KTH Computer Science and Communication

Homework I, Foundations of Cryptography 2007

Due on Feb 14 at 08.15. The general rules on homework solutions available on the course home-page apply. In particular, discussions of ideas in groups of up to at most three people are allowed but implementation should be done individually. Note that the total of the current homework (even without bonus points) is 117 points. As the thresholds for the grades given in the rules for the homeworks remain at the promised levels, please consider the extra 17 points as an opportunity rather than an extra chore.

1 (15p) Solve the cryptogram "unknown1" available on the course home-page. The clear text is in English with character "space". Note that space is handled exactly as any other letter and for instance two adjacent spaces in the cipher-text is very much different from one space.

2 (20p) Make an efficient implementation of AES. Your implementation should be submitted to the tool "Kattis" for automatic evaluation. Details about the input and output format that should be used is available at `http://kattis.csc.kth.se/problem?id=aes`.

If you have not used Kattis before, you need to have registered for the course using the `res checkin krypto07` command in order to get an account (and in this case you probably also want to have a look at the documentation about how to use Kattis, available from the URL above).

To get any points on this problem your implementation has to be accepted by Kattis. Provided this is true and your implementation runs in T seconds your score is $(250/\max(T, 10)) - 5$ rounded up to the closest integer. Note that $T \leq 10$ gives a full score while $T = 50$ gives score 0. Also, in order to get your points, you need to state the Kattis Submission ID of the code that you want to get points for in your handed in solutions.

You may use implementation ideas from the book or from other sources, but if you do then you should include references when you hand in your solution. You are not allowed to simply down-load, copy by hand, or otherwise use somebody else's implementation.

Bonus of 20 and 10 points will be awarded to the two fastest implementations. Only those programs that were turned in on time will be considered for this bonus.

3 (12p) Many cryptosystems like AES use a finite field of the type $GF(2^k)$ for some k. The situation is described for $k = 3$ in the book. We are here interested in the case $k = 4$. The following calculations should be done by hand.

3a Find a representation of the field given by an irreducible polynomial of degree 4.

3b Compute $\alpha\beta$ for $\alpha = 1 + t + t^3$ and all values of β.

3c Solve $t^x = 1 + t + t^3$.

4 (20p) Consider the following 64-bit block crypto. It runs for k rounds and uses a $64k$-bit key partitioned into k 64-bit round keys. Its heart is given by a permutation S of bytes. The 64 bits of the cleartext are loaded into an 8×8 matrix $(m_{ij})_{i,j=1}^{8}$. At each round the contents of the matrix is xored with the round key. After this, in even rounds do the following steps.

1. For $0 \leq i \leq 7$ form bytes m_i by concatenating the bits in a row, i.e., for $0 \leq j \leq 7$ the jth bit of m_i is m_{ij}.
2. Replace m_i by $S(m_i)$ i.e., for $0 \leq j \leq 7$ make the new value of m_{ij} be jth bit of $S(m_i)$.

The odd rounds are analogous replacing "rows" by "columns".
Your task is to analyze this cryptosystem.

4a For what values of k do you think you can break it for any value of the table S? Discuss unknown clear text, known clear text and chosen cleartext.

4b Is there a reasonable value of k for which you would expect it to be secure for a "good" choice of S? In particular how large a value of k would be needed to make linear cryptanalysis inefficient?

4c What method would you prefer to choose S (assuming you are looking to get a secure cryptosystem)? Discuss the two cases of you being in control of the process of choosing S and the case when you are only allowed to look at the result. The essential difference in the two scenarios is if the process uses randomness then in the second case you have to rely on somebody else's random choices.

4d (10 bonus points) Can you come up with a way to construct S to make the system breakable also for large values of k but that it still looks reasonable? Choosing S to be the identity is clearly bad but fools nobody. For a full score your tricky choice of S should have a good possibility to fool an engineer who is not a cryptographer. Can you also make a choice to fool an expert cryptographer, your bonus is doubled.

Solid reasoning giving poor estimates for the value of k is preferred over strong values of k supported by poor reasoning.

5 (20p) In the file "gskriv" on the course home page there is the input and output of the "Geheimschreiber". For details of the machine we refer to the one-page description available on the course home page. Reconstruct the wheels! The file "gskriv2" contains an encryption observed later the same day on the same communication line. Reconstructing the clear-text of this messages gives additional 20 points.

6 (15p) The entropy of a text is a measure of how information is contained in a text and the entropy of an n character long string with independent and uniformly random symbols from an alphabet of size S is $n \log S$. One reason the entropy of an n character long text in English (with spaces) is not $n \log 27$ is that letters have different frequency. An even more compelling reason is that pairs or triplets of letter do not appear with the uniform distribution. Estimate the entropy of English if it had been given by (i) independent letters, (ii) independent pairs of letters (iii) independent triplets of letters, each with the probability observed in true English. All these give upper bounds of the true entropy in English. To be more precise, under (ii) you think of English as written in a 27^2 symbol alphabet were each symbol corresponds to two ordinary letters (including space), and similarly with triplets.

Another way to get an upper bound of entropy of English is to compress typical examples of English text with common data compression programs (try at least two programs). The logic being that if a text can be compressed to t bits then it cannot have more than t bits of entropy.

Compare your upper bounds and try to comment on differences. Can you somehow get a lower bound on the entropy of English?

7 (15p) In RSA one traditionally constructs N as the product of two primes. But sometimes one sees implementations with $N = pqr$ for primes p, q and r. Firstly check that computing e and d can be done (roughly) as before also with three primes.

The advantage of using more primes is that it can speed up decryptions (and signatures). Please describe how. Do you get additional advantages of using more primes? What limits the number of primes to use?

Can you speed up encryptions?