A Tool Box of Cryptographic Functions related to the Diffie-Hellman Function*

Eike Kiltz

Lehrstuhl Mathematik & Informatik, Fakultät für Mathematik,
Ruhr-Universität Bochum, 44780 Bochum, Germany.
kiltz@lmi.ruhr-uni-bochum.de, http://www.ruhr-uni-bochum.de/lmi/kiltz/

Abstract. Given a cyclic group G and a generator g, the *Diffie-Hellman* function (DH) maps two group elements (g^a, g^b) to g^{ab}. For many groups G this function is assumed to be hard to compute. We generalize this function to the *P-Diffie-Hellman* function (P-DH) that maps two group elements (g^a, g^b) to $g^{P(a,b)}$ for a (non-linear) polynomial P in a and b. In this paper we show that computing DH is *computational equivalent* to computing P-DH. In addition we study the corresponding decision problem for the function P-DH. In sharp contrast to the computational case the decision problems for DH and P-DH can be shown to be *not generically equivalent* for most polynomials P. Furthermore we show that there is no *generic algorithm* that computes or decides the P-DH function in polynomial time.

Keywords: Complexity Theory, Generic Algorithms, Diffie-Hellman function, Square Exponent function.

1 Introduction

Let G be a cyclic finite group and let g be a generator of G. The Diffie-Hellman function, $\mathrm{DH} : G \times G \to G$ is given by $\mathrm{DH}(g^a, g^b) = g^{ab}$. This function is used, for instance, in the Diffie-Hellman cryptosystem [3]. Here two parties, say Alice and Bob, agree on a common pair (G, g), a is the private key of Alice, b is the private key of Bob, g^a is sent from Alice to Bob, g^b is sent vice-versa, and finally both of them are able to compute g^{ab}. The Computational Diffie-Hellman assumption claims that the function DH is hard to evaluate.

In this work we are generalizing the Diffie-Hellman function in the following way. Let $P(a, b)$ be a function in a and b. We define the P-Diffie-Hellman function, P-DH: $G \times G \to G$ as

$$P\text{-DH}(g^a, g^b) := g^{P(a,b)}.$$

Clearly, the Diffie-Hellman function is achieved by setting $P(a, b) = ab$. We will restrict our studies to the case where P is a *non-linear polynomial* in a and b.

* A short version of this paper appeared in the proceedings of INDOCRYPT'01 [4].

For example if $P(a,b) = a^3b + 2ab^4$ then the P-Diffie-Hellman function is defined as $P\text{-DH}(g^a, g^b) = g^{a^3b+2ab^4}$.
The function that computes $g^{(a^2)}$ from g^a is called the *Square Exponent* function. A motivation for the analysis of this variant of the Diffie-Hellman function is that certain cryptographic systems exist whose security relies on the hardness of this function. An example is a scheme for key escrow with limited time span [1]. Maurer and Wolf [5] prove the equivalence of computing the Diffie-Hellman function and computing the Square Exponent function. Further theoretical research about the Square Exponent function was done [2, 9].
Clearly computing the DH function cannot be harder than computing the P-DH function for a polynomial $P(a,b)$. In Section 3 we also show the converse direction, i.e. that computing the Diffie-Hellman function is *computational equivalent* to computing the P-DH function for non-linear polynomials $P(a,b)$. As we will see, the strength of our result will depend on the smallest prime factor of the group order. In Section 4 we study the corresponding decision problem: For random group elements g^a, g^b and (in random order) g^c and $g^{P(a,b)}$ decide between g^c and $g^{P(a,b)}$. In sharp contrast to the results in Section 3 we show that the decision problem for the Diffie-Hellman function and the P-Diffie-Hellman function are provable *not generically equivalent* for most polynomials $P(a,b)$. On the other hand we show that no efficient generic algorithm can decide the P-Diffie-Hellman function for any non-linear polynomial P. Finally, in Section 5 we give a summary of our results and mention some open problems.

2 Definitions

We say that an algorithm is *efficient* if it runs in probabilistic polynomial time. We call a function α *negligible* in n if $\alpha(n) < 1/P(n)$ holds for every polynomial P and for sufficiently large n.

P-Diffie-Hellman function. Let G be a finite cyclic group whose order $|G|$ is an n-bit integer. Let $\mathbb{Z}_{|G|}$ denote the ring of integer residue classes modulo $|G|$. Let $k = k(n)$ and $l = l(n)$ be two functions in n mapping integers to integers. Let $\mathbb{P}_l^k = \mathbb{P}_l^k(n)$ be the family of sets of all non-linear polynomials $P(a,b)$ over $\mathbb{Z}_{|G|}$ of the form

$$P(a,b) = \sum_{i,j \in \{0 \ldots l\}} c_{ij} a^i b^j$$

with coefficients $c_{ij} \in \mathbb{Z}_{|G|}$ and absolute values[1] $|c_{ij}|$ bounded by k. We restrict the polynomials $P(a,b)$ to non-linear polynomials, i.e. at least for one (i,j) with $i+j \geq 2$, $c_{ij} \neq 0$ must hold. To simplify our notation we introduce $\mathbb{P}_l := \mathbb{P}_l^{\lfloor |G|/2 \rfloor}$ (no restrictions to coefficients) and $\mathbb{P} := \mathbb{P}_{|G|-1}$.

[1] The absolute value of an element $x \in \mathbb{Z}_{|G|}$ is defined as $|x| = \min\{x, |G| - x\}$.

For a cyclic, finite group G, a fixed generator g of G and a polynomial $P \in \mathbb{P}$ we define the *P-Diffie-Hellman function*, P-DH: $G \times G \to G$ as

$$P\text{-DH}(g^a, g^b) := g^{P(a,b)},$$

where P is called the *defining polynomial* of the P-Diffie-Hellman function.

EXAMPLES of the P-DH function are:

Name	Defining polynomial	P-Diffie-Hellman function
Diffie-Hellman function [3]	$P(a,b) = ab$	$\text{DH}(g^a, g^b) = g^{ab}$
Square Exponent function [5]	$P(a,b) = a^2$	$\text{SE}(g^a) = g^{(a^2)}$
To-the-s Diffie-Hellman function	$P(a,b) = a^s$	$\text{DH}^s(g^a) = g^{(a^s)}$
-	$P(a,b) = (a+b)^n$	$P\text{-DH}(g^a, g^b) = g^{(a+b)^n}$
-	$P(a,b) = a^2b + ab^2$	$P\text{-DH}(g^a, g^b) = g^{a^2b+ab^2}$

CONSIDERED GROUP FAMILIES. For groups G whose *maximal* prime factor q of the group order is "small", computing P-DH(g^a, g^b) is easy because of the Pohlig-Hellman algorithm [6]. This algorithm computes the discrete logarithm in running time $O(\sqrt{q})$. Let $\mathcal{G} := (G_n, g_n)_{n\in\mathbb{N}}$ be a family of finite cyclic groups and generators. We define $\mathbb{G}$ as the set of all families $\mathcal{G}$, where the bitlength of the (efficiently computable) group order $|G_n|$ is of the order n. We define $\mathbb{G}(\text{nsprime}) := \{\mathcal{G} : \forall \text{polynomials } R\ \exists n_0 \forall n \geq n_0 : \text{minpf}(|G_n|) > R(n)\}$ as the set of all families $\mathcal{G}$ such that the minimal prime factor of the group order $|G_n|$ is larger than any polynomial (nsprime stands for "no small prime factor").

COMPUTATIONAL ASSUMPTIONS. Let $(G, g) = (G_n, g_n)_{n\in\mathbb{N}} = \mathcal{G}$ be a family of groups and generators[2] and let $\epsilon(n)$ be a function in n taking values in the interval $[0, 1]$. For a $P \in \mathbb{P}$ the *$\epsilon(n)$-P Computational Diffie-Hellman assumption for $\mathcal{G}$* ($\epsilon(n)$-P-CDH$(\mathcal{G})$) is: There is no efficient algorithm that, given random group elements g^a and g^b, outputs $g^{P(a,b)}$ with probability at least $\epsilon(n)$ (taken over the uniformly distributed input and coin tosses of the algorithm).
We define $\epsilon(n)$-CDH$(\mathcal{G})$ as the assumption $\epsilon(n)$-P-CDH$(\mathcal{G})$ for $P(a,b) := ab$ and $\epsilon(n)$-CSE$(\mathcal{G})$ as the assumption $\epsilon(n)$-Q-CDH$(\mathcal{G})$ for $Q(a,b) := a^2$.
The strongest possible assumption considered here that for all polynomials R there is no efficient algorithm that, given g^a and g^b, outputs $g^{P(a,b)}$ with (asymptotical) probability at least $1/R(n)$ is denoted as $\frac{1}{\text{poly}(n)}$-P-CDH$(\mathcal{G})$. Vice-versa, the weakest possible assumption that there is no efficient algorithm that, given g^a and g^b, outputs $g^{P(a,b)}$ with probability $1 - \alpha(n)$, where $\alpha(n)$ is a negligible function in n, is denoted as P-CDH$(\mathcal{G})$.
We say that assumption

- $\epsilon(n)$-P-CDH holds, if $\epsilon(n)$-P-CDH$(\mathcal{G})$ holds for every family $\mathcal{G} \in \mathbb{G}$.
- $\epsilon(n)$-P-CDH$_{\text{nsprime}}$ holds, if $\epsilon(n)$-P-CDH$(\mathcal{G})$ holds for every family $\mathcal{G} \in \mathbb{G}(\text{nsprime})$.

[2] By abuse of notation we often use (G, g) instead of $(G_n, g_n)_{n\in\mathbb{N}}$.

RELATIONS. To express relations among assumptions we will use the following notation: $A \Rightarrow B$ means that if assumption A holds, so does assumption B.Vice-versa, it also means that if there is a (polynomial-time) algorithm $\mathcal{A}_B$ breaking assumption B then we can build another (polynomial-time) algorithm $\mathcal{A}_A^{\mathcal{A}_B}$ with (oracle) access to $\mathcal{A}_B$ which breaks assumption A. $A \Leftrightarrow B$ means that $A \Rightarrow B$ and $B \Rightarrow A$, i.e. A and B are assumptions of the same (polynomial) complexity.

GENERIC ALGORITHMS (Notation of Shoup [8]). An encoding function on the additive group $(\mathbb{Z}_m, +)$ is an unknown injective map $\sigma : \mathbb{Z}_m \to \{0,1\}^n$ for some integer n. For a generic algorithm nothing is known about the structure (representation) of the underlying algebraic group. More precisely a generic algorithm $\mathcal{A}$ for $\mathbb{Z}_m$ is a probabilistic algorithm that takes as input an encoding list $(\sigma(x_1), \ldots, \sigma(x_k))$ where σ is an encoding function. Operations can only be performed via addition and subtraction oracles which given two indices i, j, return the encoding of $\sigma(x_i + x_j)$ and $\sigma(x_i - x_j)$ respectively. Both oracles can be considered as black boxes. The new encoding is then added to the encoding list. The output of the algorithm is denoted by $\mathcal{A}(\sigma; x_1, \ldots, x_k)$. An example of a generic algorithm is the Pohlig-Hellman algorithm [6] that computes the discrete logarithm.
Relations between assumptions that make only use of generic reduction algorithms are marked by the appearance of σ. For instance, $A \overset{\sigma}{\not\Rightarrow} B$ means that no efficient reduction is possible when computation is restricted to generic algorithms. And $\mathit{true} \overset{\sigma}{\Rightarrow} B$ means that assumption B is always true when computation is restricted to generic algorithms, i.e. no efficient generic algorithm can break assumption B. Note that such "impossibility statements" for generic algorithms are very weak, because problems might get substantially easier when adding an encoding to the group G. For instance, the discrete logarithm is known to be hard to compute for generic algorithms but in the additive group $(\mathbb{Z}_m, +)$ with generator 1 the discrete logarithm of $x \in \mathbb{Z}_m$ is simply given by x itself. Of course this is trivializing the computation.

3 The Computational Case

In this section we deal with assumptions regarding the computational case of the P-Diffie-Hellman function. Again we want to point out that our results are phrased in terms of assumptions. For instance P-CDH is the assumption that there is *no efficient algorithm* that computes the P-Diffie-Hellman function. Recall that to show that assumption A implies assumption B ($A \Rightarrow B$) we have to prove that if there is an algorithm $\mathcal{A}_B$ breaking assumption B then we can build another algorithm $\mathcal{A}_A^{\mathcal{A}_B}$ with (oracle) access to $\mathcal{A}_B$ which breaks assumption A. This might be a little bit confusing or contra-intuitive at first sight but it simplifies notation.

3.1 Previous Work

Theorem 1. *1.* $true \stackrel{\sigma}{\Rightarrow} \frac{1}{\mathrm{poly}(n)}$-$\mathrm{CDH}_{\mathrm{nsprime}}$ *(Shoup [8]).*
2. $true \stackrel{\sigma}{\Rightarrow} \frac{1}{\mathrm{poly}(n)}$-$\mathrm{CSE}_{\mathrm{nsprime}}$ *(Wolf [10]).*
3. $\frac{1}{\mathrm{poly}(n)}$-CDH $\Leftrightarrow$ CDH *(Shoup's Diffie-Hellman self-corrector [8]).*
4. $\frac{1}{\mathrm{poly}(n)}$-CDH $\Leftrightarrow$ $\frac{1}{\mathrm{poly}(n)}$-CSE *(Maurer and Wolf [5]) .*

Note that Theorem 1 (3) and (4) hold not only for groups whose maximal prime factor of the order is not small. In this case both assumptions are trivially wrong and therefore the relation still holds.

3.2 This Work

The following two main theorems of this section state the equivalence of the two assumptions P-CDH and Q-CDH for two defining polynomials P and Q. Note that the size of the smallest prime factor of the group order turns the balance of the strength of the two theorems. In both theorems $P(a,b) = ab$ or $Q(a,b) = ab$ is possible.

Theorem 2. *For every constant l and for every $P, Q \in \mathbb{P}_l$ we have*

$$\frac{1}{\mathrm{poly}(n)}\text{-}P\text{-}\mathrm{CDH}_{\mathrm{nsprime}} \Leftrightarrow \frac{1}{\mathrm{poly}(n)}\text{-}Q\text{-}\mathrm{CDH}_{\mathrm{nsprime}}.$$

Theorem 3. *For $l \in O(\sqrt{\log n})$ and for every $P, Q \in \mathbb{P}_l^{\mathrm{poly}(n)}$ we have*

$$P\text{-CDH} \Leftrightarrow Q\text{-CDH}.$$

No *generic* algorithm can efficiently break the P-Computational Diffie-Hellman assumption:

Theorem 4. *For every $P \in \mathbb{P}_{\mathrm{poly}(n)}$ we have*

$$true \stackrel{\sigma}{\Rightarrow} \frac{1}{\mathrm{poly}(n)}\text{-}P\text{-}\mathrm{CDH}_{\mathrm{nsprime}}.$$

The proof of Theorem 4 uses techniques due to Shoup [8] and can be found in appendix A.

Theorem 5 (P-DH self-corrector). *For every constant l and every $P \in \mathbb{P}_l$ we have*

$$\frac{1}{\mathrm{poly}(n)}\text{-}P\text{-CDH} \Leftrightarrow P\text{-CDH}.$$

3.3 Proofs

COMPUTING ROOTS IN G will be an important building stone for the proofs of our theorems. We will shortly summarize some known theoretical results from [10]. For a finite cyclic group G of known order $|G|$ let $d \in \mathbb{Z}_{|G|}$ and $x, a \in G$. Then the equation

$$x^d = a$$

has exactly $s := \gcd(|G|, d)$ different solutions[3] $x_1, \ldots, x_s$. They are called d-th roots of a and can be computed by a probabilistic algorithm in expected $O(sn^3)$ bit operations. In fact, for this algorithm to work one has to know which prime factors are shared by d and $|G|$. But in our application d is always small enough to efficiently compute this relation. Therefore a complete factorization of $|G|$ is not needed, only $|G|$ must be known.
Note that for elements $x, a \in G$ and d relatively prime to $|G|$ the equation $x^d = a$ has a unique solution x which can be computed in expected $O(n^3)$ bit operations.

Lemma 1. *For $(G, g) = (G_n, g_n)_{n \in N} \in \mathbb{G}(\text{nsprime})$ let $d \in \mathbb{Z}_{|G|}$ and $x, a \in G$. Then the equation $x^d = a$ has with overwhelming probability a unique solution x.*

Proof. We show that the fraction of d for which d is not relatively prime to $|G|$ is negligible in n. Let $m = |G| = \prod_{1 \le i \le k} p_i^{e_i}$ and p be the minimal prime factor of m. Then $m \ge p^k$ and consequently $k \le \log_p m$. The number of d's that are relatively prime to m is given by the Euler Phi function $\varphi(m)$. The following inequality holds:

$$\begin{aligned}
\frac{\varphi(m)}{m} &= \frac{1}{m} \prod_{1 \le i \le k} p_i^{e_i - 1}(p_i - 1) = \prod_{1 \le i \le k} \frac{p_i - 1}{p_i} \ge \prod_{1 \le i \le k} \frac{p-1}{p} = \left(\frac{p-1}{p}\right)^k \\
&\ge \left(\frac{p-1}{p}\right)^{\log_p m} = \frac{(p-1)^{\log_p m}}{m} \overset{(*)}{\ge} \frac{p^{\log_p m} - \log_p m \cdot p^{(\log_p m)-1}}{m} \\
&= \frac{m - \frac{m \log_p m}{p}}{m} = 1 - \frac{\log_p m}{p} \ge 1 - \frac{n+1}{p \log_2 p} \ge 1 - \frac{n+1}{p} > 1 - \frac{1}{R(n)}
\end{aligned}$$

for every polynomial $R(n)$, where inequality $(*)$ holds because $(a-1)^b \ge a^b - ba^{b-1}$. □

The following simple lemma plays a central role in our proofs. It says that once we are given an efficient algorithm that computes DH with non-negligible probability of success, then we can efficiently compute P-DH with overwhelming probability of success for any polynomial $P(a, b)$.

Lemma 2. *For every $P \in \mathbb{P}_{\text{poly}(n)}$ we have*

$$P\text{-CDH} \Rightarrow \frac{1}{\text{poly}(n)}\text{-CDH}.$$

[3] The equation $x^d = a$ implicitly tells us that there is at least one solution, x.

Proof. Fix the family $(G,g) = (G_n, g_n)_{n\in\mathbb{N}} \in \mathbb{G}$. Assume $\frac{1}{\mathrm{poly}(n)}$-CDH is wrong, i.e. there is an oracle that computes DH with success probability of at least $1/\mathrm{poly}(n)$. Use the Diffie-Hellman self-corrector of Theorem 1 (3) to get an algorithm that computes DH with overwhelming probability of success. With this reliable algorithm for DH at hand, given g^a and g^b, any monomial $g^{c_{ij}a^ib^j}$ can be computed by repeated multiplication or squaring in the exponent. Hence, P-DH can be constructed "monomial-by-monomial" (there are at most polynomial many) by addition in the exponent. This would break assumption P-CDH. Thus $P\text{-CDH} \Rightarrow \frac{1}{\mathrm{poly}(n)}\text{-CDH}$ holds. □

With this observation at hand the proof of Theorem 5 (P-DH self-corrector) is easy. Clearly $\frac{1}{\mathrm{poly}(n)}$-$P$-CDH $\Rightarrow$ P-CDH holds. To prove "$\Leftarrow$" we "detour" over DH. This will be a very frequently used strategy in our proofs. Let $P \in \mathbb{P}_l$ and let an oracle $\mathcal{O}_{P-\mathrm{DH}}$ be given that computes P-DH with non-negligible probability of success. Due to Theorem 2 we can construct an algorithm $\mathcal{A}^{\mathcal{O}_{P\text{-DH}}}$ that computes DH with non-negligible probability of success. Now apply Lemma 2.

PROOF OUTLINE of Theorem 2 and Theorem 3: Due to Lemma 2 in both cases it is sufficient to show that given an efficient algorithm that computes P-DH for a $P \in \mathbb{P}_l$ then there is an efficient algorithm that computes DH. Lemma 3 deals with the special case $P \in \mathbb{P}_2$. It can be viewed as the induction base. In Lemma 4 computing P-DH for a $P \in \mathbb{P}_N$ is reduced through a efficient algorithm to computing Q-DH for a $Q \in \mathbb{P}_{N-1}$. This lemma can be viewed as the induction step which is then applied recursively $l-2$ times. As we will see we have to take care of a blow-up of the coefficients of the polynomial Q in the induction step.

Lemma 3. *1. For $P \in \mathbb{P}_2^{\mathrm{poly}(n)}$ we have*

$$P\text{-CDH} \Leftarrow \mathrm{CDH}.$$

2. For $P \in \mathbb{P}_2$ we have

$$\frac{1}{\mathrm{poly}(n)}\text{-}P\text{-CDH}_{\mathrm{nsprime}} \Leftarrow \frac{1}{\mathrm{poly}(n)}\text{-CDH}_{\mathrm{nsprime}}.$$

Proof. We first prove part 1 of the lemma. Let $P \in \mathbb{P}_2^{\mathrm{poly}(n)}$. Because of Lemma 2 it is sufficient to show $P\text{-CDH} \Leftarrow \frac{1}{\mathrm{poly}(n)}\text{-CDH}$. Let $(G,g) = (G_n,g_n)_{n\in\mathbb{N}} \in \mathbb{G}$. Let $\mathcal{O}_{P\text{-DH}}$ be an oracle that computes P-DH, i.e. given g^a, g^b, $\mathcal{O}_{P\text{-DH}}$ outputs

$$g^{P(a,b)} = g^{c_{20}a^2+c_{21}a^2b+c_{22}a^2b^2+c_{10}a+c_{11}ab+c_{12}ab^2+c_{00}+c_{01}b+c_{02}b^2}.$$

We want to design an algorithm $\mathcal{A}^{\mathcal{O}_{P\text{-DH}}}$ that computes DH with non-negligible probability of success. The main idea of the proof is to "eliminate" any appearance of a^ib^2 and a^2b^j in the exponent for every $0 \le i,j \le 2$ by the multiplicative combination of calls to $\mathcal{O}_{P\text{-DH}}$. For this, $\mathcal{A}^{\mathcal{O}_{P\text{-DH}}}$ queries the oracle for $Y_+ = \mathcal{O}_{P\text{-DH}}(g^{a+b},g) = P\text{-DH}(g^{a+b},g)$ and $Y_- = \mathcal{O}_{P\text{-DH}}(g^{a-b},g) = P\text{-DH}(g^{a-b},g)$. Division of the two outputs yields

$$Y_+ \cdot (Y_-)^{-1} = g^{4c_2\cdot ab+2c_1\cdot b} \tag{1}$$

where all $c_i := \sum_{j=0}^{2} c_{ij}$ are known. First assume $c_2 \neq 0$. From (1)

$$y := g^{4c_2 \cdot ab} = g^{4c_2 \cdot ab + 2c_1 \cdot b} \cdot (g^b)^{-2c_1}$$

can be computed. Assume $4c_2$ is positive, otherwise invert. Now compute all $4c_2$-th roots of $g^{4ab \cdot c_2}$ (there are $s := \gcd(4c_2, |G|)$ of them) , i.e. all solutions x of the equation

$$x^{4c_2} = y, \tag{2}$$

with $x = g^{ab}$. This can be done in time $O(sn^3) = \text{poly}(n)$, because all coefficients c_{ij} are in $\text{poly}(n)$. Unfortunally we can not test which of the roots of equation (2) is the correct one because the corresponding decision problem might be hard. But since we know that one of the $s = \text{poly}(n)$ different roots if the correct one, g^{ab}, it is sufficient to output one of the roots *at random*. Hence, for the case $c_2 \neq 0$ the success probability of the $\mathcal{A}^{\mathcal{O}_{P\text{-DH}}}$ is $\epsilon(n) \geq 1/s \geq 1/(4c_2) = 1/\text{poly}(n)$.

The case $c_2 = 0$ is left. If $c_2' = \sum_{i=0}^{2} c_{i2} \neq 0$ we can swap the input and are done, too. If not, a different approach has to be taken. First note that in the case $c_2 = 0$ the polynomial $P(a, b)$ must depend on both, a and b. Again by making two queries to the oracle $\mathcal{O}_{P\text{-DH}}$ the algorithm computes $P\text{-DH}(g^{a+b}, g^2) \cdot (P\text{-DH}(g^{a-b}, g^2))^{-1}$ which leads to $g^{4d_2 \cdot ab + 2d_1 b}$ with $d_i = \sum_{j=0}^{2} c_{ij} 2^j$. Note that if $c_2 = 0$ then $d_2 \neq 0$ holds. Now continue as above. This completes the proof of part 1.

Now let $(G, g) = (G_n, g_n)_{n \in \mathbb{N}} \in \mathbb{G}(\text{nsprime})$ and let $P \in \mathbb{P}_2$. We show part 2 of the lemma. Let $\mathcal{O}_{P\text{-DH}}$ be an oracle that outputs P-DH with success probability at least $\epsilon(n) = 1/\text{poly}(n)$. First the algorithm queries the oracle for $Y_+ = \mathcal{O}_{P\text{-DH}}(g^{a+b+s}, g^u)$ and $Y_- = \mathcal{O}_{P\text{-DH}}(g^{a-b+t}, g^v)$ for random and known values s, t, u, v. Note that the queries are random and independent. Therefore the probability that both calls give the correct answer is at least $\epsilon^2(n)$. Assume this is the case, thus $Y_+ = P\text{-DH}(g^{a+b+s}, g^u)$ and $Y_- = P\text{-DH}(g^{a-b+t}, g^v)$. Now the key observation is that because the minimal prime factor of G is not too small, every coefficient in the exponent has an unique inverse with overwhelming probability (Lemma 1). In this case the inverse is efficiently computable. From

$$Y_+ = g^{c_2(a+b+s)^2 + c_1(a+b+s) + c_0} = g^{c_2(a+b)^2 + 2c_2(a+b)s + c_2 s^2 + c_1(a+b+s) + c_0}$$

for $c_i = \sum_{j=0}^{2} c_{ij} u^j$ a simple computation gives us $g^{(a+b)^2}$. For the same reason we get $g^{(a-b)^2}$ from Y_-. Again by division g^{2ab} can be computed and hence g^{ab}. We have constructed an algorithm $\mathcal{A}^{\mathcal{O}_{P\text{-DH}}}$ that computes g^{ab} with success probability of at least $\epsilon^2(n) = 1/\text{poly}(n)$. □

The next lemma can be viewed as the "induction step" to prove Theorem 3 and Theorem 2.

Lemma 4. *1. For a $P \in \mathbb{P}_N^k$ let $\mathcal{O}_{P\text{-DH}}$ be an oracle that breaks P-CDH. Then for $k' := 2kN2^N$ there is a $Q \in \mathbb{P}_{N-1}^{k'}$ and an efficient algorithm $\mathcal{A}^{\mathcal{O}_{P\text{-DH}}}$ that breaks Q-CDH making at most 3 queries to $\mathcal{O}_{P\text{-DH}}$.*

2. *For a function $\epsilon > 0$ and for a $P \in \mathbb{P}_N$ let $\mathcal{O}_{P\text{-DH}}$ be an oracle that breaks $\epsilon(n)$-P-$\text{CDH}_{\text{nsprime}}$. Then there is a $Q \in \mathbb{P}_{N-1}$ and an efficient algorithm $\mathcal{A}^{\mathcal{O}_{P\text{-DH}}}$ that breaks $\epsilon^3(n)$-Q-$\text{CDH}_{\text{nsprime}}$ making at most 3 queries to $\mathcal{O}_{P\text{-DH}}$.*

Proof. We start proving the first part of this lemma. Let $P \in \mathbb{P}_N^k$ and let $(G, g) = (G_n, g_n)_{n\in\mathbb{N}} \in \mathbb{G}$. The main idea of this proof again is to eliminate any appearance of $g^{a^N b^j}$ or $g^{a^i b^N}$ for any i, j by computing $P\text{-DH}(g^{a+b}, g) \cdot (P\text{-DH}(g^{a-b}, g))^{-1}$.
Case 1: N even.
Making *two* queries to the oracle $\mathcal{O}_{P\text{-DH}}$, algorithm $\mathcal{A}^{\mathcal{O}_{P\text{-DH}}}$ gets

$$Y_+ = P\text{-DH}(g^{a+b}, g) = g^{c_N(a+b)^N + c_{N-1}(a+b)^{N-1} + \cdots + c_0(a+b)}$$

and

$$Y_- = P\text{-DH}(g^{a-b}, g) = g^{c_N(a-b)^N + c_{N-1}(a-b)^{N-1} + \cdots + c_0(a-b)},$$

where $c_i := \sum_{j=1}^{N} c_{ij}$. Now assume $c_N \neq 0$. c_N might be 0, but if $c_N = 0$ we continue as in the case of $N - 1$[4]. Now remember N is even. Therefore division leads to $g^{Q(a,b)} = Y_+ \cdot (Y_-)^{-1}$ where

$$\begin{aligned} Q(a,b) &= c_N((a+b)^N - (a-b)^N) + c_{N-1}((a+b)^{N-1} - (a-b)^{N-1}) \\ &\quad + \sum_{i=0}^{N-2} c_i((a+b)^i - (a-b)^i)) \\ &= 2c_N\left(\binom{N}{N-1}a^{N-1}b + \binom{N}{N-3}a^{N-3}b^3 + \cdots + \binom{N}{1}ab^{N-1}\right) \\ &\quad + 2c_{N-1}\left(\binom{N-1}{N-2}a^{N-2}b + \cdots + b^{N-1}\right) + \cdots . \end{aligned}$$

Each coefficient of $a^i b^j$ of this polynomial $Q(a, b)$ is either 0 (if j is even) or $2c_{i+j}\binom{i+j}{i} \leq 2kN\binom{N}{N/2} \leq 2kN2^N =: k'$ (if j is odd). Note that the coefficients of the monomials a^N and b^N are always 0. Thus $Q(a, b) \in \mathbb{P}_{N-1}^{k'}$. Algorithm $\mathcal{A}^{\mathcal{O}_{P\text{-DH}}}$ outputs $g^{Q(a,b)} = \text{Q-DH}(g^a, g^b)$.

Case 2: N odd.
With *three* queries to the oracle $\mathcal{O}_{P\text{-DH}}$, algorithm $\mathcal{A}^{\mathcal{O}_{P\text{-DH}}}$ gets

$$g^{Q(a,b)} = P\text{-DH}(g^{a+b}, g) \cdot P\text{-DH}(g^{a-b}, g)^{-1} \cdot P\text{-DH}(g^b, g)^{-1}$$

where

$$Q(a,b) = 2c_N N a^{N-1} b - 2c_{N-1} b^{N-1} + \cdots$$

A similar computation as in the even case shows that $Q(a, b) \in \mathbb{P}_{N-1}^{k'}$ with k' defined as above. Algorithm $\mathcal{A}^{\mathcal{O}_{P\text{-DH}}}$ outputs $g^{Q(a,b)} = Q\text{-DH}(g^a, g^b)$.

[4] If $c_i = 0$ for all $1 \leq i \leq N$ then query the oracle for $Y_+ = P\text{-DH}(g^{a+b}, g^2)$ and $Y_- = P\text{-DH}(g^{a-b}, g^2)$ and continue as in Lemma 3 (2).

We continue with the second part of the lemma. Let $P \in \mathbb{P}_N$ and let $\mathcal{O}_{P\text{-DH}}$ be an oracle that computes $P\text{-DH}(g^a, g^b)$ with success probability $\epsilon(n)$. Assume N is even. The odd case again is similar. In order to randomize the input of $\mathcal{O}_{P\text{-DH}}$ chose s, t, u, v at random and uniformly distributed from $\mathbb{Z}_{|G|}$. Let $c_i := \sum_{j=1}^{N} c_{ij} u^j$ and $c'_i = \sum_{j=1}^{N} c_{ij} v^j$. Making two independent queries to the oracle $\mathcal{O}_{P\text{-DH}}$ the algorithm gets (assuming the oracle gives the correct answer twice) $g^{Q_+(a,b)} = P\text{-DH}(g^{a+b+s}, g^u)$ and $g^{Q_-(a,b)} = P\text{-DH}(g^{a-b+t}, g^v)$ with

$$\begin{aligned} Q_+(a,b) &= c_N(a+b+s)^N + c_{N-1}(a+b+s)^{N-1} + \cdots + c_0(a+b+s) \\ &= c_N\left((a+b)^N + \binom{N}{1}(a+b)^{N-1}s + \cdots + (a+b)s^N\right) \\ &\quad + c_{N-1}(a+b+s)^{N-1} + \cdots + c_0(a+b+s) \end{aligned}$$

and

$$\begin{aligned} Q_-(a,b) = c'_N\left((a-b)^N + \binom{N}{1}(a-b)^{N-1}s + \cdots + (a-b)s^N\right) \\ + c'_{N-1}(a-b+s)^{N-1} + \cdots + c'_0(a-b+s). \end{aligned}$$

Note that the probability that both queries to the oracle $\mathcal{O}_{P\text{-DH}}$ are successful is lower bounded by $\epsilon^2(n)$. Assume this is the case. Now continue as in the proof of the first part computing $g^{Q(a,b)} = g^{Q_+(a,b)\cdot c'_N} \cdot (g^{Q_-(a,b)\cdot c_N})^{-1}$ where $Q(a,b) \in \mathbb{P}_{N-1}$ and define $g^{Q(a,b)}$ as the output of the algorithm. The even case is similar instead of the fact that three calls to the oracle $\mathcal{O}_{P\text{-DH}}$ are needed. Consequently the overall success probability of $\mathcal{A}^{\mathcal{O}_{P\text{-DH}}}$ is lower bounded by $\epsilon^3(n)$. □

Now we are ready to give the proof of Theorem 3 and Theorem 2.

Proof. (of Theorem 3). Let $(G, g) = (G_n, g_n)_{n\in\mathbb{N}} \in \mathbb{G}$. For $l \in O(\sqrt{\log n})$ and $k \in \text{poly}(n)$ let $P \in \mathbb{P}_l^k$. Due to Lemma 2 it is sufficient to show $P\text{-CDH} \Leftarrow \text{CDH}$. Let $\mathcal{O}_{P\text{-DH}}$ be an oracle that computes P-DH. Now apply $(l-2)$-times Lemma 4 (1) recursively to get a polynomial Q and an efficient algorithm $\mathcal{A}^{\mathcal{O}_{P\text{-DH}}}$ that computes Q-DH. $Q \in \mathbb{P}_2^{f(k,l)}$, where

$$f(k,l) = k \cdot \prod_{i=2\ldots l} i2^i \leq k \cdot l! 2^{\frac{(l+1)l}{2}} = \text{poly}(n) \cdot \text{poly}(n) = \text{poly}(n).$$

The number of queries to $\mathcal{O}_{P\text{-DH}}$ is at most $3^{l-2} = \text{poly}(n)$. Now use Lemma 3 (1) to construct an efficient algorithm $\mathcal{B}^{\mathcal{O}_{P\text{-DH}}}$ that computes $\text{DH}(g^a, g^b)$ with overwhelming probability of success. □

Proof. (of Theorem 2). Let $P \in \mathbb{P}_l$ for a constant l. It is sufficient to show $\frac{1}{\text{poly}(n)}$-P-CDH $\Leftarrow$ $\frac{1}{\text{poly}(n)}$-CDH. Let $\mathcal{O}_{P\text{-DH}}$ be an oracle that computes P-DH with success probability $1/\text{poly}(n)$. After $l-2$ recursive applications of Lemma 4 (2) we get a $Q \in \mathbb{P}_2$ and an efficient algorithm $\mathcal{A}^{\mathcal{O}_{P\text{-DH}}}$ that computes $Q\text{-DH}(g^a, g^b)$ with success probability $(1/\text{poly}(n))^{3(l-2)} = 1/\text{poly}(n)$. Now apply Lemma 3 (2). □

4 The Decisional Case

Let $\mathcal{G} = (G_n, g_n)_{n\in\mathbb{N}} = (G, g)$ be a family of groups and generators and let $\epsilon(n)$ be a function in n. Then the *$\epsilon(n)$-P-Decision Diffie-Hellman assumption for $\mathcal{G}$* ($\epsilon(n)$-P-DDH($\mathcal{G}$)) is: There is no efficient algorithm that, given random group elements g^a, g^b and (in random order) $g^{P(a,b)}$ and another random group element g^c, identifies $g^{P(a,b)}$ with probability $1/2 + \epsilon(n)$ (taken over the input and coin tosses of the algorithm).
Let $\epsilon(n)$-P-DDH$_{\text{nsprime}}$, $\epsilon(n)$-P-DDH$_{\text{nsprime}}$, $\epsilon(n)$-P-DDH as well as the Decision Square Exponent assumption $\epsilon(n)$-DSE and the Decision Diffie-Hellman assumption $\epsilon(n)$-DDH be defined as in the computational case.

4.1 Previous Work

Theorem 6. *1. true* $\stackrel{\sigma}{\Rightarrow} \frac{1}{\text{poly}(n)}$-DDH$_{\text{nsprime}}$ *(Shoup [8]).*
2. true $\stackrel{\sigma}{\Rightarrow} \frac{1}{\text{poly}(n)}$-DSE$_{\text{nsprime}}$ *(Wolf [10]).*
3. $\frac{1}{\text{poly}(n)}$-DDH $\stackrel{g}{\not\Rightarrow} \frac{1}{\text{poly}(n)}$-DSE$_{\text{nsprime}}$ *(Wolf [10]).*
4. $\frac{1}{\text{poly}(n)}$-DDH $\Leftarrow \frac{1}{\text{poly}(n)}$-DSE *(Wolf [10]).*

4.2 This Work

We define the set of polynomials $\mathbb{P}_{2^*}$ for that the reduction from P-DDH to DDH is possible. Let $\mathbb{P}_{2^*} \subset \mathbb{P}_2$ be the set of polynomials P from $\mathbb{P}_2$ given by

$$P(a,b) = (d_{00}a + d_{01}b)(d_{10}a + d_{11}b) + c_{10}a + c_{01}b + c_{00},$$

where c_{ij} and $d_{ij} \in \mathbb{Z}_{|G|}$. Example polynomials of the set $\mathbb{P}_{2^*}$ include $P(a,b) = a^2 - b^2$ and $P(a,b) = (a+b)^2$.

We characterize the relation between the assumptions $\frac{1}{\text{poly}(n)}$-DDH and $\frac{1}{\text{poly}(n)}$-P-DDH in the following two theorems. Remember that $\mathbb{P}_1$ is the set of polynomials $P(a,b) = c_{11}ab + c_{01}a + c_{10}b + c_{00}$ satisfying $c_{11} \neq 0$.

Theorem 7. *For every $P \in \mathbb{P}_{\text{poly}(n)} \setminus \mathbb{P}_{2^*}$ and $Q \in \mathbb{P}_{2^*}$ we have*

1.
$$\frac{1}{\text{poly}(n)}\text{-DDH}_{\text{nsprime}} \stackrel{g}{\not\Leftarrow} \frac{1}{\text{poly}(n)}\text{-}P\text{-DDH}_{\text{nsprime}}.$$

2.
$$\frac{1}{\text{poly}(n)}\text{-DDH} \Leftarrow \frac{1}{\text{poly}(n)}\text{-}Q\text{-DDH}.$$

Theorem 8. *For every $P \in \mathbb{P}_{\text{poly}(n)} \setminus \mathbb{P}_1$ we have*

$$\frac{1}{\text{poly}(n)}\text{-DDH}_{\text{nsprime}} \stackrel{g}{\not\Rightarrow} \frac{1}{\text{poly}(n)}\text{-}P\text{-DDH}_{\text{nsprime}}.$$

The proof of Theorem 8 uses techniques due to Shoup [8] and can be found in appendix A.
The next theorem is a direct corollary of Theorem 6 (1) and Theorem 8.

Theorem 9. *For every $P \in \mathbb{P}_{\mathrm{poly}(n)}$ we have*

$$\mathrm{true} \overset{\sigma}{\Rightarrow} \frac{1}{\mathrm{poly}(n)}\text{-}P\text{-}\mathrm{DDH}_{\mathrm{nsprime}}.$$

4.3 Proofs

The following lemma gives an alternative characterization of $\mathbb{P}_{2^*}$.

Lemma 5. *1. If $P(a,b) \in \mathbb{P}_{2^*}$ then there are non-trivial linear combinations R, S and T of 1, a, b and $P(a,b)$ over $\mathbb{Z}_{|G|}$ that satisfy the relation*

$$\forall a, b: \quad R(1,a,b,P(a,b)) \cdot S(1,a,b,P(a,b)) = T(1,a,b,P(a,b)). \quad (3)$$

2. Let $(G,g) \in \mathbb{G}(\mathrm{nsprime})$ and let $P \in \mathbb{P}_{\mathrm{poly}(n)}$. If relation (3) is satisfied, then $P \in \mathbb{P}_{2^*}$ holds with overwhelming probability (over the choices of P).

Proof. Let $R(a,b) = r_0 + r_1 a + r_2 b + r_3 P(a,b)$, $S(a,b) = s_0 + s_1 a + s_2 b + s_3 P(a,b)$ and $T(a,b) = t_0 + t_1 a + t_2 b + t_3 P(a,b)$. Relation (3) can only hold for all a, b if $r_3 = s_3 = 0$. Thus, viewed as polynomials over a and b, relation (3) is satisfied iff $(r_1 a + r_2 b) \cdot (s_1 a + s_2 b) = u_0 + u_1 a + u_2 b + t_3 P(a,b)$, where $u_0 := t_0 - r_0 s_0$, $u_1 := t_1 - r_0 s_1 - r_1 s_0$ and $u_2 := t_2 - r_0 s_2 - r_2 s_0$. Consequently, for any $P(a,b) \in \mathbb{P}_2^*$, relation (3) can be satisfied (setting $t_3 = 1$). Now let $(G,g) \in \mathbb{G}(\mathrm{nsprime})$ and relation (3) be satisfied. Due to Lemma 1, t_3 is invertible in $\mathbb{Z}_{|G|}$ with overwhelming probability. In this case obviously $P \in \mathbb{P}_{2^*}$ holds. □

The next lemma proves Theorem 7 (2).

Lemma 6. *Let $P \in \mathbb{P}_{2^*}$ and let $\mathcal{O}_{\mathrm{DDH}}$ be a an oracles that breaks $\epsilon(n)$-DDH. Then there is an efficient algorithm $\mathcal{A}^{\mathcal{O}_{\mathrm{DDH}}}$ that breaks $\epsilon(n)$-P-DDH.*

Proof. We construct $\mathcal{A}^{\mathcal{O}_{\mathrm{DDH}}}$ as follows: Let g^a, g^b and in random order $g^{P(a,b)}$ and g^c be given, where we can not a-priori decide between the input $g^{P(a,b)}$ and g^c. Since $P \in \mathbb{P}_{2^*}$ we can compute $g^{R(a,b)}, g^{S(a,b)}$ and $g^{T(a,b)}$ with R, S, T satisfying relation (3) of Lemma 5 (1). $g^{T(a,b)}$ is computed twice, first with $g^{P(a,b)}$, second with g^c in the role of $g^{P(a,b)}$. Lets denote them as g^{T_1} and g^{T_2}. Now feed the $\epsilon(n)$-DDH oracle with the input $(g^R, g^S, g^{T_1}, g^{T_2})$ which immediately identifies which one, g^{T_1} or g^{T_2}, has been computed from $g^{P(a,b)}$. Note that we called the $\epsilon(n)$-DDH oracle only once, thus the success probability of algorithm $\mathcal{A}^{\mathcal{O}_{\mathrm{DDH}}}$ is $\epsilon(n)$. □

The next lemma plays an essential role for proofs in the context of generic algorithms.

Lemma 7 ([8]). *Let m be an integer and let $F(X_1, X_2, \cdots, X_n)$ be a non-zero polynomial in $\mathbb{Z}_m[X]$ of total degree $d \geq 1$. Then, for random $(x_1, x_2, \ldots, x_n) \in \mathbb{Z}_m^n$, the probability that $F(x_1, x_2, \ldots, x_n) = 0$ is at most d/p, where p is the smallest prime factor of m.*

The next lemma says that for $P \in \mathbb{P}_{\text{poly}} \backslash \mathbb{P}_{2^*}$ every generic algorithm that breaks $\epsilon(n)$-P-$\text{DDH}_{\text{nsprime}}$ for a non-negligible $\epsilon(n)$ needs at least super-polynomial time. It proves Theorem 7 (1). The proof uses techniques due to Shoup [8].

Lemma 8. *Let m be an integer and p the smallest prime factor of m. Let $S \subset \{0,1\}^*$ be a set of at least m binary strings. Let $P \in \mathbb{P}_l \setminus \mathbb{P}_{2^*}$. Assume that a generic algorithm $\mathcal{A}$ exists that works for groups of order m, runs in time at most T and makes calls to a (perfect)* DDH*-oracle. Let $a, b, c \in \mathbb{Z}_m$ be chosen at random, let $\sigma : \mathbb{Z}_m \to S$ be a random encoding function, and let t be a random bit. Set $w_0 = P(a,b)$ and $w_1 = c$. Then*

$$\Pr[\mathcal{A}(\sigma; 1, a, b, w_t, w_{1-t}) = t] \leq 1/2 + O(lT^2/p)$$

holds for all but a negligible fraction of all $P \in \mathbb{P}_l \setminus \mathbb{P}_{2^*}$.

Proof. The proof follows two ideas. First, if the algorithm has a slight chance to decide P-DH only by making computations in the group, then this happens by an "accident" that is not very likely to happen. And second, the algorithm has no chance to get a single bit of information from the DDH-oracle, i.e. the probability that it gets a non-trivial answer from it is very small (here Lemma 5 (2) comes to application). Hence, the oracle is useless.
Now to the formal arguments. The algorithm makes A computation steps and knows the encodings $\sigma(x_i)$ of some $x_i \in \mathbb{Z}_m$, $1 \leq i \leq A+5$. x_i clearly is of the form $x_i = R_i(a, b, P(a,b), c)$, where R_i is a linear function in a, b, $P(a,b)$ and c. We can assume that $R_i \neq R_j$ holds for all $i \neq j$. Furthermore the algorithm makes B calls to the DDH-oracle.
The algorithm may be able to output the correct bit t only by obtaining information from either of the following events $\mathcal{E}_1$ and $\mathcal{E}_2$:

- $\mathcal{E}_1$: The algorithm finds for $i \neq j$ a non-trivial collision of the form $\sigma(x_i) = \sigma(x_j)$. Note that if for all i, j one has $\sigma(x_i) \neq \sigma(x_j)$, then the algorithm has learned the random encoding of two *distinct* pairs. That clearly does not provide any information to the algorithm.
- $\mathcal{E}_2$: The algorithm gets at least one positive[5] answer for a non-trivial query to the DDH-oracle.

Let $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2$. We compute an upper bound for the probability that event $\mathcal{E}$ occurs. We give the algorithm the benefit of doubt and consider it in this case to be successful.
Case $\mathcal{E}_1$: Observe that a non-trivial collision $R_i(a,b) = R_j(a,b)$ for all a, b is never

[5] Positive in the sense that the DDH oracle correctly identifies g^{xy} given the input g^x, g^y, g^z and g^{xy}.

possible. The probability of a collision $R_i(a,b,P(a,b),c) = R_j(a,b,P(a,b),c)$ for fixed $i \neq j$ can be bounded by considering the polynomial $F(x,y,z) = R_i - R_j$. This is a polynomial of total degree at most $2l$. Due to Lemma 7 the probability that a random $(x,y,z) \in \mathbb{Z}_m^3$ is a zero of F is bounded by $2l/p$. There are $\binom{A+5}{2}$ possible distinct pairs of linear combinations R_i, R_j to consider. Consequently the probability that event $\mathcal{E}_1$ occurs is bounded by $\binom{A+5}{2} \cdot \frac{2l}{p}$.
Case $\mathcal{E}_2$: Let $i \neq j$. Due to Lemma 5 (2), with overwhelming probability over all choices of P, it is not possible to derive a (non-trivial) relation $R_i(a,b) = R_j(a,b)R_k(a,b)$. The degree of $R_j \cdot R_k - R_i$ is at most $4l$. Thus with same argument as in event $\mathcal{E}_1$ the probability that event $\mathcal{E}_2$ occurs is bounded by $4lB/p$.
In total we have $\Pr[\mathcal{E}] \leq \Pr[\mathcal{E}_1] + \Pr[\mathcal{E}_2] \leq \frac{2l(A+5)(A+4)}{2p} + \frac{4lB}{p} \leq \frac{2l(T+5)(T+4)}{2p}$, where $T = A+B$ is the number of steps of the algorithm. The overall probability that the algorithm outputs the correct bit t therefore is:

$$\Pr[\mathcal{E}] + \frac{1}{2}\Pr[\overline{\mathcal{E}}] = \frac{1}{2} + \frac{\Pr[\mathcal{E}]}{2} \leq \frac{2l(T+5)(T+4)}{4p} = O\left(\frac{lT^2}{p}\right).$$

□

5 Conclusions and Open Problems

For groups having at least one large prime factor in the (known) group order we proved that for every $P \in \mathbb{P}_{\mathrm{poly}(n)}$ no generic algorithm can compute the P-Diffie-Hellman function (Theorem 4) or decide it (Theorem 9) in probabilistic polynomial time. The relation to the Diffie-Hellman function proven in Sections 3 and 4 can be summarized as follows:
Computational Case. For every constant l and for every $P \in \mathbb{P}_l$ the following relations hold:

$$\frac{1}{\mathrm{poly}(n)}\text{-CSE} \overset{\text{Thrm. 1(4)}}{\Leftrightarrow} \frac{1}{\mathrm{poly}(n)}\text{-CDH} \overset{\text{Thrm. 2}}{\Leftrightarrow} \frac{1}{\mathrm{poly}(n)}\text{-}P\text{-CDH}$$

For $l \in O(\sqrt{\log n})$ and for every $P \in \mathbb{P}^l_{\mathrm{poly}(n)}$ the following relation holds:

$$\text{CDH} \overset{\text{Thrm. 3}}{\Leftrightarrow} P\text{-CDH}.$$

Decisional Case. The following relations hold:

$$\frac{1}{\mathrm{poly}(n)}\text{-DDH}_{\mathrm{nsprime}} \begin{cases} \overset{g}{\not\Rightarrow} \frac{1}{\mathrm{poly}(n)}\text{-}P\text{-DDH}_{\mathrm{nsprime}} & : \quad P \in \mathbb{P}_{\mathrm{poly}(n)} \setminus \mathbb{P}_1 \\ \Leftarrow \frac{1}{\mathrm{poly}(n)}\text{-}P\text{-DDH}_{\mathrm{nsprime}} & : \quad P \in \mathbb{P}_{2^*} \\ \overset{g}{\not\Leftarrow} \frac{1}{\mathrm{poly}(n)}\text{-}P\text{-DDH}_{\mathrm{nsprime}} & : \quad P \in \mathbb{P}_{\mathrm{poly}(n)} \setminus \mathbb{P}_{2^*}. \end{cases}$$

On the one hand this shows that the two assumptions $\mathrm{DDH}_{\mathrm{nsprime}}$ and P-$\mathrm{DDH}_{\mathrm{nsprime}}$ for $P \in \mathbb{P} \setminus \mathbb{P}_1$ are not *generically equivalent*. But Theorem 9 shows that they

are at least *equally hard* with respect to generic algorithms.

APPLICATIONS. As the title of this paper suggests we presented a theoretical approach of a generalization of the Diffie-Hellman function. This P-Diffie-Hellman function is provable computational equivalent to the Diffie-Hellman function for a certain class of groups. This set of functions should be viewed as a tool box. The same way the Square Exponent function was introduced as a theoretical concept first and later exploited in a cryptographic setting, we hope that one will find a useful application in some cryptographic protocols or maybe one can use it to simplify some proofs in the context of the Diffie-Hellman function.
Note that the P-Diffie-Hellman function can be adopted to use as a replacement for the Diffie-Hellman function in some applications. For instance the to-the-s Diffie-Hellman function introduced in Section 1 can be used in protocols like the scheme for key escrow with limited time span [1].

OPEN PROBLEMS: First of all it would be nice to have same more "real-world" applications of the P-Diffie-Hellman function. The summary given in the conclusions of the computational case leaves a lot of room for improvement. It would be interesting to see if one can improve the restrictions to the degree of the polynomials $P(a,b)$ from $l =$ constant to $l = \text{poly}(n)$ in Theorem 2. Specially one has to randomize the input of the P-DH oracle in the reduction what seems to be a problem. Or maybe one can even show that $\frac{1}{\text{poly}(n)}$-CDH $\Leftrightarrow$ $\frac{1}{\text{poly}(n)}$-P-CDH holds for $P \in \mathbb{P}_{\text{poly}(n)}$ as Theorem 9 indicates? We must leave that as an open question.
In [7] the *Inverse Exponent* function $\text{IE}(g^a) = g^{(a^{-1})}$ is analyzed. It is proven that this function also is computational equivalent to the Diffie-Hellman function. Consequently one might ask the question what kind of functions $f : \mathbb{Z}_{|G|} \times \mathbb{Z}_{|G|} \to \mathbb{Z}_{|G|}$ (others than polynomials) lead to f-Diffie-Hellman functions $f\text{-DH}(g^a, g^b) = g^{f(a,b)}$ that are computational equivalent to the Diffie-Hellman function. On the other hand to the best of our knowledge there is no *non-trivial* function $f(a,b)$ known that make $g^{f(a,b)}$ *easy* to compute.
Finally we want to mention that a generalization of the defining polynomial $P(a,b)$ to $P(a_1, \cdots, a_k)$ also is possible.

ACKNOWLEDGMENT. Thanks to Jürgen Forster for great help on this paper as well to Andreas Bomke, Dario Catalano, Ahmad-Reza Sadeghi, Hans-Ulrich Simon and Michael Steiner.

References

1. M. Burmester, Y. Desmedt, and J. Seberry. Equitable key escrow with limited time span (or, how to enforce time expiration cryptographically). In *Advances in Cryptology – ASIACRYPT '98*, pages 380–391, 1998.
2. D. Coppersmith and I. Shparlinski. On polynomial approximation of the Discrete Logarithm and the Diffie-Hellman mapping. *Journal of Cryptology*, 13(3):339–360, March 2000.

3. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on information Theory*, 22(6):644–654, 1976.
4. E. Kiltz. A tool box of cryptographic functions related to the Diffie-Hellman Function. *Progress in Cryptology – INDOCRYPT 2001*, 2001.
5. U. Maurer and S. Wolf. Diffie-Hellman oracles. *Proc. of CRYPTO'96. Lecture Notes in Computer Science*, 1109:268–282, 1996.
6. S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Trans. Inf. Theory*, 24:106–110, 1978.
7. A.-R. Sadeghi and M. Steiner. Assumptions related to discrete logarithms: Why subtleties make a real difference. In *Advances in Cryptology – EUROCRYPT '01*, pages 243–260, 2001.
8. V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology – EUROCRYPT '97*, pages 256–266, 1997.
9. I. Shparlinski. Security of most significant bits of g^{x^2}. *Inf. Proc. Letters*, (to appear).
10. Stefan Wolf. *Information-theoretically and Computionally Secure Key Agreement in Cryptography*. PhD thesis, ETH Zürich, 1999.

A Some Lower-Bound Proofs

Theorem 4 is a simple corollary of the following lemma. It essentially says that for $P \in \mathbb{P}_{\mathrm{poly}(n)}$ every generic algorithm that breaks $\epsilon(n)$-P-$\mathrm{CDH}_{\mathrm{nsprime}}$ for a non-negligible function ϵ needs at least super-polynomial time. The proof uses techniques due to Shoup [8].

Lemma 9. *Let m be an integer and p the smallest prime factor of m. Let $S \subset \{0,1\}^*$ be a set of at least m binary strings. Let $P \in \mathbb{P}_l$. Assume that a generic algorithm $\mathcal{A}$ exists that works for groups of order m and runs in time at most T. Let $a, b \in \mathbb{Z}_m$ chosen at random, let $\sigma : \mathbb{Z}_m \to S$ be a random encoding function. Then* $\Pr[\mathcal{A}(\sigma; 1, a, b) = \sigma(P(a,b))] \leq O((T^2 + lT)/p)$.

Proof. The algorithm makes T computation steps and knows the encodings $\sigma(x_i)$ of some $x_i \in \mathbb{Z}_m$, $1 \leq i \leq A+5$. x_i clearly is of the form $x_i = R_i(a,b)$, where R_i is a linear function in a and b. We can assume that $R_i \neq R_j$ holds for all $i \neq j$. The algorithm may be able to output $\sigma(ab)$ only by obtaining information from either of the following events $\mathcal{E}_1$ and $\mathcal{E}_2$:

$\mathcal{E}_1$: The algorithm finds for $i \neq j$ a non-trivial collision of the form $\sigma(x_i) = \sigma(x_j)$. Note that if for all i, j one has $\sigma(x_i) \neq \sigma(x_j)$, then the algorithm has learned the random encoding of two *distinct* pairs. That clearly does not provide any information to the algorithm.
$\mathcal{E}_2$: The algorithms finds a x_i satisfying $\sigma(x_i) = \sigma(P(a,b))$.

Let $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2$. We compute an upper bound for the probability that event $\mathcal{E}$ occurs. We give the algorithm the benefit of doubt and consider it in this case to be successful.
Case $\mathcal{E}_1$: Observe that a non-trivial collision $R_i(a,b) = R_j(a,b)$ for all a, b is never

possible. The probability of a collision $R_i(a,b) = R_j(a,b)$ for fixed $i \neq j$ can be bounded by considering the polynomial $F(x,y) = R_i - R_j$. This is a polynomial of total degree at most 1. Due to Lemma 7 the probability that a random $(x,y) \in \mathbb{Z}_m^2$ is a zero of F is bounded by $1/p$. There are $\binom{T+3}{2}$ possible distinct pairs of linear combinations R_i, R_j to consider. Consequently the probability that event $\mathcal{E}_1$ occurs is bounded by $\binom{T+3}{2} \cdot \frac{1}{p}$.
Case $\mathcal{E}_2$: The degree of $P(a,b) - R_i(a,b)$ is at most $2l$. There are $T+3$ such R_i to consider. With the same argument as above the probability that event $\mathcal{E}_2$ occurs is bounded by $2l(T+3)/p$.
In total we have

$$\Pr[\mathcal{E}] \leq \Pr[\mathcal{E}_1] + \Pr[\mathcal{E}_2] \leq \frac{(T+3)(T+2)}{2p} + \frac{2l(T+3)}{p} \leq O\left(\frac{T^2+lT}{p}\right).$$

□

The next lemma proves Theorem 8.

Lemma 10. *Let m be an integer and p the smallest prime factor of m. Let $S \subset \{0,1\}^*$ be a set of at least m binary strings. Let $P \in \mathbb{P}_l \setminus \mathbb{P}_1$. Assume that a generic algorithm $\mathcal{A}$ exists that works for groups of order m, runs in time at most T, makes calls to a P-DDH-oracle. Let $a, b, c \in \mathbb{Z}_m$ chosen at random, let $\sigma : \mathbb{Z}_m \to S$ be a random encoding function, and let t be a random bit. Set $w_0 = ab$ and $w_1 = c$. Then $\Pr[\mathcal{A}(\sigma; 1, a, b, w_t, w_{1-t}) = t] \leq 1/2 + O((T^2+lT)/p)$.*

Proof. The algorithm makes A computation steps and knows the encodings $\sigma(x_i)$ of some $x_i \in \mathbb{Z}_m$, $1 \leq i \leq A+5$. x_i clearly is of the form $x_i = R_i(a,b,ab,c)$, where R_i is a linear function in a, b, ab and c. We can assume that $R_i \neq R_j$ holds for all $i \neq j$. Furthermore the algorithm makes B calls to the DDH-oracle. The algorithm may be able to output the correct bit t only by obtaining information from either of the following events $\mathcal{E}_1$ and $\mathcal{E}_2$:

- $\mathcal{E}_1$: The algorithm finds for $i \neq j$ a non-trivial collision of the form $\sigma(x_i) = \sigma(x_j)$. Note that if for all i, j one has $\sigma(x_i) \neq \sigma(x_j)$, then the algorithm has learned the random encoding of two *distinct* pairs. That clearly does not provide any information to the algorithm.
- $\mathcal{E}_2$: The algorithm gets at least one positive answers for a non-trivial query to the P-DDH-oracle.

Let $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2$. We compute an upper bound for the probability that event $\mathcal{E}$ occurs. We give the algorithm the benefit of doubt and consider it in this case to be successful.
Case $\mathcal{E}_1$: Observe that a non-trivial collision $R_i(a,b) = R_j(a,b)$ for all a, b is never possible. The probability of a collision $R_i(a,b,ab,c) = R_j(a,b,ab,c)$ for fixed $i \neq j$ can be bounded by considering the polynomial $F(x,y,z) = R_i - R_j$. This is a polynomial of total degree at most 2. Due to Lemma 7 the probability that a random $(x,y,z) \in \mathbb{Z}_m^3$ is a zero of F is bounded by $2/p$. There are $\binom{A+5}{2}$ possible distinct pairs of linear combinations R_i, R_j to consider. Consequently

the probability that event $\mathcal{E}_1$ occurs is bounded by $\binom{A+5}{2} \cdot \frac{2}{p}$.
Case $\mathcal{E}_2$: For $i \neq j$ it is not possible to derive a (non-trivial) relation $R_i(a,b) = P(R_j(a,b), R_k(a,b))$. The degree of $P(R_j(a,b), R_k(a,b)) - R_i(a,b)$ is at most $2l$. With the same argument as above the probability that event $\mathcal{E}_2$ occurs is bounded by $2lB/p$.
In total we have $\Pr[\mathcal{E}] \leq \Pr[\mathcal{E}_1] + \Pr[\mathcal{E}_2] \leq \frac{2(A+5)(A+4)}{2p} + \frac{2lB}{p} \leq \frac{T^2+(4l+9)T+20}{p}$, where $T = A+B$ is the number of steps of the algorithm. The overall probability that the algorithm outputs the correct bit t therefore is:

$$\Pr[\mathcal{E}] + \frac{1}{2}\Pr[\overline{\mathcal{E}}] = \frac{1}{2} + \frac{\Pr[\mathcal{E}]}{2} \leq \frac{T^2 + (4l+9)T + 20}{p} = O\left(\frac{T^2 + lT}{p}\right).$$

□