HOW TO EXCHANGE (SECRET) KEYS

by

M. Blum

Memorandum No. UCB/ERL M81/90

13 November 1981

HOW TO EXCHANGE (SECRET) KEYS

by

Manuel Blum

Memorandum No. UCB/ERL M81/90

13 November 1981

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Preliminary version: submitted to the 23rd Annual Symposium
on Foundations of Computer Science to be held in San Francisco, May 1981

HOW TO EXCHANGE (SECRET) KEYS

Manuel Blum*

Department of Electical Engineering and Computer Sciences
University of California at Berkeley 94720
Telephone: (415)642-1662 or 642-1024
November 13, 1981

This paper presents the "Exchange of Secrets" problem and offers a protocol to solve it. Heuristic arguments are presented to explain the mechanism of the protocol and to suggest why the protocol should prove difficult to foil.

INTRODUCTION:

We show how Alice and Bob, two parties with EQUAL COMPUTING CAPABILITY and EQUAL KNOWLEDGE OF ALGORITHMS, can trade secrets WITHOUT THE USE OF AN INTERMEDIARY, WITHOUT EVEN OCCASIONAL NEED FOR A JUDGE, and with a NEGLIGIBLE PROBABILITY OF CHEATING.

Suppose Alice and Bob each have their own number, each number being a product of two large randomly chosen primes. Each of the two parties knows the prime factors of his or her own number. Each knows the other person's number but not its factorization. This paper shows how Alice and Bob can exchange their secret prime factors even when neither trusts the other. For example, if one of the numbers has been improperly selected, say is a product of three rather than two primes, this protocol hollers "cheat" before any secrets are revealed. The prime factors may be viewed as the secret keys to the RSA public-key encryption system

*Research sponsored in part by NSF grant MCS 79-03767.

[10].

This scheme might be useful in signing contracts electronically and sending certified electronic mail, ie., mail in exchange for a receipt. In both cases, the problem is to achieve the equivalent of simultaneous PARALLEL exchange of messages using sequential SERIAL communication.

In another paper along these lines, Blum and Rabin [3] give a different method to send certified mail. Their scheme makes no assumption that the two participants have equal computing capability or equal knowledge of algorithms. It can pit the vast capabilities of Professor D. H. Lehmer on the CRAY 1 (the largest currently available computer) against a staff secretary on the VAX (an ordinary computer), giving neither an advantage over the other. Their scheme, however, does allow a probability of cheating of the order of 1/100. The scheme offered here, on the other hand, can easily ensure a negligible probability of cheating of order $1/2^{100}$.

This paper ASSUMES, as does the above scheme, that factoring is hard. This means that a program cannot factor a number that is a product of two 80-(decimal)digit primes in reasonable time (not even 5 years) using the most advanced available technology (1000 CRAY-1's working in parallel) except for a negligible fraction (one in Avogadro's number) of such numbers.

Our scheme builds on and greatly extends the powerful OBLIVIOUS TRANSFER first proposed and used by Rabin [7].

THE PROBLEM:

Alice has a number na that is a product of two primes p1a and p2a. She keeps the two primes secret but gives their product na to Bob. Similarly, Bob has a number nb that is a product of two primes p1b and p2b. He keeps the two primes secret but gives their product nb to Alice. Alice and Bob would like to exchange their primes, p1a for p1b. Once an exchange takes place, each can test whether or not he or she received the other person's prime since na and nb are public.

One obvious approach to the exchange of secret primes, p1a for p1b, would be for Alice and Bob to trade p1a for p1b bit by bit. This has two problems:

1. A minor problem: if Alice goes first, Bob will be a bit ahead of her. If after Bob gets all the bits of his prime, he stops communicating with Alice, then Alice will be out one bit. This is minor because Alice can always try setting the missing bit (or bits) to 0 or to 1, testing each guess by dividing the number guessed into nb. In this way she will easily make up the loss of a bit.
2. A serious problem: neither Alice nor Bob can test the number they are receiving until all or most of it has been transmitted, at which point one of them may discover that the other cheated: perhaps Alice sent p1a to Bob, but Bob sent junk to Alice.

We show how the two can exchange their keys bit by bit in a way that neither can defraud the other. With each bit goes a PROOF that the bit is good.

APPLICATIONS:

This scheme might be used to sign contracts simultaneously. Alice constructs a product na of two large primes and Bob similarly constructs nb. Alice sends Bob her signed contract; it contains a clause asserting that the contract is valid only if Bob knows how to factor "na." Bob sends a copy of the same contract under his own signature to Alice; a corresponding clause says that the contract is valid only if Alice knows how to factor "nb." Once Alice and Bob have read each others contracts and verified each others signatures, they exchange factors.

Alice might use this scheme to send a disclosure D to Bob and get a receipt. Alice constructs na and sends Ena(D) to Bob, where Ena is Alice's public-key RSA encrypter [10] based on the product na. Bob constructs and sends nb to Alice, together with his written agreement that if Alice can factor nb, then Bob must have factored na and therefore he got whatever document D is locked inside Ena(D). Next they exchange factors.

A PROTOCOL FOR TRADING KEYS:

The protocol below is for Alice. Bob's protocol is essentially identical. Both Alice and Bob can test if an 80-digit number is prime using one of the efficient algorithms for primality of Gary Miller [6], Strassen and Solovay [11], or Rabin [9]. Each of them constructs his or her number by generating two 80-digit primes at random, i.e., repeatedly selecting 80-digit numbers at random and testing them for primality until two primes are found, then multiplying them together to form the number.

A1: Send na to Bob and (then) obtain nb.

Test if nb is even, prime, or a nontrivial integer power, $nb = m^i$ for integers $m,i > 1$ (these conditions can easily be checked in polynomial time). If so, Bob has cheated: stop.

A2: Select 100 numbers a1, a2, ..., a100 at random from Znb/2 = {1,...,[nb/2]}, where [nb/2] is the largest integer less than nb/2.

In the event that one of these numbers splits nb, stop: If nb has more than two prime factors, Bob cheated. If nb has two factors and both are prime, Alice has the desired result, but under our assumptions, this happens at most once in Avogadro's number.

In the event none of a1, a2, ..., a100 splits nb, compute and send $a1^2$ (mod nb), ..., $a100^2$ (mod nb) to Bob. Obtain $b1^2$ (mod na), ..., $b100^2$ (mod na) from Bob.

A3: Compute the four square roots mod na of each number bj^2 (mod na) received from Bob: since the prime factors of na are known (by Alice), this can be done efficiently with standard techniques, eg., see LeVeque [5] or [footnote 1]. For each j from 1 to 100, order the four square

---

[footnote 1]: Computing the squareroots of x^2 mod n for $n = p*q$ is particularly easy if p and q are congruent to 3 mod 4. (Alice and Bob could agree to choose their primes this way; by a deep theorem of de la Vallee Poisson, cf., leVeque [5], approximately half of all primes of any given length are congruent to 3 mod 4, the other half to 1 mod 4.) In that case, $x = (xp*qv + xq*pu) \bmod n$, where $xp = (x^2 \bmod n)^{(p+1)/4} \bmod p$, $xq = (x^2 \bmod n)^{(q+1)/4} \bmod q$, and $p*u + q*v = 1$. Whether or not the prime factors of n are congruent to 3 mod 4, there is an efficient probabilistic algorithm to compute squareroots mod n when the prime factorization of n is known, cf., Adleman et. al. [1] or Berlekamp [2]. Incidentally, the prime factorization of n is NECESSARY for computing square roots mod n: any probabilistic algorithm capable of computing a square root of x^2 mod n for a substantial fraction of integers x that are relatively prime to n can be used to split n efficiently: select random integers x relatively prime to n; for each x compute x^2 mod n; use the algorithm to try and obtain a square root y of x^2 mod n such that $x \neq \pm y \bmod n$; once such x,y are found, gcd (x+y,n) splits n.

roots of bj^2 (mod na) by size using ordinary integer < ; delete the largest two of the four roots. (Comment: the largest two roots are the negatives mod na of the smallest two roots. The smallest two roots lie in the set Zna/2.) Denote the smallest two square roots of bj^2 mod na arbitrarily by sqrt1(bj^2 mod na) and sqrt2(bj^2 mod na).

A4: Exchange bits of Alice's 200 (smallest) square roots for Bob's 200 square roots, starting with the least significant bits first: Alice sends the least significant bits of her 200 square roots (the 200 she computed) to Bob. Then Bob sends the least significant bits of his 200 square roots to Alice. Then Alice sends Bob the next-to-least significant bits of her 200 square roots, and so on. Any time the other seems not to be responding or to be responding slowly, a subroutine called SPLIT is invoked to try and split the other person's number.

More formally, Step A4 is done as follows:

```
BEGIN
   FOR i = 1 to |n|, |n| = max{length (na), length (nb)}.
   DO:
      FOR j = 1 to 100,
      DO:
         Send the ith least significant bit of sqrt1(bj^2 mod na)
         and sqrt2(bj^2 mod na).
      OD
      Until all 200 ith bits of sqrt1(aj^2 mod nb) and sqrt2(aj^2 mod nb)
      are received from Bob, apply subroutine "SPLIT(nb)" below.
      When the 200 bits arrive, proceed as follows:

      FOR j = 1 to 100,
      DO:
         Check that the i least significant bits of either
         sqrt1(aj^2 mod nb) or sqrt2(aj^2 mod nb),
         coincide with the i least significant bits of aj.
         If not, Bob is cheating: stop.  Otherwise continue.
      OD
   OD
END
```

SPLIT(nb): Select k from $\{1,\ldots,|n|\}$ at random.
Extend whichever of sqrt1(ak^2) or sqrt2(ak^2)
is different from ak to a number of length $|n|$, selecting the
missing bits at random.
Compute gcd((sqrt1(ak^2)) + (sqrt2(ak^2)), nb) for
each such extension. If and when this subroutine splits nb,
halt the entire protocol.

WHY THIS WORKS:

In Step A4, Alice gradually exchanges one hundred pairs of numbers with Bob. Of the hundred pairs of numbers she obtains from Bob, Alice knows beforehand exactly one number in each pair. To split nb, assuming nb is a product of two distinct odd primes, Alice need only obtain the missing number in ANY one pair. If x and y are two numbers in a pair, i.e., x = sqrt1 and y = sqrt2 are the two smallest square roots of a quadratic residue x^2 mod nb, then gcd (x+y, nb) will be a nontrivial factor of nb. More generally, if nb = $(p1b)^e$ * $(p2b)^f$ is a product of two odd prime powers, and if ai is relatively prime to nb, then gcd(sqrt1(ai^2) + sqrt2(ai^2), nb) = $(p1b)^e$ or $(p2b)^f$. This gives Alice ONE prime power factor, say $(p1b)^e$, of nb. From knowledge of $(p1b)^e$, she can obtain the prime p1b in polynomial time.

Bob cannot cheat Alice by taking nb to be even, a prime or a power of a prime, else Alice will catch him cheating at the beginning of the protocol. The number nb that Bob sends Alice must therefore be a product of two or more odd prime powers.

Suppose nb is a product of exactly two prime powers. Bob can cheat Alice ONLY by guessing, for each of the one hundred quadratic residues (squares mod nb) that he initially receives from her, which square root Alice already knows (whereupon he can send that number together with a

junk number). His chances of guessing correctly are just $1/2^{100}$ [footnote 2].

Suppose Bob's number nb is a product of k distinct odd prime powers. While k is supposed to be 2, Bob might try to cheat by taking k > 2. In that case, his chances of cheating successfully will again be at most $1/2^{100}$: this is because Alice expects two square roots from Bob for each quadratic residue she initially sent him, and one of these must be the one that Alice selected to start. If k = 2, each quadratic residue has just two square roots in Znb/2. If k > 2, however, each quadratic residue has at least four square roots [footnote 3]. In that case, Bob's chances of guessing two roots that include the square root Alice started with, and of doing this correctly for each of the 100 quadratic residues, are at best $1/2^{100}$.

If Bob stops communicating with Alice at some point, he will have at most one more bit of square root than Alice. The expected time for Alice to factor nb will therefore be at most twice the expected time for Bob to factor na. The quality of this protocol depends in part on the standard deviation in the time to factor. Only the SPLIT subroutine depends on current knowledge of algorithms. Assuming that both Alice and Bob use the same SPLIT routine (not necessarily the one given here), the EXPECTED TIME and STANDARD DEVIATION to factor the numbers will be

---

[footnote 2]: It would suffice to use 80 quadratic residues instead of 100 since $1/2^{80}$ is less than 1 in Avogadro's number, my own personal measure of a "virtual zero." 100 is used to distinguish from 80 in 80-digit primes.

[footnote 3]: If nb is a product of k distinct odd primes, each raised to some positive integer power, $nb = p1^{e1} * \ldots * pk^{ek}$, and if ai is relatively prime to nb, then ai^2 mod nb has exactly 2^k distinct square roots, cf. LeVeque [5].

the same for both parties when both have the same number of bits. This is what is meant in the introduction by the statement that both Alice and Bob have the same knowledge of algorithms. The difference of 1 bit makes Alice's expected time to factor Bob's number just half Bob's expected time to factor Alice's number.

REFERENCES

1. L. Adleman, K. Manders, G. Miller, "On Taking Roots in Finite Fields," 18th Annual IEEE Symposium on Foundations of Comp. Sci. (1977), 175-177.

2. E.R. Berlekamp, "Factoring Polynomials over Large Finite Fields," Math. of comp., Vol. 24 (1970), 713-735.

3. M. Blum and M.O. Rabin, "Mail Certification by Randomization," to appear.

4. W. Diffie and M.E. Hellman, "Privacy and Authentication: An Introduction to Cryptography," Proc. IEEE, vol. 67 no. 3 (March 1979), 397-427.

5. W.J. LeVeque, "Fundamentals of Number Theory," Addison-Wesley Pub., 1977.

.6 G.L. Miller, "Riemann's Hypothesis and a Test for Primality," J. Comput. and System Sci. vol. 13 (1976), 300-317.

7. M.O. Rabin "How to Exchange Secrets by Oblivious Transfer," Harvard Center for Research in Computer Technology, TR-81.

8. M.O. Rabin "Transaction Protection by Beacons," Harvard Center for Research in Computer Technology, TR-81.

9. M.O. Rabin, "Probabilistic Algorithm for Testing Primality," J. Number Theory, vol. 12 (1980), 128-138.

10. R.L. Rivest, A. Shamir, and L.L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Commun. ACM, vol. 21 (1978), 120-126.

11. R. Solovay and V. Strassen, "A Fast Monte-Carlo Test for Primality," SIAM J. Comput. vol. 6 (1977), 84-85.