アルゴリズム 9－2
（1989. 7. 27）

有限体における原始根の生成アルゴリズム

伊東 利哉　　辻井 重男
東京工業大学 工学部 電気・電子工学科

あらまし:

原始根生成問題とは —“ある与えられた素数 p に対して、$GF(p)$ の原始根 g を生成する問題”— と定義される。一般に、原始根生成問題は $|p|$ に関する多項式時間あるいは平均的多項式時間では解くことができないと予想されている。以下本稿では、原始根生成問題に対する確率的アルゴリズムを提案し、その諸性質 — 原始根を生成する確率, 平均実行時間, 出力される元の最小位数及び平均位数 — について解析する。また、これらの解析結果より、与えられた素数 p に対して、本稿で提案する確率的アルゴリズムは、少なくとも $1-|p|^{-k}$（k:ある自然数）以上の確率で原始元を生成し、またその実行時間は、平均的多項式時間であることを明らかにする。これより、本確率的アルゴリズムは効率良く原始根を生成し、またこれは安全なプロトコル・システムを設計する一手法を与えるものと考えられる。

How to Generate a Primitive Root Modulo a Prime

Toshiya Itoh　　Shigeo Tsujii

Department of Electrical and Electronic Engineering,
Faculty of Engineering,
Tokyo Institute of Technology
O-okayama, Meguro-ku, Tokyo 152, Japan

Abstract:

Generation of a primitive root modulo a prime is a problem that given a prime p, generate a primitive root g modulo a given prime p. In general, it is conjectured and believed that this can not be solved in polynomial or random polynomial time in $|p|$. This paper presents a randomized algorithm for generating a primitive root modulo a prime, and analyzes several properties of the randomized algorithm such as the probability that the randomized algorithm generates a primitive root modulo a given prime, the expected running time of the algorithm, the maximum lower bound for the order of the output $g \in Z_p^*$, and the expected order of the output $g \in Z_p^*$. As a result, for a given prime p, the randomized algorithm generates a primitive root g modulo a prime p with probability at least $1-|p|^{-k}$ for some integer k and runs in expected polynomial time in $|p|$. Thus the randomized algorithm efficiently generates a primitive root g modulo a given prime p and this provides us one of the ways to design secure cryptographic protocols and systems.

1 Introduction

Generation (or recognition) of a primitive root modulo a prime is a simple but is supposed to be an intractable problem in number theory. (e.g., see C18 and C19 in [1], or see 4.3 in [5].) A solution to this problem, i.e., to find an efficient algorithm for generating (or recognizing) a primitive root modulo a prime, enables us to design *secure* cryptographic protocols and systems such as the public key distribution system by Diffie and Hellman [3], the public-key cryptosystem by El-Gamal [4], etc.

Informally, "recognition of a primitive root modulo a prime" is a problem that given a prime p and any $g \in Z_p^*$, recognize whether or not $g \in Z_p^*$ is a primitive root modulo a prime p, and "generation of a primitive root modulo a prime" is a problem that given a prime p, generate a primitive root g modulo a prime p. In this paper, we use **GPR** (**RPR**) to denote "generation (recognition) of a primitive root modulo a prime" for notational simplicity. It is already known that if every prime factor of $p-1$ is given, **RPR** is solvable in polynomial time in $|p|$ (e.g., see Rem18 in [1].), where $|x|$ denotes the length of a binary encoding of an instance x, and if **RPR** is solvable in (random) polynomial time in $|p|$, **GPR** is also solvable in random polynomial time in $|p|$. (e.g., see Rem19 in [1].) In a usual case, however, it is difficult to find every prime factor of $p-1$ [8], then we have the following simple and naive open problem:

Open Problem: If every prime factor of $p-1$ is *not* known, can **RPR** or **GPR** be solved in polynomial or random polynomial time in $n(=|p|)$?

In general, it is conjectured and believed that the answer to the **Open Problem** is "*no*." (see 4.3 in [5].) If this conjecture is true, then we have no way to generate (or recognize) a primitive root modulo a prime p in polynomial or random polynomial time in $n(=|p|)$. Then we may need a more *relaxed* setting, somewhat in a practical sense, for generating (or recognizing) a primitive root modulo a prime. An informal description for our setting is; (1) we are allowed to use at most polynomial or random polynomial time computing resources, and (2) an algorithm for a given problem correctly answers with *high probability.*

In this paper, we present a random polynomial time algorithm A_{GPR} for solving **GPR** with high probability, i.e., given a prime p, the algorithm A_{GPR} correctly generates a primitive root g modulo a prime p with high probability in random polynomial time in $n=|p|$. It is worth noting that the algorithm A_{GPR} generates a primitive root modulo a prime p with high probability in random polynomial time in $|p|$, but does not necessarily generate the same one in each execution of the algorithm, and that "high probability" is in a sense of not $1-2^{-O(|p|)}$ but $1-|p|^{-O(1)}$.

The outline of this paper is as follows: After some preliminaries, section 3 presents a randomized algorithm A_{GPR} for generating a primitive root modulo a prime. Sections 4 and 5 analyze several properties of the algorithm A_{GPR} such as the probability that the algorithm A_{GPR} generates a primitive root g modulo a prime p, the *expected* running time, the maximum lower bound for the order of the output $g \in Z_p^*$, and the *expected* order of the output $g \in Z_p^*$, and show that for a given prime p, the algorithm A_{GPR} generates a primitive root g modulo a prime p with probability at least $1-|p|^{-k}$ for some integer k and runs in *expected* polynomial time in $|p|$. In addition, Section 6 describes conclusion and several remarks, and refers to a further modified problem in a slightly different setting.

2 Preliminaries

In this section, we present a formal definition for "recognition of a primitive root modulo a prime (**RPR**)" and a known result for **RPR**. For notational simplicity, we use $PRIME$ to denote a set of all primes in the rest of this paper.

Definition 1 (RPR): Given a $p \in PRIME$, for any $g \in Z_p^*$, recognize whether or not $g \in Z_p^*$ is a primitive root modulo a given prime p.

If every prime factor of $p-1$ is known, then **RPR** can be solved in polynomial time in $n(=|p|)$ by the following lemma:

Lemma 1 (see Rem18 in [1]): Let $p \in PRIME$ and let the complete factorization of $p-1$ be of the form that $p-1 = p_1^{e_1} p_2^{e_2} \cdots p_t^{e_t}$, where $p_i \in PRIME$ and $e_i \geq 1$ $(1 \leq i \leq t)$. Then $g \in Z_p^*$ is a primitive root modulo a prime p iff $g^{(p-1)/p_i} \not\equiv 1 \pmod{p}$ $(1 \leq i \leq t)$.

Proof of Sketch: It is not difficult to show the following equivalency: "$g \in Z_p^*$ is *not* a primitive root modulo a prime p." $\Leftrightarrow$ "For some $e < p-1$ such that $e|p-1$, $g^e \equiv 1 \pmod{p}$." $\Leftrightarrow$ "For some $q \in PRIME$ such that $q|p-1$, $g^{(p-1)/q} \equiv 1 \pmod{p}$." □

3 Algorithm for GPR (A_{GPR})

In this section, we give a formal definition for "generation of a primitive root modulo a prime (**GPR**)," and present a randomized algorithm A_{GPR} for solving **GPR** in the case that every prime factor of $p-1$ is *not* known.

Definition 2 (GPR): Given a $p \in PRIME$, generate a primitive root g modulo a given prime p.

Without loss of generality, $p-1$ consists of t distinct prime factors $p_1, p_2, \cdots, p_t$, where $2 = p_1 < p_2 < \cdots < p_t$. Then the description of the algorithm A_{GPR} is as follows:

Algorithm A_{GPR}:

```
Input:  $p \in PRIME$, where $n = |p|$.
Step 1: For some integer $d \geq 1$, compute every prime factor $p_i$ of $p-1$ such that $p_i \leq n^d$ $(1 \leq i \leq s \leq t)$.
        (Let the partial factorization of $p-1$ be of the form that $p-1 = p_1^{e_1} p_2^{e_2} \cdots p_s^{e_s} \cdot Q$, where $2 = p_1 < p_2 <$
        $\cdots p_s \leq n^d$ and $Q$ is generally a composite number.)
Step 2: Choose $g \in Z_p^*$ randomly, uniformly, and independently.
Step 3: Compute $u \equiv \left\{g^{(p-1)/Q} - 1\right\} \cdot \prod_{1 \leq i \leq s} \left\{g^{(p-1)/p_i} - 1\right\} \pmod{p}$.
Step 4: If $u \not\equiv 0 \pmod{p}$, then output $g \in Z_p^*$ as a primitive root modulo a prime $p$. Otherwise, go to
        Step 2.
```

Note that the **Step 1** of the algorithm A_{GPR} can be carried out in polynomial time in $n = |p|$, and the integer d plays a role to control the probability that the algorithm A_{GPR} generates a primitive root g modulo a prime p and the running time of the algorithm A_{GPR}. In **Step 4** of the algorithm A_{GPR}, if $u \equiv 0 \pmod{p}$, (randomly, uniformly, and independently chosen) $g \in Z_p^*$ is certainly *not* a primitive root modulo a prime p, because $g^{(p-1)/p_j} \equiv 1 \pmod{p}$ for some j $(1 \leq j \leq s)$, or $g^{p_1^{e_1} p_2^{e_2} \cdots p_s^{e_s}} \equiv 1 \pmod{p}$. Thus the algorithm A_{GPR} rejects $g \in Z_p^*$ in the case that $u \equiv 0 \pmod{p}$. Then how probable is the output $g \in Z_p^*$ of the algorithm A_{GPR} as a primitive root modulo a prime p? To solve this problem, the following section precisely analyzes the probability that the output $g \in Z_p^*$ of the algorithm A_{GPR} is *really* a primitive root modulo a prime p and the running time of the algorithm A_{GPR}.

4 Analyses for Algorithm A_{GPR}

Our goals in this section are to show that the algorithm A_{GPR} generates a primitive root modulo a given prime p with high probability in a sense of $1-|p|^{-O(1)}$, and to prove that the algorithm A_{GPR} runs in *expected* polynomial time in $n=|p|$.

Theorem 1: Let P_{PR} be the probability that the output $g \in Z_p^*$ of the algorithm A_{GPR} is *really* a primitive root modulo a prime p. Then $P_{PR} > 1 - \lceil n/d \log n \rceil n^{-d}$, where $n=|p|$.

Proof: Let $p-1 = p_1^{e_1} p_2^{e_2} \cdots p_t^{e_t}$, where $p_i \in PRIME$, $e_i \geq 1$ $(1 \leq i \leq t)$ and $2 = p_1 < p_2 < \cdots < p_t$. If every prime factor of $p-1$ is known, then the output of the algorithm A_{GPR} is a primitive root modulo a prime p with probability 1. (see Lemma 1.) If not, then for some integer $d \geq 1$, there exists s $(1 \leq s < t)$ such that $p_s \leq n^d < p_{s+1}$. Here we define sets D_i to be $D_i = \{a | a^{(p-1)/p_i} \equiv 1 \pmod p,\ a \in Z_p^*\}$ $(1 \leq i \leq s)$ and a set S to be $S = \{a | a^{p_1^{e_1} p_2^{e_2} \cdots p_s^{e_s}} \equiv 1 \pmod p,\ a \in Z_p^*\}$. Furthermore, we define a quantity $N(n^d)$ to be $N(n^d) = \|D_1 \cup D_2 \cup \cdots \cup D_s \cup S\|$, where $\|A\|$ denotes the *cardinality* of a set A, thus,

$$\begin{aligned}
N(n^d) &= \|D_1 \cup D_2 \cup \cdots \cup D_s \cup S\| \\
&= \|D_1 \cup D_2 \cup \cdots \cup D_s\| + \|S - (D_1 \cup D_2 \cup \cdots \cup D_s)\| \\
&= \sum_{i=1}^{s} \| D_i \| - \sum_{1 \leq i < j \leq s} \| D_i \cap D_j \| + \sum_{1 \leq i < j < k \leq s} \| D_i \cap D_j \cap D_k \| \\
&\quad + \cdots + (-1)^{s-1} \| D_1 \cap D_2 \cap \cdots \cap D_s \| + \|S - (D_1 \cup D_2 \cup \cdots \cup D_s)\|.
\end{aligned}$$

Recalling the definition of D_i $(1 \leq i \leq s)$, then we have

$$\begin{aligned}
D_i \cap D_j &= \{a | a^{(p-1)/p_i p_j} \equiv 1 \pmod p,\ a \in Z_p^*\} \ (1 \leq i < j \leq s), \\
D_i \cap D_j \cap D_k &= \{a | a^{(p-1)/p_i p_j p_k} \equiv 1 \pmod p,\ a \in Z_p^*\} \ (1 \leq i < j < k \leq s), \\
&\ \vdots \\
D_1 \cap D_2 \cap \cdots \cap D_s &= \{a | a^{(p-1)/p_1 p_2 \cdots p_s} \equiv 1 \pmod p,\ a \in Z_p^*\},
\end{aligned}$$

and thus this yields

$$\begin{aligned}
N(n^d) &= \| D_1 \cup D_2 \cup \cdots \cup D_s \| + \|S - (D_1 \cup D_2 \cup \cdots \cup D_s)\| \\
&= \sum_{i=1}^{s} \frac{p-1}{p_i} - \sum_{1 \leq i < j \leq s} \frac{p-1}{p_i p_j} + \sum_{1 \leq i < j < k \leq s} \frac{p-1}{p_i p_j p_k} + \cdots + (-1)^{s-1} \frac{p-1}{p_1 p_2 \cdots p_s} \\
&\quad + \|S - (D_1 \cup D_2 \cup \cdots \cup D_s)\| \\
&= (p-1) \left\{ \sum_{i=1}^{s} \frac{1}{p_i} - \sum_{1 \leq i < j \leq s} \frac{1}{p_i p_j} + \sum_{1 \leq i < j < k \leq s} \frac{1}{p_i p_j p_k} + \cdots + (-1)^{s-1} \frac{1}{p_1 p_2 \cdots p_s} \right\} \\
&\quad + \|S - (D_1 \cup D_2 \cup \cdots \cup D_s)\| \\
&= (p-1) \left\{ 1 - \prod_{i=1}^{s} \left(1 - \frac{1}{p_i}\right) \right\} + \|S - (D_1 \cup D_2 \cup \cdots \cup D_s)\|.
\end{aligned}$$

Note that the set $S - (D_1 \cup D_2 \cup \cdots \cup D_s)$ consists of every distinct $(p_1^{e_1} p_2^{e_2} \cdots p_s^{e_s})$*-th* primitive root of unity modulo a prime p. (see the definitions of D_i $(1 \leq i \leq s)$ and S.) Then the cardinality of the set $S - (D_1 \cup D_2 \cup \cdots \cup D_s)$ is given by

$$\|S - (D_1 \cup D_2 \cup \cdots \cup D_s)\| = \varphi\left\{ \sum_{i=1}^{s} p_i^{e_i} \right\} = \prod_{i=1}^{s} p_i^{e_i} \cdot \prod_{i=1}^{s} \left(1 - \frac{1}{p_i}\right),$$

where $\varphi(\cdot)$ denotes the Euler's totient function. The algorithm A_{GPR} randomly, uniformly, and independently chooses $g \in Z_p^*$ in **Step 2**, and rejects $g \in Z_p^*$ such that $u \equiv 0 \pmod p$ in **Step 4**. On the other hand, there exist

$\varphi(p-1)$ distinct primitive roots modulo a prime p, hence the probability P_{PR} is bounded by

$$\begin{aligned}
P_{PR} &= \frac{\varphi(p-1)}{\| Z_p^* \| - N(n^d)} \\
&= \frac{(p-1)\prod_{i=1}^{t}\left(1-\frac{1}{p_i}\right)}{(p-1)-(p-1)\left\{1-\prod_{i=1}^{s}\left(1-\frac{1}{p_i}\right)\right\} - \prod_{i=1}^{s} p_i^{e_i}\cdot\prod_{i=1}^{s}\left(1-\frac{1}{p_i}\right)} \\
&= \frac{\prod_{i=1}^{t} p_i^{e_i}\cdot\prod_{i=1}^{t}\left(1-\frac{1}{p_i}\right)}{\prod_{i=1}^{t} p_i^{e_i}\cdot\prod_{i=1}^{s}\left(1-\frac{1}{p_i}\right) - \prod_{i=1}^{s} p_i^{e_i}\cdot\prod_{i=1}^{s}\left(1-\frac{1}{p_i}\right)} \\
&= \prod_{i=s+1}^{t}\left(1-\frac{1}{p_i}\right)\frac{\prod_{i=s+1}^{t} p_i^{e_i}}{\prod_{i=s+1}^{t} p_i^{e_i}-1} > \prod_{i=s+1}^{t}\left(1-\frac{1}{p_i}\right).
\end{aligned}$$

Recalling that $2 = p_1 < p_2 < \cdots < p_s < p_{s+1} < \cdots < p_t$, then we have

$$P_{PR} > \prod_{i=s+1}^{t}\left(1-\frac{1}{p_i}\right) > \prod_{i=s+1}^{t}\left(1-\frac{1}{p_{s+1}}\right) = \left(1-\frac{1}{p_{s+1}}\right)^{t-s}.$$

Since $p > p-1 > \prod_{i=s+1}^{t} p_i^{e_i} > \prod_{i=s+1}^{t} p_{s+1} > p_{s+1}^{t-s}$ and $n^d < p_{s+1}$, this yields

$$n = |p| > \log(p-1) > (t-s)\log p_{s+1} > (t-s)d\log n,$$

and it follows that $t-s < n/d\log n$. Then,

$$P_{PR} > \left(1-\frac{1}{p_{s+1}}\right)^{t-s} > \left(1-\frac{1}{p_{s+1}}\right)^{n/d\log n} > \left(1-n^{-d}\right)^{\lceil n/d\log n\rceil},$$

because $n^d < p_{s+1}$, $t-s < n/d\log n$, and $0 < 1-1/p_{s+1} < 1$. For any $d \geq 1$ and any $n \geq 1$, it is not difficult to show that $(1-n^{-d})^{\lceil n/d\log n\rceil} > 1-\lceil n/d\log n\rceil n^{-d}$, and thus we finally have $P_{PR} > 1-\lceil n/d\log n\rceil n^{-d}$, where $n = |p|$. □

The following theorem guarantees that the randomized algorithm A_{GPR} runs in *expected* polynomial time in $n(=|p|)$.

Theorem 2: The algorithm A_{GPR} runs in expected polynomial time in $n(=|p|)$.

Proof: To show that the algorithm A_{GPR} runs in expected polynomial time in $n(=|p|)$, the probability $P_{u\not\equiv 0}$ that for any $g \in Z_p^*$, $u \not\equiv 0 \pmod p$ in **Step 4** must be analyzed. Since $g \in Z_p^*$ is randomly, uniformly, and independently chosen, $P_{u\not\equiv 0}$ is bounded by

$$\begin{aligned}
P_{u\not\equiv 0} &= \frac{\|Z_p^*\| - N(n^d)}{\|Z_p^*\|} \\
&= \frac{(p-1)-(p-1)\left\{1-\prod_{i=1}^{s}\left(1-\frac{1}{p_i}\right)\right\} - \prod_{i=1}^{s} p_i^{e_i}\cdot\prod_{i=1}^{s}\left(1-\frac{1}{p_i}\right)}{p-1} \\
&= \frac{\prod_{i=1}^{t} p_i^{e_i}\cdot\prod_{i=1}^{s}\left(1-\frac{1}{p_i}\right) - \prod_{i=1}^{s} p_i^{e_i}\cdot\prod_{i=1}^{s}\left(1-\frac{1}{p_i}\right)}{\prod_{i=1}^{t} p_i^{e_i}} \\
&= \left(1-\prod_{i=s+1}^{t} p_i^{-e_i}\right)\cdot\prod_{i=1}^{s}\left(1-\frac{1}{p_i}\right) \\
&> \frac{1}{2}\cdot\prod_{i=1}^{s}\left(1-\frac{1}{p_i}\right) > \frac{1}{2}\prod_{q\in PRIME(n^d)}\left(1-\frac{1}{q}\right),
\end{aligned}$$

where $PRIME(L) = \{q | q \in PRIME,\ q \leq L\}$. Noting the result by Mertens [6] that

$$\prod_{q \in PRIME(L)} \left(1 - \frac{1}{q}\right) = \frac{e^{-C}}{\ln L}\left\{1 + O\left(\frac{1}{\ln L}\right)\right\},$$

where C is the Euler's constant [6], we have $P_{u \not\equiv 0} > e^{-C}/(4d \log n)$ for sufficiently large n. Then the *expected* iterations of Step 4, τ, is estimated by

$$\tau = \sum_{i=1}^{\infty} i P_{u \not\equiv 0} \{1 - P_{u \not\equiv 0}\}^{i-1} = \frac{1}{P_{u \not\equiv 0}},$$

and thus, for sufficiently large n and a fixed $d \geq 1$, $\tau < 4e^{C} d \log n < n$. Hence the algorithm A_{GPR} runs in expected polynomial time in n, because for any integer $d \geq 1$, the algorithm A_{GPR} (in a brute force manner) searches every prime factor p_i $(\leq n^d)$ of $p-1$ in polynomial time in $n = |p|$, and evaluates polynomially computable congruences modulo a prime p. $\square$

5 Analyses for Order of Outputs

In this section, we analyze the order of $g \in Z_p^*$ that the algorithm A_{GPR} generates as a primitive root modulo a prime p. To do this, we prove the maximum lower bound for the order of $g \in Z_p^*$ that the algorithm A_{GPR} outputs. Furthermore, we define *expected* order with respect to the algorithm A_{GPR} and show that it is large enough. In general, any $g \in Z_p^*$ of large order modulo a prime p provides secure cryptographic protocols or systems (e.g., see [3], [4].), thus this result is not only of theoretical interest but also of practical importance.

Theorem 3: For any $g \in Z_p^*$ that the algorithm A_{GPR} generates as a primitive root modulo a prime p, the order of $g \in Z_p^* \geq p_1^{e_1} p_2^{e_2} \cdots p_s^{e_s} p_{s+1}$.

Proof: In Step 4, the algorithm A_{GPR} rejects every $g \in Z_p^*$ whose order is divisible by both each divisor of $(p-1)/p_i$ $(1 \leq i \leq s)$ and each divisor of $p_1^{e_1} p_2^{e_2} \cdots p_s^{e_s}$. Thus it can be immediately shown that the least order of $g \in Z_p^*$ that the algorithm A_{GPR} generates as a primitive root modulo a prime p is $p_1^{e_1} p_2^{e_2} \cdots p_s^{e_s} p_{s+1}$. $\square$

Definition 3: The expected order $\varepsilon(n^d)$ with respect to the randomized algorithm A_{GPR} is an ensemble average over every $g \in Z_p^*$ that the algorithm A_{GPR} generates as a primitive root modulo a prime p.

Theorem 4: The expected order $\varepsilon(n^d)$ with respect to the algorithm A_{GPR} satisfies that $\varepsilon(n^d) > (p-1)(1 - \lceil n/d \log n \rceil n^{-d})$.

Proof: In Step 4, the algorithm A_{GPR} rejects every $g \in Z_p^*$ whose order is divisible by both each divisor of $(p-1)/p_i$ $(1 \leq i \leq s)$ and each divisor of $p_1^{e_1} p_2^{e_2} \cdots p_s^{e_s}$. Since for $v | p-1$, there exist $\varphi(v)$ distinct v-*th* primitive roots of unity modulo a prime p, the expected order $\varepsilon(n^d)$ with respect to the algorithm A_{GPR} is bounded by

$$\begin{aligned}
\varepsilon(n^d) &= \frac{\sum_{i_{s+1}=0}^{e_{s+1}} \sum_{i_{s+2}=0}^{e_{s+2}} \cdots \sum_{i_t=0}^{e_t} R \cdot p_{s+1}^{i_{s+1}} p_{s+2}^{i_{s+2}} \cdots p_t^{i_t} \varphi(R \cdot p_{s+1}^{i_{s+1}} p_{s+2}^{i_{s+2}} \cdots p_t^{i_t}) - R\varphi(R)}{\| Z_p^* - N(n^d) \|} \\
&\geq \frac{R \cdot p_{s+1}^{e_{s+1}} p_{s+2}^{e_{s+2}} \cdots p_t^{e_t} \varphi(R \cdot p_{s+1}^{e_{s+1}} p_{s+2}^{e_{s+2}} \cdots p_t^{e_t})}{(p-1) - (p-1)\left\{1 - \prod_{i=1}^{s}\left(1 - \frac{1}{p_i}\right)\right\} - \prod_{i=1}^{s} p_i^{e_i} \cdot \prod_{i=1}^{s}\left(1 - \frac{1}{p_i}\right)} \\
&> \frac{R \cdot p_{s+1}^{e_{s+1}} p_{s+2}^{e_{s+2}} \cdots p_t^{e_t} \varphi(R \cdot p_{s+1}^{e_{s+1}} p_{s+2}^{e_{s+2}} \cdots p_t^{e_t})}{\cdot (p-1)\prod_{i=1}^{s}\left(1 - \frac{1}{p_i}\right)} \\
&= \frac{(p-1)\varphi(p-1)}{(p-1)\prod_{i=1}^{s}\left(1 - \frac{1}{p_i}\right)} = (p-1) \prod_{i=s+1}^{t}\left(1 - \frac{1}{p_i}\right),
\end{aligned}$$

where $R = p_1^{e_1} p_2^{e_2} \cdots p_s^{e_s}$. Thus we can show, in a way similar to the proof of Theorem 1, that $\epsilon(n^d) > (p-1)(1 - \lceil n/d \log n \rceil n^{-d})$. $\square$

6 Conclusion and Remarks

In this paper, we presented a random polynomial time algorithm A_{GPR} for generating a primitive root modulo a prime with high probability. It is worth noting that the randomized algorithm A_{GPR} is also applicable to finding a generator of $GF^*(q^m)$ or to finding a generator of a cyclic group G in almost the same way.

Here we consider a slightly *modified* setting: For any $l \geq 2$, define a set $COMP(l)$ to be $COMP(l) = \{c | c = p_1^{e_1} p_2^{e_2} \cdots p_l^{e_l},\ p_1, p_2, \cdots, p_l \in PRIME,\ p_1 < p_2 < \cdots < p_l,\ |c|/|p_1| < K, \text{ for some constant } K\}$. (Informally, $COMP(l)$ is a set of *composite* numbers with l distinct prime factors, each of which is large enough.) Then we have a modified problem, i.e., *modified* generation of a primitive root modulo a prime (**MGPR**), in the following:

Problem (MGPR): Assume that $c \in COMP(l)$ $(l \geq 2)$ and q is the least prime in an arithmetic progression $fc + 1$ $(f \geq 1)$. Then for any $c \in COMP(l)$, generate a primitive root g modulo a prime q.

A problem similar to the above arises in [10] to demonstrate possession of two factors in a zero-knowledge manner. Dirichlet's theorem (see Theorem 15 in [6].) guarantees that for any $c \in COMP(l)$, there exist *infinitely* many primes in an arithmetic progression $fc+1$ $(f \geq 1)$. Though the least prime $q (= f_{min} c + 1)$ in an arithmetic progression $fc + 1$ $(f \geq 1)$ is proved to be $q < c^2/(\log c)^k$ for every $k > 0$ (see p.218 in [9].), Heath-Brown's conjecture [7] gives us a *strong* bound that claims $f_{min} = O(|c|^2)$. Thus, with Heath-Brown's conjecture, we can use A_{GPR} as a building block to solve **MGPR** in random polynomial time in $|c|$ with overwhelming probability, i.e., with probability at least $1 - 2^{-O(|c|)}$.

Algorithm A_{MGPR}**:**

Input: $c \in COMP(l)$, where $l \geq 2$.

Step 1: Find the least prime q in an arithmetic progression $fc + 1$ $(f \geq 1)$ in a brute force way using a random polynomial time primality testing algorithm [2].

Step 2: Input (randomly, uniformly, and independently chosen) $g \in Z_q^*$ and a prime q to the algorithm A_{GPR}.

Step 3: If $u \not\equiv 0 \pmod{q}$, then output $g \in Z_q^*$ as a primitive root modulo a prime q. Otherwise go to **Step 2**.

Noting that $q = f_{min} c + 1$ and $c \in COMP(l)$, we have $q - 1 = f_{min} c = f_{min} p_1^{e_1} p_2^{e_2} \cdots p_l^{e_l}$. Thus if Heath-Brown's conjecture is true, then the complete factorization of f_{min} can be found in polynomial time in $|c|$. Here we use $P_{COMP(l)}$ to denote the probability that the output $g \in Z_q^*$ of the algorithm A_{MGPR} is *really* a primitive root modulo a prime q, then

$$\begin{aligned} P_{COMP(l)} &= \prod_{i=1}^{l} \left(1 - \frac{1}{p_i}\right) > \prod_{i=1}^{l} \left(1 - \frac{1}{p_1}\right) = \left(1 - \frac{1}{p_1}\right)^l > 1 - \frac{l}{p_1} \\ &= 1 - l \cdot 2^{-\log p_1} > 1 - 2^{-O(|p_1|)} > 1 - 2^{-O(|c|)}, \end{aligned}$$

because $p_1 < p_2 < \cdots < p_l$ and $|c|/|p_1| < K$ for some constant K. Thus we can show in a way similar to the above that for a set $COMP = \bigcup_{l \geq 2} COMP(l)$, $P_{COMP} > 1 - 2^{-O(|c|)}$, where P_{COMP} denotes the probability that for any $c \in COMP$, the output $g \in Z_q^*$ of the randomized algorithm A_{MGPR} is *really* a primitive root modulo the least prime q in an arithmetic progression $fc + 1$ $(f \geq 1)$.

Acknowledgment: The author wishes to thank Osamu Watanabe of Tokyo Institute of Technology for showing me the outline of this paper and giving me several technical comments on this work. The author would also like to thank Hiroki Shizuya of Tohoku University and Mitsunori Ogiwara of Tokyo Institute of Technology for their suggestive comments and valuable discussions on the early version of this paper.

参考文献

[1] L.M. Adleman and K.S. McCurley, "Open Problems in Number Theoretic Complexity," Perspectives in Computing, Vol.15, Discrete algorithms and Complexity, *Academic Press* (Proceedings of the Japan-US Joint Seminar), 1987, pp.237-262.

[2] H. Cohen and H.W. Lenstra, Jr., "Primality Testing and Jacobi Sums," *Math. Comp.*, Vol.42, No.165, 1984, pp.297-330.

[3] W. Diffie and M.E. Hellman, "New Directions in Cryptography," *IEEE* Trans. on Inform. Theory, Vol.IT-21, No.6, November 1976, pp.644-654.

[4] T. El-Gamal, "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithm," *IEEE* Trans. on Inform. Theory, Vol.IT-31, No.4, July 1985, pp.469-472.

[5] J. Grollmann and A.L. Selman, "Computing Measures for Public-Key Cryptosystems," *SIAM* J.Comput., Vol.17, No.2, April 1988, pp.309-335.

[6] G.H. Hardy and E.M. Wright, An Introduction to the Theory of Numbers, 5th Edition, *Clarendon Press*, Oxford, 1979.

[7] D.R. Heath-Brown, "Almost Primes in Arithmetic Progressions and Short Intervals," *Math. Proc. Cambridge Philos. Soc.*, 83, 1978, pp.357-375.

[8] H.W. Lenstra, Jr., "Elliptic Curve Factorization and Primality Testing," *Proceedings of Computational Number Theory Conference*, 1985.

[9] P. Ribenboim, The Book of Prime Number Records, *Springer-Verlag*, 1987.

[10] H. Shizuya, K. Koyama, and T. Itoh, "Zero-Knowledge Interactive Proof of Possession of Two factors," submitted to Auscrypto'90, 1989.