Cryptography

Lab 3, 15 IV 2017

Problem A.1 (5 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 (5 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. Alternatively you may assume that each IV is generated by *e.g.,* LCG or glibc random generators.

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);
```