DDH-like Assumptions Based on Extension Rings

Ronald Cramer*, Ivan Damgård†, Eike Kiltz‡, Sarah Zakarias†, Angela Zottarel†⋆

∗ CWI and Leiden University, † Aarhus University ‡ RU Bochum

Abstract. We introduce and study a new type of DDH-like assumptions based on groups of prime order q. Whereas standard DDH is based on encoding elements of $\mathbb{F}_q$ "in the exponent" of elements in the group, we ask what happens if instead we put in the exponent elements of the extension ring $R_f = \mathbb{F}_q[X]/(f)$ where f is a degree-d polynomial. The decision problem that follows naturally reduces to the case where f is irreducible. This variant is called the d-DDH problem, where 1-DDH is standard DDH. We show in the generic group model that d-DDH is harder than DDH for $d > 1$ and that we obtain, in fact, an infinite hierarchy of progressively weaker assumptions whose complexities lie "between" DDH and CDH. This leads to a large number of new schemes because virtually all known DDH-based constructions can very easily be upgraded to be based on d-DDH. We use the same construction and security proof but get better security and moreover, the amortized complexity (e.g, computation per encrypted bit) is the same as when using DDH. We also show that d-DDH, just like DDH, is easy in bilinear groups. We therefore suggest a different type of assumption, the d-vector DDH problems (d-VDDH), which are based on $f(X) = X^d$, but with a twist to avoid problems with reducible polynomials. We show in the generic group model that d-VDDH is hard in bilinear groups and that the problems become harder with increasing d. We show that hardness of d-VDDH implies CCA-secure encryption, efficient Naor-Reingold style pseudorandom functions, and auxiliary input secure encryption. This can be seen as an alternative to the known family of k-LIN assumptions.

1 Introduction

The computational Diffie-Hellman assumption (CDH, proposed by Diffie and Hellman in [DH76]), says that if one chooses random g in a finite group $\mathbb{G}$ and random exponents a, b, then given g, g^a, g^b it is hard to compute g^{ab}. The assumption was introduced as basis for the well-known Diffie-Hellman key exchange.

[⋆] The second, fourth and fifth author acknowledge support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, and also from the CFEM research center (supported by the Danish Strategic Research Council) within which part of this work was performed.

However, to get efficient cryptographic constructions one needs the stronger Decisional Diffie-Hellman assumption (DDH, studied by Naor and Reingold in [NR97]). It says that given g, g^a, g^b, the group element g^{ab} is pseudorandom, i.e., cannot be efficiently distinguished from g^c for a random c. In some groups, the DDH assumption is clearly false, but it is widely conjectured to hold when $\mathbb{G}$ is, for instance, a large prime order subgroup of $\mathbb{F}_p^*$ or an elliptic curve group.

DDH has been used as the basis for a very wide range of efficient cryptographic primitives, such as pseudorandom functions (PRF) [NR97], hash-proof systems and CCA-secure public-key encryption [CS98], leakage resilient cryptography (in particular, auxiliary input security [DGK$^+$10]), and circular secure encryption [BHHO08].

Similar efficient constructions are not known under the weaker CDH assumption (unless one assumes random oracles) and this has motivated a large body of research studying weaker variants of DDH that would still enable cryptographic constructions. A well-known example is a family of assumptions called the k-LIN assumptions (where $k = 1$ is simply the standard DDH assumption) [BBS04,HK07,Kil07,Sha07]. In the generic group model, these assumptions are known to become progressively weaker for increasing k.

In this paper we initiate a study of a new family of assumptions that form natural extensions of DDH in prime order groups: if $\mathbb{G}$ has prime order q, and we fix a generator h, then an element $g \in \mathbb{G}$ "encodes" an element $a \in \mathbb{F}_q$ namely the a for which $g = h^a$. Intuitively we can think of a copy of $\mathbb{F}_q$ sitting in the exponent, and we can add field elements by multiplying in $\mathbb{G}$, and multiply by known constants by doing exponentiation. However, if CDH is hard, we cannot do general multiplication, i.e., compute g^{ab} from g^a, g^b. If DDH is hard, we cannot even distinguish the correct result from random. Now, let us instead consider the extension ring $R_f = \mathbb{F}_q[X]/(f)$ where f is a degree-d polynomial. It is well-known that an element $\mathbf{w} \in R_f$ can be represented as a vector $(w_0, ..., w_{d-1}) \in \mathbb{F}_q^d$. We can therefore represent $\mathbf{w}$ by a tuple of d group elements $(h^{w_0}, ..., h^{w_{d-1}}) \in \mathbb{G}^d$. Addition in R_f now becomes multiplication in $\mathbb{G}^d$, and multiplication by a known constant $\mathbf{a} \in R_f$ can be done (as we shall see) by applying a linear function in the exponent. This is simply because in R_f multiplication by a constant $\mathbf{a}$ acts as a linear mapping on the vector $(w_0, ..., w_{d-1})$. More details will be given below, but the essence is that if we set $\mathbf{g} = (h^{w_0}, ..., h^{w_{d-1}}) \in \mathbb{G}^d$ and take any $\mathbf{a} \in R_f$, we can define $\mathbf{g}^{\mathbf{a}}$ in a completely natural way, namely as the d-tuple of elements in $\mathbb{G}$ that represent $\mathbf{wa}$. This leads to defining the f-DDH problem as follows: given $\left(\mathbf{g}, \mathbf{g}^{\mathbf{a}}, \mathbf{g}^{\mathbf{b}}, \mathbf{g}^{\mathbf{c}}\right)$, where $\mathbf{a}$,$\mathbf{b}$,$\mathbf{c} \in \mathbb{F}_{q^d}, \mathbf{g} \in \mathbb{G}^d$, decide if $\mathbf{c}$ is random or $\mathbf{c} = \mathbf{ab}$.

It is not hard to see that f_1-DDH and f_2-DDH are equivalent whenever R_{f_1} is isomorphic to R_{f_2}, and also that f_2-DDH is no harder than f_1-DDH where f_1 is an irreducible factor in f_2. So it is natural to consider only the case where f is irreducible of degree d, in which case $R_f = \mathbb{F}_{q^d}$. This variant is called d-DDH[1]. We show that if d_1 divides d_2, so that $\mathbb{F}_{q^{d_1}}$ is a subfield of $\mathbb{F}_{q^{d_2}}$, then d_2-DDH is

[1] The d-DDH assumption should not be confused with the previously known k-DDH assumption which is completely different and is *stronger* than DDH (see, e.g., [BB04,DY05,BMR10] for details on and applications of this assumption).

at least as hard as d_1-DDH. Conversely, we show in the generic group model that d-DDH for $d > 1$ is *harder* than DDH, and that d_2-DDH is harder than d_1-DDH if $d_1|d_2$ and $d_2 > 4(3d_1 - 2)$. Thus we get an infinite hierarchy of progressively weaker assumptions whose complexities lie "between" DDH and CDH.

From a basic research point of view, we believe this result is interesting because it contributes to understanding a very natural class of assumptions. Moreover, the proof is interesting from a technical point of view: proofs in the generic group model usually work by arguing that the adversary fails because he cannot compute expressions "in the exponent" of sufficiently high degree. This approach completely fails in our case, instead we have to solve a much harder task, namely we show that the ability to verify whether certain degree-2 equations are satisfied, does not allow verification of a different class of degree-2 equations.

From a more practical point of view, d-DDH gives us a large number of new schemes "for free" because virtually all known cryptographic constructions based on DDH can very easily be upgraded to be based on d-DDH: exactly the same construction and security proof applies but we get better security. Moreover, the amortized complexity of resulting schemes (e.g., computation per encrypted bit) is *the same* as when using DDH. We explain this in more detail in Section 5. In contrast, using the family of k-LIN assumptions is less attractive: The known DDH-based primitives have to be generalized to k-LIN and reproved from scratch, and one suffers a loss of efficiency that increases with k (also in the amortized sense).

How significant is the security advantage of using d-DDH? Given that in appropriately chosen groups, we do not know how to attack even the weakest variant, this can only be a matter of opinion. One may of course take the position that extending DDH is not useful: one can choose to believe that if DDH turns out to be easy, the algorithm will "probably" be so general that it can solve d-DDH for any d. This, on the other hand, is an argument that can made in exactly the same way against any known class of assumptions that generalize DDH, such as the k-LIN assumptions. With current state of the art, there is no way to settle this question. What our result does guarantee, however, is that if someone finds an efficient algorithm for DDH, even a non-generic one, there is no generic black-box reduction that turns it into an algorithm for 2-DDH, for instance. To render the d-DDH assumptions useless, one needs to solve the entire hierarchy using a non-generic reduction or a completely general algorithm.

We believe that in applications of cryptography, one should always minimize the risk of ones assumption being broken. And if the risk can potentially be made smaller at very little extra cost by modifying the application, there is good reason to do this. We therefore believe that using, e.g, 2-DDH instead of DDH is a 'good deal" in practice.

Everything we said so far applies to groups where no bilinear map is available, such as prime order subgroups of $\mathbb{Z}_p^*$ or compact elliptic curve based groups. In bilinear groups, however, it turns out that d-DDH, just like DDH, is easy. This fact motivates our suggestion of an alternative family of problems: we observe that by omitting some group elements from an instance of f-DDH, one can obtain a problem that is hard, even if f is reducible. Based on this, we propose the d-vector DDH (d-VDDH) assumptions, based on $f(X) = X^d$. We show in the generic group model that the d-VDDH assumption holds even in bilinear groups. In fact, it holds even given a d-linear map, which can be thought of as an oracle allowing the adversary to compute expressions of degree d in the exponent. This means that the d-VDDH assumptions become progressively weaker for increasing d. We show that the d-VDDH assumption implies CCA-secure encryption and efficient Naor-Reingold style pseudorandom functions. We also construct another cryptosystem based on the d-VDDH assumption, very similar to the BHHO scheme [BHHO08]. We show that this scheme is auxiliary input secure, a strong form of leakage resilience where full information on the secret key can be leaked, as long as the key remains hard to compute.

In bilinear groups, the family of d-VDDH assumptions can therefore be seen as an alternative to the (incomparable) family of k-linear assumptions.

A final related work that should be mentioned is [HYZX08] in which an assumption called EDDH is proposed, which is our 2-DDH assumption. This is the only prior work we know of that mentions a DDH variant based on ring extensions. It is claimed in [HYZX08] that DDH reduces to EDDH and that in the generic group model EDDH is hard, even in bilinear groups. The first result is correct, but we could not verify the proof. In this paper, we give a different proof of a more general statement. The second claim is false, and is refuted by our result that d-DDH for any d is easy in bilinear groups.

2 Preliminaries

2.1 Notation

If S is a set, we write $x \leftarrow S$ meaning that x is sampled uniformly from S. If $\mathbf{x} \in \mathbb{F}_q^m$ is a vector, we write $x[i]$ for the ith entry of $\mathbf{x}$. We say that a function $f : \mathbb{N} \to \mathbb{R}$ is negligible if, for every polynomial p, there exists an integer $n_p \in \mathbb{N}$ such that $f(n) < 1/p(n)$ for every $n > n_p$. If X and Y are two random variables, we say that X and Y are computationally indistinguishable ($X \overset{c}{\approx} Y$) if their computational distance is negligible. Furthermore, throughout the paper, vectors are denoted by bold lowercase letters.

A d-linear map $e : \mathbb{G}^d \to \mathbb{G}_T$ is an efficiently computable map such that $e(g, \dots, g) \neq 1$ and $e(g_1^{a_1}, \dots, g_d^{a_d}) = e(g_1, \dots, g_n)^{\prod a_i}$, for all g_i in $\mathbb{G}$ and for all a_i in $\mathbb{F}_q$. A d-linear group $\mathbb{G}$ is a group $\mathbb{G}$ together with a d-linear map.

3 Extension Rings and DDH

We consider here a finite field $\mathbb{F}_q$ of prime order q and its extension with a polynomial f of degree d. By this we obtain the ring $R_f = \mathbb{F}_q[X]/(f)$, where an element $\mathbf{v}$ can be written as $v_0 + \cdots + v_{d-1}X^{d-1} + (f)$. However, we can also represent $\mathbf{v}$ by the matrix $V = v_0\mathbf{I}_d + v_1 A_f + \cdots + v_{d-1}A_f^{d-1}$, where $\mathbf{I}_d$ is the d-dimensional identity matrix and A_f is the so-called companion matrix of f. The companion matrix of a monic polynomial $f = X^d + \alpha_{d-1}X^{d-1} + \cdots + \alpha_1 X^1 + \alpha_0$ is given by the $d \times d$ matrix

$$A_f = \begin{pmatrix} 0 & 0 & \cdots & 0 & -\alpha_0 \\ 1 & 0 & \cdots & 0 & -\alpha_1 \\ 0 & 1 & \cdots & 0 & -\alpha_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -\alpha_{d-1} \end{pmatrix}.$$

Action of matrices on $\mathbb{G}^d$ Given a group $\mathbb{G}$ of order q and a tuple of elements $\mathbf{g} = (g_0, \ldots, g_{d-1}) \in \mathbb{G}^d$, any matrix $M = (m_{ij})$ of dimension $n \times d$ defines a mapping $\mathbb{G}^d \to \mathbb{G}^n$ as follows:

$$\mathbf{g}^M := \left(\prod_j^d g_j^{m_{1j}}, \ldots, \prod_j^d g_j^{m_{nj}} \right). \tag{1}$$

In particular this means that R_f can act on $\mathbb{G}^d$: we write the element $\mathbf{v} \in R_f$ in its matrix representation V and compute $\mathbf{g}^{\mathbf{v}} := \mathbf{g}^V$ as above. It is straightforward to verify that this map behaves according to the standard rules for exponentiation:

$$(\mathbf{g}^{\mathbf{a}})^{\mathbf{b}} = \mathbf{g}^{\mathbf{ab}}, \quad \mathbf{g}^{\mathbf{a}}\mathbf{g}^{\mathbf{b}} = \mathbf{g}^{\mathbf{a}+\mathbf{b}}.$$

Note that this action can also be understood as implementing a product in R_f in a slightly different way: if we choose a generator h of $\mathbb{G}$, then we can write any $\mathbf{g}$ as $(g_0, \ldots, g_{d-1}) = (h^{w_0}, \ldots, h^{w_{d-1}})$. Once we fix h, we can therefore think of $\mathbf{g}$ as representing an element $\mathbf{w}$ in R_f, namely $\mathbf{w} = w_{d-1}X^{d-1} + \ldots + w_0 + (f)$. We will write this as $\mathbf{g} = h(\mathbf{w})$. It now turns out that we have

$$\mathbf{g}^{\mathbf{v}} = h(\mathbf{w})^{\mathbf{v}} = h(\mathbf{wv}).$$

This follows because we can think of R_f as a d-dimensional vector space over $\mathbb{F}_q$. In that interpretation, multiplication by $\mathbf{v}$ is a linear mapping which has a matrix, namely V. Since the action $\mathbf{g}^{\mathbf{v}}$ is defined to be multiplication by V "in the exponent", it follows that by computing $\mathbf{g}^{\mathbf{v}} = (h^{w_0}, \ldots, h^{w_{d-1}})^{\mathbf{v}}$, we are in fact multiplying $\mathbf{w}$ by $\mathbf{v}$.

3.1 The f-DDH Problem

Given the above, we can now define an new variant of the DDH problem:

Definition 1 (The f-DDH Problem). *Let f be a d-degree polynomial. Let $\mathcal{G}$ be a PPT algorithm, which given the security parameter λ, outputs the description of a group $\mathbb{G}$ of order $q = q(1^\lambda)$. Let $\mathcal{A}$ be a probabilistic algorithm that* *takes as input (a description of) $\mathbb{G}$ and a 4-tuple of elements in $\mathbb{G}^d$, and outputs 0 or 1. We say that $\mathcal{A}$ solves the f-DDH problem with advantage $\varepsilon_{\mathcal{A}}(\lambda)$, where*

$$\varepsilon_{\mathcal{A}}(\lambda) = |Pr[\mathcal{A}(\mathbb{G}, (\boldsymbol{g}, \boldsymbol{g}^{\boldsymbol{a}}, \boldsymbol{g}^{\boldsymbol{b}}, \boldsymbol{g}^{\boldsymbol{c}})) = 1] - Pr[\mathcal{A}(\mathbb{G}, (\boldsymbol{g}, \boldsymbol{g}^{\boldsymbol{a}}, \boldsymbol{g}^{\boldsymbol{b}}, \boldsymbol{g}^{\boldsymbol{ab}})) = 1]|$$

where $\boldsymbol{g} \leftarrow \mathbb{G}^d$ and $\boldsymbol{a} \leftarrow R_f, \boldsymbol{b} \leftarrow R_f, \boldsymbol{c} \leftarrow R_f$. In other words, given $\left(\boldsymbol{g}, \boldsymbol{g}^{\boldsymbol{a}}, \boldsymbol{g}^{\boldsymbol{b}}, \boldsymbol{g}^{\boldsymbol{c}}\right)$, the problem is to decide whether $\boldsymbol{c} = \boldsymbol{ab}$ or $\boldsymbol{c}$ is a random element in R_f.

Equivalently, we can think of the problem instance as being given in the alternative representation $(h(\mathbf{w}), h(\mathbf{wa}), h(\mathbf{wb}), h(\mathbf{wc}))$. This makes no difference to the adversary, as he would not be given $\mathbf{w}$ – but he knows that such a $\mathbf{w}$ exists. From the above we construct the following assumption.

Definition 2 (The f-DDH Assumption). *For any probabilistic polynomial time algorithm $\mathcal{A}$ as in Definition 1, it holds that $\varepsilon_{\mathcal{A}}(\lambda)$ is negligible as a function of λ.*

Note that this is a generalization of the DDH problem: for a polynomial f of degree 1, $R_f = \mathbb{F}_q$ and f-DDH is just the standard DDH problem in $\mathbb{G}$.

Now we look a bit closer at the polynomial f. We can distinguish between two different cases: one where f is reducible and one where f is irreducible. For the first case we have the following theorem:

Theorem 1 (f-DDH for reducible f). *Let f be a d-degree reducible polynomial and suppose f_0 divides f, then solving f-DDH is polynomial time reducible to solving f_0-DDH.*

Proof. Let d_0 and d be the degrees of f_0 and f respectively. Let us consider an element $\mathbf{w}$ in R_f. We know that $\mathbf{w}$ can be written as $w_{d-1}x^{d-1} + \cdots + w_0 + (f)$. If we map $\mathbf{w}$ to R_{f_0} by reducing modulo f_0 we get an element $\mathbf{v} = v_{d_0-1}x^{d_0-1} + \cdots + v_0 + (f_0)$. In fact, reduction modulo f_0 is a ring homomorphism $\phi : R_f \to R_{f_0}$. It particular, it is linear and therefore has a matrix M. By (1) we can let M act on $\mathbf{w}$, so we get $h(\mathbf{w})^M = h(\phi(\mathbf{w})) = h(\mathbf{v})$. Hence, M can be used to efficiently map an f-DDH instance $(h(\mathbf{w}), h(\mathbf{wa}), h(\mathbf{wb}), h(\mathbf{wc}))$ to an f_0-DDH instance $(h(\phi(\mathbf{w})), h(\phi(\mathbf{wa})), h(\phi(\mathbf{wb})), h(\phi(\mathbf{wc}))) = (h(\mathbf{v}), h(\mathbf{v}\phi(\mathbf{a})), h(\mathbf{v}\phi(\mathbf{b})), h(\mathbf{v}\phi(\mathbf{c})))$. If $\mathbf{c} = \mathbf{ab}$, then $\phi(\mathbf{c}) = \phi(\mathbf{a})\phi(\mathbf{b})$, while if $\mathbf{c}$ is uniform in R_f, then $\phi(\mathbf{c})$ is uniformly chosen in R_{f_0}. Thus, if we can solve f_0-DDH, we can solve f-DDH with the same advantage.

4 The d-DDH Problem

Theorem 1 implies that f-DDH is no harder than f_0-DDH, where f_0 is the smallest irreducible factor in f. The natural conclusion is therefore that we should only look at the irreducible polynomials. In this case we know that our ring R_f is a field, namely the extension field $\mathbb{F}_{q^d}$ where d is the degree of f. In fact, since all fields with q^d elements are isomorphic, f-DDH is equivalent f'-DDH for any f' which is also irreducible and of the same degree as f. This is because the isomorphism can be implemented as a linear mapping in the same fashion as in the proof of Theorem 1. We can thus efficiently map an f-DDH instance to an f'-DDH instance and hence the only thing that may matter to the hardness of the problem is the degree of the extension. In the following, we will talk about d-DDH. In this definition we do not fix f; we can use any d-degree irreducible polynomial and otherwise the game is the same as in Definition 1.

Theorem 2. *Let d_1 divide d_2, so $\mathbb{F}_{q^{d_1}}$ is a subfield of $\mathbb{F}_{q^{d_2}}$, then d_1-DDH is no harder than d_2-DDH.*

The proof can be found in the full version [CDK+11]. We now show that d-DDH for $d > 1$ is, in fact, harder than DDH in the generic group model. Moreover, we show that if d_1 divides d_2 and $d_2 > 4(3d_1 - 2)$, then d_2-DDH is generically harder than d_1-DDH, giving in this way a hierarchy of progressively strictly weaker assumptions. For this, we need two auxiliary results. The first is a standard result, known as the Schwartz-Zippel lemma [Sch80,Zip79]:

Theorem 3. *For a non-zero multivariate polynomial over a finite field K of degree at most t, if uniformly random and independent values are assigned to the variables, the probability that this produces a root is at most $t/|K|$.*

The second is our main technical result supporting the hardness of d-DDH. In the following, for $\mathbf{a}, \mathbf{b} \in \mathbb{F}_{q^{d_1}}$ we will use $C_k(\mathbf{a}, \mathbf{b}) \in \mathbb{F}$ to denote the k-th component of the product $\mathbf{ab} \in \mathbb{F}_{q^{d_1}}$. Moreover, for ease of notation, whenever we have $P_1, \ldots, P_d$ affine functions from $(\mathbb{F}_{q^{d_2}})^3$ to $\mathbb{F}_q$, we will denote by P the vector consisting of all the P_i's. Namely $P(X, Y, Z) = (P_1(X, Y, Z), \ldots, P_d(X, Y, Z))$. Note that here we think of $\mathbb{F}_{q^{d_2}}$ as a d_2-dimensional vector space over $\mathbb{F}$. With this notation, an expression like $C_k(P, T)(X, Y, Z)$ can be understood in a natural way as a degree-2 polynomial in the $3d_2$ coordinates of X, Y and Z.

Theorem 4. *For $i = 1, \ldots, d_1$, let $P_i, R_i, S_i, T_i : (\mathbb{F}_{q^{d_2}})^3 \to \mathbb{F}_q$ be affine functions, with $d_2 > 4(3d_1 - 2)$. Assume $F_k(P, R, S, T)(X, Y, XY) := (C_k(P, T) - C_k(R, S))(X, Y, XY)$ is the zero polynomial. Then also $F_k(P, R, S, T)(X, Y, Z)$ is the zero polynomial. In particular, if $d_1 = 1$, the above is true for any $d_2 > 1$.*

The point of this theorem is that $X, Y, Z \in \mathbb{F}_{q^{d_2}}$ represent the input that the adversary gets in the generic group model game. Once he receives these inputs, the P, R, S, T represent new group elements he can compute. They are affine functions since the adversary can only compute sums and scalar multiplications "in the exponent". The adversary is trying to decide whether $Z = XY$ or if Z is random. He can try to do this by submitting a tuple of group elements (represented by P, R, S, T) to the oracle which answers back whether this tuple is an $\mathbb{F}_{q^{d_1}}$-DDH tuple or not. In the theorem, the functions F_k represent the oracle's answer, as for each component $k = 1, \ldots, d_1$, F_k tests if the tuple is "good" or not. What the theorem says is that, no matter how the adversary computes his oracle queries, if the tuple he is submitting is "good", this was already obvious without asking the oracle because the corresponding polynomials F_k are identically zero.

The idea behind the proof is writing the functions P_i, R_i, S_i, T_i from $(\mathbb{F}_{q^{d_2}})^3$ to $\mathbb{F}_q$ as sums of affine functions mapping $\mathbb{F}_{q^{d_2}}$ to $\mathbb{F}_q$. Such affine functions can be expressed via the trace function $\mathrm{Tr} : \mathbb{F}_{q^{d_2}} \to \mathbb{F}_q$, leading to an expression which is much easier to handle. We can then start looking at the implications of $F_k(P, R, S, T)(X, Y, XY)$ being zero. We show that $F_k(P, R, S, T)(X, Y, XY)$ vanishing in $(\mathbb{F}_{d_2})^2$ implies several terms of $F_k(P, R, S, T)(X, Y, Z)$ vanish as well. Proceeding in this way, we simplify our expression further and obtain a polynomial which is a sum of products of trace functions. We show that the intersection of the kernels of these trace functions is not empty, and thus we prove that the last term surviving in $F_k(P, R, S, T)(X, Y, Z)$ actually does not depend on Z and so must be zero as well.

The complete proof of the theorem can be found in the full version [CDK$^+$11].

Theorem 5. *In the generic group model, the d_2-DDH assumption holds even when the adversary is given an oracle allowing him to solve the d_1-DDH problem, for $d_2 > 4(3d_1 - 2)$. In particular, if $d_1 = 1$, we have that d_2-DDH holds even when an adversary has access to a DDH oracle, for any $d_2 > 1$.*

Proof. Recall that an instance to the d_2-DDH problem can be written as $(h(\mathbf{w}), h(\mathbf{wa}), h(\mathbf{wb}), h(\mathbf{wc}))$ for a fixed generator h of $\mathbb{G}$ and random $\mathbf{w}, \mathbf{a}, \mathbf{b}, \mathbf{c}$ in $\mathbb{F}_{q^{d_2}}$. We will show, in the generic group model, that the problem remains hard even if the adversary is given $\mathbf{w}$. From $\mathbf{w}$, it is easy to compute $\mathbf{w}^{-1}$. So we can equivalently think of the problem as being given instead as $(h(\mathbf{x}), h(\mathbf{y}), h(\mathbf{z}))$, where the adversary now has to decide whether $\mathbf{z} = \mathbf{xy}$.

We will assume that a random bit b is chosen by the simulator, and when $b = 0$ the adversary sees $\mathbf{z} = \mathbf{xy}$, while if $b = 1$, the adversary will see a uniform $\mathbf{z}$. The theorem is proved if we can show that a polynomial-time adversary cannot guess b with non-negligible advantage over $1/2$.

Let $\mathcal{A}$ be a polynomial-time generic group adversary. As usual, $\mathcal{A}$ has access to an oracle computing the group operation and inversion. In our case, we also give $\mathcal{A}$ access to an oracle solving d_1-DDH problem. More formally, on input $g^{w_0}, \ldots, g^{w_{d_1-1}}, g^{a_0}, \ldots, g^{a_{d_1-1}}, g^{b_0}, \ldots, g^{b_{d_1-1}}, g^{c_0}, \ldots, g^{c_{d_1-1}}$, the oracle outputs 1 if $\mathbf{w}^2\mathbf{c} = \mathbf{wawb}$ in $\mathbb{F}_{q^{d_1}}$.

We consider an algorithm $\mathcal{B}$ playing the following game with $\mathcal{A}$. Algorithm $\mathcal{B}$ chooses $3d_2 + 2$ bit strings $\sigma_0, \ldots, \sigma_{3d_2+1}$ uniformly in $\{0,1\}^m$, for a sufficiently large m. These strings represent the encoded elements which algorithm $\mathcal{A}$ will work with. Internally, $\mathcal{B}$ keeps track of the encoded elements using polynomials in the ring $\mathbb{F}_q[X_1, \ldots, X_{d_2-1}, Y_0, \ldots, Y_{d_2-1}, Z_0, \ldots, Z_{d_2-1}, T_0]$. Externally, the elements that $\mathcal{B}$ gives to $\mathcal{A}$ are just bit strings in $\{0,1\}^m$. To maintain consistency, $\mathcal{B}$ creates a list L consisting of pairs (F, σ) where F is a polynomial in the ring specified above and σ is a bit string. Initially, L is set to $\{(1, \sigma_0), (X_1, \sigma_1), \ldots, (X_{d_2-1}, \sigma_{d_2-1}), (Y_0, \sigma_{d_2}), \ldots, (Y_{d_2-1}, \sigma_{2d_2-1}), (Z_0, \sigma_{2d_2}), \ldots, (Z_{d_2-1}, \sigma_{3d_2-1})\}$.
Algorithm $\mathcal{B}$ starts the game providing $\mathcal{A}$ with $\sigma_0, \ldots, \sigma_{3d_2-1}$. The simulation of the oracles goes as follows:

Group action: Given two strings σ_i, σ_j, $\mathcal{B}$ recovers the corresponding polynomials F_i and F_j and computes $F_i + F_j$. If $F_i + F_j$ is already in L, $\mathcal{B}$ returns to $\mathcal{A}$ the corresponding bit string; otherwise it returns a uniform element σ in $\{0,1\}^m$ and stores $(F_i + F_j, \sigma)$ in L.

Inversion: Given an element σ in $\mathbb{G}$, $\mathcal{B}$ recovers its internal representation F and computes $-F$. If the polynomial $-F$ is already in L, $\mathcal{B}$ returns the corresponding bit string; otherwise it returns a uniform string σ and stores $(-F, \sigma)$ in L.

d_1**-DDH:** Given $4d_1$ strings $\pi_1, \dots, \pi_{d_1}, \rho_1, \dots, \rho_{d_1}, \sigma_1, \dots, \sigma_{d_1}, \tau_1, \dots, \tau_{d_1}$ in $\mathbb{G}$, adversary $\mathcal{B}$ recovers the polynomials $P_1, \dots, P_{d_1}, R_1, \dots, R_{d_1}, S_1, \dots, S_{d_1}$, T_1, $\dots, T_{d_1}$ and returns 1 iff $C_i(P_1, \dots, P_{d_1}, T_1, \dots, T_{d_1}) = C_i(R_1, \dots, R_{d_1}, S_1, \dots, S_{d_1})$ for every $i = 1, \dots, d_1$, where C_i represents the i-th component of the product in $\mathbb{F}_{q^{d_1}}$.

After $\mathcal{A}$ queried the oracles, it outputs a bit b'. At this point, $\mathcal{B}$ chooses uniform values $\mathbf{x} = (x_1, \dots, x_{d_2-1})$, $\mathbf{y} = (y_0, \dots, y_{d_2-1})$, $\mathbf{z} = (z_0, \dots, z_{d_2-1})$ in $\mathbb{F}_{q^2_d}$ and sets $X_1 = x_1, \dots, X_{d_2-1} = x_{d_2-1}$, $Y_0 = y_0, \dots, Y_{d_2-1} = y_{d_2-1}$. Finally $\mathcal{B}$ chooses a bit b and, if $b = 1$ it sets $Z_0 = z_0, \dots,$ $Z_{d_2-1} = z_{d_2-1}$, otherwise it sets $Z_0 = C_0(\mathbf{x,y}), ..., Z_{d_2-1} = C_{d_2}(\mathbf{x,y})$.

If the simulation provided by $\mathcal{B}$ is consistent, it reveals nothing about b. This means that the probability of $\mathcal{A}$ guessing the correct value for b is $1/2$. The only way in which the simulation could be inconsistent is if, after we choose value for $\mathbf{x}, \mathbf{y}, \mathbf{z}$, either two different polynomials in L happen to produce the same value or some query to the d_1-DDH oracle is such that $C_i(P_1, \dots, P_{d_1}, T_1, \dots, T_{d_1}) - C_i(R_1, \dots, R_{d_1}, S_1, \dots, S_{d_1})$ is not the 0 polynomial, but produces 0 after assigning values.

If $b = 1$, all values for $\mathbf{x}, \mathbf{y}, \mathbf{z}$ are chosen independently, so Theorem 3 applies to show that for a single oracle query $C_i(P_1, \dots, P_{d_1}, T_1, \dots, T_{d_1}) - C_i(R_1, \dots, R_{d_1}, S_1, \dots, S_{d_1})$ or a single difference $F_i - F_j$, the probability of having 0 after assigning values is negligible because q is exponentially large and all polynomials involved have degree at most 2. Further, by the union bound, since we only have a polynomial number of polynomials to consider, the overall probability of having 0 after assigning values is also negligible.

If $b = 0$, there are two extra possibilities for inconsistency between simulation and real attack. The first is if some query to the d_1-DDH oracle satisfies that

$$C_i(P_1, \dots, P_{d_1}, T_1, \dots, T_{d_1}) - C_i(R_1, \dots, R_{d_1}, S_1, \dots, S_{d_1})(X, Y, Z) \neq 0,$$

but

$$C_i(P_1, \dots, P_{d_1}, T_1, \dots, T_{d_1}) - C_i(R_1, \dots, R_{d_1}, S_1, \dots, S_{d_1})(X, Y, XY)$$

is the 0-polynomial. This is ruled out by Theorem 4, since all the polynomials involved have degree at most 1 and can therefore be thought of as affine functions. The second potential inconsistency is if two distinct polynomials F_i, F_j in L satisfy that $(F_i - F_j)(X, Y, XY)$ is the 0 polynomial. To see that this cannot happen, note that since each F_i has degree at most 1, it can be decomposed uniquely as as $F_i(X, Y, Z) = F_i^x(X) + F_i^y(Y) + F_i^z(Z) + c_i$ for a constant c_i and polynomials $F_i^x(X), F_i^y(Y), F_i^z(Z)$ of degree at most 1 and constant term 0. A collision as described here can only happen if $(F_i^z - F_j^z)(Z) \neq 0$, but $(F_i^z - F_j^z)(XY) = 0$. This leads to a contradiction: we can assign values $Y_0 = 1, Y_1 = 0, ..., Y_{d-1} = 0$, corresponding to the 1-element in $\mathbb{F}_{q^d}$. With this assignment, we get that $(F_i^z - F_j^z)(X) = 0$, contradicting that $(F_i^z - F_j^z)(Z) \neq 0$. Having ruled out these two possibilities for inconsistency, the only remaining possibility is that an unfortunate choice of values for the variables lead to collisions, as in the $b = 1$ case. Again by Theorem 3, this happens with negligible probability since the involved polynomials have degree at most 4.

We now look at what happens to d-DDH in a bilinear group. In such a group it is well-known that DDH is easy, and we show that this is also the case for d-DDH. The EDDH assumption presented in [HYZX08] is equivalent to d-DDH for $d = 2$. It was claimed that EDDH is hard also in generic bilinear groups, which is however refuted by the following result:

Theorem 6. *d-DDH over any bilinear group can be solved in polynomial time.*

Proof. We assume that the extension field $\mathbb{F}_{q^d}$ has been constructed using some fixed irreducible polynomial f. Consider any two elements $\mathbf{x}$,$\mathbf{y} \in \mathbb{F}_{q^d}$ as vectors $\mathbf{x} = (x_0, ..., x_{d-1}), \mathbf{y} = (y_0, ..., y_{d-1})$ and write the product as $\mathbf{xy} = (z_0, ..., z_{d-1})$. Now, multiplication of $\mathbf{x}$ and $\mathbf{y}$ takes place by multiplying the polynomials $x_0 + ... + x_{d-1}X^{d-1}$ and $y_0 + ... + y_{d-1}X^{d-1}$ and reducing modulo $f(X)$. From this it follows that we can write

$$z_k = \sum \alpha_{ij}^k x_i y_j$$

for coefficients $\alpha_{ij}^k \in \mathbb{F}_q$ that depend only on $f(x)$. Now, if we are given d-tuples $h(\mathbf{x}), h(\mathbf{y})$, it follows from the above that we can efficiently compute a representation in the target group G_T of $\mathbf{xy}$. Namely, for every k, we have

$$e(h, h)^{z_k} = \prod_{ij} (e(h, h)^{x_i y_j})^{\alpha_{ij}^k} = \prod_{ij} e(h^{x_i}, h^{y_j})^{\alpha_{ij}^k}$$

and h^{x_i}, h^{y_j} can be taken directly from $h(\mathbf{x}), h(\mathbf{y})$. So if we define

$$e(h, h)(\mathbf{xy}) = (e(h, h)^{z_0}, ..., e(h, h)^{z_{d-1}})$$

what we have shown is that we can compute $e(h, h)(\mathbf{xy})$ efficiently from $h(\mathbf{x}), h(\mathbf{y})$.

Now, consider an input instance of d-DDH, in the form $h(\mathbf{w}), h(\mathbf{wa}), h(\mathbf{wb})$, $h(\mathbf{wc})$. Observe that we have $\mathbf{c} = \mathbf{ab}$ if and only if $\mathbf{wa}\ \mathbf{wb} = \mathbf{w}\ \mathbf{wc} = \mathbf{w}^2\mathbf{ab}$. It now follows immediately from the above that we can decide if $\mathbf{ab} = \mathbf{c}$ by computing $e(h, h)(\mathbf{wa}\ \mathbf{wb})$ and $e(h, h)(\mathbf{w}\ \mathbf{wc})$ and comparing the two.

Although of course not all groups are bilinear, this result nevertheless motivates looking for alternative assumptions with similar properties that can be assumed to be hard in bilinear groups. We do this in Section 6.

5 Applications of d-DDH

In this section we present a number of applications for the d-DDH assumption.

5.1 Pseudorandom Functions

We construct pseudorandom functions (PRF) from d-DDH by taking the construction from [NR97] and showing that the natural modification where we work in the extension field also gives a PRF.

Definition 3. *Let $F = \{F_k\}$ be a family of keyed functions where $F_k : A_k \to B_k$, for every k in the key space $\mathcal{K}$. We say that F is a family of pseudorandom functions if for all PPT algorithms $\mathcal{D}$, any polynomial p and large enough λ,*

$$|\Pr[\mathcal{D}^{F_k(\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{D}^{f(\cdot)}(1^\lambda) = 1]| < 1/p(\lambda),$$

where k is chosen uniformly in $\mathcal{K}$ and f is chosen uniformly from the set of functions mapping A_k to B_k.

PRF Construction We construct a function family $F = \{f_k\}$ as follows. The index k specifies a tuple $(q, \mathbb{G}^d, \mathbf{g}, \mathbf{a}_0, \ldots, \mathbf{a}_n)$ where q is a prime number, $\mathbb{G}$ is a group of order q, $\mathbf{g}$ is an element of $\mathbb{G}^d$ and $\mathbf{a}_0, \ldots, \mathbf{a}_n$ are random in $\mathbb{F}_{q^d}$. Finally, we define $f_k : \{0,1\}^n \to \mathbb{G}^d$, $f_k(x_1, \ldots, x_n) = \mathbf{g}^{\mathbf{a}_0 \prod_{x_i=1} \mathbf{a}_i}$.

Theorem 7. *Under the d-DDH assumption, the family $F = \{f_k\}$ defined above is a family of pseudorandom functions.*

The proof of the theorem follows the exact same line as in [NR97]. Essentially the proof is done by a hybrid argument in which we define a sequence of functions $\{h_i\}$ where h_0 is f_k and h_n is a uniformly random function. An adversary that distinguishes between h_0 and h_n will also distinguish between h_i and h_{i+1}, for some i, which reduces to the d-DDH problem.

5.2 Public Key Encryption

We now apply d-DDH to public key encryption. If we modify in the natural way the Elgamal [Gam84] scheme, we obtain CPA secure encryption based on d-DDH.

- $\mathsf{Gen}(1^\lambda)$: Let $\mathbb{G} \leftarrow \mathcal{G}(1^\lambda)$. Choose a random element $\mathbf{g} \leftarrow \mathbb{G}^d$ and random $\mathbf{x} \leftarrow \mathbb{F}_{q^d}$. Compute $\mathbf{h} = \mathbf{g}^{\mathbf{x}}$. The secret key is then $sk = \mathbf{x}$ and the public key is $pk = (\mathbf{h})$, where $\mathbb{G}$ can be considered a public parameter.
- $\mathsf{Enc}(pk, M)$: Let the message be $M \in \mathbb{G}^d$. Choose randomly $\mathbf{r} \leftarrow \mathbb{F}_{q^d}$. Output the ciphertext $CT = (\mathbf{g}^{\mathbf{r}}, \mathbf{h}^{\mathbf{r}} \cdot M)$.
- $\mathsf{Dec}(sk, CT)$: Write the ciphertext as $CT = (\mathbf{e}, \mathbf{c})$. Output $M' = c \cdot (\mathbf{e}^{\mathbf{x}})^{-1}$

The proof of correctness and security follows immediately as for standard Elgamal.

5.3 Applications in general

Having seen the two examples above, it should not be surprising that all DDH-based cryptographic schemes we are aware of can be based on d-DDH instead. This is basically because all involved algorithms (such as key generation, encryption, and security reduction) will work given only black-box access to a group $\mathcal{G}$ and a finite field K. We just need that for $g \in \mathcal{G}$ and $x \in K$, $g^x \in \mathcal{G}$ is well-defined and standard "axioms" such as $g^{x+y} = g^x g^y$ and $(g^x)^y = g^{xy}$ hold. The exact same scheme and security proof can be run, based on $(\mathcal{G}, K) = (\mathbb{G}, \mathbb{F}_q)$ or based on $(\mathcal{G}, K) = (\mathbb{G}^d, \mathbb{F}_{q^d})$. The only difference is that we need the d-DDH assumption in the latter case. Thus, for instance, CCA secure encryption [CS98] and circular secure or auxiliary input secure encryption [BHHO08] follow immediately from d-DDH.

5.4 Efficiency

For all constructions mentioned here, we can define a notion of amortized complexity. For a PRF, this is the computation time needed to produce a single pseudorandom group element; for an encryption scheme it is the computation time needed to encrypt a group element.

An important point is that in all applications we are aware of, the amortized complexity is essentially the same for constructions based on DDH and on d-DDH. This is because for $\mathbf{g} \in \mathbb{G}^d$ and $\mathbf{a} \in \mathbb{F}_{q^d}$, $\mathbf{g}^{\mathbf{a}}$ corresponds to a tuple of length d where each entry is an expression of the form $\prod g_i^{\alpha_i}$. By a well-known algorithm (see [Pip76]) such a value can be computed in time roughly what you need for a single exponentiation in $\mathbb{G}$.

As a concrete example, computing the PRF defined above requires essentially a single exponentiation: $\mathbf{g}^{(\mathbf{a}_0 \prod_{x_i=1} \mathbf{a}_i)}$. This produces d pseudorandom elements at amortized cost roughly 1 exponentiation in $\mathbb{G}$, which is the same cost as the DDH based version.

Various optimizations are known that save computation in the constructions we consider here. However, all the optimizations we are aware of can be applied to both variants based on DDH and d-DDH, and therefore do not affect our conclusion on the amortized complexities.

6 The Vector DDH Problem

The main observation in this section is that we can construct a problem that is generically harder than DDH by revealing only the last entry of the final vector in an f-DDH instance. In the following, we study in detail what happens if we choose f to be x^d. It turns out that there is a simple way of expressing products in $R_d = \mathbb{F}_q[X]/(x^d)$. If we take $\mathbf{x} = (x_0, \ldots, x_{d-1})$ and $\mathbf{y} = (y_0, \ldots, y_{d-1})$ in R_d, we have:

$$\mathbf{xy} = \left(x_0 y_0, \ldots, \sum_{k=0}^{i-1} x_k y_{i-1-k}, \ldots, \sum_{k=0}^{d-1} x_k y_{d-1-k} \right) . \tag{2}$$

We define the d-VDDH problem just like d-DDH, except that the problem instance is now of the form $(h(\mathbf{w}), h(\mathbf{wa}), h(\mathbf{wb}), h(\mathbf{wc})[d])$, where we recall that $\mathbf{x}[d]$ is the dth entry of the vector $\mathbf{x}$, that is x_{d-1} if we start numbering from 0.

Definition 4 (The d-VDDH Problem). *Let d be an integer. Let $\mathcal{G}$ be a PPT algorithm, which given the security parameter λ, outputs the description of a group $\mathbb{G}$ of order $q = q(1^\lambda)$. Let $\mathcal{A}$ be a probabilistic algorithm that takes as input (a description of) $\mathbb{G}$ and a 3-tuple in $\mathbb{G}^d$ plus an element in $\mathbb{G}$, and outputs 0 or 1.*

We say that $\mathcal{A}$ solves the d-VDDH problem with advantage $\varepsilon_{\mathcal{A}}(\lambda)$, where

$$\varepsilon_{\mathcal{A}}(\lambda) = |Pr[\mathcal{A}(\mathbb{G}, (\boldsymbol{g}, \boldsymbol{g}^{\boldsymbol{a}}, \boldsymbol{g}^{\boldsymbol{b}}, \boldsymbol{g}^{\boldsymbol{c}}[d])) = 1] - Pr[\mathcal{A}(\mathbb{G}, (\boldsymbol{g}, \boldsymbol{g}^{\boldsymbol{a}}, \boldsymbol{g}^{\boldsymbol{b}}, \boldsymbol{g}^{\boldsymbol{ab}}[d])) = 1]|$$

where $\boldsymbol{g} \leftarrow \mathbb{G}^d$ and $\boldsymbol{a} \leftarrow R_d, \boldsymbol{b} \leftarrow R_d, \boldsymbol{c} \leftarrow R_d$.

Definition 5 (The d-VDDH Assumption). *For any probabilistic polynomial time algorithm $\mathcal{A}$ as in Definition 4, it holds that $\varepsilon_{\mathcal{A}}(\lambda)$ is negligible as a function of λ.*

Recall the notation from Section 3: $\mathbf{g} = (g_0 \ldots, g_{d-1}) = (h^{w_0}, \ldots, h^{w_{d-1}})$. Note that we WLOG can choose $w_0 = 1$, so $h(\mathbf{w}) = (g_0, g_0^{w_1} \ldots, g_0^{w_{d-1}})$. To prove that d-VDDH is generically hard, even in d-linear groups, it is useful to do the following parameter substitution: set $\mathbf{x} = \mathbf{wa}$, $\mathbf{y} = \mathbf{wb}$. The d-VDDH problem now becomes deciding whether the last element is the dth coordinate of $\mathbf{xyw}^{-1}$ or is random.

Now, set $\mathbf{w}^{-1} = (z_0, z_1, ..., z_{d-1})$ and consider the z_i as unknowns. Since $\mathbf{ww}^{-1} = \mathbf{1} = (1, 0, ..., 0)$ we get $d-1$ equations involving the z_i's, using the product introduced in (2):

$$z_0 = 1,\ z_1 = -w_1,\ \ldots,\ z_i = -w_i - \sum_{j+l=i} z_l w_j,\ \ldots,\ z_{d-1} = -w_{d-1} - \sum_{j+l=d-1} z_l w_j$$

In particular, $z_i = -w_1 z_{i-1} - \cdots - w_{i-1} z_1 - w_i$. Hence, it can be proved by simple induction that z_i has degree i as a function of the w_j's. Now, let $p_i(\mathbf{w},\mathbf{x},\mathbf{y})$ be the ith entry of $\mathbf{w}^{-1}\mathbf{xy}$. Then $p_d(\mathbf{w},\mathbf{x},\mathbf{y})$ has degree $d+1$ in $\mathbf{w}, \mathbf{x}$, and $\mathbf{y}$.
We are now ready to prove the generic hardness of d-VDDH.

Theorem 8. *Even given a d-linear mapping, the d-VDDH holds in the generic group model.*

The proof can be found in the full paper [CDK+11].

Later, we will need a lemma stating that a generalization of the d-VDDH which considers several generators is equivalent to the original assumption.

Lemma 1. *If d-VDDH is hard for $\mathcal{G}$, then for any positive integer m*

$$\left\{(\boldsymbol{g}_1, \ldots, \boldsymbol{g}_m, \boldsymbol{g}_1^{\boldsymbol{r}}[d], \ldots, \boldsymbol{g}_m^{\boldsymbol{r}}[d]) \mid \boldsymbol{g}_i \leftarrow \mathbb{G}^d, \boldsymbol{r} \leftarrow R_d\right\} \overset{c}{\approx} \quad (3)$$

$$\left\{(\boldsymbol{g}_1, \ldots, \boldsymbol{g}_m, \boldsymbol{g}_1^{\boldsymbol{r}_1}[d], \ldots, \boldsymbol{g}_m^{\boldsymbol{r}_m}[d]) \mid \boldsymbol{g}_i \leftarrow \mathbb{G}^d, \boldsymbol{r}_i \leftarrow R_d\right\}. \quad (4)$$

The proof can be found in the full paper [CDK+11].

7 Applications of d-VDDH

In this section we discuss a number of natural application of our d-VDDH assumption. Throughout this section we will use the ring $R_d = \mathbb{F}_q[X]/(f)$ for $f = X^d$.

7.1 Public Key Encryption

It is immediate how to construct a CPA-secure encryption schemes from the d-VDDH assumption family. We now show how to extend them to chosen-ciphertext (CCA) secure schemes. Let us first recall the definition of chosen-ciphertext security for encryption schemes.

Definition 6. *A scheme* PKE *is CCA secure if for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, any polynomial* p *and large enough* λ*,*

$$\mathsf{Adv}_{\mathcal{A},h} := \left|\Pr[\mathsf{CCA}_0(\mathsf{PKE}, \mathcal{A}, 1^\lambda)] - \Pr[\mathsf{CCA}_1(\mathsf{PKE}, \mathcal{A}, 1^\lambda)]\right| < 1/p(\lambda),$$

where $\mathsf{CCA}_b(\mathsf{PKE}, \mathcal{A}, 1^\lambda)$ *is output from the following experiment:*

$$\begin{aligned}(pk, sk) &\leftarrow \mathcal{G}(1^\lambda)\\ (m_0, m_1, \mathsf{state}) &\leftarrow \mathcal{A}_1^{\mathsf{Dec}(sk,\cdot)}(1^\lambda, pk) \text{ with } |m_0| = |m_1|\\ CT^* &\leftarrow \mathsf{Enc}_{pk}(m_b),\\ \text{Output } b' &\leftarrow \mathcal{A}_2^{\mathsf{Dec}(sk,\cdot)}(1^\lambda, \mathsf{state}, CT^*)\end{aligned}$$

In the second phase the decryption oracle $\mathsf{Dec}(sk, \cdot)$ returns $\perp$ when queried on the challenge ciphertext CT^*.

We now give the construction of our CCA secure encryption scheme. Let (E, D) be a symmetric encryption scheme with key-space $K \in \mathbb{G}$. Let $\mathsf{T} : \mathbb{G}^d \to \mathbb{F}_q$ be a target collision resistant hash function (see [HK07] for a definition) and define $\hat{\mathsf{T}}(\mathbf{x}) := (\mathsf{T}(x), 0, \ldots, 0) \in R_d$. (Note that for two elements $\mathbf{x} \neq \mathbf{y}$ we have that $\hat{\mathsf{T}}(\mathbf{x}) - \hat{\mathsf{T}}(\mathbf{y})$ is invertible in R_d unless $\mathsf{T}(\mathbf{x}) = \mathsf{T}(\mathbf{y})$).

- $\mathsf{Gen}(1^\lambda)$: Let $\mathbb{G} \leftarrow \mathcal{G}(1^\lambda)$. Choose a random generator $\mathbf{g}_1 \leftarrow \mathbb{G}^d$ and random indices $\mathbf{w}, \mathbf{x}, \mathbf{y} \leftarrow R_d$. Compute $\mathbf{g}_2 = \mathbf{g}^{\mathbf{w}}, \mathbf{u} = \mathbf{g}^{\mathbf{x}}, \mathbf{v} = \mathbf{g}^{\mathbf{y}}$. The secret key is then $sk = \mathbf{w}, \mathbf{x}, \mathbf{y}$ and the public key is $pk = (\mathbb{G}, \mathbf{g}_1, \mathbf{g}_2, \mathbf{u}, \mathbf{v})$.
- $\mathsf{Enc}(pk, M)$: Choose randomly $\mathbf{r} \leftarrow R_d$. Compute $\mathbf{c}_1 = \mathbf{g}^{\mathbf{r}}$ and $\mathbf{c}_2 = (\mathbf{u}^{\mathbf{t}}\mathbf{v})^{\mathbf{r}}$, where $\mathbf{t} = \hat{\mathsf{T}}(\mathbf{c}_1) \in R_d$. Compute the symmetric part as $C = \mathsf{E}_K(m)$, where $K = \mathbf{g}_2^{\mathbf{r}}[d]$. Output the ciphertext $CT = (\mathbf{c}_1, \mathbf{c}_2, C)$.
- $\mathsf{Dec}(sk, CT)$: Write the ciphertext as $CT = (\mathbf{c}_1, \mathbf{c}_2, C)$. If $\mathbf{c}_1^{\mathbf{x}\cdot\mathbf{t}+\mathbf{y}} \neq \mathbf{c}_2$ then return $\perp$. Otherwise return $\mathsf{D}_K(C)$, where $K = \mathbf{c}_1^{\mathbf{w}}[d]$.

It is easy to see that correctness follows by the definition of the public/secret key and by the correctness of the symmetric scheme. To prove the theorem we need that symmetric scheme is secure in the sense of authenticated encryption. That is, it acts as a one-time pad plus any decryption query (with respect to a uniform random key) is rejected. We refer again to [HK07] for a formal definition.

Theorem 9. *If* (E, D) *is a symmetric encryption scheme secure in the sense of authenticated encryption,* T *is a target collision resistant hash function and the* d*-VDDH holds in* $\mathbb{G}$*, then the encryption scheme is IND-CCA secure.*

The proof is exactly the same as Theorem 2 in [HK07] where an encryption scheme is proved CCA secure from the DDH assumption. We give some intuition about the proof.

The difficulty in the security reduction is that an adversary against the d-VDDH assumption has to answer the decryption queries and hence has to distinguish between consistent ciphertexts (i.e., ciphertexts for that $\mathbf{c}_1^{\mathbf{xt}+\mathbf{y}} = \mathbf{c}_2$ holds) and inconsistent ones, without knowing $\mathbf{w} = \log_{\mathbf{g}_1} \mathbf{g}_2$. The simulator inputs $(\mathbf{g}_1, \mathbf{g}_2, \mathbf{c}_1^* = \mathbf{g}_1^{\mathbf{r}}, K^*)$ and wants to distinguish $K^* = \mathbf{g}_2^{\mathbf{r}^*}[d]$ from a uniform element in $\mathbb{G}$. In the simulation the values $\mathbf{u}, \mathbf{v}$ from the public-key are set-up such that the tuple $\mathbf{c}_1^*, \mathbf{c}_2^*$ can be used as the challenge ciphertext for some efficiently computable $\mathbf{c}_2^*$ and the value K^* as the symmetric key. More precisely, we define $\mathbf{u} = \mathbf{g}_1^{\mathbf{x}_1}\mathbf{g}_2^{\mathbf{x}_2}, \mathbf{v} = \mathbf{g}_1^{\mathbf{y}_1}\mathbf{g}_2^{-\mathbf{t}^* \cdot \mathbf{x}_2}$ for uniform $\mathbf{x}_1, \mathbf{y}_1 \in R_d$, $\mathbf{x}_2 \in R_d^*$ and $\mathbf{t}^* = \hat{\mathsf{T}}(\mathbf{c}_1^*)$. By construction, the corresponding real session key is $\mathbf{g}_2^{\mathbf{r}^*}[d]$ so breaking the indistinguishability of the scheme is equivalent to solving the d-VDDH problem. It leaves to deal with the decryption queries for $CT = (\mathbf{c}_1, \mathbf{c}_2, C)$. The simulator is not able to distinguish consistent from inconsistent ciphertexts. However, for ciphertexts with $\mathbf{t} = \hat{\mathsf{T}}(\mathbf{c}_1) \neq \mathbf{t}^*$ (these are the interesting cases) the simulator implements an alternative decryption algorithm by computing the symmetric key as $K = (\mathbf{c}_1 \mathbf{c}_2^{-\mathbf{x}_1\mathbf{t}+\mathbf{y}_1})^{(\mathbf{x}_2(\mathbf{t}-\mathbf{t}^*))^{-1}}[d]$. (Note that by the properties of $\hat{\mathsf{T}}$, $\mathbf{x}_2(\mathbf{t} - \mathbf{t}^*) \in R^*$ so its inverse is well-defined.) This has the following consequences.

It is easy to verify that if the queried ciphertext is consistent then the alternative decryption algorithm yields the correct symmetric key $K = \mathbf{c}_1^{\mathbf{w}}$. If the queried ciphertext is inconsistent then the alternative decapsulation algorithm yields one single symmetric key K that is uniformly distributed over $\mathbb{G}$. (The probability space is taken over all possible $\mathbf{x}_1, \mathbf{x}_2, \mathbf{y}_1$ that yield $\mathbf{u}, \mathbf{v}$ from the public-key given to the adversary.) Returning this key K to the adversary would completely determine the simulator's secret key and hence also the virtual symmetric key K' for the next decapsulation query. However, this key K is used to decrypt the symmetric part C of the decryption query and by the authenticity property of the latter this will always lead to a rejection. Hence the decryption query is answered correctly and no information about the secret key is leaked which makes it possible to apply the same argument again.

7.2 Generalized BHHO Encryption

In this section we define a public-key encryption scheme which is heavily inspired by the scheme in [BHHO08]. Here, however, the cryptosystem is based on d-VDDH, instead of DDH.

Let λ be the security parameter and $m = m(\lambda)$ be a parameter of the scheme. The encryption scheme is $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.

- $\mathsf{Gen}(1^\lambda)$: Let $\mathbb{G} \leftarrow \mathcal{G}(1^\lambda)$. Choose a vector of uniformly random generators $\mathbf{g} = (\mathbf{g}_1, \ldots, \mathbf{g}_m), \mathbf{g}_i \leftarrow \mathbb{G}^d$ and random bit string $\mathbf{s} = (s_1, \ldots, s_m) \leftarrow \{0,1\}^m$. Compute $\mathbf{y} = \prod_{i=1}^m \mathbf{g}_i^{(s_i,0,\ldots,0)}$, where $(s_1, 0, \ldots, 0)$ is viewed as an element in R_d. The secret key is then $sk = \mathbf{s}$ and the public key is $pk = (\mathbb{G}, \mathbf{g}, \mathbf{y})$, where $\mathbb{G}$ and $\mathbf{g}$ can be considered public parameters.
- $\mathsf{Enc}(pk, M)$: Let the message be $M \in \mathbb{G}$. Choose randomly $\mathbf{r} \leftarrow R_d$. Compute $f_i = \mathbf{g}_i^{\mathbf{r}}[d]$ and output the ciphertext $CT = (f_1, \ldots, f_m, \mathbf{y}^{\mathbf{r}}[d] \cdot M)$.
- $\mathsf{Dec}(sk, CT)$: Write the ciphertext as $CT = (f_1, \ldots, f_m, c)$. Output $M' = c \cdot (\prod_{i=1}^m f_i^{s_i})^{-1}$

Correctness of decryption follows since

$$\begin{aligned}\prod_{i=1}^m f_i^{s_i} &= \prod_{i=1}^m (\mathbf{g}_i^{\mathbf{r}}[d])^{s_i} = \prod_{i=1}^m \left(g_{i1}^{r_d} \cdot g_{i2}^{r_{d-1}} \cdots g_{id}^{r_1}\right)^{s_i} = \prod_{i=1}^m \left(g_{i1}^{r_d s_i} \cdot g_{i2}^{r_{d-1} s_i} \cdots g_{id}^{r_1 s_i}\right) \\ &= \prod_{i=1}^m (g_{i1}^{s_i}, \ldots, g_{id}^{s_i})^{(r_1,\ldots,r_d)}\,[d] = \prod_{i=1}^m (g_{i1}, \ldots, g_{id})^{(s_i,0\ldots,0)(r_1,\ldots,r_d)}\,[d] \\ &= \prod_{i=1}^m \mathbf{g}_i^{(s_i,0,\ldots,0)\mathbf{r}}[d] = \mathbf{y}^{\mathbf{r}}[d]\end{aligned}$$

CPA security in the usual sense follows immediately from Lemma 1. We will, however, argue that the scheme is also leakage resilient in the auxiliary input model.

Auxiliary Input Security The definition of security w.r.t auxiliary inputs is exactly as in [DGK+10].

Definition 7. *A scheme* PKE *is CPA secure w.r.t. auxiliary inputs from a function class* $\mathcal{H}$ *if for any function* $h \in \mathcal{H}$*, any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, any polynomial* p *and large enough* λ*,*

$$\mathsf{Adv}_{\mathcal{A},h} := \left|\Pr[\mathsf{CPA}_0(\mathsf{PKE}, \mathcal{A}, 1^\lambda, h)] - \Pr[\mathsf{CPA}_1(\mathsf{PKE}, \mathcal{A}, 1^\lambda, h)]\right| < 1/p(\lambda),$$

where $\mathsf{CPA}_b(\mathsf{PKE}, \mathcal{A}, 1^\lambda, h)$ *is output from the following experiment:*

$$\begin{aligned}(pk, sk) \leftarrow& \mathsf{Gen}(1^\lambda) \\ (m_0, m_1, \mathsf{state}) \leftarrow& \mathcal{A}_1(1^\lambda, pk, h(sk, pk)) \text{ with } |m_0| = |m_1| \\ CT^* \leftarrow& \mathsf{Enc}_{pk}(m_b), \\ \text{Output } b' \leftarrow& \mathcal{A}_2(1^\lambda, \mathsf{state}, CT^*)\end{aligned}$$

The functions we will consider are those where the secret key is hard to compute even given the leakage. More precisely, $\mathcal{H}_{ow}(f(k))$ consists of all PT functions $h : \{0,1\}^{|sk|+|pk|} \to \{0,1\}^*$ s.t. given $h(sk, pk)$ (for $(sk, pk) \leftarrow \mathsf{Gen}(1^\lambda)$), no PPT algorithm can find sk with probability greater that $f(k)$. A scheme secure w.r.t auxiliary inputs from $\mathcal{H}_{ow}(f(k))$ is called $f(k)$-AI-CPA secure.

We are now ready to state the theorem about the security of our scheme.

Theorem 10. *Let $m = (4 \log q^d)^{1/\varepsilon}$, for some $\epsilon > 0$. Assuming that d-VDDH is hard for $\mathcal{G}$, the scheme above is $\left(2^{-m^\varepsilon}\right)$-AI-CPA secure.*

The complete details of the proof of Theorem 10 are given in the appendix of the full version [CDK$^+$11]. Based on Lemma 1, it follows the exact same lines as in the proof in [DGK$^+$10].

There is a trade-off between the ciphertext size and the hardness of the leakage functions that we can protect against. Obtaining security against functions that are 2^{-m^ε}-hard to invert, requires that $m = (4 \log q^d)^{1/\varepsilon}$ instead of $m = (4 \log q)^{1/\varepsilon}$, which is a polynomial overhead in the ciphertext size.

We point out that, even though this generalized version of BHHO schemes is auxiliary input secure, KDM security does not follow using our implementation with d-VDDH assumption.

7.3 Pseudorandom Functions

In this section we present a construction for pseudorandom functions (see Definition 3) based on the d-VDDH assumption. This construction is a modification of the DDH-based one in [NR97].

PRF Construction We construct a function family $F = \{f_k\}$ as follows. The index k specifies a tuple $(q, \mathbb{G}, g_1, g_2, e, \mathbf{a}_0, \ldots, \mathbf{a}_n)$ where q is a prime number, $\mathbb{G}$ is a group of order q, g_1, g_2 are two generators of $\mathbb{G}$, $e : \mathbb{G}^2 \to \mathbb{G}_T$ is a bilinear map and $\mathbf{a}_0, \ldots, \mathbf{a}_n$ are random in R_2. For any such index k we denote $t_1 = e(g_1, g_1), t_2 = e(g_2, g_1)$ and $\mathbf{t} = (t_1, t_2)$. Finally, we define $f_k : \{0,1\}^n \to \mathbb{G}_T$, $f_k(x_1, \ldots, x_n) = \mathbf{t}^{\mathbf{a}_0 \prod_{x_i=1} \mathbf{a}_i}[2]$.

Theorem 11. *Under the 2-VDH assumption, the family $F = \{f_k\}$ defined above is a family of pseudorandom functions.*

We refer to the full version [CDK$^+$11] for the proof of this theorem.

References

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2004.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.

[BHHO08] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision diffie-hellman. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 108–125. Springer, 2008.

[BMR10] Dan Boneh, Hart William Montgomery, and Ananth Raghunathan. Algebraic pseudorandom functions with improved efficiency from the augmented cascade. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM Conference on Computer and Communications Security*, pages 131–140. ACM, 2010.

[CDK$^+$11] Ronald Cramer, Ivan Damgaard, Eike Kiltz, Sarah Zakarias, and Angela Zottarel. Ddh-like assumptions based on extension rings. Cryptology ePrint Archive, Report 2011/280, 2011. http://eprint.iacr.org/.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 1998.

[DGK$^+$10] Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In Daniele Micciancio, editor, *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 361–381. Springer, 2010.

[DH76] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[DY05] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In Serge Vaudenay, editor, *Public Key Cryptography*, volume 3386 of *Lecture Notes in Computer Science*, pages 416–431. Springer, 2005.

[Gam84] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO*, pages 10–18, 1984.

[HK07] Dennis Hofheinz and Eike Kiltz. Secure hybrid encryption from weakened key encapsulation. In *CRYPTO*, pages 553–571, 2007.

[HYZX08] Huawei Huang, Bo Yang, Shenglin Zhu, and Guozhen Xiao. Generalized elgamal public key cryptosystem based on a new diffie-hellman problem. In Joonsang Baek, Feng Bao, Kefei Chen, and Xuejia Lai, editors, *ProvSec*, volume 5324 of *Lecture Notes in Computer Science*, pages 1–21. Springer, 2008.

[Kil07] Eike Kiltz. Chosen-ciphertext secure key-encapsulation based on gap hashed diffie-hellman. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 282–297. Springer, 2007.

[NR97] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *FOCS*, pages 458–467, 1997.

[Pip76] Nicholas Pippenger. On the evaluation of powers and related problems (preliminary version). In *FOCS*, pages 258–263, 1976.

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27:701–717, October 1980.

[Sha07] Hovav Shacham. A Cramer-Shoup encryption scheme from the Linear Assumption and from progressively weaker Linear variants. Cryptology ePrint Archive, Report 2007/074, February 2007. http://eprint.iacr.org/.

[Sta96] Markus Stadler. Publicly verifiable secret sharing. In *EUROCRYPT*, pages 190–199, 1996.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *EUROSAM*, pages 216–226, 1979.