HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

CRYPTOGRAPHY
Project #1: Password manager

Hoàng Duy Anh	20214944
Nguyễn Trọng Hiếu	20210330
Nguyễn Quốc Trung	20214976

Supervisor: Ph.D. Trần Vĩnh Đức
School: Information and Communications Technology

HANOI, 04/2024

1 Short-answer Questions

1. **Question:** Briefly describe your method for preventing the adversary from learning information about the lengths of the passwords stored in your password manager.

   Answer: We used the *createCipheriv* function in **crypto** library with default *setAutoPadding()* so all password will have the same length.

2. **Question:** Briefly describe your method for preventing swap attacks. Provide an argument for why the attack is prevented in your scheme.

   Answer:

   - The key-value-store in our password stores a pair of HMAC hashed domain name, and an authentication encrypted password concatenated with the HMAC hash of its domain name (in this case AES-256/GCM with IV).
   - keychain = **{HMAC(name), IV || AES-256-GCM(HMAC(name) || password)}**

   Proof by contradition: Assume there's an adversary that can successfully perform the swap attack. This means that the adversary can swap the record for

   HMAC(name), IV1 || AES-256-GCM(HMAC(name1) || password1)

   and **HMAC(name), IV2 || AES-256-GCM(HMAC(name2) || password2)**

   such that the new records are

   HMAC(name), IV || AES-256-GCM(HMAC(name1) || password2)

   and **HMAC(name), IV || AES-256-GCM(HMAC(name2) || password1)**

   However, this means that the adversary broke AES-256-GCM() (i.e. AES-256/GCM) since the ciphertext integrity property was violated because he was able to submit a new ciphertext

   IV || AES-256-GCM(HMAC(name1) || password2

   or **IV || AES-256-GCM(HMAC(name2) || password1**

   that the challenge accepted. We have reached a contradiction. Thus, the scheme protects againts a swap attack.

3. **Question:** In our proposed defense against the rollback attack, we assume that we can store the SHA-256 hash in a trusted location beyond the reach of an adversary. Is it necessary to assume that such a trusted location exists, in order to defend against rollback attacks? Briefly justify your answer.

   Answer: Indeed, it is essential to presume the existence of a reliable location. This is because our integrity strategy, which pads the HMAC(name) to the padded

password, only guards against swap attacks if we presume that the hash is not kept in a safe storage place. The stored "AES-256-GCM(password || HMAC(name)" entry can be replaced by an adversary with an earlier "AES-256-GCM(password || HMAC(name)" entry, and since the HAMC(name) will still match, the password manager won't be able to recognize it.

4. **Question:** Because HMAC is a deterministic MAC (that is, its output is the same if it is run multiple times with the same input), we were able to look up domain names using their HMAC values. There are also randomized MACs, which can output different tags on multiple runs with the same input. Explain how you would do the look up if you had to use a randomized MAC instead of HMAC. Is there a performance penalty involved, and if so, what?

   Answer:

   Being a PRF and existentially unforgeable, HMAC meets the criteria for generating keys. It's okay to swap out HMAC with the randomized MAC if it meets these requirements. However, to perform lookups with a randomized MAC, the database would need to store both the domain name and the MAC value(s) generated for it. When looking up a domain, its MAC value would need to be re-calculated and compared to the stored MAC values for potential matches, rather than doing a direct value lookup.

   This would involve a performance penalty compared to using a deterministic MAC like HMAC. With HMAC, the lookup process is a simple value comparison. But with a randomized MAC, the following additional steps would be needed for each lookup:

   - Re-compute the MAC value for the domain.
   - Scan the database, retrieving and re-computing MAC values for all entries.
   - Compare the freshly computed MAC to each stored MAC value to find potential matches.

   This is more computationally expensive than a single lookup. It also requires more memory accesses as each database entry's MAC needs to be re-computed. So there would be an increase in both computation time and memory/storage usage compared to using a deterministic MAC like HMAC.

5. **Question:** In our specification, we leak the number of records in the password manager. Describe an approach to reduce the information leaked about the number of records. Specifically, if there are k records, your scheme should only leak log2 (k) (that is, if k1 and k2 are such that log2 (k1) = log2 (k2), the attacker should not be able to distinguish between a case where the true number of records is k1 and another case where the true number of records is k2).

Answer: One approach could be to establish a high maximum limit on the total number of passwords allowed. The password manager would always take up the same amount of disk space, padding unused areas with random bits. However, this design would not make optimal use of memory and storage. It is important to note that setting an upper limit on the number of entries does place a cap on how many passwords could be stored in the manager. But this restriction would be acceptable as long as the limit is sufficiently high that a user is very unlikely to ever reach the maximum.

6. **Question:** What is a way we can add multi-user support for specific sites to our password manager system without compromising security for other sites that these users may wish to store passwords of? That is, if Alice and Bob wish to access one stored password (say for nytimes) that either of them can get and update, without allowing the other to access their passwords for other websites.

   Answer: Each user's passwords should be encrypted using a unique encryption key derived from their master password or other secure authentication mechanism. This ensures that each user's passwords remain secure and inaccessible to others. Implement access control mechanisms to manage which users have access to the shared websites. We can use role-based access control (RBAC) or access control lists (ACLs) to achieve this. For the passwords stored in the shared websites, generate a shared encryption key that is shared among the authorized users. This shared encryption key should be securely distributed to the authorized users, perhaps using a key exchange protocol or a secure communication channel.