COMP 537 **Fall 2024**

Problem Set 1 Submission

Name: Jeffrey Zhu, Juntao Li

Problem 1 Suppose an adversary is able to perform a swap attack (as described in Section 3). Show how such an adversary can win the note taking security game. Note that you must say why the adversary you construct is admissible for the security game.
The attack goes the following:

1. The adversary first makes an insertion query t_1 with notes note_{e0}=note_{e1}=n_0, which constructs the triple $\langle\ t_1, \text{note}_{e0}, \text{note}_{e1}\rangle$. Now the challenger would store the pair $<t_1, n_0>$, because $\text{note}_b = n_0$. For the second insertion, the adversary specifies $<t_2, n_1, n_2>$ where $n_1 \neq n_2$. Now the note_b becomes

   - If $b = 0$, $\text{note}_b = n_1$.
   - If $b = 1$, $\text{note}_b = n_2$

   So the challenger would store $<t_2, \text{note}_b>$ in the database.

2. The adversary can perform the swap attack by requesting a serialize query, which would give challenger the opportunity to swap the notes t_1, t_2. Consequently, the adversary sends back the deserialize query with the following changes:

   - $< t_1, \text{note}_b >$
   - $< t_2, n_0 >$

3. Now the adversary can request a retrieve query for t_1, which is admissible since the last insertion query satisfies the condition $\text{note}_{e0} = \text{note}_{e1} = n_0$. However, instead of returning $< t_1, n_0 >$, the challenger would now return either $< t_1, n_1 >$ or $< t_1, n_2 >$ due to the swap attack.

4. Finally, the adversary would be able to determine b by examining the retrieved note.

   - If the note is n_1, then $b = 0$.
   - If the note is n_2, then $b = 1$.

Problem 2
Suppose an adversary is able to perform a rollback attack (as described in Section 3). Show how such an adversary can win the security game. As before, you should say why the adversary you construct is admissible.

The adversary can first do an insertion of $<$ "$title$", $m_0, m_1 >$. They then do the insertion $<$ "$title$", $m_2, m_2 >$. They then perform a rollback attack, and then asks for a retrieval of "title". The challenger will return either m_0 or m_1, which allows the adversary to make their guess.
The challenger is admissible, because the last insertion on the title was the repeated message m_2, but in this case, we didn't care about m_2 at all.

Problem 3

Briefly describe your method for checking passwords.

The helper _encrypt and _decrypt methods were created in order to use AES-GCM encryption. When making a new database, salt is generated, and a source key is generated from the password and salt. When the data is dumped, the AES-GCM decryption in _encrypt is used to create a serialized, essentially ciphertext using the salt as a nonce_source, and the source_key as the key, and then adding the salt to the beginning of the serialized data. Then, when attempting to access the data by passing the data through the constructor, the salt is extracted from the first 16 bytes of data, and is used with PBKDF2HMAC in order to create the same source key. Then, AES-GCM _decrypt is ran using salt as the nonce_source, and the source_key as the key, which should be the same if the same salt and password were generated from the data. Since the nonce is deterministically created from nonce_source, the nonce will be the same. Then, if an error is thrown, that must mean that the password was wrong, or the salt or data were edited.

Problem 4

Briefly describe your method for preventing the adversary from learning information about the lengths of the notes stored in your note-taking application.

Padding was added to the end of every note. Knowing that each note could have a total of 2048 bytes, a padding length was calculated by simply subtracting the length of the plaintext from 2048, since every plaintext is a string, the plaintext should be an integral number of bytes. Then padding is done using 0x00, or null bytes, all the way until the max length of 2048 bytes. Then, once the padded plaintext is encrypted, every ciphertext will have the same size.

Problem 5

Briefly describe your method for preventing swap attacks (Section 3). Provide an argument for why the attack is prevented in your scheme.

Each title and note pair is encrypted using the title as the source for the nonce in AES-GCM encryption. If the encrypted note is decrypted using the wrong nonce, which in this case mean's the wrong title, an exception is thrown, and the note will not be shown.

Problem 6

Optional feedback. How much time did you spend on this assignment? Did you find it too easy/hard or just right?

Probably around 4-5 hours. Most of the work was really reading through documentation, and seeing how everything fit together.

Problem 7

Optional feedback. Please let us know if you have any feedback on the design of this assignment or on the course in general.

Nothing in particular.