C2107 Tutorial 4 (Hash, mac)
Solution

Chang E.-C., School of Computing, NUS

September 17, 2025

1. (*Pitfalls: "pseudo random" could be "deterministic".*)

   To generate the IV and key, Bob insecurely employed H by giving a fixed string of 160 zeros as the seed s. Although hash function (e.g SHA family) is often called "pseudo random", note that it is "deterministic". Hence, the attacker can simply derive the key by repeating Bob's key generation process, i.e. by taking the leading 128 bits of $\texttt{H}(\texttt{H}(000\ldots000))$.

2. (*Still insecure: (1) seed entropy too low. (2) programming error.*)

   - If an adversary knows the time, which is possible in practice, then he/she can derive the key.
     If the adversary knows only the approximate time, still he/she can exhaustively search all possible times.

   - Even if the adversary knows nothing about the time, it is still possible to brute force the variable s. This is since `int` data type in C is only either 2-byte (16-bit) or 4-byte (32-bit) long depending on the platforms used. It is feasible to exhaustive search 32 bits.

3. (*Information on secret key is used to derive other data, instead of solely for encryption.*)

   Bob's implementation still insecure as an attacker can still find the key k used. Notice that the key k is derived by applying H to the 160-bit $x_1 = v \,||\, r$, where v is the 128-bit IV and r is a 32-bit string. Since the attacker knows the IV v, then he/she will just need to guess the generated r. Given that r is only 32 bits, the attacker is therefore able to exhaustively search r. For each r, construct $x_1' = v \,||\, r$, and compute $x_2' = \texttt{H}(x_1')$. Then, test whether the first 128-bit of x_2' is the correct key k.

4. (*Online vs offline attack*)

   (a) Yes. According to RFC 4086 recommendation (lecture note), 30 bits is sufficient to be secure against online attack. Following the guideline in our course, 49 bits is sufficient. There are a total of $(26 \times 2 + 10 = 62)$ alphanumeric symbols. So each character in the password contribute to at least 5.9 bits. 10 characters would be 59 bits and hence more than sufficient. (To be rigorous, note that total number of passwords is 62^{10}. Assuming each password is equally likely, the entropy would be $\log_2 62^{10} > 10 \times 5.9 = 59$.)

(b) 10 characters not sufficient. It is possible to carry out offline attack. According to the assumption of attack scenario, attacker can first sniff and get c, h. Next, the attacker exhaustively search for the password p s.t. (1) $k = \texttt{SHA3}(p)$, (2) $r = \texttt{DEC}(k, c)$, (3) $h = mac(k, r)$ without interacting with the system. The p that meets the three equality tests must be the correct password. The entropy of the password is $\log_2 62^{10} < 5.96 \times 6 = 59.6$. Using the guideline in Tutorial 3, at least 128 bits are required.

(c) Wrong implementation. If r is always 0, the mac h is always the same. So, the attacker can simply conduct a *Replay Attack*. Just send in previous h.
(Remark: This demonstrate replay attack. Simply sniff and replay as it is.)
(Optional: r supposes to provide freshness in the authentication protocol.)

(d) Unfortunately, WPA2 personal employ a similar protocol and is vulnerable to offline dictionary attack. So, a longer password required.

(e) Remark. To mitigate exhaustive search on the password, it is common to deploy a hash function that is intentionally designed to be very slow. Such hash function is aka Key Derivation Function (KDF). More on KDF in next tutorial.

(f) Optional Remarks.

i. Most wifi access point deploy WPA2 to secure the wireless connection. There are variants of WPA2 that uses LEAP or PEAP (LEAP is vulnerable to offline attack and PEAP is secure against offline attacks). Fortunately, NUS uses PEAP. To prevent the offline attack, PEAP first makes sure that the server is authentic, and then all communication is protected by a key generated by "authenticated key-exchange" (to be covered later). The password (or hash of it) can then next send via the secured channel. For this, we need the client to know the server's public key, and thus the need of "certificate".

ii. This question considers a passive eavesdropper who sniff the h and r. In practice, an attacker might setup a fake client or server. Unlike the passive eavesdropper, the fake client (or server) doesn't have to follow the protocol and thus might be stronger.

5. (*Stream cipher is malleable*)

Recap the security requirement of mac: *After seen multiple valid pairs of messages and their corresponding mac, it is still difficult for the attacker to forge a mac for message not seen before.*

One way to show/prove the insecurity of a mac is by giving a successful attack. That is, we have to give an algorithm/attack that, when given

many pairs of message and mac, the algorithm can generate a new message and its valid mac.

Suppose the attacker knows a pair of valid message m and its mac

$$t = mac(k, m) = \text{Enc}_k(H(m)) = r \oplus H(m)$$

where r is the pseudorandom sequence generated by the stream cipher. The attacker next chooses another message $\widetilde{m}$ where $\widetilde{m} \neq m$. The attacker computes

$$\widetilde{t} = t \oplus H(\widetilde{m}) \oplus H(m)$$

Note that $\widetilde{t}$ is a valid mac for $\widetilde{m}$, since

$$\widetilde{t} = t \oplus H(\widetilde{m}) \oplus H(m) = r \oplus H(m) \oplus H(\widetilde{m}) \oplus H(m) = r \oplus H(\widetilde{m})$$

So, the attacker has constructed a new pair of message $\widetilde{m}$ and its valid mac $\widetilde{t}$. In other words, the attacker can forge a valid pair that it has not seen before. This violates the security definition and thus not a secure mac.

Remark. There are constructions of mac that use encryption. That is, certain choices of encryption can give a secure mac.