Institut for Matematik og Datalogi
Syddansk Universitet

March 22, 2011
JFB

Cryptology – F11 – Week 8

Announcement

I have graded your second assignment.

Lecture, March 15

We finished with chapter 4, skipping the subsection on the Merkle–Damgård construction. We covered SHA–256 and then the last two sections of chapter 4. Then we returned to chapter 7, covering up through section 7.4.2.

Lecture, April 4

We will discuss subliminal channels and begin on chapter 8, covering up through section 8.2 and section 8.4.

Lecture, April 7

We will cover sections 8.3 and 7.6 (the latter from some notes).

Problem session April 11

1. Do problem 4.12. For part (b), you can find a (1,1)-forger. Skip the difficult case mentioned.

2. Let p be an odd prime and g_0 and g_1 be generators of Z_p^*. Consider the following two functions: $f_0(x) = g_0^x \pmod{p}$ and $f_1(x) = g_1^x \pmod{p}$. Use these two functions to create a hash function which will hash an arbitrary length message down to a value in Z_p^*. Can you make it secure under the assumption that the discrete log problem is infeasible?

3. Do problem 6.21 in the textbook.

4. Do problem 7.1 in the textbook. (You might want to look at the notes on the course home page on number theory to recall how to solve linear congruences.)

5. In the discussion of the Schnorr signature scheme on page 286, it says that to find a qth root of 1 modulo p, one should begin with a primitive element α_0 of Z_p^* and compute $\alpha_0^{(p-1)/q}$. (Recall that p and q are both primes.)

    a. Why is this correct? What subgroup does the result generate?

    b. How long does it take to do this computation?

    c. Is it necessary that α_0 be a primitive element?

Assignment due Wednesday, April 27, 14:15

Note that this is part of your exam project, so it must be approved in order for you to take the exam in June, and you may not work with others not in your group. If it is late, it will not be accepted (though it could become the assignment you redo). You may work in groups of two or three.

Write a program which implements part of the index calculus algorithm for finding discrete logarithms modulo a prime. Your program should create congruences such as those in section 6.2.4 of the textbook with

$$\alpha^{x_j} \equiv p_1^{a_{1j}} p_2^{a_{2j}} \cdots p_B^{a_{Bj}}.$$

Your program should also choose primes of different lengths at random (see the Java class BigInteger) and make sure that you have a generator.

Recall that one can find a prime p where $p-1$ is factored by first choosing the factors of $p-1$ and then checking if p is prime. A reasonable way to do this is to first choose the length of $p-1$. Then randomly choose the length of the first prime factor (between 2 and the length of $p-1$ minus 1. Find a prime p_1 of this length. This determines the length of $(p-1)/p_1$. Continue, randomly finding factors of this.

You should test your program using different size factor bases and different size primes. There should be a trade-off here.

To deal with the long numbers necessary, you may use Java, there is a class in java.math called BigInteger which should be efficient and easy to use. There is documentation available at

http://java.sun.com/javase/6/docs/ api/.

You may use the standard methods provided there.

Please turn in your program, some output and a report. You should send me your program and any extra files via e-mail (they can just be attachments in `pine`).

If you prefer another language to Java, please come talk with me by Thursday, April 7 (preferably earlier).