Homework sheet 2, due 12 January 2017 at 13:45

Submit your homework by encrypted and signed email. This time a different member of your group should handle the submission; ideally you should cc your group members. Do not forget to attach your public key if I don't have it yet.

1. Alice is a web merchand offering encrypted connections using semi-static DH in $\mathbb{F}_{103}^*$ in the subgroup of order $\ell = 51$ generated by 2.

   For this exercise you should not use your computer for more functions than a pocket calculator offers you; in particular make sure to give full details when computing inverses and exponentitations.

   (a) Verify that 2 has order 51, justify your computation and try to use not too many multiplications and squarings.

   (b) Alice's public key is $h_A = 30$. Use the baby-step giant-step algorithm to compute an integer a between 0 and 50 so that $g^a = h_A$, i.e. compute the discrete logarithm of Alice's key. Solutions using brute-force search for a will not be accepted. Make sure to verify your result by computing g^a.

2. In SSLv3 one of the two options for symmetric encryption is DES in CBC mode. To protect against message forgery a message authentication code MAC is used. SSLv3 uses the MAC-then-encrypt approach, thus a message m first gets encoded as $M = m||\text{MAC}(m)||\text{pad} = M_1 \dots M_{\ell-1} M_\ell$ and then encrypted using DES with CBC. The padding pad is chosen so that the total length of M in bytes is a multiple of 8 (to match the block size of DES) and that the last byte states the length of the padding (including this byte) in bytes. Note, the latter means that there always has to be a padding, even if $m||\text{MAC}(m)$ has length a multiple of 8. There are no further requirements on how the padding is chosen. Upon receiving a ciphertext C, a computer will decrypt the message M, read the last byte to learn the length of the padding to identify m and MAC(m), and finally verify the MAC. If this verification fails the computer will close the connection.

   (a) Just as a reminder of how CBC works, write how you decrypt the last block of the ciphertext.

   (b) Assume that $C = C_0 C_1 \dots C_{\ell-1} C_\ell$ is a ciphertext so that the C_ℓ block comes entirely from the encryption of pad. The first block C_0 contains the IV. What is the value of the last byte in M_ℓ? Show how this gives you a method that for each $0 < i < \ell$ you can test whether the last byte of M_i matches a publicly available value (computed from the C_i).

   To give a concrete example let C_0 =`01 23 45 67 89 AB CD EF`, $C_{\ell-1}$ =`12 34 56 78 9A BC DE F0` (in hex) and (like above) let C_ℓ come entirely from padding. What value of the last byte of M_1 can you test for?

3. In RC4 we need to swap two states. This is easiest to do using an extra variable, i.e. we copy $S[i]$ to `dummy`, copy $S[j]$ to $S[i]$ and finally copy `dummy` to $S[j]$. To save on storage space one might have the idea to implement the swap in the following three steps:

   (a) $S[i] \leftarrow S[i]$ xor $S[j]$

   (b) $S[j] \leftarrow S[i]$ xor $S[j]$

   (c) $S[i] \leftarrow S[i]$ xor $S[j]$

   Explain first why this usually computes the correct $S[i]$ and $S[j]$. Now assume that this piece of code does the swap in the second part of the code (after the key setup). Explain why this can go wrong and state (with explanation) the expected number of steps until this goes wrong for the first time. Explain what happens long term with this implementation. Note that there are multiple possibilities of what happens. I don't expect a full analysis.

4. This exercise expects you to brute force RC4 at "export-cipher" strength (40 bit keys). Through some side-channel information you learn that this key was set up for 2WF80 and that the first byte `key`[0] = 80. Find a key that could have produced the following output sequence:

   130, 189, 254, 192, 238, 132, 216, 132, 82, 173.