Testing - Rypto
April 25, 2017

Table of Contents

Preface

"Tietorakenteet ja algoritmit" – excercise.

Rypto is a software, which can encrypt and decrypt.

Testing arrangements

Unit testing

Unit testing is done with CUnit framework and gradle.

Other tests

The rest of the tests are implemented as Bourne shell scripts. For these tests, Bourne shell or compatible command interpreter, command-line utilities cmp, expr, dd, and special device /dev/zero are needed. In Unix, Mac OS, and Linux environments these are present on most of the installations. In Microsoft Windows, additional tools (e.g. Cygwin) are needed.

Test cases

Unit tests

AES_makeword

The makeword test case is: 0x01, 0x02, 0x03, 0x04 → 0x01020304.

AES_RotWord, AES_SubWord

The RotWord and SubWord test cases were extracted from the standard[FIPS197], pp. 27, first line of the table.

AES_KeyExpansion

The three Key Schedule test cases were obtained from[SAMIAM].

Selected test cases were the following:

- A key with all bits zero.
- A key with all bits one.
- A key with all bytes different.

AES_AddRoundKey, AES_SubBytes, AES_ShiftRows, AES_MixColumns

Test cases were taken from the standard, pp. 33, first possible cases.

AES_Inv*

Test cases were generated from standard version test cases by inverting input and output.

AES_encrypt, AES_decrypt

Test cases were taken from the standard, pp. 35-

Integration tests

Integration tests are located in the directory tests.

The test cases are encryption and decryption using one key, and reference files created with OpenSSL 1.0.2k.

The tests are implemented in a Bourne shell script integration-tests.sh.

The test files are as follows:

File	Description
key	The key used in encryption and decryption.
plain.1	Plaintext file, length 1
plain.10	Plaintext file, length 10
plain.16	Plaintext file, length 16
plain.1506	Plaintext file, length 1506
openssl-options	General openssl options used to generate ciphertext files.
openssl.1	plain.1 encrypted (and padded) w/ OpenSSL
openssl.10	plain.10 encrypted (and padded) w/ OpenSSL
openssl.16	plain.16 encrypted (and padded) w/ OpenSSL
openssl.1506	plain.1506 encrypted (and padded) w/ OpenSSL
openssl.nopad.16	plain.16 encrypted without padding w/ OpenSSL (not used in tests)

Table: Integration test files

The integration tests are run in the following fashion. This is done for each plaintext file.

1. The plaintext file is encrypted with rypto.
2. The resulting ciphertext file is compared with one produced with OpenSSL.
3. If there are differences, a failure is reported.
4. The ciphertext file produced with OpenSSL is decrypted with rypto.
5. If there are differences, a failure is reported.

In the end, number of succesful and failed tests is reported.

Performance tests

Performance tests are located in the directory tests.

Used test cases are files containing 100,000, 1,000,000 and 10,000,000 bytes. Tests are run in such a fashion that encryption and decryption are timed for each file size.

The tests are implemented in a Bourne shell script performance-tests.sh.

How to repeat tests

Unit tests

The static libcunit.a must be linked to directory libs/ in the project root for tests to run.

Say

```
cradle build
```

from the command line. If the cradle tool is not installed, command

```
./gradlew build
```

might work instead.

Integration tests

Integration tests are located on directory tests.

Build the project as in previous chapter.

Run the integration tests by issuing command

```
sh integration-tests.sh
```

in directory tests.

Performance tests

Performance tests are located on directory tests.

Build the project.

Run the performance tests by issuing command

```
sh performance-tests.sh
```

in directory tests.

Test results

Unit tests

Unit tests – passed on both development machine (Mac OS Sierra) and melkki (Ubuntu Linux).

Integration tests

Integration tests – passed on both development machine (Mac OS Sierra) and melkki (Ubuntu Linux).

Performance tests

References

FIPS197: U.S. Department of Commerce/National Institute of Standards and Technology, Federal Information Processing Standard, FIPS PUB 197 Advanced Encryption Standard (AES), 2001
SAMIAM: Trenholme, Sam, Rijndael's key schedule, 2016, http://www.samiam.org/key-schedule.html