# On the complexity of Decomposition Attack

Koh-ichi Nagao
nagao@kanto-gakuin.ac.jp

Dept. of Engineering, Kanto Gakuin Univ.,
1-50-1 Mutsuura Higashi Kanazawa-ku Yokohama 236-8501, Japan

**Abstract.** In recent researches, it is discovered that index calculus is useful for solving the discrete logarithm problems (DLP) of the groups of the Jacobian of curves (including elliptic curve) over finite field, which are widely used to cryptosystems. In these cases, the probability that an element of the group is written by the summation of $N$ elements of large primes and factor bases is $O(1)$ where $N$ is some pre-fixed constant. So the situation is little different to the normal index calculus and it is proposed that it should be called another name, "decomposition attack". In decomposition attack, first, some relations are collected and the graph, whose vertexes are the set of large primes and whose edges are the relations, is considered and the elimination of large prime is done by using this graph. However, in the proposed algorithm, the randomness of the graph, which is difficult to define, is needed. In this paper, we first formulate the decomposition attack and next propose a new algorithm, which does not require the randomness of the graph and its worst complexity can be estimated.

**Keywords** Index Calculus, Decomposition Attack, Discrete logarithm problem

## 1 Introduction

Index calculus is widely used for solving DLP of a finite group. Recently, Gaudry[2] shows that for the DLP of a Jacobian group over finite field $G = \mathbf{Jac}(C/\mathbb{F}_q)$, taking the set of large primes $B_0 = \{P - \infty \,|P \in C(\mathbb{F}_q\}$, index calculus works well. Latter, the author [7] and Gaudry et al. [3] independently show that the technique of 2-large prime elimination is useful for decreasing its expected complexity. Moreover, Gaudry [4](elliptic curve case) and the author [8] (hyperelliptic curve case) show that for the DLP of a Jacobian group over finite extension field $G = \mathbf{Jac}(C/\mathbb{F}_{q^n})$, taking the set of large primes $B_0 = \{P - \infty \,|P \in C(\mathbb{F}_{q^n}, x(P) \in \mathbb{F}_q\}$, index calculus works well and also show that the technique of the 2-large prime elimination is useful in this algorithm. In the 2-large prime elimination, the graph, whose vertexes are the set of large primes and whose edges are the relations, which are collected in the computation, is considered. However, in the pre-proposed algorithm [7] [3], the randomness of the graph is needed and it is difficult to define. In this paper, we propose a new algorithm, which does not require the randomness of the graph and its worst complexity can be estimated. Let $G$ be a finite group whose order $|G|$ is prime. Here, we shortly note the summary of the 2-large prime elimination. Let $B$, $B_0$ be fixed subsets of $G$ such that $B \subset B_0 \subset G$. An element of $B$ (resp. $B_0 \backslash B$) is called factor base (resp. large prime) in the theory of index calculus. In the computation of the index calculus, a family of the vectors, indexed by the elements of $B_0$, $\overrightarrow{v}_i = \prod_{b \in B_0} v_{i,b} \in \mathbb{A}^{|B_0|}(\mathbb{Z}/|G|\mathbb{Z})\,(i = 1, 2, .., N_2(> |B_0|))$ are collected. Each vector satisfies the property: 1) For each vector $\overrightarrow{v}_i$, the number of non-zero

terms is at most fixed positive integer $N$ (i.e. $\#\{b \in B_0 | v_{i,b} \neq 0\} \leq N$). By using linear algebra of sparse matrix, we have some $\{s_i \in \mathbb{Z}/|G|\mathbb{Z}\,(i = 1, 2, .., N_2)\}$ such that $\sum_{i=1}^{N_2} s_i \overrightarrow{v}_i = 0$ and $\exists i, s_i \neq 0$. Its complexity is around $O(N_2^2)$. However, if one only collects the vectors satisfying that the number of non-zero terms of large prime is at most 2 (i.e. $\#\{b \in B_0 \backslash B | v_{i,b} \neq 0\} \leq 2$), and the value of these terms are 1(i.e, $v_{i,b} = 0$ or $1, for\, \forall b \in B_0 \backslash B$), the elimination of large prime is simply done as follows: Let $\mathbf{Graph} = (\mathbf{Edge}, \mathbf{Vertex})$ be the graph whose vertexes are large primes and the edge $\overline{bb'}$ exists if and only if there is a vector $\overrightarrow{v}_i$ with $v_{i,b} = v_{i,b'} = 1$. For simplicity, (only in Introduction), put $B = \{b_1, ..., b_{|B|}\}$, $B_0 \backslash B = \{b_{|B|+1}, ..., b_{B_0}\}$ and $\overrightarrow{v}_i = \{v_{i,1}, .., v_{i,|B|}, v_{i,|B|+1}, .., v_{i,|B_0|}\}$.

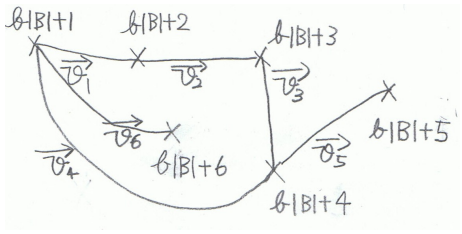


**Fig. 1.** Example of even length loop

For example, let

$$\begin{cases} \overrightarrow{v}_1 = (*****, 1, 1, 0, 0, 0, 0), \overrightarrow{v}_2 = (*****, 0, 1, 1, 0, 0, 0), \overrightarrow{v}_3 = (*****, 0, 0, 1, 1, 0, 0), \\ \overrightarrow{v}_4 = (*****, 1, 0, 0, 1, 0, 0), \overrightarrow{v}_5 = (*****, 0, 0, 0, 1, 1, 0), \overrightarrow{v}_6 = (*****, 1, 0, 0, 0, 0, 1) \end{cases}$$

where "****" is the terms of factor bases. Then the graph is written by Fig 1. This graph contains an even length loop $l = \overrightarrow{v}_1 \overrightarrow{v}_2 \overrightarrow{v}_3 \overrightarrow{v}_4$, and from even length loop, the elimination of large prime terms is done by $Eli(l) := \sum_{i=1}^{\text{length of loop}} (-1)^{i-1} \overrightarrow{v}_i = \overrightarrow{v}_1 - \overrightarrow{v}_2 + \overrightarrow{v}_3 + \overrightarrow{v}_4 = (****, 0, 0, 0, 0, 0, 0)$.

Moreover, suppose 1) $|B| \leq O(\sqrt{|B_0|}, 2)$ there are sufficiently large number of even small length($\leq O(\log |B_0|)$) loops, and 3) these loops can be computed in $O(|B_0|) \times \mathbf{poly}(\log |B_0|)$ complexity. So the computation of $\{s_i\}$ is reduced to solving sparse linear algebra of the size around $|B| \times |B|$ and total complexity is estimated by $O(|B_0|) \times \mathbf{poly}(\log |B_0|)$ which is very small. In [8], [3], there is proposed algorithm of collecting enough even small length loops, which works well under the assumption that 1) $N_2 = \text{Const} \times |B_0|$ where Const is some large constant, and 2) the graph is random. However, the randomness of the graph, which is hard to define, is needed for this algorithm. So, not only the worst complexity but also the guarantee that the algorithm returns the answer is not known. In this paper, we propose a modified index calculus algorithm whose worst complexity is known. In the concept and word of graph theory, we show the following theorem.

**Theorem 1.** *Let* $\mathbf{Graph} = (\mathbf{Edge}, \mathbf{Vertex})$ *be a non-oriented graph (multi path allowed, path to a vertex to itself not allowed, disjoint graph allowed). Assume* $|\mathbf{Edge}| - 4|\mathbf{Vertex}| > M'$ *for a positive integer* $M'$. *Then* **Graph** *contains a family of even length loops* $\{l_j | j = 1, 2, 3.., M''\}$ *and a family of edges* $\{d_j | j = 1, 2, .., M''\}$ *satisfying the following properties:*
*i)* $M'' \geq M'$, *ii) Length of* $l_j$ *is* $< 4 + 4 \log_2 |\mathbf{Edge}|$,
*iii) Edge* $d_j$ *only once appears in the loop* $l_j$ *and does not appear in the loop* $l_{j'}$ *for any* $j' > j$. *Moreover, there is an algorithm computing* $\{l_j\}$ *and* $\{d_j\}$ *from* **Graph**, *whose complexity is estimated by* $|\mathbf{Edge}| \times \mathbf{poly}(\log |\mathbf{Edge}|)$.

## 2 Discrete logarithm problem of Finite Group

Let $G$ be a finite group whose order $|G|$ is a prime.

**Definition 1 (DLP)** *Let $a, b \in G$ such that $a = nb$ for unknown $n \in \mathbb{Z}/|G|\mathbb{Z}$. DLP is the problem finding unknown $n$.*

For a while, we prepare some notations and assumptions of the complexity. Let $q$ be an input size. For a function $f(q)$, we define the notation of the complexity upto $\mathbf{poly}(\log q)$ term by $\frac{r_2}{(\log q)^{r_1}} \leq \tilde{O}(f(q)) \leq r_3(\log q)^{r_4}, \quad r_i \in \mathbb{R}_{>0}$. Also, for functions $f(q), g(q)$, we define $f(q) \approx g(q) \iff \tilde{O}(f(q)/g(q)) = 1$. Further, we will assume that i) $|G| \leq \tilde{O}(q^r)$ for some $r \in \mathbb{R}_{>0}$, ii) Data length of $g \in G$ and group operation of $G$ is in $\tilde{O}(1)$, iii) Group order $|G|$ is prime and known. Let $B_0$, $B$ be fixed subsets of $G$ such that $B \subset B_0 \subset G$. An element of $B$ is called factor base and an element of $B_0 \backslash B$ is called large prime in index calculus. Also let $N$ be a fixed small positive integer.

**Definition 2 (Decomposed factor)** *Let $g \in G$. $g$ is written by $g = \sum_{i=1}^{N} g_i$,*
*i) $g_i \in B_0$, resp. ii) $g_1, ..., g_{N-2} \in B$, $g_{N-1}, g_N \in B_0$,*
*resp. iii) $g_1, ..., g_{N-1} \in B$, $g_N \in B_0$, resp. iv) $g_1, ..., g_N \in B$,*
*$g$ is called i) potentially decomposed element, resp. ii) 2-almost decomposed element, resp. iii) almost decomposed element, resp. iv) decomposed element. In these cases, the set $\{g_i\}_{i=1}^{N}$ is called decomposed factor of $g$.*

Further, we will assume the following assumption, which is true for the index calculus of a Jacobian of a curve, for some $N$.

**Assumption 1 (Assumption of Index Calculus)** *a)The computation, whether $g$ is potentially decomposed element or not and the computation of $\{g_i\}$, is done by the complexity $\tilde{O}(1)$.*
*b) The probability that $g(\in G)$ is i) potentially decomposed element, resp. ii) 2-almost decomposed element, resp. iii) almost decomposed element, resp. iv) decomposed element is $O(1)$, resp. $O((|B|/|B_0|)^{N-2})$, resp. $O((|B|/|B_0|)^{N-1})$, resp. $O((|B|/|B_0|)^{N})$.*

## 3 Universe

In order to solve the DLP $a = nb$ ($a, b \in G$, $n \in \mathbb{Z}/|G|\mathbb{Z}$), we prepare the algebraic structure named Universe. Throughout this paper, finite group $G$, its two subsets $B_0$,$B$, and $a, b(\in G)$ in DLP are fixed.

**Definition 3 (Universe)** *Put*

$$\mathcal{U} := \{u = (g_u, \overrightarrow{v}_u, \overrightarrow{r}_u) \,|\, g_u \in G, \overrightarrow{v}_u = \prod_{b \in B_0} v_{u,b} \in \mathbb{A}^{|B_0|}(\mathbb{Z}/|G|\mathbb{Z}),$$
$$\overrightarrow{r}_u = (r_1, r_2) \in \mathbb{A}^2(\mathbb{Z}/|G|\mathbb{Z}) \text{ such that } g_u = r_1 a + r_2 b = \sum_{b \in B_0} v_{u,b}\, b\,\}.$$

*The set $\mathcal{U}$ is called Universe and for an element of $u \in \mathcal{U}$, the length of $u$ is defined by $lu(u) := \#\{b \in B_0 | v_{u,b} \neq 0\}$.*

**Definition 4** *Let* $\mathcal{U}_0$, $\mathcal{U}_1$, $\mathcal{U}_2$ *be the subset of* $\mathcal{U}$ *defined by the following:*
$\mathcal{U}_0 := \{u \in \mathcal{U} |\, v_{u,b} = 0 \text{ for } \forall b \in B_0 \backslash B\}$,
$\mathcal{U}_1 := \{u \in \mathcal{U} |\, v_{u,b} = 0 \text{ or } 1 \text{ for } \forall b \in B_0 \backslash B, \#\{b \in B_0 \backslash B |\, v_{u,b} = 1\} = 1\}$,
$\mathcal{U}_2 := \{u \in \mathcal{U} |\, v_{u,b} = 0 \text{ or } 1 \text{ for } \forall b \in B_0 \backslash B, \#\{b \in B_0 \backslash B |\, v_{u,b} = 1\} = 2\}$
*Moreover. let* $\mathcal{U}'_0 (\subset \mathcal{U}_0)$, $\mathcal{U}'_1 (\subset \mathcal{U}_1)$, $\mathcal{U}'_2 (\subset \mathcal{U}_2)$ *be the subset of* $\mathcal{U}$ *defined by* $\mathcal{U}'_i := \{u \in \mathcal{U}_i |\, lu(u) = N\}$, $(i = 0, 1, 2)$.

Let $r_1, r_2 \in \mathbb{Z}/|G|\mathbb{Z}$ and suppose that $r_1 a + r_2 b$ is decomposed element, resp. almost decomposed element, resp. 2-almost decomposed element. Also let $\{g_i\}_{i=1}^N$ be the decomposed factor of $r_1 a + r_2 b$ (i.e., $r_1 a + r_2 b = g_1 + g_2 + ... + g_N$, $g_i \in B_0$) Put $\mathbf{DEC}(r_1, r_2) = (g, \overrightarrow{v}, \overrightarrow{r})$ where $g = r_1 a + r_2 b$, $v_b = \begin{cases} 1, b \in \{g_i\} \\ 0, \text{otherwise} \end{cases}$ and $\overrightarrow{r} = (r_1, r_2)$. So, we easily have that $\mathbf{DEC}(r_1, r_2)$ is an element of $\mathcal{U}'_0$, resp. $\mathcal{U}'_1$, resp. $\mathcal{U}'_2$. Further, we will make an algebraic structure of $\mathcal{U}$.

**Definition 5 (support)** *For any* $u \in \mathcal{U}_2$, *put* $\sup(u) := \{b \in B_0 \backslash B |\, v_{u,b} \neq 0\}$.
*For any* $u \in \mathcal{U}_1$, *put* $\sup(u) := \{b \in B_0 \backslash B |\, v_{u,b} \neq 0\} \cup \{\infty\}$.

Note that for any $u \in \mathcal{U}_1 \cup \mathcal{U}_2$, $\sup(u)$ is a subset of $B_0 \backslash B \cup \{\infty\}$ consists of 2 elements.

**Definition 6 (Linear Sum)** *Let* $\{t_i \in \mathcal{U} | i = 1, 2, ..., n\}$ *be a list of Universe and let* $\{\lambda_i \in \mathbb{Z}/|G|\mathbb{Z} \,|\, i = 1, 2, ..., n\}$ *be a list of number. Linear sum of* $\{t_i\}$ *by* $\{\lambda_i\}$, *which is written by* $\sum_{i=1}^n \lambda_i t_i$, *is the element of* $\mathcal{U}$ *satisfying the following: 1)* $g_{\sum_{i=1}^n \lambda_i t_i} = \sum_{i=1}^n \lambda_i g_{t_i}$, *2)* $\overrightarrow{v}_{\sum_{i=1}^n \lambda_i t_i} = \sum_{i=1}^n \lambda_i \overrightarrow{v}_{t_i}$, *3)* $\overrightarrow{r}_{\sum_{i=1}^n \lambda_i t_i} = \sum_{i=1}^n \lambda_i \overrightarrow{r}_{t_i}$.

By using this definition, we will define the elimination of large prime and factor base.

**Definition 7 (Elimination of Large Prime)** *Let* $\{u_i \in \mathcal{U}'_1 \cup \mathcal{U}'_2 |\, i = 1, 2, .., N_2\}$ *be a list of the elements in* $\mathcal{U}'_1 \cup \mathcal{U}'_2$ *and let* $M'$ *be a positive integer such that* $M' \geq |B| + 1$. *Elimination of large primes from* $\{u_i\}$ *is a computation of a list of linear sum* $\{w_j \in \mathcal{U}_0 | j = 1, 2, ..., M''\}$ *satisfying the following:*
*1)* $M'' \geq M'$, *2)* $w_j \in \mathcal{U}_0$ *is written by* $w_j = \sum_{i=1}^{N_2} \alpha_{i,j} u_i, \quad (\alpha_{i,j} \in \mathbb{Z}/|G|\mathbb{Z})$,
*3)For any* $j$, *there is some* $D(j)$ *such that* $\alpha_{D(j),j} \neq 0, \alpha_{D(j),j'} = 0$ *for any* $j' > j$, *4)For any* $j$, $\#\{i \,|\, \alpha_{i,j} \neq 0\} \leq O(\log N_2)$.

In section **6**, we will show the following claim is directly from Theorem 1.

**Claim 2** $\{w_j\}$ *and* $\{u_{D(j)}\}$ *in Definition 7 are computable with* $\tilde{O}(N_2)$ *complexity under the assumption* $N_2 > 4|B_0 \backslash B| + 4 + M'$.

**Definition 8 (Elimination of Factor Base)** *Let* $\{w_j \in \mathcal{U}_0 | j = 1, 2, ..., M'\}$ *be a list of* $\mathcal{U}_0$. *Elimination of factor base from* $\{w_j\}$ *is a computation of a linear sum* $u_o = \sum_{j=1}^{M'} \beta_j w_j \in \mathcal{U}_0, \qquad (\beta_j \in \mathbb{Z}/|G|\mathbb{Z})$ *satisfying the following:*
*1)* $\overrightarrow{v}_{u_o} = \overrightarrow{0}$, *(from this,* $g_{u_o} = 0$ *is reduced),*
*2) There is some* $J$ *such that* $\beta_J \neq 0$, $\beta_j = 0$ *for any* $j < J$.

We see easily that the computation of the elimination of factor base can be done by linear algebra of the size around $|B| \times |B|$ matrix. Assume that $\{w_j\}$ is obtained from elimination of large prime. So, from the property 3) of Definition 7, the linear algebra, which is used to eliminate the factor base, is sparse and its complexity is $\tilde{O}(|B|^2)$. Put $s_j := \sum_{j=1}^{M'} \beta_j \alpha_{i,j}$. So, we see easily $u_o = \sum_{i=1}^{N_2} s_i u_i$ and $s_{D(J)} = \sum_{j=1}^{M'} \beta_j \alpha_{D(J),j} = \beta_J \alpha_{D(J),J} \neq 0$. From the definition, $u_o = (0, \overrightarrow{0}, \overrightarrow{r}_{u_o})$ where $\overrightarrow{r}_{u_o} = \sum_{i=1}^{N_2} s_i \overrightarrow{r}_{u_i}$. So, remark that if $u_i \in \mathcal{U}_1' \cup \mathcal{U}_2'$ are selected randomly, the probability of $\overrightarrow{r}_{u_o} = \overrightarrow{0}$ is $\frac{1}{|G|}$ and it is negligible. The following algorithm 1 is an algorithm solving DLP of finite group.

**Algorithm 1** Solving DLP of finite group

**Input:** $a, b \in G$ s.t. $a = nb$ for some unknown $n \in \mathbb{Z}/|G|\mathbb{Z}$. **Output:** find $n$.
1: Collecting 2-almost and almost decomposed elements
2: $U_1 \leftarrow \{\}$, $U_2 \leftarrow \{\}$
3: **while** $|U_1 \cup U_2| <$ suitable number $N_2$ **do**
4: $(r_1, r_2) \leftarrow$ random pair of $\mathbb{Z}/|G|\mathbb{Z}$,.
5: **IF** $r_1 a + r_2 b$ being 2-almost decomposed **Then** $U_2 \leftarrow U_2 \cup \{\mathbf{DEC}(r_1, r_2)\}$
6: **IF** $r_1 a + r_2 b$ being almost decomposed **Then** $U_1 \leftarrow U_1 \cup \{\mathbf{DEC}(r_1, r_2)\}$
7: Put $\{u_i | \, i = 1, 2, ..., N_2\} = U_1 \cup U_2$, $M' = |B| + 1$
8: Elimination of large prime Compute $\{w_j = \sum_{i=1}^{N_2} \alpha_{i,j} u_i \in \mathcal{U}_0 | \, j = 1, 2, .., M'\}$
9: Elimination of factor base (by solving sparse linear algebra)
10: Compute $u_o = \sum_{j=1}^{M'} \beta_j w_j = (0, \overrightarrow{0}, \overrightarrow{r}_{u_o})$
11: $(r_1, r_2) \leftarrow \overrightarrow{r}_{u_o}$, return $-r_2/r_1 \bmod |G|$

Note that the set of factor bases $B$ is usually unconditional subset of $B_0$ and its size $|B|$ can be coordinated.

**Theorem 3.** *The complexity of solving DLP by Algorithm 1 is minimized at* $|B| \approx |B_0|^{(N-1)/N}$*, and it is estimated by* $\tilde{O}(|B_0|^{(2N-2)/N})$.

*Proof.* From Assumption 1, the probability that for randomly chosen $r_1, r_2 \in \mathbb{Z}/|G|\mathbb{Z}$, $\mathbf{DEC}(r_1, r_2)$ is in $\mathcal{U}_1 \cup \mathcal{U}_2$ is $O(|B/B_0|^{N-2}) + O(|B/B_0|^{N-1}) = O(|B/B_0|^{N-2})$. From Claim 2, we must collect $N_2 = O(|B_0|)$ number of $\mathbf{DEC}(r_1, r_2) \in \mathcal{U}_1 \cup \mathcal{U}_2$. Then we see that the cost for collecting $U_1 \cup U_2$ is $\tilde{O}(|B_0/B|^{N-2} \cdot |B_0|)$. From this Claim, we also have the cost of the elimination of large prime is $\tilde{O}(|B_0|)$ and it is smaller than that of collecting $U_1 \cup U_2$ part. The cost of the elimination of factor base is $\tilde{O}(|B|^2)$, since in this computation, sparse linear algebra of the size around $|B| \times |B|$ is used. So, rebalancing the cost of collecting $U_1 \cup U_2$ part and elimination of factor base part, we have $|B_0/B|^{N-2} \cdot |B_0| \approx |B|^2$. Thus, $|B| \approx |B_0|^{(N-1)/N}$ is obtained and we have desired result.

## 4 Graph and Global variables of the System

Let $U_1 (\subset \mathcal{U}_1')$ and $U_2 (\subset \mathcal{U}_2')$ be the sets of Universe, which are collected in the former part of Algorithm 1. Further, we fix $U_1$, $U_2$ and exceed the discussion. For the purpose of the elimination of large prime, we consider the graph $\mathbf{Graph} = (\mathbf{Edge}, \mathbf{Vertex})$ such that
1) **Edge** is a subset of $U_1 \cup U_2$,
2) **Vertex** is the subset of $B_0 \backslash B \cup \{\infty\}$, which appears in $\sup(u)$ for some

$u \in$ **Edge**, (i.e., **Vertex** $= \cup_{u \in \mathbf{Edge}} \sup(u)$),
3) Edge $u \in$ **Edge** contains 2 vertexes $\{b_1, b_2\} = \sup(u)$.

**Definition 9 (Citation)** *For any* $b \in$ **Vertex**, *put* $cite(b) := \{u \in \mathbf{Edge} | b \in \sup(u)\}$.

Note that cite($b$) is the set of edges which contain vertex $b$. At first, the graph is initialized by **Edge** $\leftarrow U_1 \cup U_2$, **Vertex** $\leftarrow \cup_{u \in U_1 \cup U_2} \sup(u)$ and latter, some edges and therefore the vertexes, which have no edge, are removed from graph.

In the computation of the elimination of large prime, we use some global variables of the system.

**Definition 10 (List of the Global Variables of the Graph)**
*Table* **SUP** $:= \{\sup(u) | \, u \in U_1 \cup U_2\}$, *Table* **CITE** $:= \{cite(b) | \, b \in B \backslash B \cup \{\infty\}\}$
*Set* **Edge**$(\subset U_1 \cup U_2)$ *and its cardinality* $|\mathbf{Edge}|$
*Set* $V_i := \{b \in B \backslash B \cup \{\infty\} | \, \#cite(b) = i\}$ *and its cardinality* $|V_i|$

In these variables, vertexes of the graph is considered as **Vertex** $= \cup_{i \geq 1} V_i$.

In the whole computation, we seek the loop of **Graph**, which start form some base point $b$. In order to preserve the number of discovered loop, the global variable **Counter** is prepared. The value of **Counter** is initialized by 0. If a loop $l$ is discovered, in order not to re-found the same loop, some one edge $u_l$, which appears only once in $l$, is deleted from **Graph** and the value of the **Counter** is incremented. Suppose $c_l$ is the value of the **Counter**, when the edge $l$ is found. The 3-ple $ex(l) := (l, u_l, c_l)$ is called the extended data of found loop $l$. In the computation, we collect the even length loops. By this mean, we prepare the global variable **LP**, which is the list of the collection of the extended data of even length loops. **LP** is initialized by [] and if an even length loop is found, its extended data is inserted to **LP**. Suppose that two odd length loops $l_1$,$l_2$, which start from the same base point, is found. The conjunction of $l_1$,$l_2$, which is simply written by $l_1 + l_2$, is an even length loop. Put $c_{l_1+l_2} := \min(c_{l_1}, c_{l_2})$,
$u_{l_1+l_2} := \begin{cases} u_{l_1} & \text{if } c_{l_1} < c_{l_2} \\ u_{l_2} & \text{if } c_{l_1} > c_{l_2} \end{cases}$, and $ex(l_1 + l_2) := (l_1 + l_2, u_{l_1+l_2}, c_{l_1+l_2})$,

From the notations of extended data, we have the following two lemmas.

**Lemma 1.** *Let* $l_1$,$l_2$ *be found odd length loops start from the same point.*
*i)* $c_{l_1} = c_{l_1+l_2}$ *if* $c_{l_1} < c_{l_2}$,
*ii) The edge* $u_{l_1+l_2}$ *appears only once in the loop* $l_1 + l_2$.

**Lemma 2.** *Let* $l_3$,$l_4$ *be even length loops, which are 1) found even length loop(s) or 2) even length loop(s) obtained by conjunction(s) of two odd length loops. Suppose* $c_{l_3} < c_{l_4}$. *Then,* $u_{l_3}$ *appears only once in the loop* $l_3$ *and does not appears in* $l_4$.

We prepare the global variable **ROL**($b$), which is indexed by $b \in B_0 \backslash B \cup \{\infty\}$. **ROL**($b$) is initialized by $\emptyset$. If an odd loop $l$, which starts from $b$, is first found, **ROL**($b$) is updated by $ex(l)$. Moreover, suppose **ROL**($b$) $\neq \emptyset$ and let $(l_{old}, u_{old}, c_{old})$ be the stored data in **ROL**($b$). Thus, we have an even length loop $l_{old} + l$. And so, the extended data $ex(l_{old} + l)$ is inserted to **LP** and **ROL**($b$) is also updated by $ex(l)$.

**Definition 11 (List of the Global Variables of the loop)**
*List* **LP** *list of the collection of 3-ple (extended data of even loop)*
*List* **ROL** $:= \{\mathbf{ROL}(b) | b \in B \backslash B \cup \{\infty\}\}$ *list of recent odd loop start from* $b$.

*Counter* **Counter** *The number of found loops (Not include the number of even loop obtained by conjunction)*

In the computation, we collect the even length loops by the policy of Algorithm 2.

**Algorithm 2** Policy of the Collection of Even length Loops

Global Variable **Counter**,**LP**,**ROL**($b$) are used
**Counter** $\leftarrow 0$, **LP** $\leftarrow []$
**for all** $b \in B_0 \backslash B \cup \{\infty\}$ **do** Put **ROL**($b$) $\leftarrow \emptyset$
**while** Some Condition **do**
  Seeking the loop of **Graph**
  **if** Even length loop $l$ is found **then**
    select one edge $u_l$ which appears once in $l$, delete $u_l$ from **Graph**, **Counter** $++$
    insert $(l, u_l,$ **Counter**$)$ to **LP** sorted by 3rd coordinate
  **if** Odd length loop $l$ start from $b$ is found **then**
    **if ROL**($b$) $\neq \emptyset$ **then**
      $(l_{old}, u_{old}, c_{old}) \leftarrow$ **ROL**($b$)
      insert $(l + l_{old}, u_{old}, c_{old})$ to **LP** sorted by 3rd coordinate
    select one edge $u_l$ which appears once in $l$, delete $u_l$ from **Graph**, **Counter** $++$
    **ROL**($b$) $\leftarrow (l, u_l,$ **Counter**$)$

**Lemma 3.** *Let* $e_1 = (l_1, u_{l_1}, c_{l_1})$, $e_2 = (l_2, u_{l_2}, c_{l_2})$ $(e_1 \neq e_2)$ *be two extended data in* **LP** *obtained from Algorithm 2. Then,* $c_{l_1} \neq c_{l_2}$.

*Proof.* When one of the $\{l_1, l_2\}$ is a found even length loop, it is trivial. Suppose $l_1$, $l_2$ are the even loops obtained by conjunction. Note that from the construction of Algorithm 2, $c_{l_1}$ is the value of **Counter** of **ROL**($b$) for some $b$ and when the conjunction loop $l_1$ in made, the value of **Counter** of **ROL**($b$) is updated. So, it is never used further.

**Lemma 4.** *Let LP be the list obtained from Algorithm 2. Put* $M'' := |\mathbf{LP}|$ *and* $\{(l_j, d_j, c_j) | \, j = 1, 2, 3..., M''\} := \mathbf{LP}$. *Then,* $d_j$ *appears only once in* $l_j$ *and does not appear* $l_{j'}$ $(j' > j)$.

*Proof.* Remark that the list $\{(l_j, d_j, c_j)\}$ is sorted by the value of $c_j$. From Lemma 2 and Lemma 3, we have desired result.

Initialization of global valiables is given by Algorithm 3.

**Algorithm 3** Initialization of Global variables of the Sysyem

**Input:** $U_1$, $U_2$, **Output:** Global Variables of the System
**Edge** $\leftarrow U_1 \cup U_2$
**for all**$u \in U_1 \cup U_2$ **do** Compute sup($u$) and store in **SUP**
**for all** $b \in B_0 \backslash B \cup \{\infty\}$ **do** Put cite($b$) $\leftarrow \{\}$
**for all** $u \in U_1 \cup U_2$ **do** $(b_1, b_2) \leftarrow$ sup($u$), cite($b_1$) $\leftarrow$ cite($b_1$) $\cup \{u\}$, cite($b_2$) $\leftarrow$ cite($b_2$) $\cup \{u\}$
**for all** $V_i$, $(i \geq 0)$ **do** Put $V_i \leftarrow \{\}$, $|V_i| \leftarrow 0$
**for all** $b \in B_0 \backslash B \cup \{\infty\}$ **do** $i \leftarrow \#$cite($b$), $V_i \leftarrow V_i \cup \{b\}$, $|V_i| ++$
**LP** $\leftarrow []$, **Counter** $\leftarrow 0$
**for all** $b \in B_0 \backslash B \cup \{\infty\}$ **do ROL**($b$) $\leftarrow \emptyset$

**Lemma 5.** *The complexity of initializing global variables of the system by Algorithm 3 is* $\tilde{O}(|U_1 \cup U_2|)$.

## 5 Operation of Graph

Here, we prepare the two operations of graph **Graph** = (**Edge**, **Vertex**). First one is the operation deleting edge $u_0$ from **Graph**. This operation is done by the algorithm 4. From Algorithm 4, we easily have the following lemma.

**Algorithm 4** Deleting edge $u_0$ from graph

**Input:** $u_0$, **Output** Update Global Variables of the graph
1: **Edge** $\leftarrow$ **Edge**$\backslash\{u_0\}$, $(b_1, b_2) \leftarrow \sup(u_0)$
2: **for** $i = 1, 2$ **do**
3: $n \leftarrow \#\text{cite}(b_i)$, $V_n \leftarrow V_n \backslash \{b_i\}$, $|V_n| - -$, $V_{n-1} \leftarrow V_{n-1} \cup \{b_i\}$, $|V_{n-1}| + +$

**Lemma 6.** *The complexity of deleting one edge from graph is* $\tilde{O}(1)$.

Second operation of graph is called $n$-gleton. Let $n$ be a small positive integer. $n$-gleton is the operation that delete the vertexes which have only less than or equals to $n$ edges(i.e. $\{b \in \textbf{Vertex} | \#\text{cite}(b) \leq n\}$) and delete the edges which contains deleted vertexes and continue these operations recursively. So after this operation, any vertex $b$ have more than $n + 1$ edges (i.e.,$\#\text{cite}(b) \geq n + 1$ for $\forall b \in \textbf{Vertex}$). $n$-gleton is done by the Algorithm 5. In Algorithm 5, delete

**Algorithm 5** $n$-gleton

**Input:** $n$, **Output** Update Global Variables of the graph
1: **while** $b \in \cup_{i=1}^{n} V_i$ **do**
2: $C \leftarrow \text{cite}(b)$
3: **for all** $u \in C$ **do**
4: delete $u$ from graph (note that $V_i$'s are updated and recursive deleting is done)

of the vertexes is automatically done in the sub-deleting edge operation. From Algorithm 5, we easily have the following lemma.

**Lemma 7.** *i) The complexity of* $n$*-gleton is* $\tilde{O}$(*number of deleted edges*).
*ii) The value* $|\textbf{Edge}| - n|\textbf{Vertex}|$ *is unchanged or increasing after* $n$*-gleton operation.*

## 6 Chain and Loop of Graph

In this section, we define the chain and loop of the **Graph** = (**Edge**, **Vertex**).

**Definition 12 (Chain)** *Chain is a sequence of Edges* $c = u_1 u_2 ... u_n$ ($u_i \in$ **Edge**) *such that there are some vertexes* $b_0, b_1, ..., b_n \in$ **Vertex** *satisfying* $\sup(u_i) = \{b_{i-1}, b_i\}$. *Moreover, the length of the chain* $l_c(c)$ *is defined by the length of the sequence (i.e.,* $l_c(c) = n$*). The vertexes* $b_0$ *(resp.* $b_n$*) is called start point (resp. end point) of chain* $C$ *and they are written by* $start(c) = b_0, end(c) = b_n$.
*For the generality, the* 0*-chain, whose length is* 0*, is also considered.*

Note that the start point and end point are not unique (however, the pair of the start point and end point exists from the definition). Also note that each edge $u \in$ **Edge** is considered as a chain of length 1, whose $\{start(u), end(u)\} = \sup(u)$. For the generality, put $start(0\text{-chain}) = end(0\text{-chain}) = b$ for $\forall b \in$ **Vertex**.

**Definition 13 (Arithmetic Operation of Chain)** *For a chain* $c = u_1u_2...u_{n-1}u_n$, *put its reversal by* $rev(c) := u_nu_{n-1}...u_2u_1$. *For two chains* $c_1 = u_1u_2...u_n$ *and* $c_2 = u_{n+1}u_{n+2}...u_m$ *such that* $end(c_1) = start(c_2)$, *put there conjunction by* $c_1 + c_2 := u_1u_2..u_nu_{n+1}..u_m$.

Note that we easily have $start(rev(c)) = end(c)$, $end(rev(c)) = start(c)$, $start(c1+c_2) = start(c_1)$, and $end(c_1 + c_2) = end(c_2)$ form the definition. For any chain $c$, (0-chain) $+ c$, $c +$ (0-chain) are considered by $c$.

**Definition 14 (Loop)** *Loop is a chain* $l = u_1u_2...u_n$ *such that its start point and end point can be taken the same vertex (i.e.,* $start(l) = end(l)$*).*

**Definition 15 (Large Prime Elimination via Even Length Loop)** *Let* $l = u_1u_2..., u_{2n}$ *be an even length loop. Put*

$$Eli(l) := \sum_{i=1}^{l_c(l)} u_i \times (-1)^{i-1} = (\sum_{i=1}^{l_c(l)}(-1)^{i-1}g_{u_i}, \sum_{i=1}^{l_c(l)}(-1)^{i-1}\overrightarrow{v}_{u_i}, \sum_{i=1}^{l_c(l)}(-1)^{i-1}\overrightarrow{r}_{u_i})$$

**Lemma 8.** *For an even length loop* $l$, $Eli(l) \in \mathcal{U}_0$.

*Proof.* Let $l = u_1u_2..., u_{2n}$ and let $b_0, b_1.., b_{2n}$ be the vertexes such that $\sup(u_i) = \{b_{i-1}, b_i\}$. Since $l$ is a loop, $b_0 = b_{2n}$ holds. For vectors $\overrightarrow{v}_1, \overrightarrow{v}_2 \in \mathbb{A}^{|B_0|}(\mathbb{Z}/|G|\mathbb{Z})$, the equivalent relation $\equiv$ is defined by $\overrightarrow{v_1} \equiv \overrightarrow{v_2} \Leftrightarrow \overrightarrow{v_1} - \overrightarrow{v_2} \in \mathcal{U}_0$. For any $b' \in$ **Vertex**, put $\overrightarrow{V}_{b'} \in \mathbb{A}^{|B_0|}(\mathbb{Z}/|G|\mathbb{Z})$ by $V_{b',b'} = 1, V_{b',b} = 0$ for any $b \neq b'$. So we have, $\overrightarrow{v}_{u_i} \equiv \overrightarrow{V}_{b_{i-1}} + \overrightarrow{V}_{b_i}$ and $\overrightarrow{v}_{Eli(l)} \equiv (\overrightarrow{V}_{b_0} + \overrightarrow{V}_{b_1}) - (\overrightarrow{V}_{b_1} + \overrightarrow{V}_{b_2}) + ... - (\overrightarrow{V}_{b_{2n-1}} + \overrightarrow{V}_{b_{2n}}) = \overrightarrow{V}_{b_0} - \overrightarrow{V}_{b_{2n}} = \overrightarrow{0}$. So, we have $Eli(l) \in \mathcal{U}_0$.

Here, we show that Theorem 1 induces Claim 2 and then Theorem 3. Assume that Theorem 1 is true. We apply Theorem 1 to the **Graph** $=$ (**Edge**, **Vertex**) whose **Edge** is $\{u_i | i = 1, 2, ..., N_2\}$ and whose **Vertex** is $\cup_{i=1}^{N_2} \sup(u_i) \subset B_0 \setminus B \cup \{\infty\}$. Note that $N_2 =$ |**Edge**| and |**Vertex**| $\leq |B_0 \setminus B| + 1$. On the other hands, $N_2 > 4|B_0 \setminus B| + 4 + M'$ holds from assumption of Claim 2. So, we have |**Edge**| $- 4|$**Vertex**$| > M'$, which is the assumption of Theorem 1. Suppose that $\{l_j | j = 1, 2, ..., M''\}$ and $\{d_j | j = 1, 2, ..., M''\}$ are the lists which are obtained from Theorem 1. Also put $w_j := Eli(l_j)$ $(j = 1, 2, .., M'')$. Here, we must show that the list $\{w_i\}$ and $\{d_j\}$ satisfy the properties of Definition 7. From lemma 8, we have $w_j \in \mathcal{U}_0$. From its construction, $w_j$ is written by the form $w_j = \sum_{i=1}^{N_2} \alpha_{i,j}u_i$. For any $j$, the value $\#\{i | \alpha_{i,j} \neq 0\}$ is the same or smaller than $N \times l_c(l_j) < 4N + 4N \log_2 |N_2| = O(\log |N_2|)$. Also put $D(j)$ by $u_{D(j)} = d_j$, and $D(j)$ satisfied the property iii)of Defintion 7, since $\{l_j\}$ and $\{d_j\}$ satisfy the property iii) of Theorem 1. So we have Claim 2 and therefore have Theorem 3, which comes from Claim 2. So, further in this paper, we will prove Theorem 1.

## 7 Computing $L_n$

**Definition 16 (Distance of vertexes)** *Let* $b, b'$ *be vertexes of the* **Graph** $=$ (**Edge**, **Vertex**). *Suppose that*
*1) There is some chain* $c$ *such that* $\{start(c), end(c)\} = \{b, b'\}$, $l_c(c) = n$, *and*
*2) There is no chain* $c$ *such that* $\{start(c), end(c)\} = \{b, b'\}$, $l_c(c) = n - 1$.
*Then we say the distance of* $b$ *and* $b'$ *is* $n$ *and it is denoted by* $dist(b, b') = n$.

Further, we fix the base point $b_0 \in \mathbf{Vertex}$.

**Definition 17** *Put* $L_n := \{b \in \mathbf{Vertex} | \, dist(b, b_0) = n\} \quad (n = 1, 2, 3, .., )$. *For the generality, also put* $L_0 := \{b_0\}$.

Let $b \in L_n$. From the definition of the distance, there is some (generally not unique) chain $c_b$ such that $start(c_b) = b_0, end(c_b) = b$, $l_c(c_b) = n$. Such $c_b$ is called associate chain of $b \in L_n$.

**Definition 18** *Put* $\tilde{L}_n := \{(b, c_b) | \, b \in L_n\} \quad (n = 1, 2, 3, ...)$. *For the generality, also put* $\tilde{L}_0 := \{(b_0, 0\text{-}chain)\}$, *where* 0*-chain is the chain of length* 0.

We will compute $L_n$ and $\tilde{L}_n$ $(n \geq 1)$ under the assumption that $L_i, \tilde{L}_i$ $(i = 0, 1, 2, .., n-1)$ are already computed. Suppose $b \in L_{n-1}$ and $u \in \mathrm{cite}(b)$. Put $\{b'\} := sup(u) \backslash \{b\}$. From the definition of $L_i$, $b' \in \cup_{i=\max(0,n-2)}^{n} L_i$. So suppose $b' \notin \cup_{i=\max(0,n-2)}^{n-1} L_i$, we easily have $dist(b_0, b') = n$ and $c_{b'} = c_b + u$. Thus, we have $L_n = \cup_{b \in L_{n-1}} \cup_{u \in \mathrm{cite}(b)} \sup(u) \setminus \cup_{i=\max(0,n-2)}^{n-1} L_i$. Further, in the process of the computation of $L_n$, we seek and collect the loops of the graph. Note that the global variables **LP**,**Counter**,**ROL**$(b)$ are used for the collection of the loops.
**case 1(odd length loop)** (cf. Fig 2) Let $b, b' (\neq b) \in L_{n-1}$. Suppose that there

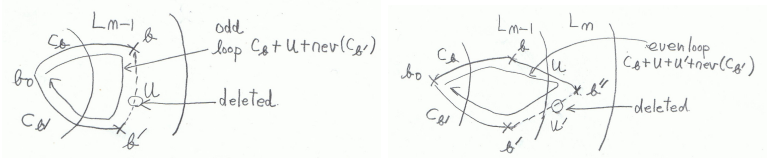


**Fig. 2.** Odd length loop(left) and Even length loop(right)

is some $u \in \mathrm{cite}(b) \cap \mathrm{cite}(b')$. Then we see $\sup(u) = \{b, b'\}$ and there is an odd length $(2n-1)$ loop $c_b + u + rev(c_{b'})$. In Algorithm 6, we collect such odd length loop in the list $\mathbf{OL}_n$ (it means Odd Loop) and delete the edge $u$ from the graph in order not to collect the same loop. Note that the vertexes of the graph is unchanged under the operation of deleting $u$, since $u$ is a edge of some loop. Also note that the distance from the base point is unchanged under the operation of deleting $u$.

Suppose that two odd length loops $l_1, l_2$ with the same base point are found, one has even length loop $l_1 + l_2$. By this mean, when an odd length loop $l_1$ is found, the 3-ple of the data $(l_1, d_1, c_1)$, where $l_1$ is the data of odd length loop, $d_1$ is the data of deleted edge, and $c_1$ is the value of counter when the edge is deleted, is substituted to the global variable $\mathbf{ROL}(b_0)$. Latter, when another odd length loop $l_2$ whose 3-ple is $(l_2, d_2, c_2)$ is found, $(\mathbf{ROL}(b_0)[1] + l_2, \mathbf{ROL}(b_0)[2], \mathbf{ROL}(b_0)[3])$ is appended to the list **LP** and $\mathbf{ROL}(b_0)$ is updated by $(l_2, d_2, c_2)$.

**case 2(even length loop)** (cf. Fig 2) Let $b, b' (\neq b) \in L_{n-1}$. Suppose that there are some $u \in \mathrm{cite}(b)$, $u' \in \mathrm{cite}(b')$, $b'' \in L_n$ such that $\sup(u) = \{b, b''\}$, $\sup(u') = \{b', b''\}$. Then there is an even length $(2n)$ loop $c_b + u + u' + rev(c_{b'})$.

In the Algorithm 6, we collect such even length loop in the list $\mathbf{EL}_n$ (it means Even Loop) and delete the edge $u'$, which is discovered latter, from the graph. Also the 3-ple$(c_b + u + u' + rev(c_{b'}), u'$, the value of **Counter**) is inserted to the list **LP**. Similarly, note that the set of vertexes and the distance from the base point is unchanged under the operation of deleting $u'$.

The following Algorithm 6 computes $L_n$, $\tilde{L}_n$, collects loops and updates the global variables of the system.

**Algorithm 6** Computing $L_n$

**Input:** $b_0$,$n$,$L_i, \tilde{L}_i$ $(i = 0, 1, 2, ..., n-1)$
**Output:** $L_n, \tilde{L}_n$, $\mathbf{OL}_n$, $\mathbf{EL}_n$, Update Global Variables of the system
$\mathbf{OL}_n \leftarrow \{\}$, $\mathbf{EL}_n \leftarrow \{\}$, $L_n \leftarrow \{\}$, $\tilde{L}_n \leftarrow \{\}$
**for all** $(b, c_b) \in \tilde{L}_{n-1}$ **do**
  **for all** $u \in \mathrm{cite}(b)$ **do**
    $\{b'\} \leftarrow \sup(u) \backslash \{b\}$
    **if** $b' \in L_{n-1}$ **then**
      // Odd loop is found
      $\mathbf{OL}_n \leftarrow \mathbf{OL}_n \cup \{c_b + u + rev(c_{b'})\}$, delete $u$ from Graph,**Counter** + +
      **if** $\mathbf{ROL}(b_0) = \emptyset$ **then**
        $\mathbf{ROL}(b_0) \leftarrow (c_b + u + rev(c_{b'}), u, \mathbf{Counter})$
      **else**
        $(l', u', c') \leftarrow \mathbf{ROL}(b_0)$
        insert $(l' + c_b + u + rev(c_{b'}), u', c')$ to **LP** sorted by 3rd coordinate
        $\mathbf{ROL}(b_0) \leftarrow (c_b + u + rev(c_{b'}), u, \mathbf{Counter})$
    **if** $b' \notin \cup_{i=\max(0,n-2)}^{n-1} L_i$ **then**
      **if** $b' \notin L_n$ **then**
        // New vertex in $L_n$ is found
        $L_n \leftarrow L_n \cup \{b'\}$, $\tilde{L}_n \leftarrow \tilde{L}_n \cup \{(b', c_b + u)\}$
      **else**
        // Even loop is found
        $\mathbf{EL}_n \leftarrow \mathbf{EL}_n \cup \{c_{b'} + u + rev(c_b)\}$, delete $u$ from Graph,**Counter** + +
        insert $(c_{b'} + u + rev(C_b), u, \mathbf{Counter})$ to **LP** sorted by 3rd coordinate

From the construction of Algorithm 6, we easily have the following lemma.

**Lemma 9.** *i) The number of the decrease of the edges of the graph in Algorithm 6 equals to* $|OL_n| + |EL_n|$.
*ii) No vertex is removed by the operation of Algorithm 6,*
*iii) The complexity of Algorithm 6 is* $\tilde{O}(|OL_n| + |EL_n| + |L_n|)$,
*iv) Let* $b \in$ **Vertex**. *The distance* $dist(b, b_0)$ *is unchanged by the operation of Algorithm 6.*

## 8 Proposed Algorithm and Proof of Theorem1

In this section, we propose an improved algorithm(Algorithm 7, 8, 9), which does not need the randomness of the graph and its worst complexity can be estimated. Let $U_1(\subset \mathcal{U}_1')$, $U_2(\subset \mathcal{U}_2')$ be the sets of Universe and let $M'(\geq |B|+1)$ be a positive integer. Suppose $|U_1 \cup U_2| \geq 4|B_0 \backslash B| + 4 + M'$ and global variables of the **Graph** is initialized by $U_1 \cup U_2$. Put $M = M' + |B_0 \backslash B| + 1$. Sub algorithm 1 (Algorithm 8) returns at least $M$ number loops which are collected in $\mathbf{OL} \cup \mathbf{EL}$ and at least $M'$ number of even length loops which are collected in the list **LP**. Thus, main algorithm (Algorithm7) returns a set $U_0(\subset \mathcal{U}_0)$ such that $|U_0| \geq M'$.

**Algorithm 7** Proposed algorithm(main)

**Input:** $U_1 \cup U_2$, $M'$ such that $|U_1 \cup U_2| \geq 4|B_0 \backslash B| + 4 + M'$
**Output:** $U_0 (\subset \mathcal{U}_0)$ such that $|U_0| \geq M'$
1: Init system by $U_1 \cup U_2$ (Not only the variables of **Graph**,**LP**,**RE**,**ROL** are init.)
2: **call** sub algorithm 1 (Algorithm 8)
3: $U_0 \leftarrow []$
4: **for** $j = 1, 2, ..., |\mathbf{LP}|$ **do** $(l_j, d_j, c_j) \leftarrow LP[j]$, append $Eli(i_j)$ to $U_0$

**Algorithm 8** Proposed algorithm(sub algorithm 1)

**Input:** $M'$, **Output: OL**,**EL**, Update global variables of the system
1: Assume $|\mathbf{Edge}| - 4|\mathbf{Vertex}| > M'$
2: $\mathbf{OL} \leftarrow \{\}$, $\mathbf{EL} \leftarrow \{\}$, $M \leftarrow M' + |B_0 \backslash B| + 1$
3: **while** $|\mathbf{OL}| + |\mathbf{EL}| < M$ **do**
4: **call** 3-gleton algorithm(Algorithm 5, $n = 3$)
5: **IF Graph** $= \emptyset$ break (Remark that this does not happen from Lemma 14)
6: **call** sub algorithm 2 (Algorithm 9,Update **OL**, **EL** and global variables)

In sub algorithm 1(Algorithm 8) and sub algorithm 2(Algorithm 9), **OL** is the set of the collection of odd length loops and **EL** is the set of the collection of even length loops, whose loops are obtained in the sub computing $L_n$ algorithm(Algorithm 6). In sub algorithm 1(Algorithm 8), first, **Graph** is changed by 3-gleton and next the operation of sub algorithm 2(Algorithm 9) is done. In sub algorithm 2(Algorithm 9), starting from random base point $b_0$, $L_1, L_2, ..., L_n$ are computed. In this process, loops, which have start point $b_0$, are collected. After the process of sub algorithm 2(Algorithm 9) finishes, process returns to sub algorithm 1(Algorithm 8) and also the operation of 3-gleton and sub algorithm 2 are done again until system collect enough number of loops. First we prove some lemmas associated with sub algorithm 2(Algorithm 9). Suppose that $L_1, L_2, ..., L_n$ are the sets of vertexes obtained in Algorithm 9.

**Lemma 10.** *i)* $|L_{n-1}| < \sum_{i=1}^{n} |L_i| < 4|L_{n-1}|$, *ii)* $n < 1 + \log_2 |\mathbf{Vertex}|$.

*Proof.* From inequalities $|L_i| \geq 2|L_{i-1}|\,(i \leq n-1)$ and $|L_n| < 2|L_{n-1}|$, we have the inequality i) and $2^{n-1} \leq |L_{n-1}| < |\mathbf{Vertex}|$.

**Lemma 11.** $|\mathbf{Vertex}|, |\mathbf{Edge}| + |\mathbf{OL}| + |\mathbf{EL}|$ *are unchanged after the operation of Algorithm 9.*

*Proof.* From Lemma 9 i) and ii), this lemma is directly obtained.

**Lemma 12.** *Let* $\triangle|\mathbf{OL}|$ *and* $\triangle|\mathbf{EL}|$ *be the increases of* $|\mathbf{OL}|$ *and* $|\mathbf{EL}|$, *after the operation of Algorithm 9.*
*i)* $\triangle|\mathbf{OL}| + \triangle|\mathbf{EL}| \geq 2$, *ii)The complexity of Algorithm 9 is* $\tilde{O}(\triangle|\mathbf{OL}| + \triangle|\mathbf{EL}|)$.

*Proof.* First, we remark that $\#\text{cite}(b) \geq 4$ for $\forall b \in \mathbf{Vertex}$, since 3-gleton operation is done just before the Algorithm 9. From Lemma 9 iii), the complexity of Algorithm 9 is $\tilde{O}(\sum_{i=1}^{n} |L_i| + \triangle|\mathbf{OL}| + \triangle|\mathbf{EL}|)$, since $\triangle|\mathbf{OL}| = \sum_{i=1}^{n} |\mathbf{OL}_i|$ and $\triangle|\mathbf{EL}| = \sum_{i=1}^{n} |\mathbf{EL}_i|$. So, in order to prove ii), it is sufficient to show $O(\triangle|\mathbf{OL}| + \triangle|\mathbf{EL}|) \geq O(|L_{n-1}|)$, since $O(|L_{n-1}|) = O(\sum_{i=1}^{n} |L_i|)$ from Lemma 10. Further we prove $O(\triangle|\mathbf{OL}| + \triangle|\mathbf{EL}|) \geq O(|L_{n-1}|)$ and $\triangle|\mathbf{OL}| + \triangle|\mathbf{EL}| \geq 2$ by case analysis.
**In case of** $n = 1$. There is a single vertex $b$ such that $L_1 = \{b\}$, since $|L_1| < 2|L_0| = 2$. So, for any $u \in \text{cite}(b_0)$, we have $\sup(u) = \{b_0, b\}$ and $\triangle|\mathbf{EL}| = \#\text{cite}(b_0) - 1 \geq 3$. Thus, we have $\triangle|\mathbf{OL}| + \triangle|\mathbf{EL}| \geq 3 > 2 > 1 = |L_0|$ and desired result.

**Algorithm 9** Proposed algorithm(sub algorithm 2)

**Input:** $M, M', \mathbf{OL}, \mathbf{EL}$, **Output:** $\mathbf{OL}$,$\mathbf{EL}$, Update global variables of the system
1: Assume $\#\mathrm{cite}(b) \geq 4$ for $\forall b \in \mathbf{Vertex}$ (It holds, since 3-gleton is operated)
2: $b_0 \leftarrow \mathrm{random}(\mathbf{Vertex})$, $n \leftarrow 0$, $L_0 \leftarrow \{b_0\}$, $\tilde{L}_0 \leftarrow \{(b_0, 0\text{-chain})\}$
3: **repeat**
4: $n++$
5: **call** computing $L_n$ algorithm (Algorithm 6).
6: (c.f. Algorithm 6 compute $\mathbf{OL}_n, \mathbf{EL}_n, L_n, \tilde{L}_n$ and update global variables)
7: $\mathbf{OL} \leftarrow \mathbf{OL} \cup \mathbf{OL}_n$, $\mathbf{EL} \leftarrow \mathbf{EL} \cup \mathbf{EL}_n$,
8: **until** $|L_n| < 2|L_{n-1}|$

**In case of** $n \geq 2$. Now, suppose **Graph** is the graph before the operation of Algorithm 9. Let $b \in L_{n-1}$ and $u \in \mathrm{cite}(b)$. Put $\{b'\} := \sup(u) \backslash \{b\}$. From the definition of distance, $b' \in L_{n-2} \cup L_{n-1} \cup L_n$. Put $\mathrm{cite}_2(b) := \{u \in \mathrm{cite}(b) |\ \sup(u) \backslash \{b\} \subset L_{n-2}\}$, $\mathrm{cite}_1(b) := \{u \in \mathrm{cite}(b) |\ \sup(u) \backslash \{b\} \subset L_{n-1}\}$, $\mathrm{cite}_0(b) := \{u \in \mathrm{cite}(b) |\ \sup(u) \backslash \{b\} \subset L_n\}$.
So, we have $\cup_{i=0}^{2} \mathrm{cite}_i(b) = \mathrm{cite}(b)$, $\sum_{i=0}^{2} \#\mathrm{cite}_i(b) = \#\mathrm{cite}(b) \geq 4$, and $\mathrm{cite}_2(b) \geq 1$, since $b \in L_{n-1}$ and there exists some edge which contains $b$ and some element in $L_{n-2}$. From the construction of the loop, we also have( cf Fig 3)
$|\mathbf{EL}_{n-1}| = \sum_{b \in L_{n-1}} (\mathrm{cite}_2(b) - 1)$, $|\mathbf{OL}_n| = \frac{1}{2} \sum_{b \in L_{n-1}} \mathrm{cite}_1(b)$,
$|\mathbf{EL}_n| = (\sum_{b \in L_{n-1}} \mathrm{cite}_0(b)) - |L_n|$. Thus we have
$\triangle|\mathbf{OL}| + \triangle|\mathbf{EL}| \geq |\mathbf{EL}_{n-1}| + |\mathbf{OL}_n| + |\mathbf{EL}_n| \geq \frac{1}{2}|\mathbf{EL}_{n-1}| + |\mathbf{OL}_n| + \frac{1}{2}|\mathbf{EL}_n|$
$> \frac{1}{2} \sum_{b \in L_{n-1}} \#\mathrm{cite}(b) - \frac{1}{2}|L_{n-1}| - \frac{1}{2} \cdot 2|L_{n-1}| = \frac{1}{2}|L_{n-1}| \geq 1$ and desired result.

Further, we show that the sub algorithm 1(Algorithm 8) satisfies the condition of Theorem 1.

**Lemma 13.** *In Algorithm 8, the inequation* $|\mathbf{Edge}| - 3|\mathbf{Vertex}| + |\mathbf{OL}| + |\mathbf{EL}| > M$ *always holds.*

*Proof.* Just after Algorithm 8 start, since $|\mathbf{Edge}| - 3|\mathbf{Vertex}| > M' + |\mathbf{Vertex}| = M$, $|\mathbf{OL}| = |\mathbf{EL}| = 0$, this inequation holds. From Lemma 7, 3-gleton operation increases the value $|\mathbf{Edge}| - 3|\mathbf{Vertex}|$ and so, after 3-gleton operation, this inequation also holds. On the other hands, from Lemma 11, the operation of Algorithm 9 keeps the value $|\mathbf{Vertex}|, |\mathbf{Edge}| + |\mathbf{OL}| + |\mathbf{EL}|$. So, this inequality also hold after the operation of Algorithm 9 and we have desired result.

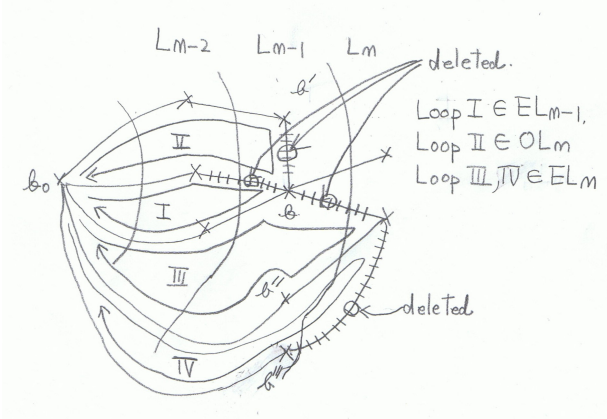


**Fig. 3.** Proof of Lemma 12

**Lemma 14.** *Empty graph (i.e.,the situation* $\mathbf{Graph} = \emptyset$*) does not appears in Algorithm 8.*

*Proof.* Suppose $\mathbf{Graph} = \emptyset$(i.e.,$|\mathbf{Vertex}| = |\mathbf{Edge}| = 0$). From Lemma 13, we have $|\mathbf{OL}| + |\mathbf{EL}| > M$. It is a contradiction, since $n$-gleton algorithm keeps the value $|\mathbf{OL}|, |\mathbf{EL}|$ and $|\mathbf{OL}| + |\mathbf{EL}| < M$ is loop condition. (Also note that sub algorithm 2 (Algorithm 9) does not returns empty graph, since it keeps $|\mathbf{Vertex}|$ form Lemma 11).

**Lemma 15.** *Algorithm 8 must stop and its complexity is estimated by* $\tilde{O}(|\mathbf{Edge}|)$.

*Proof.* After once operation of sub algorithm 2 (Algorithm 9), the value $|\mathbf{Edge}|$ decreases at least 2, since $|\mathbf{Edge}| + |\mathbf{OL}| + |\mathbf{EL}|$ is unchanged and $|\mathbf{OL}| + |\mathbf{EL}|$ increases at least 2 from Lemma 11 and Lemma 12. Also $n$-gleton algorithm (Algorithm 5) keeps or decreases the value $|\mathbf{Edge}|$. Suppose that Algorithm 8 does not stop. So, we have $\mathbf{Graph} = \emptyset$, since the value $|\mathbf{Edge}|$ decrease at least 2 by once loop and goes to 0. It is a contradiction to Lemma 14.

From Lemma 11 and Lemma 12, the complexity of sub algorithm 2 (Algorithm 9) is $\tilde{O}$(number of deleted edges). On the other hands, from Lemma 7, the complexity of $n$-gleton algorithm (Algorithm 5) is also $\tilde{O}$(number of deleted edges). So, the total cost of this algorithm is estimated by $\tilde{O}(|\mathbf{Edge}|)$.

**Lemma 16.** *Suppose* $\mathbf{LP}$ *is the list which are computed in Algorithm 8. Put* $M'' := |\mathbf{LP}|$*, and* $\{(l_j, d_j, c_j) | j = 1, 2, 3..., M''\} := \mathbf{LP}$*. Then,* $\{l_j\}$*,and* $\{d_j\}$ *satisfy the condition of Theorem 1.*

*Proof.* The number of even loops in $\mathbf{LP}$ obtained by conjunction is $\geq |\mathbf{OL}| - |B_0 \backslash B \cup \{\infty\}|$. So, we have $|\mathbf{LP}| \geq |\mathbf{EL}| + (|\mathbf{OL}| - |B_0 \backslash B \cup \{\infty\}|) = M - |B_0 \backslash B \cup \{\infty\}| = M'$, which is the property i) of Theorem 1. From Lemma 10 ii), the number $n$ which appears sub algorithm 2 (Algorithm 9), is smaller than $1 + \log_2 |\mathbf{Vertex}|$. On the other hands, the length of the loop, collected in $\mathbf{LP}$, is smaller than $4n$. So we have property ii) of Theorem 1. Property iii) of Theorem 1 is directly from Lemma 4, since Algorithm 8 obeys the policy of the collection of even length loop of Algorithm 2. Thus, we finish the proof of Theorem 1.

## 9 Conclusion

In this paper, in the case that an element of group is written by the summation of $N$ large primes and factor bases in $O(1)$ probability, we formulate the index calculus and propose a new algorithm, which does not require the randomness of the graph and its worst complexity can be estimated.

## References


1. C. Diem, On the discrete logarithm problem in class groups, preprint, 2009.
2. P.Gaudry, An algorithm for solving the discrete log problem on hyperelliptic curves, *Eurocrypt 2000*, LNCS 1807, Springer-Verlag, 2000, pp. 19–34.
3. P. Gaudry, E. Thomé, Thériault, C. Diem, A double large prime variation for small genus hyperelliptic decomposed attack, Math. Comp. 76, 2007, pp.475–492.
4. P. Gaudry, Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem, Journal of Symbolic Computation, Vol.44, **12**, 2009, pp. 1690–1702.
5. R. Granger, F. Vercauteren, On the Discrete Logarithm Problem on Algebraic Tori, Advances in Cryptology, CRYPTO 2005, LNCS 3621, Springer-Verlag, 2005, pp. 66-85.
6. B. A. LaMacchia, A. M. Odlyzko, Solving large sparse linear systems over finite fields, *Crypto '90*, LNCS 537, Springer-Verlag, 1990, pp. 109–133.
7. K. Nagao, Index calculus for Jacobian of hyperelliptic curve of small genus using two large primes, Japan Journal of Industrial and Applied Mathematics, **24**, no.3, 2007.
8. K. Nagao, Decomposition Attack for the Jacobian of a Hyperelliptic Curve over an Extension Field, 9th International Symposium,ANTS-IX., Nancy, France, July 2010, Proceedings LNCS 6197,Springer, pp.285–300, 2010.
9. N. Thériault, Index calculus for hyperelliptic curves of small genus, ASIACRYPT2003, LNCS 2894, Springer-Verlag, 2003, pp. 75–92.
10. D. H. Wiedemann, Solving sparse linear equations over finite fields, *IEEE Trans. Inform. Theory*, **IT-32**, no.1, 1986, pp.54–62.