Project 2

Yanshu Hong, Nat Roth
CS255: Cryptography and Computer Security

March 11, 2015

Question 1. Consider a man-in-the-middle attack where the attacker intercepts the message l from the server. Because SHA-256 can be computed by anyone, the attacker can just reply with SHA-256(l) and thereby the server will identify the attacker as the actual client.

In the public-private-key scheme, the attacker does not know the secret key. Therefore, he is not able to produce a valid response even if he knows l.

Question 2. Because once we have pinned our CA certificate, the TLS connection will be established only if the server can present a certificate that is signed by our CA. Therefore, if we have accepted the connections, we know that the server's certificate is validly signed by our CA. In other words, the issuer field must be valid.

However, we still have to check the subject and the validity period because they can be incorrect independent of the CA.

Question 3. The current scheme reads in *sec_key_base*64 and parses it into one 128-bit salt and the encryption of the client secret key. The KDF hash-function will generate a 128-bits GCM key from the entered password and the salt. Then the GCM key will be used to decrypt the client secret key.

Insecure Implementation 1. If the system stores the password in plain text with the salt and the encrypted client secret key, when the disk is leaked to the attacker or the loaded memory is exposed to the attacker, the attacker can run the entire process by himself (in particular, using the password and the salt to run KDF and then using the generated key to decrypt the client secret key). The current scheme is secure against this attack because it does not store the password anywhere.

Insecure Implementation 2. Say we attempt to speed up the loading process. The system takes the first 16 bits of the password and feed that into the KDF function with the salt. This implementation is vulnerable to brute-force attack because the attacker can just try all $2^{16} = 65536$ possible passwords and he can find the right one and thus decrypt the user secret key. Since the current scheme has variable length password and the password can be long, the brute-force attack is not feasible.

With the same intention, the system might be implemented with a different hash function that is much faster than KDF. If the hash function is too fast, a password-dictionary-based brute-force attack might be able to guess out the user password and then recover the user secrete key.

Question 4. **(a)** Using the symmetric encryption scheme does not need a trusted third party. In this example, we need a trusted CA to issue a certificate for the server and this might be time and money-consuming.

Moreover, the MAC-based signature is generally much shorter than the public-key-secret-key digital signature.

(b) If we're using the symmetric encryption scheme, every pair of client and server should have a different key. If that is not the case, for example, one user has the same key with Google and Facebook server, then Facebook can forge messages so that they appear to be signed by Google and the user can not distinguish because both servers share the same MAC key. Therefore, every client should store the keys for all servers and all servers should store the keys for all clients. This might be impossible if each has only a small amount of trusted storage.

Question 5. **(a)** The client will send in a password to the server and the server will calculate the salted hash of the password and compare it with the stored value. If they match, the server can identify the client. The information that should be stored in server's database will be the salt and the salted hash of the password. We should not directly store the clients' passwords because if the database has been leaked or compromised, the attacker can figure out the passwords easily. But if we store the salted hash, even the database has been leaked, the attacker can't figure out the passwords. And the table-lookup algorithm takes a long time.

(b) If the attacker can have a "bad" certificate even for a short time, the attacker can pretend to be a valid server that attempts to establish connections with a client. Because the client will think the attacker is a valid server, he will presumably send in his correct password to the attacker who can then store it. After the certificate expires, the attacker can still use the client's password to connect to the server in the future if the system uses simple password authentication and the client does not change his password.