Cybersecurity MSc: Applied Cryptography 2022-23

2nd Project: The speed of digital signatures

November 28, 2022 Due date: January 1, 2023

Changelog

- v1.0 - Initial version.

1 Introduction

Digital signatures can be used for many purposes. We can use them to associate the identity of signers to documents, and we can also use them to authenticate communication endpoints. While in the first case performance is not a critical issue, it the second case it may be so.

When a communication endpoint authenticates itself with a digital signature, it usually has to do so each time a new endpoint establishes an interaction. And the time it takes to create and validate a digital signature has impact in each of the peers, but not in the same way.

The time it takes to create and validate a digital signature depends in the first place on the technology used. Some technologies are naturally faster than others, one reason being the size of the keys they require to get a similar security level. But the algorithms' implementation also has an impact.

In this project we want to evaluate the performance of different signature algorithms implemented in different languages, possibly with different cryptographic libraries.

2 Homework

The work consists in measuring the performance of signature creation and validation algorithms using the following algorithm and setup variations:

- RSA signatures:
  - With 3 different key sizes: 1024, 2048 and 4096 bits.
  - With PKCS #1 and PSS paddings. For PSS use always the same configuration (MGF function, digest function, salt, label, etc.),
- ECDSA (Elliptic Curve Digital Signature Algorithm):
  - With 3 different curves types (e.g. NIST P, K(oblitz) and B curves);
  - With 3 different curves allowing small, medium and large keys.

Consider the same digest function (e.g. SHA-256) to compute and validate all the signatures. **Do not include the time to generate the digest of the signed data in the performance figures**.

With these variations, and considering signature generations and validations separately, you should compute $2 \times (3 \times 2 + 3 \times 3) = 30$ performance results per library.

Regarding the use of libraries, select only one for each of the following languages: C (or C++), Java and Python.

In C, you can use these library implementations:

- The OpenSSL crypto library;
- The Nettle library.

In C++, you can use this library implementations:

- The Crypto++ library.

In Java, you can use these library implementations:

- The Java Runtime Environment;
- IAIK JCE;
- Bouncy Castle Crypto Library.

In Python, you can use these library implementations:

- Cryptography;
- PyCrypto.

Use only one computer for all the evaluations, and don't forget to mention its characteristics (CPU model, clock frequency, etc.) in the report. You can use any operating system, but use always the same to reduce sources of entropy.

2.1 Performance evaluations

For the performance evaluations use always the same algorithms and keys for all programming languages. You can create the keys first (with a program written in any language) and reuse them in all programs, regardless of their language.

For an accurate performance evaluation perform a loop with n iterations were the same operation is executed m times. Evaluate the elapsed time it takes to execute the m consecutive operations and select the smallest one from the n observed. Divide it by m and you will have an upper bound for the time each operation takes.

Note: select an m so that the time taken by that number of consecutive operations is one order of magnitude bigger that the measurement precision. However, don't make it too big, otherwise the elapsed time will be "polluted" by other computer operations (interruptions, context switches, etc.).

Note: ideally, you should perform the evaluations in a computer without graphical interface, to reduce the amount of events that can add "noise" to your measurements.

3 Evaluation

The project will be evaluated as follows:

- Performance measurements in each language (25% each);
- Written report, with a complete explanations of the strategies followed and the results achieved: 25%

4 Homework delivery

Send your code to the course teachers through Elearning (a submission link will be provided). Include a small report, with no more than 6 pages, describing the implementation (not a complete copy of the code developed!). Code snippets may be used to illustrate your implementation.

Every piece of code imported from anywhere must be stated in the report and in the code itself. Failure to do so will be penalised.

5 References

- *Digital Signature*, `https://en.wikipedia.org/wiki/Digital_signature`