A SHORT NOTE ON DISCRETE LOG PROBLEM IN $\mathbb{F}_p^*$

HABEEB SYED

Abstract. Let p be a odd prime such that 2 is a primitive element of finite field $\mathbb{F}_p$. In this short note we propose a new algorithm for the computation of discrete logarithm in $\mathbb{F}_p^*$.

Introduction

Consider a finite field $\mathbb{F}_q$ (also denoted by $\mathrm{GF}(q)$), where $q = p^r$, p is a prime and $r \in \mathbb{N} := \{1, 2, 3, \ldots\}$. Let α be a primitive element of $\mathbb{F}_q$ i.e., generator of the multiplicative cyclic group $\mathbb{F}_q^*$. For arbitrary element $b \in \mathbb{F}_q^*$ computing $n \in \mathbb{N}, n \leq q - 1$ such that

$$b = \alpha^n \mod p \tag{1}$$

is known as *discrete log problem* (DLP) in $\mathbb{F}_q^*$. Discrete log computation in finite fields is an important problem mainly due to applications of these groups in cryptography. Beginning with Diffie-Hellman key exchange protocol [3], El ElGamal encryption/signature scheme [4] the DLP in $\mathbb{F}_q^*$ has been used as basic mathematical primitive in many cryptographic schemes, and security of these systems depend on difficulty of DLP in respective $\mathbb{F}_q^*$. It is rather difficult to give even reasonably good list of references to all the work involving DLP in $\mathbb{F}_q^*$, however [8, 6] are good to begin with.

In the last couple of decades DLP in $\mathbb{F}_q^*$ has been studied extensively and several algorithms have been proposed for the computation same. Most efficient algorithm for the computation of DLP is the one based on Number Field Sieve [5, 9]. See also [2, 7] for results which are not computationally oriented but certainly give insight into the problem of DLP in $\mathbb{F}_q^*$. In this short note we are focused on odd primes p for which 2 is primitive element of $\mathbb{F}_p$. For such primes we propose a new algorithm to compute discrete logarithm in $\mathbb{F}_p^*$. The proposed algorithm is based on elementary properties of finite fields and is purely theoretical in nature. Further, complexity of the algorithm is exponential, and as such it is not being suggested for any computational purposes. This short note has two sections. In section 1 we begin with basic results needed and then explain the algorithm in detail. In section 2 we analyze the complexity of the algorithm.

Key words and phrases. Finite Fields, Discrete Log Problem.

1. The Algorithm

In reminder of this note p denotes odd prime and $r \in \mathbb{N}$. By $\log_\alpha b = n$ we mean n as in (1). We begin with following simple results.

1.1. **Mini Lemma.** *Let $a, b \in \mathbb{F}_q^*$, $(q = p^r)$ be such that $a + b = 0(\bmod p)$, then for any primitive element α of $\mathbb{F}_q$ we have,*

$$\log_\alpha a - \log_\alpha b = \log_\alpha b - \log_\alpha a = \frac{q-1}{2} \bmod (q-1).$$

Proof. For any $a, b \in \mathbb{F}_q^*$ we have,

$$a + b = 0(\bmod p) \Longleftrightarrow \frac{a}{b} = \frac{b}{a} = -1(\bmod p).$$

Computing discrete logarithm with respect to any primitive element α of $\mathbb{F}_q$, we have,

$$\log_\alpha \frac{a}{b} = \log_\alpha \frac{b}{a} = \log_\alpha a - \log_\alpha b = \log_\alpha(-1).$$

Now the conclusion follows from the simple observation,

$$\log_\alpha(-1) = \frac{q-1}{2} \bmod (q-1). \tag{2}$$

□

1.2. *Remark.* The result (2) is true in more generality: Let $\mathbf{G}$ be a finite cyclic group of even order say, $2m$. Suppose α is a primitive element of $\mathbf{G}$. It is easy to see that the element $\beta = \alpha^m$ is the only non-trivial element fixed by all automorphisms of $\mathbf{G}$. This implies that the discrete logarithm of β is independent of primitive element α of $\mathbf{G}$ and is equal to m. In case of $\mathbf{G} = \mathbb{F}_q$, we have (2).

The proposed algorithm depends on above lemma and following simple fact:

Fact 1. Let $a, b \in \mathbb{N}$, $1 < a, b < p$ be such that $a + b = p$, then precisely one of a, b is divisible by 2.

Before we explain the algorithm we remind that this algorithm computes n in (1) when p is a odd prime such that 2 is a primitive element of $\mathbb{F}_p$. A necessary condition for such a thing to happen is that $p \equiv \pm 3(\bmod 8)$ [1, Chap 4]. Next we explain the proposed algorithm with the help of simple example.

Example. Consider the cyclic group $\mathbb{F}_{37}^*$ which is generated by 2. Suppose we want to find $\log_2 3$. Noting that all the operations are performed $\bmod 37$, the proposed algorithm works as follows

We have $3 + 34 = 37$ and hence

$$\begin{aligned} 3 = -34 =& 2 \cdot (-17) = 2 \cdot 20 \\ =& 2 \cdot (4 \cdot 5) = 2^3 \cdot 5 = 2^3 \cdot (-32) \\ =& 2^3 \cdot 2^5 \cdot (-1) = 2^8 \cdot 2^{18} = 2^{26} \end{aligned}$$

We have $\log_2(3) = 26$.

Now we are ready to state the algorithm.

Algorithm 1

INPUT: Element b of $\mathbb{F}_p^*$

OUTPUT: Discrete Log of b to base 2

Initialize Out $= 0$
if $b = 1$ **then**
  return 0
end if
while $b \neq 1$ **do**
  Find the max power k of 2 that divides b
  if $k = 0$ **then**
    $b = p - b$
    Out $=$ Out $+ (p-1)/2 \pmod{(p-1)}$
  else
    $b = b/(2^k)$, Out $=$ Out $+ k \pmod{(p-1)}$
  end if
end while

Next we prove that the Algorithm 1 converges.

Proof. Suppose we want to compute $\log_2 b$ $(b \in \mathbb{N}, 1 < b < p)$ in $\mathbb{F}_p^*$. Let $b = 2^r b'$, b' not divisible by 2, then $\log_2 b = r + \log_2 b'$, and hence if needed we can replace b by b' and assume that b is not divisible by 2. Since we are assuming that 2 is primitive element of $\mathbb{F}_p^*$, there exists t such that $1 < t < p - 1$ and

$$b \equiv 2^t \pmod{p}. \tag{3}$$

Let $b_0 = p - b$. Since b is not divisible by 2 we have that b_0 is divisible by 2. Let $b_0 = 2^r b_1$ where $r \in \mathbb{N}$ and b_1 is not divisible by 2. If $b_1 = 1$, we are done. Otherwise,

Claim. *$r < t$.*

Suppose not; let $r = t + s, s \in \mathbb{N}$, then from (3) we have,

$$(-b_1)2^r \equiv 2^t \pmod{p} \Longrightarrow p \text{ divides } 2^t + b_1 2^{t+s} = 2^t(1 + b_1 2^s) \tag{4}$$

and hence p divides $(1+b_1 2^s)$. On the other hand $(1+b_1 2^s) < b+b_0 = p$ and hence the only way p can divide $(1+b_1 2^s)$ is if $1+b_1 2^s = 0$ in $\mathbb{Z}$, which clearly is not the case.

So we have

$$-b_1 \equiv 2^{t-r} (\operatorname{mod} p). \tag{5}$$

Now we are back to (3) with $b = p - b_1$ and $t = t - r$. Thus, after at most t iterations the algorithm stops and returns value of $\log_2 b$. □

2. Analysis of The Algorithm

Throughout this section p denotes odd primes for which 2 is primitive element of $\mathbb{F}_p$. For a given $b \in \mathbb{F}_p^*$, to compute $\log_2 b$, Algorithm.1 repeats steps $(6)-(8)$ each time replacing b by $p-b'$ until $b' = \pm 1 (\operatorname{mod} p)$. The space requirements to execute the algorithm are not significant, but the order of growth of computations is $\mathrm{O}(2^{(p-1)/2})$. This algorithm does not give any advantages over the existing algorithms in terms of complexity. Our computational experiments with the algorithm suggested that while implementation of the algorithm worst case scenario (in terms of time taken to compute) occurred while calculating $\log_2((p-1)/2)$. However one can easily check,

$$\log_2(\frac{p-1}{2}) = \frac{p-3}{2}$$

Acknowledgments

Author is thankful to Mr. Ramanjulu for his remarks as well as for his help in coding and computations.

References

[1] Baker, A., *A Concise Introduction to the Theory of Numbers*, Cambridge University Press, 1984.

[2] Coppersmith, D., Shparlinski, I.E., *On polynomial approximation of the discrete logarithm and the Diffie-Hellman mapping*, Journal of Cryptology, **93** (2000), 387-399.

[3] Diffie, W., Hellman, M., *New directions in cryptography*, IEEE Transactions of Information Theory, **22** (1976), 644654.

[4] El Gamal, T., *A public-key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Transactions of Information Theory, **31** (1985), 469-472.

[5] Gordon, D.M., *Discrete logarithms in* GF(p) *using the number field sieve*, SIAM J. Discrete Math., **6:1** (1993), 124138.

[6] Menezes, A., Okamoto, T., Vanstone, S., *Reducing elliptic curve logarithms to logarithms in a finite field*, IEEE Transactions of Information Theory, **39** (1993), 16391646.

[7] Mullen, G.L., White, D., *A polynomial representation for logarithms in GF(q)*, Acta Arithmetica, **47** (1986), 255-261.

[8] Odlyzko, A.M., *Discrete logarithms in finite fields and their cryptographic significance*, pp. 224-314 in Advances in Cryptology: Proceedings of EUROCRYPT 84, (T. Beth, N. Cot, and I. Ingemarsson eds.), pp.224-314 Springer-Verlag, Lecture Notes in Computer Science 209, 1985.

[9] Schirokauer, O., *Using number fields to compute logarithms in finite fields,* Mathematics of Computation. **69** (2000), 1267-1283.

Information Security Group
Computational Research Laboratories Limited
Pune, 411 016 - INDIA
E-mail address: habeeb@crlindia.com