Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Networks 49 (2005) 476–491
Computer Networks
www.elsevier.com/locate/comnet

A note on efficient implementation of prime generation algorithms in small portable devices

Chenghuai Lu *, Andre L.M. Dos Santos

Georgia Institute of Technology, College of Computing, 801 Atlantic Drive, Atlanta, GA 30309, United States

Received 23 December 2003; received in revised form 2 December 2004; accepted 17 December 2004
Available online 23 March 2005

Responsible Editor: G. Schaefer

Abstract

This paper investigates existing prime generation algorithms on small portable devices, makes optimizations and compares their efficiencies. It shows by comparing the performances that the bit array algorithm is the most efficient among all the existing prime generation algorithms. The paper further optimizes the implementation of the bit array algorithm by using an optimal parameter in the prime generations, namely the small prime set for its sieving procedure. A method for estimating the optimal small prime set for the bit array algorithm is provided. The paper gives generalized bit array algorithms which are able to find primes with special constraints, i.e., DSA primes and strong primes. Finally, the algorithms are implemented in a smart card and a PDA for validation. It shows that there is very little efficiency sacrifice for generating special primes with respect to generating random primes. It also shows that using optimal sets of small primes for prime generations will result in 30–200% efficiency improvement.

Keywords: Public key cryptography; Smart card; Primality test; Sieve procedure

1. Introduction

Cryptographic functions based on public key cryptography [6,19] have gained increasing attention from the research and commercial communities, as well as from end users. The use of public key cryptography can add security to a wide variety of applications. Especially, public key cryptography is a valuable tool for simplifying key management and enabling secure communications. Recently, there is a strong trend to use public key cryptography in small portable devices such as smart cards and handheld PDA's to enable them to perform secure transactions. Those devices should be able to implement one or multiple

* Corresponding author. Tel.: +1 404 822 8849.
E-mail address: lulu@cc.gatech.edu (C. Lu).

1389-1286/$ - see front matter
doi:10.1016/j.comnet.2004.12.007

public key cryptographic systems. Examples of public key algorithms that can be used by portable devices are RSA [17], Diffie–Hellman [7] and DSA [15]. Among them, RSA is the most popular public key cryptosystem and has been widely deployed in many portable devices to support protocols (e.g. communication protocols).

Many of the currently available portable devices possess a limited amount of hardware resource and computing power. Consequently, the processing speed attained by implementing public key cryptographic functions on those devices could be much slower than those on desktop computers. Because of this, there have been many studies on using specially designed hardware and software approaches to overcome the limited hardware resource and improve the performance of certain cryptographic functions [10,13,16].

This paper focuses on one of the important problems in public key cryptosystems—the generation of large random primes on resource-constrained devices. Due to the nature of the procedure for prime generation, which will be discussed later in the paper, the generation of large random primes is very time costly. For instance, the generation of 1024-bit primes, which can correspond to 2048-bit RSA key pairs, may cost several minutes to accomplish in devices like smart cards. For some applications, a user may need a higher security level that requires generating even larger primes, e.g. 2048-bit primes. In this case, the time needed for generating 2048-bit random primes is usually much more.[1] One of the reasons contributing to the low performance of large prime generation is the hardness of finding efficient primality testing algorithms. Meanwhile, another reason is that the existing prime generation algorithms and the implementations are not sufficiently investigated, particularly for small portable devices.

There have been a number of prime generation algorithms implemented on small portable devices, with performances varying widely. Unfortunately, most implementations simply use any prime generation algorithm available to them without noticing the big performance variations among them. As a result, some of the implementations run for an unreasonable amount of time. The total time for key generation can be very long and sometimes unacceptable, especially when a group of keys need to be generated or if some low-end tamper resistant devices are used. In addition, many small portable devices have limited storage space, particularly non-persistent storage that is used to store temporary values. The problem is aggravated by the fact of having several applications competing for the limited storage. Naturally, prime generation algorithms should optimize their storage requirements, what is not considered in the usual publicly available prime generation algorithms. Therefore, it is necessary to optimize the performance and storage requirements of prime generation algorithms for small portable devices.

This paper investigates existing prime generation algorithms, makes optimizations and compares their efficiencies in terms of time and memory space required. Hence, it provides a good reference for software engineers who implement large prime generations on small portable devices where resource is limited. The paper initially discusses one of the most used ways for prime generation—incremental search. Then, several optimizations are made to the incremental search prime generation algorithm. The study of performances shows that the table lookup and bit-array algorithms are the most efficient among all the algorithms examined. In addition, the storage requirements are compared. The bit array algorithm requires significantly less memory than the table lookup algorithm and therefore is the best choice among the algorithms, when both time and memory efficiencies are considered.

One of the important issues in optimizing incremental search prime generation algorithms is in choosing a small prime set (SPS) for the sieve procedure. The paper analyzes the factors that affect the choices of the optimal SPS sets when generating different sizes of primes on portable devices and proposes a method that can predicate those values instead of exhaustively searching for them. It is shown by experiments that the efficiencies of prime generations can be improved by 30% in the worst case and 200% in the best case by using optimal SPS sets, compared with using some

[1] It could be more than 10 times as much as that for generating 1024-bit primes.