crypto-condor

Compliance testing for cryptographic primitives

Julio Loayza Meneses

July 4th 2024 @ Pass the SALT 2024

Quarkslab

whoami

- R&D engineer @ Quarkslab
- Cryptography
- End-of-master internship in 2023 that resulted in this presentation
  - Thank you Dahmun, Angie, and Quarkslab!

@julioloayzam

Let’s define some terms

Cryptographic primitive

Cryptographic primitives are low-level cryptographic algorithms that can be used to construct other algorithms or protocols. Example: AES used in the TLS protocol.

Let’s define some terms

Cryptographic primitive

Cryptographic primitives are low-level cryptographic algorithms that can be used to construct other algorithms or protocols. Example: AES used in the TLS protocol.

Compliance testing

Cryptographic primitives are described in documents called specifications.
➔ We want to ensure that implementations behave as the algorithm that is described.

Compliance testing

How?

We can use **test vectors**: sets of algorithm inputs and their associated outputs.

- Deterministic algorithms always return the same output when given the same input.
- Example: SHA3-256

```
msg = 01020304
md = 966DBDCBD0E0348FAA1CCBCE5A62B8E73B0D08955D666DB82243B303D9BD9502
```

Compliance testing

How?

We can use **test vectors**: sets of algorithm inputs and their associated outputs.

- Deterministic algorithms always return the same output when given the same input.
- Example: SHA3-256

```
msg = 01020304
md = 966DBDCBD0E0348FAA1CCBCE5A62B8E73B0D08955D666DB82243B303D9BD9502
```

Why?

For audits and certifications, the implementations **must** conform to the spec.

State of the art

Project Wycheproof

- Implements several attacks against popular cryptographic primitives.
- Most attacks are provided as test vectors $\Rightarrow$ we can integrate them!
- No ready-to-use tool except for Java libraries.

State of the art

Project Wycheproof

- Implements several attacks against popular cryptographic primitives.
- Most attacks are provided as test vectors $\Rightarrow$ we can integrate them!
- No ready-to-use tool except for Java libraries.

Example: ECDSA

- Signatures are a couple of integers (r, s).
- Implementations must check that $r, s \in [1, n-1]$ *(n is the order of the base point)*.
- `pubkey = 3059301306072A86...8D1A974E7341513E`
  `msg = 313233343030`
  `sig = 3006020100020100`
- Checks if `sig` $\Leftrightarrow (0, 0)$ is accepted.

crypto-condor

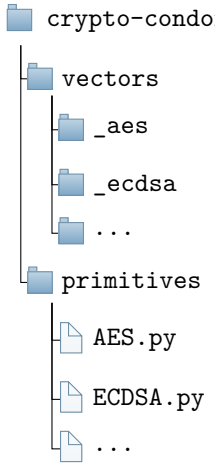
- Open-source Python library for compliance testing of implementations of cryptographic primitives.
- Available on PyPI:
  `python -m pip install crypto-condor`
- Provides a Python API and comes with a CLI.
- Includes guides on the primitives supported.

crypto-condor's logo

Method guides

- Provide a rundown of key information about the primitive.
  - Parameters, modes of operation, different variations...
- Include the corresponding rules and recommendations by the ANSSI.
- Are available in the documentation:
  `https://quarkslab.github.io/crypto-condor/latest/index.html`

The modules



- Sources of vectors:
  - NIST CAVP ➜ **compliance**.
  - Project Wycheproof ➜ **resilience**.
  - Specifications (RFCs, etc.)
- Each primitive has its own module.
- Each module has test functions.

Two approaches

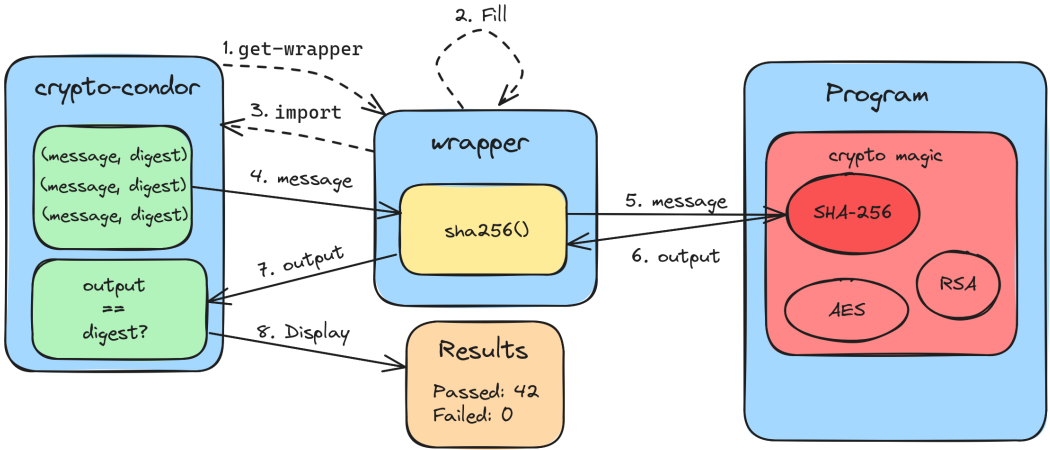
With implementation:

- The test vectors.
- To agree on the function signature.

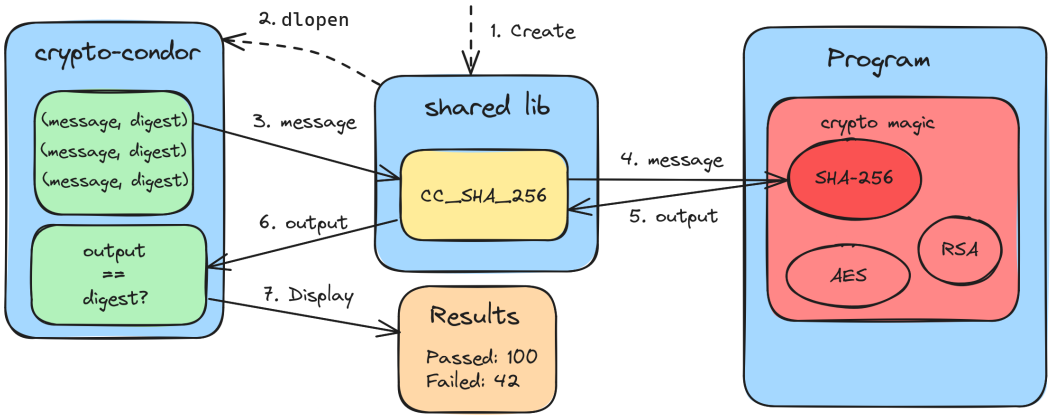
With output:

- Input/output values.
- An internal implementation.

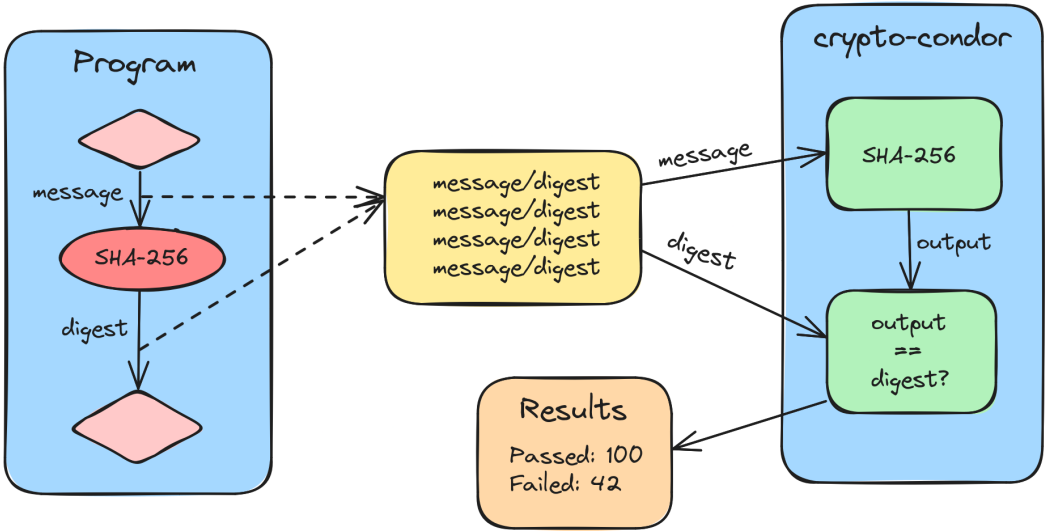
First approach: using a wrapper



First approach again: using a harness



Second approach: using the output



Use-case example: CRY.ME

- A “secure messaging application based on the Matrix protocol containing many cryptographic vulnerabilities deliberately introduced for educational purposes.”
- Developed by the ANSSI and CryptoExperts.
- Presented at SSTIC 2023.
- `https://github.com/ANSSI-FR/cry-me`

Demo

Thank you

Contact information:

Repo: `https://github.com/quarkslab/crypto-condor`

Email: contact@quarkslab.com

Website: `https://www.quarkslab.com`

Quarkslab

How to add primitives

Adding new primitives

Here are some guidelines on how to add a new primitive. To get started, the handy `utils/add_primitive.py` script creates templates of most of the necessary files:

```
python utils/add_primitive.py <primitive name>
```

From here on out, we'll use AES as an example.

Test vectors

First, there are the test vectors. It creates a directory named `_AES` to store the source files, protobuf descriptors, parsing script, and the serialized vectors. We mainly use test vectors from NIST CAVP and Project Wycheproof, though we may use other sources when needed, such as RFC 3686 for AES-CTR vectors.