Cryptographic Techniques

Jennifer Seberry

Department of Computer Science
University College
University of New South Wales

Centre for Communications Security Research

Australian Defence Force Academy
Canberra, ACT, 2600
AUSTRALIA

The public key cryptosystem published by Diffie and Hellman in 1976 is still of great interest today. Of the public key systems that have been proposed the RSA scheme of Rivest, Shamir and Adelman is of most current interest.

We shall briefly describe this system (more can be found in references 2 and 7) and then describe a method to obtain suitable primes for use in the scheme.

The RSA Scheme

To implement this scheme a person (traditionally Bob) makes himself a set of three large numbers: m, E and D, (the modulus, public key and secret key respectively) with the following properties

$$\text{if } y = x^E \pmod{m} \quad \text{then} \quad x = y^D \pmod{m}$$

for all numbers x in the range $(0, m - 1)$.

The numbers E and m are published, and someone else (traditionally Alice) who wishes to send a secret message x (regarded for the purposes of encryption as a large integer) to Bob, calculates y from x and sends to Bob the cryptogram y. Since Bob knows D he can recover the message. Anyone else wishing to eavesdrop must find D, or else discover x some other way. Both of these recourses appear to be computationally infeasible for suitable choices of parameters.

Bob makes m, E and D as follows. He chooses two very large primes p and q with p and q of roughly equal size (of say 500-600 bits each). This choice of p and q is what will concern us here. Bob chooses E at random, relatively prime to $\gcd(p - 1, q - 1)$, and then finds D by solving

$$ED = 1 \pmod{(p - 1)(q - 1)}$$

which he can do easily and quickly using Euclid's algorithm (see reference 7). Finally he forms m by choosing

$$m = pq.$$

A potential eavesdropper must, it seems, first find D, which appears to require the determination of p and q, which in turn seems to imply that he must be able to factorize m. To factorize $m = pq$ where p and q are very large primes of say 500-600 bits each is one of the hardest known common problems (see references 2 and 3).

Strong primes give more cryptographic security

The advanced techniques a cryptanalyst might use (see references 2, 3 and 6) break down when p (and similarly q) is not only prime but has the property that $p - 1$ has a large prime factor, say r, and $p + 1$ has a large prime factor, say s (see reference 6). The cryptanalyst's task is made

even more difficult if $r - 1$ should have a large prime factor, say t, as well. For logistical reasons it is also necessary to be able choose p in some sense at random but with a given number of bits.

Thus there is considerable urgency to solve the problem of finding primes with these desirable properties. However very little has been published on the way to do so. The method we describe is due to John Gordon (reference 5). It finds the large prime factors r, s and t separately and incorporates them in the construction. The extra conditions imposed on p and q add only 19% to the task of finding p and q .

A prime p will be called a *strong prime* if it satisfies the following seven conditions:

(i) p is large

(ii) p is prime

(iii) p is chosen at random in response to a seed

(iv) p has a given number of bits

(v) $p - 1$ has a large prime factor, say r

(vi) $p + 1$ has a large prime factor, say s

(vii) $r - 1$ has a large prime factor, say t .

Since we wish p, r and s all to be large we are only interested in odd primes p whose properties are

$$p = 2jr + 1 \qquad (\text{or } p = 1 \pmod{2r})$$

$$p = 2ks - 1 \qquad (\text{or } p = (s - 1) \pmod{2s})$$

$$r = 2Lt + 1 \qquad (\text{or } pr = 1 \pmod{2t})$$

for some j, k and L where r, s and t are primes.

Gordon's technique

Gordon proposes the following steps:

(i) choose random seeds a and b

(ii) from a and b generate random primes s and t

(iii) from t construct r

(iv) from r and s construct p .

Find r and s: Finding random primes r and s which are of a specified number of bits (n) and greater than a given seed is relatively straightforward. Starting with a random seed a , we find the first prime s (or t) greater than a .

The time to find s (or t) in this way using Knuth's Algorithm - P (see reference 3), is dominated by the time to perform modular exponentiations which take, on average, approximately

$$T_{exp}(n) = cTn^3/w \text{ seconds}$$

where c is a constant of size about 8, T is the time (in seconds) for one instruction and w is the word size in bits. If we quickly eliminate by trial division all multiples of primes less than, say,

256 it is necessary to examine only about $0.07n$ numbers on average before finding a prime (see reference 5), and so the time needed to find s (or t) (ignoring the time for quick eliminations) is about $0.07n$ times cTn^3/w, i.e. about

$$T_{prime}(n) = cvTn^4/w$$

when we have found s (or t) it is unlikely to have more bits than the seed a. We can virtually ensure this by picking our value of a in the range $(2^{n-1}, 2^{n-1}+2^{n-2}-1)$. This ensures a starts with the two binary digits 10 which leaves a run of 2^{n-2} integers in which to find a prime before increasing the number of bits.

Find r: We now wish to find a prime of the form $2Lt+1$. We search through $(2Lt+1)$-space for successive values of L.

We are likely to exhaust about $nLn(2)/2 = 0.35n$ sucessive values of L before finding r (see reference 5). Every time L doubles another bit is added to $2Lt+1$. A naive search will almost certainly result in a prime of too many bits.

A more sophisticated approach is to choose $2t$ to be, say, $d(n) = \log_2(n)$ bits shorter than the desired length of r (see below), then starting with unity, to add in successive multiples of $2t$ until the desired length of r is achieved and then to begin checking primality at each subsequent addition of $2t$. In this way, L is unlikely to double during the search for primes. This certainty of success can be increased by using a larger $d(n)$.

Find p: We now wish, given primes r and s, to find a prime p, close in size to a given number of bits, and satisfying

$$p = 2jr+1 = 2ks-1 \qquad \text{for some } j \text{ and } k$$

or

$$p = 1 \pmod{2r} = (2s-1) \pmod{2s} \qquad (1)$$

The key to finding primes with these properties is the following theorem.

Theorem: If r *and* s *are odd primes, then* p *satisfies*

$$p = 1 \pmod{2r} = (s-1) \pmod{2s}$$

if and only if p *is of the form*

$$p = p_0 + 2krs \qquad (2)$$

where

$$p_0 = u(r,s) \qquad : u(r,s) \text{ odd}$$

$$= u(r,s) + rs \qquad : u(r,s) \text{ even}$$

and

$$u(r,s) = (s^{r-1} - r^{s-1}) \pmod{rs} \qquad (3)$$

Proof: Integers which satisfy (1) also satisy the weaker condition

$$p = jr+1 = ks-1 \qquad \text{for some } j \text{ and } k \qquad (4)$$

Numbers which satisfy (4) are alternatively odd and even. The integers which satisy (1) are just the odd-valued solutions of (4). We now show that numbers satisfying (4) are of the form $u(r,s) + krs$. Solving (4) is just a special case of the Chinese Remainder Theorem (see reference 7).

Consider the number $u(r,s)$ of the form of (3) above. Now by Fermat's Theorem (see reference 7) we have $s^{r-1} = 1 \pmod r$, and similarly $r^{s-1} = 1 \pmod s$. Also, of course, $s^{r-1} = 0 \pmod s$, and $r^{s-1} = 0 \pmod r$. Finally, $rs = 0 \pmod r = 0 \pmod s$. Thus $u(r,s)$ satisfies (4).

We now show that numbers not of the form $u(r,s) + krs$ cannot satisfy (4).

Let u and w satisfy (4) and consider the difference

$$u - w = 1 \pmod r - 1 \pmod r = 0 \pmod r = kr$$

$$= (s - 1) \pmod s - (s - 1) \pmod s = 0 \pmod s = k'r$$

for some k and k'. Thus $u - w$ is a multiple of the lcm(r, s), which is rs since r and s are prime. Since $u(r,s)$ satisfies 4 u and w must be of the form $u(r,s) + krs$. #

Finding $u(r,s)$ and hence p_0 requires two exponentiations at a cost of $2T_{exp}$. Finding p amounts to finding a prime in $(p_0 + 2krs)$-space and the same considerations apply in this case as did to the search for r in $(2Lt + 1)$-space, namely that we should start with p_0 and add successive multiples of $2rs$ until the desired size is reached, then check for primality at each subsequent addition.

Size of p, r, s and t: The time spent searching for primes dominates. We need to search for p, of n bits, and for t, r and s, each of roughly $n/2$ bits. Altogether, ignoring all times except those spent searching for primes, the time spent to find p will average:

$$T_{prime}(n) + 3T_{prime}(n/2) = 19/16 T_{prime}(n)$$

$$= 1.19 \times 0.07\, cvTn^4/w.$$

This represents an increase of only 3/16 (=19%) over the time to find a random prime of given size n bits.

An implementation by Gordon

Using a 1 MHz clock microcomputer (Apple II) with an efficient arithmetic package (CyMAS) Gordon found t with 81 decimal digits, r with 84 decimal digits and s with 84 decimal digits which gave an RSA key with 336 decimal digits (1116 bits) in a few hours.

Ueli Maurer's Algorithm

At Eurocrypt 1989 Ueli Maurer of Switzerland's ETH gave details of a proposed scheme which uses smaller primes to first construct intermediate primes to use to find primes for the RSA algorithm. No implementation details are currently at hand but this method also promises to be a fast, reliable generator of strong primes.

References

[1] W. Diffie and M. E. Hellman, "New directions in cryptography", *IEEE Trans.*, 1976, **IT-22**, pp 644-654.

[2] R. L. Rivest, A. Shamir and L. Adelman, "A method for obtaining digital signatures and public key cryptography", *Commun. ACM.*, 1978, **21**, pp 121-126.

[3] D. E. Knuth, *The Art of Computer Programming, Vol 2, Semi-numerical Algorithms,* (Addison-Wesley, 1982, 2nd edn.).

[4] D. M. Burton, *Elementary Number Theory,* (Allyn and Bacon, 1980).

[5] J. A. Gordon, "Strong RSA keys", *Electronics Letters,* 1984, **20, No 12,** 514-516.

[6] W. B. Muller, "Permutation polynomials and public key cryptosystems," *Proc. Eurocrypt -84,* Paris, April, 1984.

[7] J. Seberry and J. Pieprzyk, *Cryptography: An Introduction to Computer Security,* (Pre