TMA4155 — Exercise 8

Kristian Gjøsteen

2006-10-16

NB! This exercise will be in the computer lab.

The numbers used in this exercise are available in the Maple worksheet `numbers08.mws`.

Task 1 — Attacking RSA

In this task, the public key is (`t1_n`, 7).

a. We have a ciphertext `t1_c1`. You have tricked the owner of the public key into decrypting the ciphertext `t1_c2` under his secret key, and the decryption is `t1_m2`. Find the decryption of `t1_c1` and verify that it is correct.

Hint: $\texttt{t1\_c2} \equiv 10000000\,\texttt{t1\_c1} \pmod{\texttt{t1\_n}}$, and $10000000 = 10^7$.

(This is a variant over Task 4 on the test exam for 2002.)

b. You have tricked the owner into giving you the decryption of a number of small primes `t1_l[i]` as `t1_m[i]`. Decrypt the ciphertext `t1_c`.

Hint: If x has large prime factors, then perhaps $x + kn$ only has small prime factors, for small values of k, such as -2, -1, 1, 2.

Maple hint: `ifactor(x,easy);` can be used to find all the small prime factors of `x`.

Task 2 — RSA with padding

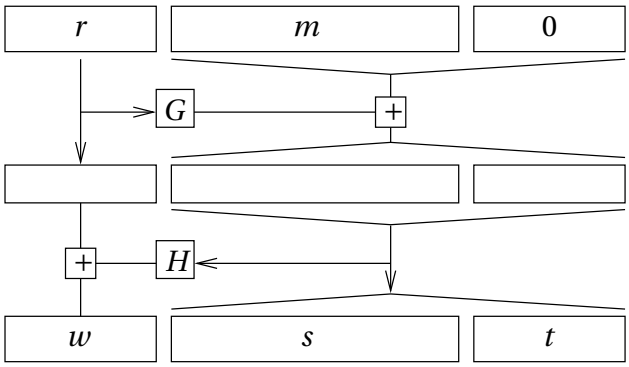
As we have seen, straight RSA has numerous security problems. One way to protect against such security problems is with a *padding scheme*. A padding scheme consists of two algorithms encode and decode. encode is a probabilistic algorithm that takes as input the message and outputs the padded message. decode is a deterministic algorithm that takes as input the padded message and outputs the message itself, or an error.

We are going to look at one possible padding scheme to use when encrypting with RSA. (To simplify computations in Maple, this padding scheme operates on decimal numbers, not on bit strings.)

The message m will have $l = 80$ decimal digits. We shall use $k = 35$ digits for random padding and checksum.

The primes p_1 and p_2 are given as `t2_p1` and `t2_p2`. Define the functions $G' : \mathbb{Z} \to \{1, \ldots, p_1\}$ og $H' : \mathbb{Z} \to \{1, \ldots, p_2 - 1\}$ til å være $G'(x) = (-730101010101010101)^x \bmod p_1$ og $H'(x) = 2^x \bmod p_2$. Then we define $G(x) = G'(x) \bmod 10^{k+l}$ and $H(x) = H'(x) \bmod 10^k$.

The encode algorithm is a Feistel-like construction:



The additions are done modulo 10^{l+k} and 10^k, respectively. For every padding, a new r is chosen uniformly at random from $\{0, \ldots, 10^k - 1\}$.

Note: This padding scheme is extremely inefficient and may very well be insecure. To get efficiency and security, better G and H functions must be chosen.

a. Give a diagram for the corresponding decode algorithm.

b. Choose a message m and apply the padding scheme. Do it again with a new r. Does the result change? Extract the message from both paddings to verify that the padding works.

c. Make a small change to a padded message (you can for example change a digit). Extract the message. How could you detect that it has changed?