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 *computationally equivalent* to computing P-DH. In addition we study the corresponding decision problem. 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.

1 Introduction

Let G be a cyclic finite group and let g be a generator of G. The Diffie-Hellman function, 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.

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 [2,8].

Clearly computing the P-DH function cannot be harder than computing the 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 provably *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. Finally, in Section 5 we 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 $\mathrm{Z}_{|G|}$ denote the ring of integer residue classes modulo $|G|$. Let $k = k(n)$ and $l = l(n)$ be two functions mapping integers to integers. Let $\mathrm{P}_l^k = \mathrm{P}_l^k(n)$ be the family of sets of all non-linear polynomials $P(a, b)$ over $\mathrm{Z}_{|G|}$ of the form $P(a, b) = \sum_{i,j \in \{0 \ldots l\}} c_{ij} a^i b^j$ with coefficients $c_{ij} \in \mathrm{Z}_{|G|}$ and absolute values $|c_{ij}|$ 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 $\mathrm{P}_l := \mathrm{P}_l^{\lfloor |G|/2 \rfloor}$ (no restrictions to coefficients) and $\mathrm{P} := \mathrm{P}_{|G|-1}$.

For a cyclic, finite group G, a fixed generator g of G and a polynomial $P \in \mathrm{P}$ 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]	$P(a, b) = ab$	$\mathrm{DH}(g^a, g^b) = g^{ab}$
Square Exponent function [5]	$P(a, b) = a^2$	$\mathrm{SE}(g^a) = g^{(a^2)}$
To-the-s Diffie-Hellman function	$P(a, b) = a^s$	$\mathrm{DH}^s(g^a) = g^{(a^s)}$
-	$P(a, b) = a^2b + ab^2$	$P\text{-DH}(g^a, g^b) = g^{a^2b+ab^2}$

Considered Group Families. Let $\mathcal{G} := (G_n, g_n)_{n \in \mathrm{N}}$ be a family of finite cyclic groups and generators. We define 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 $\mathrm{G(nsprime)} := \{\mathcal{G} : \forall \text{polynomials } R\ \exists n_0 \forall n \geq n_0 : \mathrm{minpf}(|G_n|) > R(n)\}$ 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 $(G, g) = (G_n, g_n)_{n \in \mathrm{N}} = \mathcal{G}$ be a family of groups and generators and let $\epsilon(n)$ be a function in n taking values in the interval $[0, 1]$. For $P \in \mathrm{P}$ 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 assumption 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 $\frac{1}{\mathrm{poly}(n)}$-$P$-CDH($\mathcal{G}$). Vice-versa, the 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 $\mathcal{G} \in \mathrm{G}$. We say that $\epsilon(n)$-P-CDH$_{\mathrm{nsprime}}$ holds, if $\epsilon(n)$-P-CDH($\mathcal{G}$) holds for every family $\mathcal{G} \in \mathrm{G(nsprime)}$.

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 efficient algorithm $\mathcal{A}_B$ breaking assumption B then we can build another efficient algorithm $\mathcal{A}_A^{\mathcal{A}_B}$ with (oracle) access to $\mathcal{A}_B$ which breaks assumption A.

Generic Algorithms (Notation of Shoup [7]). An encoding function on the additive group $(\mathrm{Z}_m, +)$ is an unknown injective map $\sigma : \mathrm{Z}_m \to \{0, 1\}^n$ 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 Z_m is a probabilistic algorithm that takes as input an encoding list $(\sigma(x_1), \dots, \sigma(x_k))$ 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. The new encoding is then added to the encoding list. The output of the algorithm is denoted by $\mathcal{A}(\sigma; x_1, \dots, x_k)$. An example of a generic algorithm is the Pohlig-Hellman algorithm 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 *true* $\overset{\sigma}{\Rightarrow} B$ means that there is 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.

3 The Computational Case

3.1 Previous Work

Theorem 1. *1. true* $\overset{\sigma}{\Rightarrow}$ $\frac{1}{\mathrm{poly}(n)}$-$\mathrm{CDH}_{\mathrm{nsprime}}$ *(Shoup [7]).*

2. true $\overset{\sigma}{\Rightarrow}$ $\frac{1}{\mathrm{poly}(n)}$-$\mathrm{CSE}_{\mathrm{nsprime}}$ *(Wolf [9]).*

3. $\frac{1}{\mathrm{poly}(n)}$-CDH $\Leftrightarrow$ CDH *(Shoup's Diffie-Hellman self-corrector [7]).*

4. $\frac{1}{\mathrm{poly}(n)}$-CDH $\Leftrightarrow$ $\frac{1}{\mathrm{poly}(n)}$-CSE *(Maurer and Wolf [5]) .*

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.

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

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

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

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

No *generic* algorithm can efficiently break the assumption $\frac{1}{\mathrm{poly}(n)}$-$P$-$\mathrm{CDH}_{\mathrm{nsprime}}$:

Theorem 4. *For every $P \in \mathrm{P}_{\mathrm{poly}(n)}$ we have: true* $\overset{\sigma}{\Rightarrow}$ $\frac{1}{\mathrm{poly}(n)}$-$P$-$\mathrm{CDH}_{\mathrm{nsprime}}$.

The proof of Theorem 4 uses techniques due to Shoup [7] and can be found in the full version of this paper [4].

Theorem 5 (P-DH self-corrector). *For every constant l and every $P \in \mathrm{P}_l$ we have:* $\frac{1}{\mathrm{poly}(n)}$-$P$-CDH $\Leftrightarrow$ P-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 [9]. For a finite cyclic group G of known order $|G|$ let $d \in \mathrm{Z}_{|G|}$ and $x, a \in G$. Then the equation

$$x^d = a$$

has exactly $s := \gcd(|G|, d)$ different solutions $x_1, \ldots, x_s$ (there must be at least one, x). 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 compute this relation. Therefore a complete factorization of $|G|$ is not needed, only $|G|$ must be known. The proof of the following simple lemma can be found in the full version [4]:

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

The following central lemma says that once we are given an algorithm that computes DH with *non-negligible* probability of success, then we can compute P-DH with *overwhelming* probability of success for any polynomial $P(a, b)$. Recall that, for instance, P-CDH is the assumption that there is *no efficient algorithm* that computes the P-Diffie-Hellman function.

Lemma 2. *For every $P \in \mathrm{P}_{\mathrm{poly}(n)}$ we have: P-CDH $\Rightarrow \frac{1}{\mathrm{poly}(n)}$-CDH.*

Proof. Fix the family $(G, g) = (G_n, g_n)_{n \in \mathrm{N}} \in \mathrm{G}$. Assume $\frac{1}{\mathrm{poly}(n)}$-CDH is wrong, i.e. there is an oracle that computes DH with non-negligible probobility of success. 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 $g^{c_{ij} a^i b^j}$ 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 brakes assumption P-CDH. $\square$

With this observation at hand the proof of Theorem 5 (P-DH self-corrector) is easy. Clearly "$\Rightarrow$" holds. To prove "$\Leftarrow$" we "detour" over DH. This will be a very frequently used strategy in our proofs. Let $P \in \mathrm{P}_l$ 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 $\mathcal{A}^{\mathcal{O}_{P-\mathrm{DH}}}$ 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 algorithm that computes P-DH for a $P \in \mathrm{P}_l$ then there is an algorithm that computes DH. Lemma 3 deals with the special case $P \in \mathrm{P}_2$. It can be viewed as the induction base. In Lemma 4 computing P-DH for a $P \in \mathrm{P}_l$ is reduced through an efficient algorithm to computing Q-DH for a $Q \in \mathrm{P}_{l-1}$. 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 $P \in \mathrm{P}_2^{\mathrm{poly}(n)}$ we have: P-CDH $\Leftarrow$ CDH.*
2. For $P \in \mathrm{P}_2$ we have: $\frac{1}{\mathrm{poly}(n)}$-$P$-CDH$_{\mathrm{nsprime}} \Leftarrow \frac{1}{\mathrm{poly}(n)}$-CDH$_{\mathrm{nsprime}}$.

Proof. We first prove part 1 of the lemma. Let $P \in \mathrm{P}_2^{\mathrm{poly}(n)}$. Because of Lemma 2 it is sufficient to show P-CDH $\Leftarrow \frac{1}{\mathrm{poly}(n)}$-CDH. Let $(G, g) = (G_n, g_n)_{n \in \mathrm{N}} \in \mathrm{G}$. 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 $C = Y_+ \cdot (Y_-)^{-1} = g^{4c_2 \cdot ab + 2c_1 \cdot b}$ where all $c_i := \sum_{j=0}^{2} c_{ij}$ are known. First assume $c_2 \neq 0$. Now $g^{4c_2 \cdot ab} = C \cdot (g^b)^{-2c_1}$ 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 $s := \gcd(4c_2, |G|)$), i.e. all solutions of the equation

$$x^{4c_2} = g^{4ab \cdot c_2}, \tag{1}$$

with $x = g^{ab}$. This can be done in time $O(sn^3) = \mathrm{poly}(n)$, because for all coefficients, $c_{ij} = \mathrm{poly}(n)$ holds. Now output one of the roots of equation (1) at random, one of them is the correct one, g^{ab}. Hence, for the case $c_2 \neq 0$ the success probability of the $\mathcal{A}^{\mathcal{O}_{P\text{-DH}}}$ is $\epsilon(n) \ge 1/s \ge 1/(4c_2) = 1/\mathrm{poly}(n)$.
In the case $c_2 = 0$ we query the oracle for $P\text{-DH}(g^{a \pm b}, g^2)$ or $P\text{-DH}(g^{a \pm b}, g^3)$. As shown in the full paper [4] at least one of those queries leads to a successful computation of g^{ab}. This completes the proof of part 1.

Now let $(G, g) = (G_n, g_n)_{n \in \mathrm{N}} \in \mathrm{G}(\text{nsprime})$ and let $P \in \mathrm{P}_2$. 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 $\epsilon(n) = 1/\mathrm{poly}(n)$. 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

$$Y_+ = g^{c_2(a+b+s)^2 + c_1(a+b+s) + c_0} = g^{c_2(a+b)^2 + 2c_2(a+b)s + c_2s^2 + c_1(a+b+s) + c_0}$$

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 $\epsilon^2(n) = 1/\mathrm{poly}(n)$. □

The next lemma is the "induction step" to proof Theorems 3 and 2.

Lemma 4. *1. For a $P \in \mathrm{P}_l^k$ let $\mathcal{O}_{P\text{-DH}}$ be an oracle that breaks P-CDH. Then for $k' := 2kl2^l$ there is a $Q \in \mathrm{P}_{l-1}^{k'}$ 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 $P \in \mathrm{P}_l$ let $\mathcal{O}_{P\text{-DH}}$ be an oracle that breaks $\epsilon(n)$-P-$\mathrm{CDH}_{\mathrm{nsprime}}$. Then there is a $Q \in \mathrm{P}_{l-1}$ and an efficient algorithm $\mathcal{A}^{\mathcal{O}_{P\text{-DH}}}$ that breaks $\epsilon^3(n)$-Q-$\mathrm{CDH}_{\mathrm{nsprime}}$ making at most* 3 *queries to $\mathcal{O}_{P\text{-DH}}$.*

Proof. We start proving the first part of this lemma. Let $P \in \mathrm{P}_l^k$ and let $(G, g) = (G_n, g_n)_{n \in \mathrm{N}} \in \mathrm{G}$. The main idea of this proof again is to eliminate any appearance of $g^{a^l b^j}$ or $g^{a^i b^l}$ for any i, j by computing $P\text{-DH}(g^{a+b}, g) \cdot (P\text{-DH}(g^{a-b}, g))^{-1}$.

Case 1: l even. Making *two* queries to the oracle $\mathcal{O}_{P\text{-DH}}$, algorithm $\mathcal{A}^{\mathcal{O}_{P\text{-DH}}}$ gets

$$\begin{aligned} Y_+ &= P\text{-DH}(g^{a+b}, g) = g^{c_l(a+b)^l + c_{l-1}(a+b)^{l-1} + \cdots + c_0(a+b)} \\ \text{and} \quad Y_- &= P\text{-DH}(g^{a-b}, g) = g^{c_l(a-b)^l + c_{l-1}(a-b)^{l-1} + \cdots + c_0(a-b)}, \end{aligned}$$

where $c_i := \sum_{j=1}^{l} c_{ij}$. Now assume $c_l \neq 0$. c_l might be 0, but in this case continue with the same trick as in the proof of Lemma 3 (1). Because l is even division leads to $g^{Q(a,b)} = Y_+ \cdot (Y_-)^{-1}$ where

$$\begin{aligned} g^{Q(a,b)} &= g^{c_l((a+b)^l - (a-b)^l) + c_{l-1}((a+b)^{l-1} - (a-b)^{l-1}) + \sum_{i=0}^{l-2} c_i((a+b)^i - (a-b)^i))} \\ &= g^{2c_l(\binom{l}{l-1}a^{l-1}b + \binom{l}{l-3}a^{l-3}b^3 + \cdots + \binom{l}{1}ab^{l-1}) + 2c_{l-1}(\binom{l-1}{l-2}a^{l-2}b + \cdots + b^{l-1}) + \cdots}. \end{aligned}$$

Each coefficient of $a^i b^j$ of this polynomial $Q(a, b)$ is either 0 (if j is even) or $2c_{i+j}\binom{i+j}{i} \leq 2kl\binom{l}{l/2} \leq 2kl2^l =: k'$ (if j is odd). Note that the coefficients of the monomials a^l and b^l are always 0. Thus $Q(a, b) \in \mathrm{P}_{l-1}^{k'}$. Algorithm $\mathcal{A}^{\mathcal{O}_{P\text{-DH}}}$ outputs $g^{Q(a,b)} = \text{Q-DH}(g^a, g^b)$.

Case 2: l 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 $Q(a, b) = 2c_l l a^{l-1} b - 2c_{l-1} b^{l-1} + \cdots$. A similar computation as in the even case shows that $Q(a, b) \in \mathrm{P}_{l-1}^{k'}$ with k' defined as above. Algorithm $\mathcal{A}^{\mathcal{O}_{P\text{-DH}}}$ outputs $g^{Q(a,b)} = Q\text{-DH}(g^a, g^b)$. This completes the proof of the first part.

The proof of the second part of the lemma can be found in [4]. □

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

Proof (of Theorem 3). Let $(G, g) = (G_n, g_n)_{n \in \mathrm{N}} \in \mathrm{G}$. For $l \in O(\sqrt{\log n})$ and $k \in \mathrm{poly}(n)$ let $P \in \mathrm{P}_l^k$. Due to Lemma 2 it is sufficient to show P-CDH $\Leftarrow$ 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. $Q \in \mathrm{P}_2^{f(k,l)}$, where $f(k, l) = k \cdot \prod_{i=2\ldots l} i2^i \leq k \cdot l!2^{\frac{(l+1)l}{2}} = \mathrm{poly}(n) \cdot \mathrm{poly}(n) = \mathrm{poly}(n)$. The number of queries to $\mathcal{O}_{P\text{-DH}}$ is at most $3^{l-2} = \mathrm{poly}(n)$. Now use Lemma 3 (1) to construct an efficient algorithm $\mathcal{B}^{\mathcal{O}_{P\text{-DH}}}$ that computes $\mathrm{DH}(g^a, g^b)$ with overwhelming probability of success. □

The proof of Theorem 2 is similar and can be found in the full paper [4].

4 The Decisional Case

Let $\mathcal{G} = (G_n, g_n)_{n\in\mathrm{N}} = (G, g)$ 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 $\mathcal{A}$ 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 $\mathcal{A}$). Let $\epsilon(n)$-P-DDH$_{\mathrm{nsprime}}$, $\epsilon(n)$-P-DDH$_{\mathrm{nsprime}}$, $\epsilon(n)$-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* $\overset{\sigma}{\Rightarrow} \frac{1}{\mathrm{poly}(n)}$-DDH$_{\mathrm{nsprime}}$ *(Shoup [7]).*

2. true $\overset{\sigma}{\Rightarrow} \frac{1}{\mathrm{poly}(n)}$-DSE$_{\mathrm{nsprime}}$ *(Wolf [9]).*

3. $\frac{1}{\mathrm{poly}(n)}$-DDH $\overset{\sigma}{\not\Rightarrow} \frac{1}{\mathrm{poly}(n)}$-DSE$_{\mathrm{nsprime}}$ *(Wolf [9]).*

4. $\frac{1}{\mathrm{poly}(n)}$-DDH $\Leftarrow \frac{1}{\mathrm{poly}(n)}$-DSE *(Wolf [9]).*

4.2 This Work

We define the set of polynomials P_{2^*} for which the reduction from P-DDH to DDH is possible. Let $\mathrm{P}_{2^*} \subset \mathrm{P}_2$ be the set of polynomials $P(a, b)$ given by

$$P(a,b) = (d_{00}a + d_{01}b)(d_{10}a + d_{11}b) + c_{10}a + c_{01}b + c_{00}, \quad c_{ij}, d_{ij} \in \mathrm{Z}_{|G|}.$$

Example polynomials of P_{2^*} include $P(a, b) = a^2 - b^2$ and $P(a, b) = (a + b)^2$.

We characterize the relation between the assumptions $\frac{1}{\mathrm{poly}(n)}$-DDH and $\frac{1}{\mathrm{poly}(n)}$-$P$-DDH in the following two theorems. Remember that 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 $P \in \mathrm{P}_{\mathrm{poly}(n)} \setminus \mathrm{P}_{2^*}$ and $Q \in \mathrm{P}_{2^*}$ we have:*

1. $\frac{1}{\mathrm{poly}(n)}$-DDH$_{\mathrm{nsprime}} \overset{\sigma}{\not\Leftarrow} \frac{1}{\mathrm{poly}(n)}$-$P$-DDH$_{\mathrm{nsprime}}$.

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

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

$$\frac{1}{\mathrm{poly}(n)}\text{-DDH}_{\mathrm{nsprime}} \overset{\sigma}{\not\Rightarrow} \frac{1}{\mathrm{poly}(n)}\text{-}P\text{-DDH}_{\mathrm{nsprime}}.$$

The proof of Theorem 8 uses techniques due to Shoup [7] and can be found in the full paper [4]. The next theorem is a direct corollary of Theorem 7 (1).

Theorem 9. *For every $P \in \mathrm{P}_{\mathrm{poly}(n)}$ we have: true* $\overset{\sigma}{\Rightarrow} \frac{1}{\mathrm{poly}(n)}$-$P$-DDH$_{\mathrm{nsprime}}$.

4.3 Proofs

The following lemma gives an alternative characterization of P_{2^*}.

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

$$\forall a,b: \quad R(1,a,b,P(a,b)) \cdot S(1,a,b,P(a,b)) = T(1,a,b,P(a,b)). \quad (2)$$

2. Let $(G,g) \in \mathrm{G}(\text{nsprime})$ and let $P \in \mathrm{P}_{\text{poly}(n)}$. If relation (2) is satisfied, then $P \in \mathrm{P}_{2^*}$ 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 (2) can only hold for all a,b if $r_3 = s_3 = 0$. Thus, viewed as polynomials over a and b, relation (2) 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 $P(a,b) \in \mathrm{P}_2^*$, relation (2) can be satisfied (setting $t_3 = 1$). Now let $(G,g) \in \mathrm{G}(\text{nsprime})$ and relation (2) be satisfied. Due to Lemma 1, t_3 is invertible in $\mathrm{Z}_{|G|}$ with overwhelming probability. In this case obviously $P \in \mathrm{P}_{2^*}$ holds. □

The next lemma proves Theorem 7 (2).

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

Proof. We construct $\mathcal{A}^{\mathcal{O}_{\text{DDH}}}$ as follows: Let g^a, g^b and in random order $g^{P(a,b)}$ and g^c be given. Since $P \in \mathrm{P}_{2^*}$ we can compute $g^{R(a,b)}, g^{S(a,b)}$ and $g^{T(a,b)}$ with R, S, T satisfying relation (2) 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 $\mathcal{O}_{\text{DDH}}$ 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 $\mathcal{O}_{\text{DDH}}$ only once, thus the success probability of algorithm $\mathcal{A}^{\mathcal{O}_{\text{DDH}}}$ is $\epsilon(n)$. □

The next lemma says that for $P \in \mathrm{P}_{\text{poly}} \setminus \mathrm{P}_{2^*}$ 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 [7].

Lemma 7. *Let $m = m(n)$ be a family of integers whose smallest prime factors $p = p(n)$ are (asymptotically) lower bounded by a polynomial $R(n)$. Let $S \subset \{0,1\}^*$ be a set of at least m binary strings. Let $P \in \mathrm{P}_l \setminus \mathrm{P}_{2^*}$. Let $\mathcal{A} = \mathcal{A}(n)$ be generic algorithms that work for groups of order m, run in time at most $T = T(n)$ and make calls to a (perfect) DDH-oracle. Let $a,b,c \in \mathrm{Z}_m$ be chosen at random, let $\sigma : \mathrm{Z}_m \to S$ be a random encoding function, and let t be a random bit. Set $w_0 = P(a,b)$ and $w_1 = c$. Then $\Pr[\mathcal{A}(\sigma; 1, a, b, w_t, w_{1-t}) = t] \leq 1/2 + O(lT^2/p)$.*

Proof (Sketch). 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. See the full version [4] for a formal treatment of the proof. □

5 Conclusions and Open Problems

We presented a theoretical approach of a generalization of the Diffie-Hellman function. This P-Diffie-Hellman function is provably computationally equivalent to the Diffie-Hellman function for a certain class of groups. As the title of this paper suggests 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-DH function can replace the DH 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: As mentioned above it would be nice to have some more "real-world" applications of the P-Diffie-Hellman function.

The results in the computational case leave a lot of room for improvement. It would be interesting to see if one can improve our results to show that $\frac{1}{\text{poly}(n)}$-CDH $\Leftrightarrow$ $\frac{1}{\text{poly}(n)}$-P-CDH holds for $P \in \mathrm{P}_{\text{poly}(n)}$. In [6] the *Inverse Exponent* function $\text{IE}(g^a) = g^{(a^{-1})}$ is proven to be computationally equivalent to the DH function. Consequently one might ask the question what kind of functions f (others than polynomials) lead to f-Diffie-Hellman functions $f\text{-DH}(g^a, g^b) = g^{f(a,b)}$ that are computationally equivalent to the DH function. Also, a generalization of the defining polynomial $P(a, b)$ to $P(a_1, \cdots, a_k)$ 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 (full version). *Manuscript*, 2001.
5. U. Maurer and S. Wolf. Diffie-Hellman oracles. *Proc. of CRYPTO'96. Lecture Notes in Computer Science*, 1109:268–282, 1996.
6. 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.
7. V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology – EUROCRYPT '97*, pages 256–266, 1997.
8. I. Shparlinski. Security of most significant bits of g^{x^2}. *Inf. Proc. Letters*, (to appear).
9. Stefan Wolf. *Information-theoretically and Computionally Secure Key Agreement in Cryptography*. PhD thesis, ETH Zürich, 1999.