Cryptography

Lab 2, 3 IV 2017, max 10 pts.

Problem A.1 (3 pts) Implement a program which encrypts/decrypts selected file(s) on disk. The program takes as inputs:

- mode of encryption, at least: OFB/CTR/CBC... (it has to support `AES_cbc_encrypt`, you can use *openssl*),
- path to a keystore,
- key identifier.

Password to the keystore has to be read from a config file or from a command line.

Prepare unit tests for each supported mode of encryption.

The program needs to support two modes:

encryption oracle on input consisting q messages: $\langle m^1, \ldots, m^q \rangle$ it returns it ciphertexts.

challenge – on input m_0, m_1 your program picks independently, uniformly at random a bit b and returns a ciphertext c_b of a message m_b.

Problem A.2 (7 pts) Implement a CPA-distinguisher which is capable of winning a CPA-experiment with probability 1 a modified version of `AES_cbc_encrypt`.

You may assume that the program from the previous program generates consecutive IVs by incrementing its value by 1, each time it is run.

You can achieve this by modifying the value *ivec* in (*include/openssl/aes.h*):

void *AES_cbc_encrypt*(const unsigned char **in*, unsigned char **out*,
size_t *length*, const AES_KEY **key*,
unsigned char **ivec*, const int *enc*);

Problem B (5 pts) Write a program and try to find which of the DES' S-boxes are the most linear.(The definition of the DES can be found here `http://www.itl.nist.gov/fipspubs/fip46-2.htm`).

You need to compute the probability that the following equation holds:

$$S_i(z) \oplus S_i(z \oplus x) = y,$$

for all $i = 1, \ldots, 8$, $x \in \{0,1\}^6$ i $y \in \{0,1\}^4$ and random $z \in \{0,1\}^6$.

Problem C (5 pts) Write a distinguisher for a selected PRNG (f – LCG, glibc). The distinguisher uses as a subroutine a predictor which you implemented for Lab 1. It will be given a bit-string x which comes either from a uniform distribution ($x \leftarrow \{0,1\}^n$) or from an output of the PRNG generator ($x = f(s)$ where s is uniform and you know f). Your program needs to tell in which way x was generated.