The Security of Polynomial Information of Diffie-Hellman Key

Yao Wang[1,2,3] and Kewei Lv[1,2](✉)

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
{wangyao,lvkewei}@iie.ac.cn

[2] Data Assurance Communication Security Research Center, Chinese Academy of Sciences, Beijing 100093, China

[3] University of Chinese Academy Sciences, Beijing 100049, China

Abstract. In this paper, we study the relations between the security of Diffie-Hellman (DH) key and the leakage of polynomial information of it again. Given a fixed sparse polynomial $F(X)$ and an oracle, which returns value of polynomial of DH key i.e., $F(g^{xy})$ when called by g^x and g^y, we obtain a probabilistic algorithm to recover the key. It is an extension of Shparlinski's result in 2004. This shows that finding polynomial information of DH key is as difficult as the whole key again. Furthermore, we study a variant of DH problem given 2 and g^y to compute 2^y and the n-DH problem with this method respectively, and obtain similar results.

Keywords: Diffie-Hellman key · m-sparse polynomial · Polynomial information · n-DH problem

1 Introduction

In 1976, Diffie and Hellman proposed a practical method to agree on a secret key over an insecure channel called Diffie-Hellman (DH) key exchange protocol. Let $g \in F_p^*$ be an element of multiplicative order t. In DH key exchange protocol over F_p^*, two parties calculate g^a, g^b respectively, where $a, b \in [0, t-1]$ and exchange them to form their common key $K = g^{ab}$. The element K has the same bit length n as p and n is chosen to make this protocol secure. Since then, many new cryptosystems have been proposed based on DH protocol.

In general, after the key exchange protocol is finished, both parties need to switch to a private key cryptosystem. For practicality and speed, they may wish to use a block cipher and therefore need to derive a much shorter bit string from K. A natural way would be to use a block of bits from g^{ab}. So when we analyze the security of DH key, bit security is an important aspect. Boneh and Venkatesan proved that a part (32 bits) of the most significant bits (MSB) is as secure as the whole (1024 bits) key. They showed that finding $n^{1/2}$ MSB of K is as difficult as the whole key in [2,3]. A detailed survey of several other results of this type of problem has been given in [4].

S. Qing et al. (Eds.): ICICS 2015, LNCS 9543, pp. 71–81, 2016.
DOI: 10.1007/978-3-319-29814-6_7

Verheul [5] studies another aspect of DH key. Assume $q = p^t$ *and* $\gamma \in F_q$ is a generator of a group. If $g(x) = \sum_{i=0}^{t} a_i x^i$ is an irreducible polynomial of degree t in $F_p[X]$, then we can describe the extension field F_q as $F_p[x]/g(x)$, i.e., each element f in F_q can be uniquely written modulo $g(x)$, as a polynomial of degree $< t$. In this setting, for any i less than t, let f_i denote the i-th coefficient of an element f. There exists a function that would be a linear mapping from F_q onto F_p and its value is the coefficient f_i. [5] proves that this function can be expressed as $F(X) = \Sigma_{i=1}^{m} c_i X^{e_i} (c_i \in F_q)$ and c_i can be easily determined. [5] also studies the security of polynomial information of DH key and proves that finding coefficients f_i (i.e., polynomial information $F(\gamma^{xy})$) of DH key γ^{xy} is as difficult as the whole key.

As an application of [5,6] gives a variant of DH scheme. In this variant, both parties send each other the minimal polynomials of γ^x, γ^y rather than the element themselves and the exchanged key is some coefficient of non-constant term of the minimal polynomial of γ^{xy}. This coefficient can be expressed as $F(\gamma^{xy})$ and the polynomial F must have a very large degree, such that it is unfeasible to find γ^{xy} by solving the equation $F(\gamma^{xy}) = A$. [5,6] proves that if we are given an oracle which for each pair (γ^x, γ^y) returns $F(\gamma^{xy})$, then one can construct a polynomial time algorithm to recover γ^{xy}. This shows that the variant are at least as secure as the original DH scheme over a multiplicative group of F_q. So the security of polynomial information of DH key is closely related to the security of DH key.

Shparlinski's result [1] is a generalization of [5,6]. It studies the security of polynomial transformations of DH key. Indeed, this polynomial transformation is a value of given polynomial function of the key. And [1] also extends [5] to the unreliable oracle case, that is, the oracle returns correct result only for a certain very small fraction of inputs and an error message for other inputs. Then an algorithm is given making expected number of calls of the oracle, to return γ^{xy}. It is deterministic when correct answers could be obtained from oracle. But it requires that the error answers of oracle could be identified. Moreover, in [1], only one part of the input to oracle is random and the other is fixed.

Here we improve the oracle and algorithm in [1] to get a probabilistic algorithm to recover DH key. In this improvement, not only the error from oracle could be not identified, but also both parts of inputs are random. In our algorithm, we use the Chebyshev inequality to identify error answers of the oracle. And for the two parts random inputs (γ^x, γ^y) of the oracle, we use the Markov inequality to find a good y which makes us have a sufficient advantage in receiving correct answers from the oracle taking over x only. Thus we can solve a nonsingular system of linear equations to recover DH key. As corollaries, we study two special cases. Finally, we use the same method to study variants of DH problem, i.e., given 2 and g^y trying to recover the key 2^y and the n-DH problem.

2 Preliminaries

In order to show our algorithms, we need an estimate on the number of zeros of polynomials from [1] and two important inequalities. Let F_q be a finite field of q elements and F_p^* be a multiplicative subgroup of F_q, where p is a prime.

Lemma 1 ([1]). *For $m \geq 2$, elements $a_1, a_2, \ldots, a_m \in F_q^*$ and integers $e_1, \ldots, e_m$, an element $\theta \in F_q$ of multiplicative order t. We denote by W the number of solutions of the equation $\sum_{i=1}^{m} a_i \theta^{e_i u} = 0, u \in [0, t-1]$. Then $W \leq 3t^{1-1/(m-1)} D^{1/(m-1)}$, where $D = min_{1 \leq i \leq m} max_{j \neq i} gcd(e_j - e_i, t)$.*

Let $E(\xi)$ be the expected value of a random variable ξ and $D(\xi)$ be the variance value of ξ. So $E_\xi[g(\xi)]$ denotes the expected value of a random variable $g(\xi)$, where the function g only depends on the distribution of ξ.

Theorem 1 *(Markov). For a positive c and a random variable ξ upper bounded by M, $Pr[\xi \geq E(\xi)/c] \geq M^{-1}(1 - 1/c)E(\xi)$.*

Theorem 2 *(Chebyshev). For an arbitrary positive δ, $Pr[|\xi - E(\xi)| \geq \delta] \leq D(\xi)/\delta^2$.*

3 The Security of Polynomial Information of DH Key

Let γ be an element in F_q of multiplicative order t. We consider an m-sparse polynomial $F(X) = \sum_{i=1}^{m} c_i X^{e_i} \in F_q[X]$, where $c_1, \ldots, c_m \in F_q^*$ and $e_1, \ldots, e_m$ are pairwise distinct modulo t.

3.1 The Polynomial Information from an Imperfect Oracle

Let $0 < \varepsilon \leq 1$. Assume there exists an oracle $O_{F,\varepsilon}$ satisfying that, given values of (γ^x, γ^y) to the oracle, it returns correct values of $F(\gamma^{xy})$ for at least εt^2 pairs $(x, y) \in [0, t-1]^2$ and returns a random element of F_p^* for other pairs of $(x, y) \in [0, t-1]^2$. The case $\varepsilon = 1$ is a noise-free oracle which had been considered in [5]. So the following discussion only involves in the case of $\varepsilon < 1$.

Here we try to construct a nonsingular system of linear equations using polynomial information from the oracle. We firstly study how to select the coefficient matrix of this equation system.

Given $\theta \in F_p^*$, for a vector $\overrightarrow{u} = (u_1, u_2, \ldots, u_m)$, we say that $\overrightarrow{u}$ is good if $det(\theta^{e_i u_j})_{i,j \neq 1}^{m} \neq 0$. We set $U = \{\overrightarrow{u} | \overrightarrow{u} \ is\ good\}$. Here we estimate the possibility of finding a good $\overrightarrow{u}$.

Assume that for some $k (2 \leq k \leq m)$, we have already found $k-1$ elements $u_1, u_2, \ldots, u_{k-1} \in [0, t-1]$ with

$$det(\theta^{e_i u_j})_{i,j \neq 1}^{k-1} \neq 0 \tag{1}$$

We select element $u_k \in [0, t-1]$ until

$$det(\theta^{e_i u_j})_{i,j \neq 1}^{k} \neq 0 \tag{2}$$

We know that if the determinant (2) vanishes then u_k is a solution of equation

$$\triangle_1^{e_k u_k} + \cdots + \triangle_k^{e_1 u_k} = 0 \tag{3}$$

where, by (1), $\triangle_1 = det(\theta^{e_i u_j})_{i,j}^{k-1} \neq 0$.

Applying Lemma 1, the number of elements $u_k \in [0, t-1]$ satisfying (3) is at most $3t^{1-1/(k-1)}$. So the probability of finding $u_k \in [0, t-1]$ which satisfy (2) is at least $1 - 3t^{-1/(k-1)}$. If we select $u_1, u_2, .., u_m \in [0, t-1]$ uniformly and independently at random to get the vector $\overrightarrow{u} = (u_1, u_2, \ldots, u_m)$, then $Pr[\overrightarrow{u} \in U] \geq \prod_{i=2}^{m}(1 - 3t^{-\frac{1}{i-1}})$.

Since both parts of inputs to this oracle are random, for input (γ^x, γ^y) of oracle, using idea of [1], we hope that we could choose a set of good y with a high probability such that, for each good y, we have a sufficient advantage in receiving correct answers from the oracle taking over x only. Then we can query oracle by randomizing the γ^x-component, fixing a good y. Thus, we can obtain a probabilistic algorithm to recover γ^{xy} with a high probability.

Let ε_y be the average success probability of $O_{F,\varepsilon}$, taken over random x for a given y. Thus, $E_y[\varepsilon_y] = \varepsilon$. For $k = \lceil \log \frac{2}{\varepsilon} \rceil$, we say that y is *j-good* if $\varepsilon_y \in [2^{-j}, 2^{-j+1})$, $j = 1, 2, \ldots, k$. Let $S_j = \{y | y \text{ } is \text{ } j\text{-}good\}$ (thus we ignore any y satisfying $\varepsilon_y < \varepsilon/2$). By Theorem 1, for $c = 2, M = 1$, $Pr[\varepsilon_y \geq \frac{\varepsilon}{2}] \geq \frac{\varepsilon}{2}$.

If all j satisfy $Pr[(y+v) \in S_j] < \frac{\varepsilon \cdot 2^{j-2}}{k}$, then

$$\frac{\varepsilon}{2} \leq \sum_{j=1}^{k} 2^{-j+1} Pr[(y+v) \in S_j] < \sum_{j=1}^{k} 2^{-j+1} \cdot \frac{\varepsilon \cdot 2^{j-2}}{k} = \frac{\varepsilon}{2},$$

which is a contradiction. So there must exist j such that $Pr[(y+v) \in S_j] \geq \frac{\varepsilon \cdot 2^{j-2}}{k}$. It means that we could find a suitable v such that $y + v$ is *j-good*.

Now we present a probabilistic algorithm (Algorithm 1) to look for a suitable v.

Algorithm 1. On input (t, γ^y, B), given oracle $O_{F,\varepsilon}$, output (v, ε_{y+v}).

```
1: Choose a random $v \in [0, t-1]$ and set $count := 0$;
2: For a sufficiently large integer $B = poly(k)$, choose $r_1, r_2, \ldots, r_B$ randomly.
   Using $(\gamma^{r_i}, \gamma^{y+v})$ to call the oracle, it returns $A_i$, $i = 1, 2, .., B$;
3: For every $r_i$, compute $\gamma^{r_i(y+v)}$ and if $\gamma^{r_i(y+v)}$ equals $A_i$, set $count = count + 1$;
4: Compute the approximate value of $\varepsilon_{y+v} = \frac{count}{B}$. If $\varepsilon_{y+v} \geq \frac{\varepsilon}{2}$, outputs
   $(v, \varepsilon_{y+v})$, else aborts.
```

In Algorithm 1, for every r_i, we can compute correct values of $\gamma^{r_i(y+v)}$ to determine whether the output A_i of the oracle in step 2 is correct. So in step 4, we can get an approximate value of ε_{y+v} for some v. Because $k = \lceil \log \frac{2}{\varepsilon} \rceil$, we know that if y is *j-good*, that is, $\varepsilon_{y+v} \in [2^{-j}, 2^{-j+1})$, the value of ε_{y+v} must satisfy $\varepsilon_{y+v} \geq \frac{\varepsilon}{2}$. Thus if Algorithm 1 finds a suitable v satisfying $\varepsilon_{y+v} \geq \frac{\varepsilon}{2}$, we can get j and $y + v \in S_j$ with the probability of at least $\frac{\varepsilon}{2}$.

Based on Algorithm 1, we get Algorithm 2 which can output DH key by calling oracle $O_{F,\varepsilon}$.

Algorithm 2. On input $(t, \delta, \gamma, \gamma^x, \gamma^y)$, for $0 < \delta < 1$, given oracle $O_{F,\varepsilon}$, output γ^{xy}.

```
1: Run Algorithm 1 to find some $v \in [0, t-1]$ which satisfies $y + v \in S_j$ and
   compute $\theta = \gamma^{y+v}$;
2: Find some $\overrightarrow{u} = (u_1, u_2, \ldots, u_m)$ satisfying $\overrightarrow{u} \in U$;
3: For every component $u_j (j = 1, 2, \ldots, m)$ of $\overrightarrow{u}$, using $(\gamma^{x+u_j}, \gamma^{y+v})$ to call
   the oracle, it returns $A_j$;
4: Solve the system of equations $\sum_{i=1}^m c_i(\theta^{x+u_j})^{e_i} = A_j, j = 1, 2, \ldots, m$ to get
   a candidate value of $\gamma^{xy}$;
5: Choose some $n$ satisfying $n \geq \frac{1-2^{-jm}}{2^{jm}\delta^2(1-\varepsilon)}$. Repeat steps 2,3,4 $n$ times to get
   a list $\mathcal{L}$ of candidate values of $\gamma^{xy}$. If some $\gamma^{xy}$ appears $z (z \leq (2^{-jm} + \delta)n)$
   times in the list $\mathcal{L}$ and $|\frac{z}{n} - 2^{-jm}| \leq \delta$, output this $\gamma^{xy}$. Otherwise, choose
   another $n$ to redo this step.
```

Theorem 3. *Let t be a prime, $m \geq 2$ and an m-sparse polynomial $F(X) = \sum_{i=1}^m c_i X^{e_i} \in F_q[X]$, where $c_1, \ldots, c_m \in F_q^*$ and $e_1, \ldots, e_m$ are pairwise distinct modulo t. Given an oracle $O_{F,\varepsilon}$, Algorithm 2 can output DH key with a probability at least $\frac{\varepsilon}{2} \cdot (1 - \frac{1}{n\delta^2 \cdot 2^{jm}}) \cdot (1 - 3t^{-\frac{1}{m-1}})^{m-1}$ in time polynomial in (mn, B) by making $mn + B$ calls to the oracle. In particular, if $t \geq (\frac{3}{1-2^{-\frac{1}{m-1}}})^{m-1}$ and $\delta \geq (\frac{1}{n \cdot 2^{jm}})^{\frac{1}{2}}$, DH key could be found with a probability of at least $\frac{\varepsilon}{4}$.*

Proof. After running steps 1, 2, 3 of Algorithm 2, we can get a nonsingular system of linear equations

$$(\theta^{e_i u_j})_{i,j=1}^m \cdot \begin{pmatrix} c_1\theta^{xe_1} \\ \vdots \\ c_m\theta^{xe_m} \end{pmatrix} = \begin{pmatrix} A_1 \\ \vdots \\ A_m \end{pmatrix}$$

Because the coefficient matrix is non-singular, we can solve the system of equations to get the values of $(c_1\theta^{xe_1}, \ldots, c_m\theta^{xe_m})$. Because $m \geq 2$ and $e_1, \ldots, e_m$ are pairwise distinct modulo t, at least one of $e_1, \ldots, e_m$ is relatively prime to t. So we can find an integer $f_i \in [0, t-1]$ satisfying $e_i f_i \equiv 1 (mod\ t)$ and compute $\theta^x = (\theta^{xe_i})^{f_i}$, that is, $(\gamma^{y+v})^x$. Thus we can get a candidate value of γ^{xy} from $(\gamma^{y+v})^x = \gamma^{xy} \cdot \gamma^{xv}$.

Then we estimate the probability of which step 5 outputs a correct value of DH key. When we use $(\gamma^{x+u_i}, \gamma^{y+v})$ to call the oracle, it returns the correct value of $F(\gamma^{(x+u_i)(y+v)})$ with probability at least 2^{-j}. So after repeating steps 2, 3, 4 one time, we can get the correct γ^{xy} with probability at least 2^{-jm}. From Theorem 2, with the increasing number of repetitions n in step 5, the value of $\frac{z}{n}$ infinitely close to 2^{-jm}. Thus the value of δ should be chosen as small as possible, but at this time, the value of n will be very big. In order to keep the efficiency of the algorithm, one should make a trade-off in choosing the value of $\delta \in (0, 1)$. Here by Theorem 2, we know that the output of step 5 is correct with the probability of $Pr[|\frac{z}{n} - 2^{-jm}| \leq \delta] \geq 1 - \frac{2^{-jm}(1-2^{-jm})}{n\delta^2}$.

The success of the Algorithm 2 means that steps 1, 2, 5 run successfully. So the successful probability of Algorithm 2 is:

$$\begin{aligned} Pr[Algorithm\ 2\ succeeds] \geq & \frac{\varepsilon}{2} \cdot (1 - \frac{2^{-jm}(1-2^{-jm})}{n\delta^2}) \cdot \prod_{i=2}^{m}(1 - 3t^{-\frac{1}{i-1}}) \\ \geq & \frac{\varepsilon}{2} \cdot (1 - \frac{1}{n\delta^2 \cdot 2^{jm}}) \cdot (1 - 3t^{-\frac{1}{m-1}})^{m-1} \end{aligned}$$

Obviously, Algorithm 1 run in polynomial time $poly(B)$ and steps 2, 3, 4, 5 of Algorithm 2 are done in polynomial time $poly(mn)$. So Algorithm 2 is done in time polynomial in (mn, B). When Algorithm 2 succeeds, we run the step 1 one time with B calls to the oracle and repeat step 2, 3, 4 n times with mn calls. Thus, we make the totally number of $mn + B$ calls to the oracle.

If $t \geq (\frac{3}{1-2^{-\frac{1}{m-1}}})^{m-1}$, we know $\prod_{i=2}^{m}(1 - 3t^{-\frac{1}{i-1}}) \geq \frac{1}{2}$. In order to output DH key with a probability at least $\frac{\varepsilon}{4}$, one can choose the value of δ at least $(\frac{1}{n \cdot 2^{jm}})^{\frac{1}{2}}$. This completes the proof.

3.2 Further Discussions on Another Two Cases

From Sect. 3.1, we can get two special cases. In these cases, we give two algorithms which can recover DH key by calling two special oracles respectively.

Assume that there is a special oracle $\tilde{O}_{F,\varepsilon}$ satisfying that, for every $x \in [0, t-1]$, when we use (γ^x, γ^y) to make a call of the oracle, it returns the correct value of $F(\gamma^{xy})$ for at least εt values of $y \in [0, t-1]$ and returns a random element of F_p^* for other values of $y \in [0, t-1]$. Here the error output from oracle $\tilde{O}_{F,\varepsilon}$ could not be identified.

We give Algorithm 3 to recover DH key by calling oracle $\tilde{O}_{F,\varepsilon}$.

Algorithm 3. On input $(t, \delta, \gamma, \varepsilon, \gamma^x, \gamma^y)$, for $0 < \delta < 1$, given oracle $\tilde{O}_{F,\varepsilon}$, output γ^{xy}.

```
1: Set $\theta = \gamma^x$. Find some $\overrightarrow{u} = (u_1, u_2, \ldots, u_m)$ satisfying $\overrightarrow{u} \in U$;
2: For every component $u_j (j = 1, 2, \ldots, m)$ of $\overrightarrow{u}$, using $(\gamma^x, \gamma^{y+u_j})$ to call the
   oracle, it returns $A_j$;
3: Solve the system of equations $\sum_{i=1}^{m} c_i(\theta^{y+u_j})^{e_i} = A_j, j = 1, 2, \ldots, m$ to get
   a candidate value of $\gamma^{xy}$;
4: Choose some $n$ satisfying $n \geq \frac{\varepsilon^m(1-\varepsilon^m)}{(1-\varepsilon)\delta^2}$. Repeat steps 1, 2, 3 $n$ times to get
   a list $\mathcal{L}$ of candidate values of $\gamma^{xy}$. If some $\gamma^{xy}$ appears $z(z \leq (\varepsilon^m + \delta)n)$
   times in the list $\mathcal{L}$ and $|\frac{z}{n} - \varepsilon^m| \leq \delta$, output this $\gamma^{xy}$. Otherwise, choose
   another $n$ to redo this step.
```

Theorem 4. *Let t be a prime, $m \geq 2$ and an m-sparse polynomial $F(X) = \sum_{i=1}^{m} c_i X^{e_i} \in F_q[X]$, where $c_1, \ldots, c_m \in F_q^*$ and $e_1, \ldots, e_m$ are pairwise distinct modulo t. Given an oracle $\tilde{O}_{F,\varepsilon}$, Algorithm 3 can output DH key with a probability of at least $(1 - \frac{\varepsilon^m}{n\delta^2}) \cdot (1 - 3t^{-\frac{1}{m-1}})^{m-1}$ in time polynomial in mn by making mn calls to the oracle.*

Proof. The proof is similar to Theorem 3, except that the probability of success is different. Using $(\gamma^x, \gamma^{y+u_j})$ to call the oracle, it returns the correct value of $F(\gamma^{x(y+u_j)})$ with probability at least ε. So after repeating steps 1, 2, 3 one time, we can get the correct γ^{xy} with probability at least ε^m. By Theorem 2, for some $0 < \delta < 1$, the output of step 4 is correct with the probability of $Pr[|\frac{z}{n} - \varepsilon^m| \leq \delta] \geq 1 - \frac{\varepsilon^m(1-\varepsilon^m)}{n\delta^2}$.

The success of Algorithm 3 means that steps 1, 4 run successfully. So the successful probability of Algorithm 3 is:

$$\begin{aligned} Pr[Algorithm\ 3\ succeeds] \geq & (1 - \frac{\varepsilon^m(1-\varepsilon^m)}{n\delta^2}) \cdot \prod_{i=2}^{m}(1 - 3t^{-\frac{1}{i-1}}) \\ \geq & (1 - \frac{\varepsilon^m}{n\delta^2}) \cdot (1 - 3t^{-\frac{1}{m-1}})^{m-1} \end{aligned}$$

Obviously, when the Algorithm 3 succeeds, we repeat step 1, 2, 3 n times. Thus, we make the totally number of mn calls to the oracle.

Assume that there is another special oracle $\hat{O}_{F,\varepsilon}$ satisfying that, given values of (γ^x, γ^y) to the oracle, it returns correct values of $F(\gamma^{xy})$ for at least εt^2 pairs $(x, y) \in [0, t-1]^2$ and returns an error message for other pairs of $(x, y) \in [0, t-1]^2$. The oracle $\hat{O}_{F,\varepsilon}$ makes two parts of inputs randomize instead of only one part in [1].

Here we give Algorithm 4 to recover DH key by calling oracle $\hat{O}_{F,\varepsilon}$.

Algorithm 4. On input $(t, \gamma, \gamma^x, \gamma^y)$, given oracle $\hat{O}_{F,\varepsilon}$, output γ^{xy}.

```
1: Run Algorithm 1 with the oracle $\hat{O}_{F,\varepsilon}$ to find some $v \in [0, t-1]$ satisfying
   $y + v \in S_j$ and compute $\theta = \gamma^{y+v}$;
2: Find some $\overrightarrow{u} = (u_1, u_2, \ldots, u_m)$ satisfying $\overrightarrow{u} \in U$. For every component
   $u_j (j = 1, 2, \ldots, m)$ of $\overrightarrow{u}$, computing $\gamma^{x+u_j}$. Use $(\gamma^{x+u_j}, \gamma^{y+v})$ to call the
   oracle, it returns value of $F(\gamma^{(x+u_j)(y+v)}) = A_j$.
3: Solve the system of equations $\sum_{i=1}^{m} c_i(\theta^{x+u_j})^{e_i} = A_j, j = 1, 2, \ldots, m$ to get
   the value of $\gamma^{xy}$, output this $\gamma^{xy}$.
```

Theorem 5. *Let t be a prime, $m \geq 2$ and an m-sparse polynomial $F(X) = \sum_{i=1}^{m} c_i X^{e_i} \in F_q[X]$, where $c_1, \ldots, c_m \in F_q^*$ and $e_1, \ldots, e_m$ are pairwise distinct modulo t. Given an oracle $\hat{O}_{F,\varepsilon}$, Algorithm 4 can output DH key with a probability of at least $\frac{\varepsilon}{2^{jm+1}} \cdot (1 - 3t^{-\frac{1}{m-1}})^{m-1}$ in time polynomial in (m, B) by making $m + B$ calls to the oracle.*

Proof. The proof is similar to Theorem 3 except for the probability of success. The success of Algorithm 4 means that steps 1, 2 run successfully. Step 2 can find a suitable u_j such that oracle returns the correct values of $F(\gamma^{(x+u_j)(y+v)})$ and $det(\theta^{e_i u_j})_{i,j=1}^{m} \neq 0$. So step 2 runs successfully with probability of at least $(2^{-j})^m \prod_{i=2}^{m}(1 - 3t^{-\frac{1}{i-1}})$.

Thus, the successful probability of Algorithm 4 is:

$$\begin{aligned} Pr[Algorithm\ 4\ succeeds] \geq & \frac{\varepsilon}{2} \cdot (2^{-j})^m \cdot \prod_{i=2}^{m} (1 - 3t^{-\frac{1}{i-1}}) \\ \geq & \frac{\varepsilon}{2^{jm+1}} \cdot (1 - 3t^{-\frac{1}{m-1}})^{m-1} \end{aligned}$$

When Algorithm 4 succeeds, we run the step 1 one time with B calls to the oracle and the step 2 one time with m calls. Thus, we make the totally number of $m + B$ calls to the oracle.

Algorithms 2–4 all show that finding polynomial information of DH key is as difficult as the whole key.

4 Some Variants of DH Problem and Their Polynomial Information Security

In this section, we present some variants of DH problem, such as, $DH_g(2, g^y)$, the n-DH problem and Multiple DH problem. For these variants, we give algorithms and theorems with the similar method to Sect. 3 respectively. All theorems show that finding polynomial information of DH key of these variants is also as difficult as the whole key.

4.1 DH Problem $DH_g(2, g^y)$

In [2], there is a new variant of the DH key exchange protocol. Say Alice and Bob wish to perform secret key over p. Alice picks a random number x in the range $[1, p-1]$ such that $gcd(x, p-1) = 1$, computes $g = 2^x (mod\, p)$ and sends g to Bob. Bob picks a random number y in $[1, p-1]$ and sends g^y to Alice. The key they agree on is $\alpha = 2^y (mod\, p)$. Clearly Bob can compute this value. Alice can compute this value since $2^y = g^{yx^{-1}} (mod\, p)$. So this variant of DH can be described as knowing 2 and g^y to recover the key 2^y, denoted $DH_g(2, g^y)$. There is an oracle $O_{F,\varepsilon}$ whose definition is the same as Sect. 3.1.

Corollary 1. *Given the oracle $O_{F,\varepsilon}$, 2^y can be recovered from $(2, g^y)$ running Algorithm 2.*

Proof. We use $(2, g^y)$ as the inputs of the oracle $O_{F,\varepsilon}$. Then running Algorithm 2 given input $(p, \delta, g, 2, g^y)$, we can get the value of 2^y.

4.2 The n-DH Problem

In [7], Cash, Kiltz and Shoup proposed a new computational problem and named it the twin Diffie-Hellman (twin DH) problem with the meaning that given a random triple of the form $(\gamma^{x_1}, \gamma^{x_2}, \gamma^{y}) \in F_q^3$, compute $\gamma^{x_1 y}$ and $\gamma^{x_2 y}$. [8] presented a modification of the twin DH problem by extending the number of the (ordinary) DH instances from 2 to an arbitrary integer n, and name it the n-DH problem. The n-DH problem is that given a random $n+1$ tuple of the form $(\gamma^{x_1}, \ldots, \gamma^{x_n}, \gamma^{y}) \in F_q^{n+1}$, compute $(\gamma^{x_1 y}, \ldots, \gamma^{x_n y})$.

Assume that there is a n-DH oracle $O^n_{F,\varepsilon}$ satisfying that, for every $x_i \in [0, t-1]$, $i = 1, 2, .., n$, given the values of $(\gamma^{x_i}, \gamma^{y})$, it returns correct $F(\gamma^{x_i y})$ for at least εt values of $y \in [0, t-1]$ and returns an error message for other values of $y \in [0, t-1]$.

Here we can construct an algorithm using similar method to recover n-DH key by calling oracle $O^n_{F,\varepsilon}$.

Algorithm 5. On input $(t, \gamma, \gamma^{x_1}, \ldots, \gamma^{x_n}, \gamma^{y})$, given oracle $O^n_{F,\varepsilon}$, output $(\gamma^{x_1 y}, \ldots, \gamma^{x_n y})$.

```
1: Set $\theta = \gamma^{x_1}$.Find some $\overrightarrow{u} = (u_1, u_2, \ldots, u_m)$ satisfying $\overrightarrow{u} \in U$. For every component $u_j$ of $\overrightarrow{u}$, using $(\theta, \gamma^{y+u_j})$ to call the oracle, it returns value of $F(\theta^{y+u_j}) = A_j$. Solve the system of equations $\sum_{i=1}^{m} c_i(\theta^{y+u_j})^{e_i} = A_j, j = 1, 2, \ldots, m$ to get the value of $\gamma^{x_1 y}$;
2: Replace $\theta$ with $\gamma^{x_2}$ and repeat step 1, we can get the value of $\gamma^{x_2 y}$;
3: Repeatedly calculate as step 2 until getting all values of $\gamma^{x_i y}$, then output values of $(\gamma^{x_1 y}, \ldots, \gamma^{x_n y})$.
```

Theorem 6. *Let t be a prime, $m \geq 2$ and an m-sparse polynomial $F(X) = \sum_{i=1}^{m} c_i X^{e_i} \in F_q[X]$, where $c_1, \ldots, c_m \in F_q^*$ and $e_1, \ldots, e_m$ are pairwise distinct modulo t. Given the oracle $O^n_{F,\varepsilon}$, Algorithm 5 can output n-DH key with a probability of at least $\varepsilon^{mn} \cdot (1 - 3t^{-\frac{1}{m-1}})^{(m-1)n}$ in time polynomial in mn by making mn calls to the oracle.*

Proof. It can easily imply from Theorem 5 that there exists an algorithm one can get value of $\gamma^{x_1 y}$ with a probability of at least $\varepsilon^{m} \cdot (1 - 3t^{-\frac{1}{m-1}})^{(m-1)}$. Algorithm 5 is n repeats of Algorithm 4, so it can output the value of n-DH key with a probability of at least $\varepsilon^{mn} \cdot (1 - 3t^{-\frac{1}{m-1}})^{(m-1)n}$.

4.3 Multiple DH Problem

Based on Sect. 4.2, we define another variant of DH problem. It can be described as knowing $\gamma^{x_1}, \gamma^{x_2}, \ldots, \gamma^{x_n}$ to recover the key $\gamma^{\prod_{i=1}^{n} x_i}$.

Assume that there is a Multiple DH oracle $O^M_{F,\varepsilon}$ satisfying that, for every $x_1, x_2, \ldots, x_n \in [0, t-1]$, given values of $(\gamma^{x_1}, \gamma^{x_2}, \ldots, \gamma^{x_n})$, it returns correct $F(\gamma^{\prod_{i=1}^{n} x_i})$ for at least εt values of $x_n \in [0, t-1]$ and returns an error message for other values of $x_n \in [0, t-1]$.

Here we can construct a recursion algorithm using similar method to Sect. 3 to recover Multiple DH key by calling oracle $O^M_{F,\varepsilon}$.

Algorithm 6. On input $(t, \gamma, \gamma^{x_1}, \ldots, \gamma^{x_n})$, given oracle $O^M_{F,\varepsilon}$, output $\gamma^{\prod_{i=1}^{n} x_i}$

```
1: Find some $\overrightarrow{u} = (u_1, u_2, \ldots, u_m)$ satisfying $\overrightarrow{u} \in U$. Set $\theta_1 = \gamma^{x_1}, \theta_2 = \gamma^{x_2}$. For every component $u_j$ of $\overrightarrow{u}$, using $(\gamma, \ldots, \gamma, \theta_1, \theta_2 \cdot \gamma^{u_j})$ to call the oracle, it returns value of $F(\theta_1^{x_2+u_j}) = A_j$. Solve the system of equations $\sum_{i=1}^{m} c_i(\theta_1^{x_2+u_j})^{e_i} = A_j, j = 1, 2, \ldots, m$ to get the value of $\gamma^{x_1 x_2}$;
```

2: Replace θ_1, θ_2 with $\gamma^{x_1 x_2}, \gamma^{x_3}$ and repeat step 1, we can get the value of $\gamma^{x_1 x_2 x_3}$;

3: Recursively calculate as step 2 until getting the value of $\gamma^{\prod_{i=1}^{n} x_i}$, then output this value of $\gamma^{\prod_{i=1}^{n} x_i}$.

Theorem 7. *Let t be a prime, $m \geq 2$ and an m-sparse polynomial $F(X) = \sum_{i=1}^{m} c_i X^{e_i} \in F_q[X]$, where $c_1, \ldots, c_m \in F_q^*$ and $e_1, \ldots, e_m$ are pairwise distinct modulo t. Given the oracle $O_{F,\varepsilon}^{M}$, Algorithm 6 which given $(\gamma^{x_1}, \gamma^{x_2}, \ldots, \gamma^{x_n})$ makes the expected number of at most $2m\varepsilon^{-1}(n-1)$ calls of the oracle, it returns the value of $\gamma^{\prod_{i=1}^{n} x_i}$.*

Proof. Theorem 5 has proved that there exists an algorithm one can get value of $\gamma^{x_1 x_2}$. We can easily know it needs the expected number of at most $2m\varepsilon^{-1}$ calls of the oracle. Algorithm 6 is $n-1$ recursions of Algorithm 4, so it can output the value of $\gamma^{\prod_{i=1}^{n} x_i}$ by the expected number of at most $2m\varepsilon^{-1}(n-1)$ calls of the oracle.

5 Conclusion

In this paper, we study the relations between security of DH key and its polynomial information, and give several algorithms to recover DH key γ^{xy} for different DH problems. These algorithms construct systems of equations to recover DH key by making polynomial number of calls to oracle to find polynomial information of DH key with a certain probability. And all these algorithms imply that finding polynomial information of DH key is as difficult as the whole key.

References

1. Shparlinski, I.E.: Security of polynomial transformations of the Diffie-Hellman key. Finite Fields Appl. **10**(1), 123–131 (2004)
2. Boneh, D., Venkatesan, R.: Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 129–142. Springer, Heidelberg (1996)
3. Vasco, M.I.G., Shparlinski, I.E.: On the security of Diffie-Hellman bits. In: Proceedings of the Workshop on Cryptography and Computer Number Theory, Singapore, 1999, pp. 257–268. Birkhauser, Basel (2001)
4. Vasco, M.I.G., Naslund, M.: A survey of hard core functions. In: Proceedings of the Workshop on Cryptography and Computational Number Theory, Singapore, 1999, pp. 227–256. Birkhauser, Basel (2001)
5. Verheul, E.R.: Certificates of recoverability with scalable recovery agent security. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 258–275. Springer, Heidelberg (2000)
6. Brouwer, A.E., Pellikaan, R., Verheul, E.R.: Doing more with fewer bits. In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 321–332. Springer, Heidelberg (1999)

7. Cash, D.M., Kiltz, E., Shoup, V.: The twin Diffie-Hellman problem and applications. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 127–145. Springer, Heidelberg (2008)
8. Chen, L., Chen, Y.: The *n*-Diffie-Hellman problem and its applications. In: Lai, X., Zhou, J., Li, H. (eds.) ISC 2011. LNCS, vol. 7001, pp. 119–134. Springer, Heidelberg (2011)