COMP 3109 Final Project: Real World Use of Cryptographic Libraries

Due date: December 8, 2022, via Brightspace
Bonus marks if feedback is given by November 8, 2022, and for early project submissions (see bottom of page for details)

TL;DR

Implement the missing methods in this file. Use only pynacl, tink, and built-in Python 3.9 libraries. Submit a single Python file via Brightspace.

Introduction

The final project involves using two modern cryptographic libraries that you might encounter in your professional future. Your goal is to use both libraries to compare and contrast the ways in which they enable developers to use the cryptographic primitives learned in COMP 3109. The two libraries that were chosen for this project are: NaCl and Tink.

Installing Python 3.9:

If you already have Python 3.10 or later installed, you'll need to install Python 3.9, since 3.10 is currently unsupported by Tink. To determine what version of Python you are using:

```
python3 --version  //for Ubuntu
```

To install Python 3.9 on Ubuntu using APT, run the commands below. A new binary called `python3.9` will be available in your path. Update your aliases to use this binary instead of the system default `python3` or `python`.

```
sudo add-apt-repository ppa:deadsnakes/ppa
sudo apt install python3.9 python3.9-distutils
python3.9 -m pip install pynacl tink protobuf==3.20.*
```

NaCl

NaCl (pronounced Salt) is a modern crypto library designed by Daniel Bernstein and his team to make development of cryptographic code more usable and more secure, ideally avoiding errors. The library only supports a small set of primitives and algorithms, but these are chosen to be secure. The way in which the algorithms can be used is also limited, allowing developers to stick to defaults and be confident that their choices are not wrong.

PyNaCl is a Python binding to libsodium, which implements the NaCl library. It can be installed directly from PyPi by doing:

```
pip3 install pynacl
```

Your project must use PyNaCl, so take a moment to read through the website and API documentation.

Google Tink

Tink is another multi-platform, multi-language crypto library that provides APIs that are secure, easy to use, and hard to misuse. Like NaCl, the library is opinionated about what algorithms should be used, and how they are invoked. Tink also has strong opinions about key management and use.

The Python implementation of this library (known simply as tink) can be installed from PyPi (see documentation) by running:

```
pip3 install tink
```

Individual work

The project will be submitted individually. Each student must submit code they wrote themselves. While students may work in groups to brainstorm/discuss ideas, each student is ultimately required to attempt their own implementation. Code similarity checks will be done on all submissions, and potential plagiarism will be manually reviewed. If compelling evidence of copying is found, students may be contacted to provide additional evidence of independent work. Egregious cases will be directly sent to the Dean of Science.

Requirements

- While you may work locally, your program should run on the Openstack instances. Please ensure this is the case before submitting to avoid problems with the grader.
- Your development environment should be configured to use Python version 3.9 (due to issues with tink and 3.10)
- Your submission cannot import additional third-party libraries beyond `pynacl` and `tink`. Python built-ins are OK.
- You may implement additional helper methods, but please do not modify the method signatures provided.

- You may add comments to your code, but we will not read them unless something does not work.

Grading

- To grade your submission, we'll use `importlib` to load your Python code as a library, and run a set of tests to verify that you've implemented the methods correctly. E.g., can we encrypt a plaintext with a generated key and then decrypt it? Do we get a new key every time we call the key generator? How long are digital signatures? etc.

- While running tests, we'll use Python tracing to ensure each method invokes the correct library. We'll mock `pynacl` and `tink` methods to verify that they are invoked correctly by your code.
- You must submit your Python source for grading. Any attempts to trick the auto-grader will be treated as academic dishonesty.
- Each of the 22 methods below will be awarded max 2 points each as follows:
  - 0 - not submitted, not working (major errors), uses wrong library
  - 1 - partially implemented, incorrectly implemented (minor errors)
  - 2 - implemented correctly
- Total points: 44

Method	Points	PyNaCl	Tink
Generate secret keys	4	2	2
AEAD Encryption	4	2	2
AEAD Decryption	4	2	2
Encryption Keypair Generation	4	2	2
Hybrid Encryption	4	2	2
Hybrid Decryption	4	2	2
Signature Keypair Generation	4	2	2
Message signing	4	2	2
Signature verification	4	2	2
MAC creation	4	2	2
MAC verification	4	2	2
Total	44	22	22

Bonus points

Note: The bonus marks below may be combined for a total max project grade of 52/44.

Project feedback - Up to 4 bonus marks (i.e., max project grade 48/44) for students who submit feedback on the project writeup by November 8 at 11:59 pm. This includes feedback on (but not limited to):

- Clarity of project requirements
- Errors and omissions in method signatures
- Errors and omissions in data types
- Issues identified in using the required libraries (installation, configuration, invocations, etc.)

Feedback must be submitted (either by email or Discord) to the instructor by the November 8 deadline above.

Early submissions - 4 bonus marks (i.e., max project grade 48/44) for students who submit the project by November 15th at 11:59 pm.