Q1 *The Lorenzo Von Matterhorn* **(0 points)**

Barney needs to make sure that no attackers can access his highly sensitive, top secret playbook tricks!

For each password scheme, select all true statements. Assume that:

- Each user has a unique username, but not necessarily a unique password.
- All information is stored in a read-only database that both the server and the attacker can access.
- The server has a symmetric key K not known to anyone else. The server also has a secret key SK not known to anyone else, and a corresponding public key PK that everyone knows.
- An *operation* is defined as one of the following actions: hash, encryption, decryption, and HMAC.
- The attacker does not have access to a client UI; therefore, online attacks are not possible.

Q1.1 For each user, the database contains $\mathsf{username}$ and $\mathsf{H}(\mathsf{password})$, where H is a cryptographic hash function.

- ■ If a user inputs a username and password, the server can verify whether the password is correct
- ■ Given the information in the database, the attacker can verify that a given username and password pairing is correct.
- ☐ The server can list all plaintext passwords by computing at most one operation per user
- ■ An attacker can list all passwords by computing at most one operation per possible password
- ☐ None of the above

Solution:

A: The server can hash the password to check that it matches the hash in the database.

B: The attacker can hash the password (hashes aren't keyed) and check that it matches the hash in the database.

C: The server cannot compute one operation to reverse a hash.

D: The attacker can conduct an offline brute-force attack, hashing every possible password and comparing to the hashes in the database.

Q1.2 For each user, the database contains username and $\mathsf{HMAC}(K, \mathsf{password})$.

- [x] If a user inputs a username and password, the server can verify whether the password is correct
- [ ] Given the information in the database, the attacker can verify that a given username and password pairing is correct.
- [ ] The server can list all plaintext passwords by computing at most one operation per user
- [ ] An attacker can list all passwords by computing at most one operation per possible password
- [ ] None of the above

Solution:

A: The server can HMAC the password to check that it matches the HMAC in the database.

B: An attacker cannot compute the output of HMAC without knowing the key input K.

C: The server cannot compute one operation to reverse HMAC.

D: The attacker cannot compute HMACs without knowing the key input K.

Q1.3 For this subpart, Enc denotes an IND-CPA secure symmetric encryption function.

For each user, the database contains username and $\mathsf{Enc}(K, \mathsf{password})$.

- [x] If a user inputs a username and password, the server can verify whether the password is correct
- [ ] Given the information in the database, the attacker can verify that a given username and password pairing is correct.
- [x] The server can list all plaintext passwords by computing at most one operation per user
- [ ] An attacker can list all passwords by computing at most one operation per possible password
- [ ] None of the above

Solution:

A: The server can encrypt the password to check that it matches the password in the database.

B: An attacker cannot encrypt passwords without knowing the key input K.

C: The server can decrypt the password in the database.

D: An attacker cannot encrypt passwords without knowing the key input K.

Q1.4 For this subpart, RSA denotes RSA encryption without OAEP padding.

For each user, the database contains username and RSA(PK, password).

- [x] If a user inputs a username and password, the server can verify whether the password is correct
- [x] Given the information in the database, the attacker can verify that a given username and password pairing is correct.
- [x] The server can list all plaintext passwords by computing at most one operation per user
- [x] An attacker can list all passwords by computing at most one operation per possible password
- [ ] None of the above

Solution:

A: The server can encrypt the password to check that it matches the password in the database.

B: The attacker can also encrypt passwords, because everybody knows the public key.

C: The server can decrypt the password in the database.

D: An attacker can encrypt every possible password and compare the encryptions to the encryptions in the database.

Q1.5 Consider a modification to the scheme in the first subpart: Instead of storing H(password) per user, we now store H(password||salt) per user.

Assume that concatenation does not count as an operation. Compared to the original scheme, which of the following algorithms for generating salts would force the attacker to compute more operations to list all passwords? Select all that apply.

- [x] A 128-bit value, randomly generated per user
- [x] A 128-bit counter, starting at 0 and incremented per user
- [x] A 128-bit counter, starting at a random number and incremented per user
- [ ] None of the above

Solution: Salts need to be unique per user. If salts are unique, then the attacker needs to hash the dictionary of passwords once per user, instead of once for all users.

Q1.6 Which of these hash algorithms makes the scheme in the first subpart most secure against offline brute-force attacks? Briefly explain (10 words or fewer).

○ MD5 ○ SHA2-256 ● Argon2Key (PBKDF2)

Solution: Hashes need to be slow to increase the amount of time a dictionary attack takes. MD5 and SHA2-256 are faster hashes, and Argon2Key (PBKDF2) is a slower hash.

Q2 *So, you want a secure key?* **(16 points)**

Alice wants to create a secure channel of communication with a server. From CS 161, Alice remembers that the best way of communicating is to somehow end up with a shared, symmetric key, but has no idea how this process works.

Assume that there exists a certificate authority (CA) actively sending certificates to many clients. Assume that Mallory, a MITM attacker, and Eve, an eavesdropper, can both exist in all communication channels for all subparts unless otherwise specified.

Q2.1 (2 points) Alice first wants to authenticate the CA before authenticating the server. She remembers that certificates provide authenticity, so she exchanges the following messages with the CA:

1. Alice queries the CA for the CA's public key and receives PK_{CA}.
2. Alice queries the CA for the server's public key and receives $\{\text{“The server's public key is } PK_S\text{”}\}_{SK_{\text{CA}}^{-1}}$.

Can Mallory trick Alice into accepting a different public key PK'_S **of Mallory's choosing** as the server's public key without being detected?

● Yes ○ No

> **Solution:** A MITM attacker could be present within the Alice–CA channel and therefore provide a public key (PK_{CA}) that is different from the CA's actual public key, to Alice. Since there is no root of trust, there is no way for Alice to know that she received an incorrect key.

For the rest of this question, assume that instead of querying the CA for their public key, Alice has the CA's correct public key, PK_{CA}, hardcoded into her computer.

Q2.2 (5 points) When Alice queries the CA for the server's public key, the CA sends $\{\text{“The server's public key is } PK_S\text{”}\}_{SK_{\text{CA}}^{-1}}$.

Can Mallory trick Alice into accepting a different public key PK'_S, **not necessarily of Mallory's choosing**, as the server's public key without being detected?

If you mark “Yes”, provide an attack that would accomplish this goal. If you mark “No”, explain why not in 2 sentences or fewer.

● Yes ○ No

> **Solution:** Mallory can replay any certificate that has been sent over the channel previously. Two potential examples include a certificate of a different server that has previously been sent over the channel by the CA, or sending an old public key of *this* server.

Q2.3 (5 points) When Alice queries the CA for the server's public key, the CA selects a random number x between 1 and 20 and sends $\{\text{“The server's public key is } PK_S\text{”}\|x\}_{SK_{\text{CA}}^{-1}}$.

Can Mallory trick Alice into accepting a different public key PK'_S, **not necessarily of Mallory's choosing**, as the server's public key without being detected?

If you mark “Yes”, provide an attack that would accomplish this goal. If you mark “No”, explain why not in 2 sentences or fewer.

● Yes ○ No

Solution: The addition of x here doesn't really change anything from the previous subpart since the value of x can be repeated. An attacker could still replay a certificate that was sent over the channel.

Q2.4 (4 points) Alice has received some public key PK_S from the CA, but she doesn't trust that PK_S belongs to the server. Which of the following messages can the server send to convince Alice that she is talking to the legitimate server? Select all that apply.

☐ The server sends $H(PK_S)$

☐ The server sends $\mathsf{Sign}(SK_S, H(PK_S))$

☐ The server sends $H(SK_S||PK_S)$

☐ The server randomly generates a symmetric key K and sends $(\mathsf{Sign}(SK_S, K), \mathsf{HMAC}(K, PK_S))$

■ None of the above

Solution: This question is an issue of the server trying to authenticate themselves to Alice. If the server sends $H(PK_S)$, it doesn't authenticate themselves to Alice since any MITM could replace the hash with their own hash. If the server signs the hash of their public key, since Alice still does not have the server's actual public key, this doesn't work either. For example, an attacker who has sent Alice PK'_S could replace the signature with $\mathsf{Sign}(SK'_S, H(PK'_S))$. Finally, if the server generates a symmetric key, the same issue (and attack) with option 2 still applies.

Q3 *Chegg Certificates* **(7 points)**

Chegg uses a certificate chain in order to verify tutors. When tutors post responses, they attach a digital signature of their response along with their certificate. Students can verify the authenticity of a response by verifying the certificate and using the public key in the certificate to verify the signature.

The certificate chain is below. Assume that the Chegg Root Certificate Authority (CA) is hardcoded into students' browsers.

1. Identity: Director of Chegg Recruiting (Verified by Chegg Root CA)
2. Identity: Campus Chegg Recruiter (Verified by Director of Chegg Recruiting)
3. Identity: Authorized Tutor (Verified by Campus Chegg Recruiter)

Q3.4 (4 points) EvanBot is not a valid tutor, but wants to create a fake tutor response with a valid signature. Which of these attacks would allow Bot to accomplish this?

- ☐ Steal the public key of the Campus Chegg Recruiter
- ■ Steal the private key of the Director of Chegg Recruiting
- ■ Steal the private key of the Chegg Root CA
- ☐ Steal the certificate of an authorized tutor
- ☐ None of the above

Solution: Stealing the private key of any individual in the certificate chain above the Authorized Tutor level would allow EvanBot to construct a valid certificate and pose as an Authorized Tutor. Certificates and public keys are public, so stealing one doesn't help you forge a message.

Q3.5 (3 points) EvanBot gains access to the private key of Dave, who is an authorized tutor. Which of the following can EvanBot do?

- ☐ Post a valid response as Nick, an existing tutor
- ■ Post a valid response as Dave
- ☐ Create and sign a certificate for Raluca, a new tutor
- ☐ None of the above

Solution: Acquiring a private key to a user at the base level of the certificate only permits an attacker to spoof authenticated messages on behalf of that particular user.