Institut for Matematik og Datalogi
Syddansk Universitet

March 11, 2008
JFB

Cryptology – F08 – Week 8

Lecture, March 7

We covered chapter 6.

Lecture, March 13

We will cover the McEliece Cryptosystem (copied from the earlier edition of the textbook), introduce digital signatures from chapter 7, and start on chapter 4 (skipping the subsection on the Merkle–Damgård construction).

Lecture, April 7

We will first cover SHA–256 and then cover the last two sections of chapter 4. Then we will return to chapter 7, covering up through section 7.4.2.

Lecture, April 11

We will begin on chapter 8.

Problem session April 10

1. Finish any problems from the last set which we haven't finished.
2. Do problem 4.1 in the textbook. For part (d), use the fact that the left-hand side in (c) is at least zero.
3. Do problem 4.6.
4. Do problem 4.12. For part (b), you can find a (1,1)-forger. Skip the difficult case mentioned.

5. 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?

6. Do problem 6.21 in the textbook.

7. 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.)

8. In the discussion of the Schnorr signature scheme on page 286, it saysthat 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 Thursday, April 17, 12:15

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.

A slightly different (from the one I presented in class) version of the polynomial time test for primality is given in the paper by the original authors, which can be found at:

http://www.cse.iitk.ac.in/users/manindra/algebra/primality_v6.pdf.

Write a program which implements this algorithm and compare the performance of the algorithm to that of the Miller-Rabin primality test. In your report, include a brief explanation of some of the intuition behind the algorithm (not a complete proof of correctness) and an explanation of how you implemented the various parts of the algorithm. Explain how you did your testing (comparison to Miller-Rabin) and the results you obtained.

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`). But, in addition to the e-mail, I would like printed copies of everything.

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