgamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (1985)
8. Gjøsteen, K.: A New Security Proof for Damgård's ElGamal. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 150–158. Springer-Verlag, San Jose, CA, USA (Feb 13–17, 2006)
9. Kiltz, E., Pietrzak, K., Stam, M., Yung, M.: A New Randomness Extraction Paradigm for Hybrid Encryption. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 590–609. Springer-Verlag, Colgone, Germany (Apr 26–30, 2009)
10. Maurer, U.M.: Abstract Models of Computation in Cryptography. In: Smart, N.P. (ed.) WCC 2005. LNCS, vol. 3796, pp. 1–12. Springer-Verlag, Cirencester, UK (Dec 19–21, 2005)
11. Naor, M.: On Cryptographic Assumptions and Challenges. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 96–109. Springer-Verlag, Santa Barbara, USA (Aug 17–21, 2003)
12. Okamoto, T., Pointcheval, D.: The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 104–118. Springer-Verlag, Cheju Island, Korea (Feb 13–15, 2001)
13. Shoup, V.: Lower Bounds for Discrete Logarithms and Related Problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer-Verlag, Konstanz, Germany (11–15 May 1997)
14. Tsiounis, Y., Yung, M.: On the Security of ElGamal-Based Encryption. In: Imai, H., Zheng, Y. (eds.) PKC 1998. LNCS, vol. 1431, pp. 117–134. Springer-Verlag, Pacifico Yokohama, Japan (5–6 Feb 1998)
15. Wu, J., Stinson, D.R.: On the Security of the ElGamal Encryption Scheme and Damgård's Variant. Tech. Rep. 2008/200, International Association for Cryptologic Research (2008), available at http://eprint.iacr.org/2008/200