600.445/600.646 Practical Cryptographic Systems March 31, 2025

Assignment 4

Instructor: Matthew Green Due: 11:59 pm, 13th April 2025

The assignment must be completed individually. You are permitted to use the Internet and any printed references, though your answers and code must be your own!

Supplementary material. As part of this assignment, you are provided with two Python source files: • `problem.py` implements the challenges and the test harness. • `solution.py` provides a starter template for your solution. Please refer to these files for more details.

Submission instructions. This assignment includes both written and programming components. Submit your answers to the written portion as a single PDF file under "Assignment 4 - Written" on Gradescope. For the programming part, submit your completed `solution.py` file under "Assignment 4 - Programming" on Gradescope.

1 Programming

Problem 1

In class, we've explored several examples where flaws in the design or implementation of cryptographic primitives and protocols can lead to serious security vulnerabilities. In this assignment, you will apply some of those ideas to mount an attack on TwoDrive—a cloud storage service provider that has made some questionable implementation choices.

TwoDrive allows registered users to store and update data on its cloud servers using a client application developed by TwoDrive. Specifically, TwoDrive offers the following services.

- **Updating user storage:** Registered users can *securely* modify their cloud storage. To enable this, TwoDrive requires users to register their public key with the server during the registration process. When users wish to update their storage, they first authenticate themselves to the server using the registered key-pair and establish a secure channel using a handshake protocol. The new storage data is then transmitted over this secure channel.

  TwoDrive authenticates the user and establishes the secure channel using the Noise handshake protocol[1]. Key features of the Noise protocol include its modular design, emphasis on Diffie-Hellman key agreement, and support for formal verification[2]. Since storage updates involve one-way communication from the user to the server—and both parties already possess each other's public keys—TwoDrive uses the `K` pattern for the handshake (see the Noise protocol specification (Section 7.4) for details).

- **Check for client software updates:** TwoDrive also allows users to check for updates to the client software. Since this process does not involve any sensitive information, the software update status messages are sent in the clear and can be queried by anyone, not just registered users.

[1] https://noiseprotocol.org/noise.html
[2] https://noiseexplorer.com/

To ensure authenticity, the server includes an ECDSA signature on the status update message, allowing users to verify that the response is legitimate.

However, as mentioned earlier, TwoDrive's implementation of these services do not follow best practices for implementing cryptographic protocols. The file `problem.py` contains an implementation of the server. Parts of the source code that may be relevant to the questions below are marked with "Point of Interest" comments.

1. Implement the `compute_ecdsa_sk` function in `solution.py`. This function recovers the server's ECDSA secret key.

   [30 points]

2. Implement the `modify_user_storage` function in `solution.py`. This function should allow an attacker (who is not a registered user) to modify the storage of a registered user.

   [50 points]

2 Written

Problem 2

Describe your approach to solving Problem 1 and explain why it works.

[5 points]

Problem 3

After a security audit, TwoDrive fixed the implementation flaws that allowed the attack in Problem 1.1. However, users have subsequently reported that their cloud storage is occasionally rolled back to an older version. What could be the cause of this issue? Describe the approach used by the attacker.

[7 points]

Problem 4

Suppose TwoDrive used the `KK` pattern (see Noise protocol specification (Section 7.5)) instead of the `K` pattern for handshake. Would your approach to solving Problem 1.2 still work in this case? Why or why not?

[8 points]