An Algorithm For Selecting a Generator of Z_p^*

Shaohua Zhang[1] Gongliang Chen[2] Huanguo Zhang[3] Chunhong Wang[1]
(1. Wuhan Maritime Communications Research Institute, CSIC.
2. School of Information Security Engineering of Shanghai Jiaotong University. Shanghai
3. School of Computing Science of Wuhan University. Wuhan)

Abstract *How to select a generator of Z_p^* is an important problem in cryptographic applications, where p is an odd prime. In this paper, we mainly consider the especial case that p is a safe prime, and give an algorithm for selecting a generator of Z_p^*, and find all generators of Z_p^*, where p is a safe prime. Our algorithm is more faster than the algorithm in [1]. Based on the proposed algorithm, one could find all generators of Z_p^* as well, where p is a perfect prime.*

Key words *safe prime perfect prime generator primitive root discrete logarithms*

Introduction

To selects a generator of Z_p^*, where p is a prime, usually, we use the straightforward algorithm in [2]. But we need the complete factorization of $p-1$ for the algorithm to work. This may be difficult when $p-1$ is very large. No polynomial–time algorithm is known for finding generators, or even for testing whether an element is a generator, of a finite group Z_p^* (or a finite field F_q) if factorization of $p-1$ (or $q-1$) is unknown. V. Shoup[3] considered the problem of deterministically generating in polynomial time a subset of F_q that contains a generator, and presented a solution to the problem for the case where the characteristic p of F_q is small (e.g. $p=2$). U. Maurer[4] discussed how his algorithm can be used to generate the parameters (p,α), where p is a provable prime and α is a generator of Z_p^*. In this paper, we mainly consider the especial case that p is a safe prime. A safe prime p is a prime of the form $p=2q+1$, where q is a prime. There is the algorithm that selects a generator of Z_p^* in [1], where p is a safe prime. Here, we give an algorithm that selects a generator of Z_p^*. Our algorithm is not only in polynomial time but also very instant. In fact, it has a running time of only a few seconds on a computer.

Algorithm Selecting a generator of Z_p^*, where p is a safe prime.

INPUT: a safe prime p

Computer $r=p(\mathrm{mod}\,8)$;

If $r=3$, then 2 is a generator of Z_p^*, otherwise $\frac{p-1}{2}$ is a generator of Z_p^*;

OUTPUT: a generator of Z_p^*

The above–mentioned algorithm is true. We have the following theorem:

Theorem Let $p=2q+1$ be a safe prime. If $p\equiv 3(\mathrm{mod}\,8)$, then all generators of Z_p^* are $2^1,2^3,2^5,\mathrm{L}, 2^{q-2},2^{q+2},\mathrm{L}, 2^{2q-3}$ and $2^{2q-1}(\mathrm{mod}\,p)$. Otherwise, all generators of Z_p^* are $q^1,q^3,q^5,\mathrm{L}, q^{q-2},q^{q+2},\mathrm{L}, q^{2q-3}$ and $q^{2q-1}(\mathrm{mod}\,p)$.

Based on the above–mentioned theorem and algorithm, one could find all generators of Z_p^* as well, where p is a perfect prime.

Proof of the theorem

Proof Since $p=2q+1$ is a safe prime, hence q is a prime. Since q is a prime, hence $q\equiv 1(\mathrm{mod}\,4)$ or $q\equiv 3(\mathrm{mod}\,4)$. If $q\equiv 1(\mathrm{mod}\,4)$, then $p\equiv 3(\mathrm{mod}\,8)$, $\left(\frac{2}{p}\right)=-1$. Hence $2^q\neq 1(\mathrm{mod}\,p)$. We also have $2^2\neq 1(\mathrm{mod}\,p)$, thus 2 is a primitive root $\bmod p$, and so is 2^a, if $(a,2q)=1$. If $q\equiv 3(\bmod 4)$, then $p\equiv 7(\bmod 8)$. If $q^q\equiv 1(\bmod p)$, then $\left(\frac{q}{p}\right)=1=\left(\frac{p}{q}\right)(-1)^{\frac{q-1}{2}\frac{p-1}{2}}=-1$. It is impossible. Since $q^2\neq 1(\bmod p)$, hence q is a primitive root $\bmod p$, and so is q^a, if $(a,2q)=1$. We know that there are $q-1$ primitive roots $\bmod p$. Let $a=1,3,5,L,q-2,q+2,L$, $2q-3$ and $2q-1$, then it shows immediately that theorem is true.

We know that p is a safe prime if and only if $2^{p-1}\equiv 1(\bmod p)$. See [5]. By our algorithm, now one could know that 20000000000000002559 is a safe prime, which has a primitive root 10000000000000001279, and 80000000000000001239 is a safe prime, too, which has a primitive root 40000000000000000619.

Now we consider how to select a generator of Z_p^*, where p is a perfect prime. A perfect prime p is a prime of the form $p=2q+1$, where q is a safe prime. For example, 23 is a perfect prime. Note that 16421 is a prime, $32843=2\times 16421+1$ and $65687=2\times 32843+1$ are primes as well. Therefore 65687 is a perfect prime, too. Chuan-kun Wu[6] proved an interesting result that some perfect primes are not perfect. Based on the our algorithm, one could simply prove the following result:

Corollary 1 If $p=2q+1$ is a perfect prime, then all generators of Z_p^* are

$q^1,q^3,q^5,L,q^{q-2},q^{q+2},L,q^{2q-3}$ and $q^{2q-1}(\bmod p)$.

Corollary 2 A safe prime has a prime primitive root, and so has a perfect prime.

Concluding remarks

As you know, some public-key systems require lager primes. For example, to guard against the Pohlig-Hellman -algorithm for computing discrete logarithms, a security requirement is that p is a random prime satisfying that $p-1$ should contain a large prime factor. Also see [1, page108]. Another familiar example is the requirement of primes p and q for an RSA modulus $n=pq$. The discovery of these public-key systems led to the consideration of several additional constraints on the choice of a prime p which is necessary to ensure the resulting systems safe from cryptanalytic attack, and the notion of a safe prime, a perfect prime and a strong prime were defined. In [5], we preliminarily explore how to generate and determine quickly this kind of especial primes. (As for the definition and property of a strong prime, see [7], [8].) In this paper, we mainly consider how to find primitive roots of this kind of especial primes. Undoubtedly, the problem we consider is important in cryptographic applications. Many mathematicians, cryptologist and cryptanalyst are all interested in it. On the other hand, although we have found primitive roots of a safe prime or a perfect prime, how to find primitive roots of a strong prime is still open. It will be our next goal of this line of research.

References

[1] A.Menezes, P.Van Oorschot and S. Vanstone, Handbook of Applied Cryptography. CRC Press , 1997, pp.164.

[2] Henri Cohen, A Course in Computational Algebraic Number Theory, Graduate Texts in Mathematics 138, Springer-Verlag, Berlin, Heidelberg, New-York, 1996. pp.25.

[3] V.Shoup, Searching for Primitive Roots in Finite Fields. Mathematics of Computation. 1992(58), pp.369-380.

[4] U.Maurer, Fast Generation of Prime Numbers and Secure Public-key Cryptographic Parameters. Journal of Cryptology, 1995(8), pp.123-155.

[5] Zhang Shaohua, Chen Gongliang, Zhou Guangming, Yang jinfeng, Determination of the Primality of 2kp+1.数学杂志, 2005 , 25(1), pp.16.

[6] Chuan-kun Wu, Differential Factoring for Integers. See Kefei Chen Edited, Progress on Cryptography, 25Years of Cryptography in China. Boston/Dordrecht/London: Kluwer Academic Publishers. 2004, pp.25.

[7] J.Gordon, Strong RSA Keys. Electronics Letters.1984, 20: 514~516.

[8] J.Gordon, Strong Primes are Easy to Find. Advances in Cryptology-Proceedings of EUROCRYPT. 1985,84(LNCS 209), 216~223.