INTSEC VT 2025 - Challenge 3.4:
Difficulty: Interesting (Hard +)

Important: No AI tools are allowed to solve this challenge.

Scenario

Alice is working at a café that has a surveillance camera in a corner of the main room. One evening after closing the café, while she reviews the images from the camera, she notices that some customers often bring their laptop to work from the café. Depending on where they sit in the room she can see both their screens and keyboards. She also notices that two customers, Bob and Charlotte, access their online bank (Acedia Bank) from the café.

According to a hacker friend of Alice, Denis, Acedia Bank uses a two-factors authentication scheme to verify the identity of its customers. One factor is simply the customer's ID (a 10-digit number) and a customer-chosen password. The second factor of authentication is a One Time Password generated by a small dongle that the bank provides its customers with. If a customer makes 3 unsuccessful attempts with a correct customer ID and customer password but an incorrect OTP, the bank freezes the account and contacts the customer to reset their customer password and change their dongle.

Denis tells Alice he successfully reverse-engineered how the dongle works. It uses an algorithm called TinyH OTP to generate passwords (4 hexadecimal characters) based on a counter and a hidden OTP master password (unique for each dongle) stored on the dongle but never shown to the user.

TinyH-OTP

The algorithm works as follows:

$$\text{TinyH-OTP}(mp, n) = \text{TinyH}^n(mp) \oplus mp$$

where

- mp: master password as a hex string (16 bits),
- n: unsigned integer step (16 bits) initialized at 0 and incremented before any One Time Password generation.
- $\oplus$ is the bitwise XOR operator.

$\text{TinyH}^n(mp)$ corresponds to n successive applications of TinyH on mp. For example:

$$\text{TinyH}^2(mp) = \text{TinyH}(\text{TinyH}(mp))$$

$$\text{TinyH}^3(mp) = \text{TinyH}(\text{TinyH}(\text{TinyH}(mp))).$$

TinyH is a small hash algorithm based on SHA1. It takes the 16 most significant bits (4 hex characters) of SHA1:

$$\text{TinyH}(a) = \text{SHA1}(a) \gg 144.$$

where

- $\gg$ is the bitshift operator.

Denis even provides a Python script implementing the algorithm for reference (See `tiny_h_otp.py`).

Known Data

Alice goes back to the footage from the surveillance camera and she is able to work out that:

- Bob connected twice in a row (he got logged out for inactivity and logged in directly after) to his online bank in the café with the following credentials:
  - customer ID: `1944533760`
  - customer password: `wka1ée3ç'`
  - OTPs: first `f25a`, then `cbc9`

Assuming Alice has solid skills and shaky ethical principles, and would like to log in to Bob's bank account, what OTP should she use, assuming Bob has not logged in since the day he was recorded at the café?

- Charlotte connected to her online bank in the café twice in a row as well, but another customer walked in front of the camera during her second log in. From the footage, Alice was only able to recover the first character of the second OTP used by Charlotte. On the other hand, Alice noticed that Charlotte pulled out the dongle from it's original package. While the package was already open, Alice makes the assumption that the dongle is fairly knew and has probably been used less than a hundred times.
  - customer ID: `5197821583`
  - customer password: `$*alh,zgzs°`
  - OTP: `1ccf`, `1***`

Assuming, Charlotte has not logged in since the day she was recorded at the cafe, which OTPs should Alice try to use to access Charlotte's bank account (with some assumptions)? What is the probability she is successful, should her assumptions hold?