Diffie-Hellman key exchange

A. The idea

Suppose two people, Alice and Bob [traditional names], want to use insecure email to agree on a secret "shared key" that they can use to do further encryption for a long message. How is that possible? The so-called Diffie-Hellman method provides a way. This method is one of the ingredients of SSL, the encryption package that is part of the Netscape browser.

These notes are a little more detailed than in class, for clarity, but on the exam you are responsible only for doing exercises like Problem 8.2.

B. The mod function

The main ingredient is the "remainder" or "modulo" or "mod" function, denoted `%` in Perl. For example, `25%10` is `5` (say "25 mod 10 is 5") and `25%16` is `9` ("25 mod 16 is 9"). For `n%10`, the result will always be one of `0,1,...,9`.

As you can see, any positive integer modulo 10 is just the last digit in base 10: `1537%10` is `7`, etc. You can think of "modulo 10" for positive integers as meaning "ignore all decimal digits except the last one".

Doing "modular arithmetic" with "modulus" 10 means doing addition, subtraction, and multiplication (including powers) where you only care about the remainder modulo 10. You can use some other modulus m instead of 10, as long as it's the same through the whole problem. It works very smoothly.

The "as often as you want" principle: If you are doing modular arithmetic to find an the answer modulo m, you can take the remainder modulo m as often as you want during the calculations, without changing the answer.

Example 1. To find 1537 x 4248 modulo 10, you could multiply out and take the last digit, but a better way would be to replace 1537 by 7 and 4248 by 8 to start, find 7 x 8 = 56, and then take 56 mod 10 to get 6 as the answer.

A handy standard notation is to write $a \equiv b$ (mod m) if a and b have the same remainder modulo m. This is read "a is congruent to b modulo m". In this notation the example just mentioned looks like this: $1537 \times 4248 \equiv 7 \times 8 = 56 \equiv 6$ (mod 10).

Example 2. Find 2^8 (mod 11).

One solution. $2^8 = 256$; 11 goes into 256 with quotient 23 and remainder 3.

Another solution. Find 2^2, 2^4, 2^8 by squaring repeatedly, but take remainders mod 11 each chance you get: $2^2 = 4$, $2^4 = 4^2 = 16 \equiv 5$, $2^8 \equiv 5^2 = 25 \equiv 3$.

Example 3. Find **all** the powers of 2 up to 2^{10}, each modulo 11.

Solution. Keep doubling, taking remainders modulo 11 whenever possible:

2, 4, 8, $16 \equiv 5$, 10, $20 \equiv 9$, $18 \equiv 7$, $14 \equiv 3$, 6, $12 \equiv 1$ (mod 11). So the answer is 2, 4, 8, 5, 10, 9, 7, 3, 6, 1.

Notice that the powers of 2 run through all possible remainders modulo 11, except 0. We say 2 is a "generator" modulo 11. There is a theorem that if you take a **prime** modulus, then there is always **some** generator, and in fact 2 often works. If 2 doesn't, maybe 3 will.

C. The Diffie-Hellman method

The idea of Diffie and Hellman is that it's easy to compute powers modulo a prime but hard to reverse the process: If someone asks **which** power of 2 modulo 11 is 7, you'd have to experiment a bit to answer, even though 11 is a small prime. If you use a huge prime istead, then this becomes a very difficult problem even on a computer. Steps:

1. Alice and Bob, using insecure communication, agree on a huge prime p and a generator g. They don't care if someone listens in.
2. Alice chooses some large random integer $x_A < p$ and keeps it secret. Likewise Bob chooses $x_B < p$ and keeps it secret. These are their "private keys".
3. Alice computes her "public key" $y_A \equiv g^{x_A}$ (mod p) and sends it to Bob using insecure communication. Bob computes his public key $y_B \equiv g^{x_B}$ and sends it to Alice. Here $0 < y_A < p$, $0 < y_B < p$.

   As already mentioned, sending these public keys with insecure communication is safe because it would be too hard for someone to compute x_A from y_A or x_B from y_B, just like the powers of 2 above.

4. Alice computes $z_A \equiv {y_B}^{x_A}$ (mod p) and Bob computes $z_B \equiv {y_A}^{x_B}$ (mod p). Here $z_A < p$, $z_B < p$.

   But $z_A = z_B$, since $z_A \equiv {y_B}^{x_A} \equiv (g^{x_B})^{x_A} = g^{(x_A x_B)}$ (mod p) and similarly $z_B \equiv (g^{x_A})^{x_B} = g^{(x_A x_B)}$ (mod p). So this value is their **shared secret key**. They can use it to encrypt and decrypt the rest of their communication by some faster method.

   In this calculation, notice that the step ${y_B}^{x_A} \equiv (g^{x_B})^{x_A}$ involved replacing g^{x_B} by its remainder y_B, (in the reverse direction) so we were really using the "as often as you want" principle.

D. Notes (not on final exam)

- It's easy to see why the "as often as you want" principle works for modular arithmetic with positive integers in base 10. In Example 1, imagine doing the multiplication with paper-and-pencil arithmetic, but ignoring everything except the last digit. You get

```
   ...7
 x ...8
-------
   ...6
  ....
 ....
....
-------
......6
```

  In other words, you can multiply and then take the last digit, or you can take remainders early, by saving just the 7 and 8, taking their product, and saving its last digit.
- Congruences work fine for negative numbers if you always use a remainder that is positive or 0; for

example, $-13 \equiv 7 \pmod{10}$ because -20 is a multiple of 10 and -13 is 7 larger. The `%` operation in Perl works this way but the same operation in C and C++ does not.

- The notation $\equiv$ is meant to suggest =, because several properties of $\equiv$ are similar to those of =. For example, $a \equiv b$ and $b \equiv c$ give $a \equiv c$ (all mod m).
- Another interesting fact is that modulo 11, we have $2^{10} \equiv 1$, $3^{10} \equiv 1$, $4^{10} \equiv 1, \ldots, 10^{10} \equiv 1$, and of course $1^{10} = 1$ to start with. More generally, there is a theorem saying that for **any** prime p and for any a from 1 to p-1 we get $a^{p-1} \equiv 1 \pmod{p}$.
- The Diffie-Hellman method works best if $p = 2q+1$ where q is also a prime. (For example, 5 and 11 are prime and $11 = 2 \times 5 + 1$.) Then half the integers 1,2,...,p-1 are generators, and it is possible to check whether g is a generator just by seeing whether $g^q \equiv -1 \pmod{p}$.
- Diffie-Hellman does have a weakness: If an intruder Charlie can intercept and resend email between Alice and Bob, then the intruder can pretend to be Bob for Alice and pretend to be Alice for Bob, substituting his own y_C and tricking each of Alice and Bob into having a shared secret key with him. There are ways to fix this problem.
- The Diffie-Hellman method illustrates the concept of "public-key cryptography", where people can give out public information that enables other people to send them encrypted information.

E. An example

For Diffie-Hellman to be secure, it is desirable to use a prime p with 1024 bits; in base 10 that would be about 308 digits. An example, expressed in hexadecimal, is

p= de9b707d 4c5a4633 c0290c95 ff30a605 aeb7ae86 4ff48370 f13cf01c 49adb9f2 3d19a439 f743ee77 03cf342d 87f43110 5c843c78 ca4df639 931f3458 fae8a94d 1687e99a 76ed99d0 ba87189f 42fd31ad 8262c54a 8cf5914a e6c28c54 0d714a5f 6087a172 fb74f481 4c6f968d 72386ef3 45a05180 c3b3c7dd d5ef6fe7 6b0531c3

z= 56c03667 f3b50335 ad532d0a dcaa2897 a02c0878 099d8e3a ab9d80b2 b5c83e2f 14c78cee 664bce7d 209e0fd8 b73f7f68 22fcdf6f fade5af2 ddbb38ff 3d2270ce bbed172d 7c399f47 ee9f1067 f1b85ccb ec8f43b7 21b4f980 2f3ea51a 8acd1f6f b526ecf4 a45ad62b 0ac17551 727b6a7c 7aadb936 2394b410 611a21a7 711dcde2

To compute with huge integers like these, you need a special "multiple precision" software package, because the built-in arithmetic on computer chips handles only 32 or 64 bits.

F. Solution to the homework problem

Here p=11, g=2, $x_A = 9$, $x_B = 4$. So $y_A = 2^{x_A} = 2^9 \pmod{11}$.

You can find this most easily by finding $2^2 = 4$, $2^4 = 4^2 = 16 \equiv \pmod{11}$, $2^8 = (2^4)^2 \equiv 5^2 = 25 \equiv 3 \pmod{11}$, and finally $2^9 = 2 \times 2^8 \equiv 2 \times 3 = 6$. So $y_A = 6$.

Similary, $2^{x_B} = 2^4 = 16 \equiv 5 \pmod{11}$, so $y_B = 5$.

The secret shared key z_A is the remainder of $y_B{}^{x_A} = 5^9 \pmod{11}$. So find $5^2 = 25 \equiv 3 \pmod{11}$, $5^4 = (5^2)^2 \equiv 3^2 = 9 \pmod{11}$, $5^8 = (5^4)^2 \equiv 9^2 = 81 \equiv 4 \pmod{11}$, $5^9 = 5 \times 5^8 \equiv 5 \times 4 = 20 \equiv 9 \pmod{11}$. As a check, z_B is the remainder of $y_A{}^{x_B} = 6^4 \pmod{11}$. $6^2 = 36 \equiv 3 \pmod{11}$ so $6^4 = (6^2)^2 \equiv 3^2 = 9 \pmod{11}$, which checks. So $z_A = z_B = 9$.

Next Up Previous

Next: About this document **Up:** No Title **Previous:** No Title

Kirby A. Baker
Sun Mar 21 19:36:51 PST 1999