Assignment 1 - Files encryption

Security

Jämes Ménétrey
james.menetrey@unine.ch

27 October 2023

1 Assignment instructions

Develop a program with an interface (CLI or GUI) that encrypts and decrypts files using many known-secure algorithm(s).

1.1 Symmetric ciphers

Your program must support multiple block cipher modes of operation:

1. Electronic codebook (ECB)
2. At least one "classic" mode of operation (*i.e.*, CTR, CBC, CFB, OFB)
3. At least one authenticated encryption with associated data (AEAD) mode of operation (*e.g.*, AES-GCM, ChaCha20-Poly1305)

1.2 Asymmetric ciphers

This second part is dedicated to better understand how Web browsers communicate using the HTTPS protocol. Your program is extended to leverage the asymmetric cipher RSA, similarly to the HTTPS requests performed by Web browsers. Since asymmetric ciphers are slower than symmetric ciphers, Web browsers only rely on public-key cryptography to establish a symmetric key. Afterwards, the remaining communication relies on symmetric ciphers. We mimic this practice in this exercise, so you will extend your program to support 3 new features: the generation of keypairs, the asymmetric encryption and decryption of files.

1.2.1 Keypairs generation

Your program must be able to generate asymmetric keypairs (*i.e.*, public and private keys) for RSA, which are stored in two files: the public and private keys files. The resulting files are then used as inputs for the asymmetric encryption and decryption features.

1.2.2 Asymmetric encryption

Your program must support the asymmetric encryption of files using this procedure:

1. Receive a **public** key as an argument
2. Generate a symmetric key for an AEAD cipher
3. Encrypt the body of the file with the symmetric AEAD cipher
4. Encrypt the symmetric key with the **public** asymmetric key
5. Append the encrypted symmetric key and the encrypted body of the file
6. Save this result as the file

As a result, the body of the file is now encrypted symmetrically, and the decryption key for the body is encrypted using the asymmetric public key. Therefore, only the owner of the private key can decrypt the body of the file.

1.2.3 Asymmetric decryption

Your program must also support the asymmetric decryption of files using this procedure:

1. Receive a **private** key as an argument
2. Extract the encrypted symmetric key and the encrypted content from the file
3. Decrypt the symmetric key with the **private** asymmetric key
4. Decrypt the encrypted body of the file using the symmetric key
5. Save the decrypted body as the file

The result of the decryption must be the same file initially encrypted. This means that the symmetric key can be discarded at the end of the decryption process.

For the sake of completeness, **you must implement the textbook RSA algorithm by hand, including the secure generation of the keypairs**.[1]

[1] In the context of this assignment, it is requested to implement *Textbook RSA*, which is the initial specifications of RSA as presented during the lecture. Nowadays, *Textbook RSA* is considered insecure, because various attacks can compromise the security of the plaintext. Therefore, RSA encryption is usually paired with the encryption padding scheme *RSA-OAEP*, which provides additional security measures. Implementing *RSA-OAEP* is optional in this assignment and counts as a bonus.

2 Hand-in

The deadline for this assignment is 11 October at 14:15 (2 weeks).

- To be submitted to Ilias:
  - Source code of *your* assignment (this assignment must be solved alone).
  - Readme file briefly mentioning how to compile and run your program, which dependencies it requires, etc.
  - All the files have to be packed in an archive in a standard format (`zip`, `tar.gz`), named following this exact pattern, in lowercase letters only:
    `security23_as1_<your family name>.<extension>`
    For example, if your name were to be *Homer J. Simpson*, you would use the following filename for this assignment: `security23_as1_simpson.tar.gz`

3 Demonstration

You must demonstrate your solution to the assistant during the exercise session, at the latest during the week that follows the deadline.

It is **mandatory** for each student to demonstrate their submission!

During the demonstration, you will show why using ECB is a bad idea and the advantages of AEAD over "classic" modes of operation.

4 Grading

These criteria will be taken into account to grade your work.

- Hand-in according to the instructions
- Readme file allows knowing how to test your work
- Functionalities of your program
  - Encryption and decryption using ECB
  - Encryption and decryption using with "classic" mode(s) of operation
  - Encryption and decryption using AEAD cipher(s)
  - Generation of keypairs for RSA
  - Encryption and decryption using RSA and an AEAD cipher
  - Implementation of RSA by hand
- Choice of the encryption algorithm(s)
- In-class demonstration
  - Demonstration showing why using ECB is a bad idea
  - Demonstration showing that AEAD is better than "classic" modes of operation
- Software quality (code quality, *etc.*)
- "User friendliness", *e.g.*, help for command line flags, humanly-understandable console output, *etc.*

5 Notes

You can use your favorite programming language for the assignments of this course, as long as it is a programming language readily available on the GNU/Linux or Windows operating systems.[2] The suggested language for this assignment is Python with the PyCryptodome library[3].

[2] You can use any of the languages in the following list. If you want to use another language, please check with the TA. List in alphabetical order: Bash, C, C++, C#, Go, Java, Kotlin, Python, Ruby, Rust, Scala.

[3] `https://pycryptodome.readthedocs.io/en/latest/index.html`