

# Cryptography

## Lab 5

1. 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 (generate a keystore using *e.g.*: `keytool` (<https://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html>),
- 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:  $\langle m^1, \dots, m^q \rangle$  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  $m_b$ .

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

You may assume that the program from the previous problem generates consecutive *IV*s 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 the function `AES_cbc_encrypt()` (see: `include/openssl/aes.h`):

---

```
                    aes.h
55 void AES_cbc_encrypt(const unsigned char *in, unsigned char *out,
56                     size_t length, const AES_KEY *key,
57                     unsigned char *ivec, const int enc);
```

---