

## The Diffie-Hellman key-exchange protocol

This protocol allows two users to choose a common secret key, for DES or IDEA, say, while communicating over an insecure channel (with eavesdroppers).

The two users agree on a common large prime  $p$  and a constant value  $a$ , which may be publicly known and available to everyone. It is best if the smallest exponent  $e > 0$  for which  $a^e \equiv 1 \pmod{p}$  is  $e = p - 1$ , but the protocol will work if  $e < p - 1$  provided  $e$  is still large.

Alice secretly chooses a random  $x_A$  in  $0 < x_A < p - 1$  and computes  $y_A = a^{x_A} \pmod{p}$ . Bob secretly chooses a random  $x_B$  in  $0 < x_B < p - 1$  and computes  $y_B = a^{x_B} \pmod{p}$ .

Alice sends  $y_A$  to Bob. Bob sends  $y_B$  to Alice. An eavesdropper, knowing  $p$  and  $a$ , and seeing  $y_A$  and  $y_B$ , cannot compute  $x_A$  or  $x_B$  from this data unless he can solve the Discrete Logarithm Problem quickly.

Alice computes  $K_A = y_B^{x_A} \bmod p$ . Bob computes  $K_B = y_A^{x_B} \bmod p$ .

Then

$$K_A \equiv a^{x_A \cdot x_B} \equiv K_B \pmod{p}$$

and  $0 < K_A, K_B < p$ , so  $K_A = K_B$ .

Alice and Bob choose certain agreed-upon bits from  $K_A$  to use as their key for a single-key cipher like DES or IDEA.

Although this protocol provides secure communication between Alice and whoever is at the other end of the communication line, it does not prove that Bob is the other party. To guarantee that Bob is at the other end, they would have to use a signature system like RSA.

## Discrete Logarithms

The Diffie-Hellman key exchange, the ElGamal public key cryptosystem, the Pohlig-Hellman private key cryptosystem and the Digital Signature Algorithm could all be broken if we could compute discrete logarithms quickly, that is, if we could solve the exponential congruence  $a^x \equiv b \pmod{p}$  easily.

By analogy to ordinary logarithms, we may write  $x = \log_a b$  when  $p$  is understood from the context. These discrete logarithms enjoy many properties of ordinary logarithms, such as  $\log_a bc = \log_a b + \log_a c$ , except that the arithmetic with logarithms must be done modulo  $p - 1$  because  $a^{p-1} \equiv 1 \pmod{p}$ .

Neglecting powers of  $\log p$ , the congruence may be solved in  $O(p)$  time and  $O(1)$  space by raising  $a$  to successive powers modulo  $p$  and comparing each with  $b$ . It may also be solved in  $O(1)$  time and  $O(p)$  space by looking up  $x$  in a precomputed table of pairs  $(x, a^x \pmod{p})$  sorted by the second coordinate.

Shanks' "giant step-baby step" algorithm is a meet-in-the-middle method which solves the congruence in  $O(\sqrt{p})$  time and  $O(\sqrt{p})$  space as follows. Let  $m = \lceil \sqrt{p-1} \rceil$ . Compute and sort the  $m$  ordered pairs  $(j, a^{mj} \bmod p)$ , for  $j$  from 0 to  $m-1$ , by the second coordinate. Compute and sort the  $m$  ordered pairs  $(i, ba^{-i} \bmod p)$ , for  $i$  from 0 to  $m-1$ , by the second coordinate. Find a pair  $(j, y)$  in the first list and a pair  $(i, y)$  in the second list. This search will succeed because every integer between 0 and  $p-1$  can be written as a two-digit number  $ji$  in radix  $m$ . Finally,  $x = mj + i \bmod p - 1$ .

There are faster ways to solve  $a^x \equiv b \pmod{p}$ . Here is one method. Choose a factor base of primes (fixed set of small primes)  $p_1, \dots, p_k$ . Perform the following precomputation which depends on  $a$  and  $p$  but not on  $b$ . For many random values of  $x$ , try to factor  $a^x \pmod{p}$  using the primes in the factor base. Save at least  $k + 20$  of the factored residues:

$$a^{x_j} \equiv \prod_{i=1}^k p_i^{e_{ij}} \pmod{p} \text{ for } 1 \leq j \leq k + 20,$$

or equivalently

$$x_j \equiv \sum_{i=1}^k e_{ij} \log_a p_i \pmod{p-1} \text{ for } 1 \leq j \leq k + 20.$$

When  $b$  is given, perform the following main computation to find  $\log_a b$ . Try many random values for  $s$  until one is found for which  $ba^s \bmod p$  can be factored using only the primes in the factor base. Write it as

$$ba^s \equiv \prod_{i=1}^k p_i^{c_i} \pmod{p}$$

or

$$(\log_a b) + s \equiv \sum_{i=1}^k c_i \log_a p_i \pmod{p-1}.$$

Use linear algebra to solve the linear system of congruences modulo  $p-1$  for  $\log_a b$ .

One can prove that the precomputation takes time

$$\exp((1 + \epsilon(p))\sqrt{\log p \log \log p}),$$

where  $\epsilon(p) \rightarrow 0$  as  $p \rightarrow \infty$ , while the main computation takes time

$$\exp((0.5 + \epsilon(p))\sqrt{\log p \log \log p}),$$

where  $\epsilon(p) \rightarrow 0$  as  $p \rightarrow \infty$ .