Problem Set 6–Diffie-Hellman and the discrete Log Problem

CSCI3381-Cryptography

Due November 11, 2014

This time you have to do all the problems to get full credit. There are only four, each worth 25 points.

Let's recall briefly what the fundamental algorithms are. In the Diffie-Hellman key agreement protocol, Alice and Bob agree publicly on a large prime p and a primitive element g mod p. Alice generates a secret value $1 \leq x < p$, and Bob similarly generates $1 \leq y < p$.

Alice sends $g^x \mod p$ to Bob, and Bob sends $g^y \mod p$ to Alice. Each of them couples this information they receive with their secret information to compute the shared secret $g^{xy} \mod p$.

All our problems concern the variant of Diffie-Hellman called the *El Gamal public key cryptosystem.* Here, Alice keeps x as her secret key and published $(p, g, g^x \mod p)$ as her public encryption key. Messages are integers m in the range $1 \leq m < p$. To encrypt this message, Bob generates a random $1 \leq y < p$ and sends

$$(\alpha = g^y \mod p, \beta = (g^{xy} \cdot m) \mod p)$$

to Alice. To decrypt, Alice takes the first component α and her secret value x to compute $g^{xy} \mod p$, then computes its inverse $g^{-xy} \mod p$, and multiplies this by the second component β of Bob's message to recover m.

In the problems below, the messages are brief ASCII texts that have been converted, as usual, into sequences of bytes and then into long integers. In Problems 2-4 you should convert the decrypted value back into text. All the parameters and messages are given on the accompanying web page.

Your solution should be a carefully reasoned account of how you solved the problems, together with the fragments of code you used and the ASCII solutions to the decryption problems. Now would be an excellent time to put aside those charming handwritten documents, and photos thereof, and use either LaTeX or the equation editor in Microsoft Word. You should also include a `.py` file that contains all the code, and which, when run, generates all the answers.

1. Verify that p is prime and that $g = 5$ is a primitive element mod p. You can use `mrpt.py` to verify that it is prime. To show that g is a primitive element, you should verify that $p = 2q + 1$ for a prime q and verify that $g^q \mod p \neq 1$ and $g^2 \mod p \neq 1$ (and be able to explain why this shows that g is primitive.

2. Alice's public and private parameters and Bob's message pair (α_1, β_2) are given on the Web page. Decrypt Bob's message.

3. Bob sends Alice another pair (α_2, β_2). It is later revealed that the message sent was

```
Now my charms are all o'erthrown and what strength I have's mine own.
```

Subsequently, Eve intercepts yet another message (α_2, β_3), and notices that it has the same first component as the previous message. This is because Bob made the fatal error of reusing his random value y. Decrypt this new message.

4. I did not give you just any old pair (p, g). This is the pair used by the current Discrete Log recordholder. The website also contains the value $\alpha_4 = g^y \bmod p$ whose discrete log was successfully computed, and the discrete log y itself. Eve intercepts Bob's communciation (α_4, β_4) to Alice, and uses this priceless discrete log information to decrypt the message.

You are assuming here that the value whose discrete log was known is sent as the first component of Bob's message to Alice. What would be the result if this value were posted as the first component of Alice's public key?

The moral of problem 3, if you didn't catch it, is that Bob cannot reuse his secret value y. The moral of problem 4 is that p must be large enough to beat the best technology for computing discrete logs. In this example, p has 180 decimal digits, roughly 600 bits. For this reason, 1024 bits is considered an appropriate size.