NUS CS-3235: Computer Security September 18, 2020

Assignment 1

Lecturer: Reza Shokri *Student: Student A0180257E*

1 Think like an Adversary

1.1

1. Sniffing
   - Assumptions
     - Device first sends messages to a WiFi router, which then relays the message to the portal, but the traffic between the device and the router is not encrypted, so the credit card detail is sent in clear.
   - Attack Mechanism
     - Attacker sniffs the packet containing the credit card details.
2. Accessing Router Traffic as Admin
   - Assumptions
     - Attacker is the WiFi router owner and can access encrypted traffic between device and router in clear by logging into the router.
   - Attack Mechanism
     - Attacker logs into router to obtain the packet containing the credit card details.
3. Impersonation with Replay Attack
   - Assumptions
     - When device is establishing secure connection with the portal server, it only verifies authenticity of server by verifying, with the public key from server certificate, a signed message from the server.
     - Attacker has previously sniffed such a message from the server.

  - Attacker can direct traffic from devices to itself, rather than to the server.
- Attack Mechanism
  - When the device sends ClientHello to connect to the portal server, it is directed to the attacker instead, who impersonates as the server and sends the server signed message at an appropriate juncture.
  - Upon receiving the signed message for verification, device is able to verify that the message is indeed signed with server private key, believes it has connected to the server, and proceeds to establish shared key with attacker.
  - Attacker can then encrypt all incoming traffic, including the credit card details.

4. Downgrade Attack
   - Assumptions
     - Device and portal server support weaker export ciphersuites.
   - Attack Mechanism
     - When device is establishing secure connection with the portal server, attacker intercepts ServerHello and replaces secure ciphersuite with a weaker one, eg ExportRSA, ExportDHE.
     - Device accepts this and proceeds to establish shared key with attacker.
     - Attacker can then decrypt all incoming traffic, including the credit card details.
5. DNS Spoofing
   - Assumptions
     - Attacker is able to reply to DNS requests to resolve portal server hostname to the IP address of his own website instead.
   - Attack Mechanism

  - When device attempts to obtain IP address of the portal server, attacker sends a fake reply redirecting device to his own site.
  - User then enters credit card detail into attacker's own site, allowing him to obtain the credit card details.

6. Evil Twin: Fake WiFi access point
   - Assumptions
     - Attacker is able to set up a free WiFi access point with same name as that of the cafe.
   - Attack Mechanism
     - When device uses the attacker's WiFi to connect the portal, attacker can sniff messages from the device to the portal, similar to the first attack.
7. Shoulder Surfing
   - Assumptions
     - Attacker is able to watch user key into device without appearing suspicious, and can see clearly what the user is typing.
     - User types Card Verification Value (CVV) in clear.
   - Attack Mechanism
     - Attacker watches user key in credit card details and obtains it.

2 Perfectly Secret or Not

2.1

The scheme is not perfectly secret. Example:

$$P(M = m) = \frac{1}{5}, \quad P(C = c)\frac{1}{5}, \quad P(K = k) = \frac{1}{6}$$

$$\begin{aligned} P(M = 1|C = 1) &= \frac{P(M = 1, C = 1)}{P(C = 1)} \\ &= \frac{P(M = 1, K = 0) + P(M = 1, K = 5)}{P(C = 1)} \\ &= 5\,(\frac{1}{5} \times \frac{1}{6} + \frac{1}{5} \times \frac{1}{6}) \\ &= \frac{1}{3} \\ &\neq \frac{1}{5} = P(M = 1) \end{aligned}$$

2.2

This scheme is not perfectly secret because the key space is less that of the message space, by theorem in Lecture 3 slide 4.

Example:

$$M = \{0, 1\}^2, K : \{00, 11\}$$

$$P(M = 00|C = 10) = 0 \neq P(M = 00)$$

3 Permutation Cipher

3.1

For $b \in \{0, 1\}$, let W_b represent the scenario of adversary guessing 1 in experiment b, and let m_b be the message sent by adversary in experiment b, as illustrated in the attack game below:

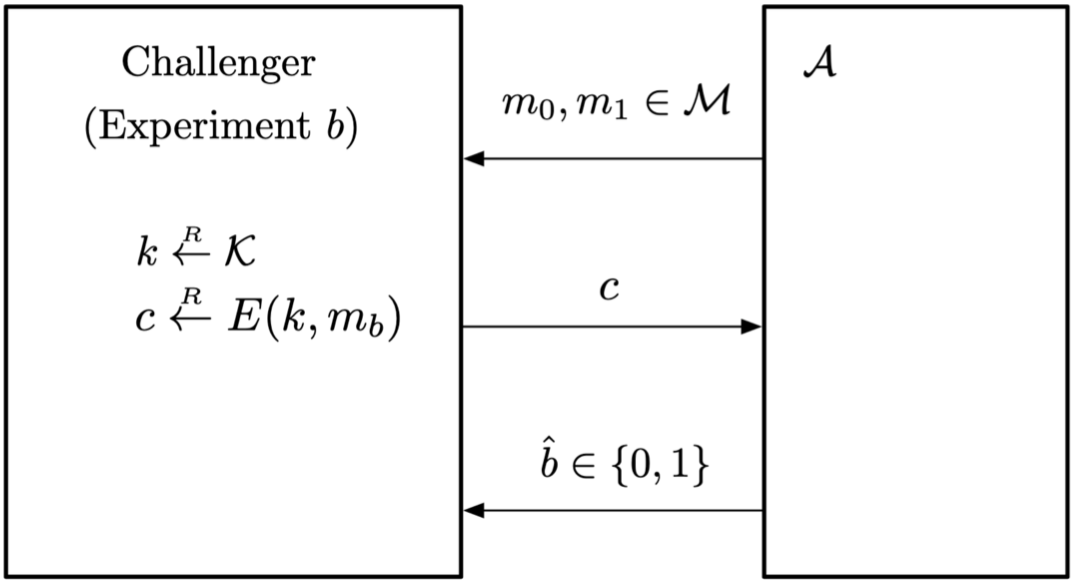

Figure 1: Lecture 3 Slide 17

Adversary advantage $= |P(W_0) - P(W_1)|$

Let $m_0 = [0]^l, m_1 = [1]^l$. For each message, as all bits are the same, permuting the bits results in no change:

$$\forall k \in K, E(k, m_b) = c_b = m_b$$

As such, adversary can always distinguish between c_0, c_1, and thus always determine the experiment number correctly. For example, in experiment 0, adversary will receive c_0, which will be the same as m_0, so adversary can tell this is experiment 0.

Therefore, advantage $= |P(W_0) - P(W_1)| = 0 - 1 = 1$ and this cipher is not semantically secure.

4 Collision Resistant Hash Functions

The hash function H is secure if it is collision resistant: given m_1, an attacker cannot find another message m_2 such that $H(m_2) = H(m_1)$ in negligible time.

4.1

$H(m) := (H_1(m), H_2(m))$ is secure.

Proof by Contradiction:

1. Premise: Let CRHF H_1 be secure, H_2 be broken.
2. Suppose H is NOT secure $\rightarrow$ given m_1, attacker can find m_2 s.t. $H(m_2) = H(m_1) = (H_1(m_1), H_2(m_1)) = (H_1(m_2), H_2(m_2))$.
3. $H(m_1)$ can be split into $H_1(m_1), H_1(m_2)$, and $H(m_2)$ can be split into $H_2(m_1), H_2(m_2)$.
4. Since H_2 is broken, given m_1, attacker is able to find m_2 s.t. $H_2(m_2) = H_2(m_1)$.
5. Since H is not secure by line 2, this also means the attacker can use the same m_2 s.t. $H_1(m_1) = H_1(m_2)$, to allow $(H_1(m_1), H_2(m_1)) = (H_1(m_2), H_2(m_2))$.
6. This means H_1 is no longer secure.
7. Contradiction with premise.

The same argument applies if H_1 is broken and H_2 is still secure.
Therefore, H is still secure if either H_1, H_2 is secure.

4.2

$H'(m) = H_1(H_2(m))$ is not secure if one of its component hash functions is not secure.

Proof by Construction:

1. Case 1: Let CRHF H_1 be secure, H_2 be broken.

1.1 This means given m_1, attacker can find m_2 s.t. $H_2(m_1) = H_2(m_2)$

1.2 Thus $H_1(H_2(m_1)) = H_1(H_2(m_2))$ because hash functions are deterministic.

1.3 Effectively, given m_1, attacker has found m_2 s.t. $H'(m_1) = H'(m_2)$, even if H_1 is still secure.

1.4 As such, H' is not secure even if H_1 is secure and H_2 is not.

2. Case 2: Let CRHF H_2 be secure, H_1 be broken.

2.1 This means given $m_1 = H_2(m_1')$, attacker can find m_2 s.t.

$H_1(m_1) = H_1(m_2)$.

2.1.1 m_2 in this case need not be the result of $H_2(m_2')$, $m_1' \neq m_2'$, since H_2 is secure.

2.2 So even if H_2 is secure, line 2.1 already means H' is not secure: given $m_1 = H_2(m_1')$, attacker could find m_2 s.t. $H(m_1') = H_1(m_1) = H_1(m_2)$

As such, H' need not be secure even if one of H_1 or H_2 is secure.

5 Multicast MACs

5.1

Assumption: Secure MAC means that the keys are drawn randomly and uniformly from a very large keyspace such that it is infeasible to find the key by brute force, and that each key is generated independent of others.

$$P(K_n = k_n | K_1 = k_1, K_2 = k_2, ..., K_{n-1} = k_{n-1}) = P(K_n = k_n)$$

Where K_i represents the random variable for the key assigned to user i.

As such, even if $n-1$ users collude, the probability of them guessing K_n will not change. Since the MAC is secure, it is computationally infeasible to obtain a valid tag for a given message without knowing the key, and the probability of guessing K_n is not high in the first place, so they would not be able to send a unauthentic message with a valid tag.

5.2

Assumption: U_1 does not collude with anyone else, only knows his own keys J_1 and does not know what other keys there are in the key space.

Since there are 4 keys, and no two users have the same key, there is at least 1 key in J_6 that is unknown to U_1, and thus he cannot guess J_6 ($P(\text{guessing } J_6 = 0)$). As such, assuming the MAC requires both keys, and given that the MAC is secure, generating a valid tag without knowing the keys is computationally infeasible, hence this scheme is secure if U_1 does not collude.

5.3

Assumption: user only knows his own keys and does not know what other keys there are in the key space. Keys are drawn randomly and uniformly from a very large keyspace such that it is computationally infeasible to find the key by brute force.

1. Case 1: Let U_1, U_2 collude, and let the two sets of keys be different.
    - 1.1 As there are only 4 keys, U_1 will then know all possible keys.
    - 1.2 He then knows J_6 is made up of two of the 4 keys from $\{J_1, J_2\}$.
    - 1.3 As there are 6 possible subsets of 2 keys from these 4 keys ($\binom{4}{2} = 6$), but 2 of them are J_1, J_2, so excluding these 2, U_1 can guess J_6 with probability $\frac{1}{6-2} = \frac{1}{4} > 0$.
2. Case 2: Let U_1, U_2 collude, and let the two sets of keys have 1 shared key.
    - 2.1 Even though only 3 out of 4 keys are known and there is 1 more unknown key, there is now a chance that J_6 is made up of the two unshared keys of J_1, J_2.
        - 2.1.1 Let $J_1 = \{k_1, k_2\}, J_2 = \{k_2, k_3\}$, one of the remaining subsets must be $\{k_1, k_3\}$.
    - 2.2 Out of the 6 subsets of keys, J_1, J_2 are used, and one of the 4 remaining subsets $J_3, ..., J_6$ is the subset from line 2.1.1, so J_6 has $\frac{1}{4}$ chance of being $\{k_1, k_3\}$.
    - 2.3 There is thus now a non-zero chance of U_1 guessing J_6. Or if U_1 is not targeting U_6 specifically, since $\{k_1, k_3\}$ must be one of $J_3, .., J_6$, U_1 knows the set of keys for one user in $U_3, ..., U_6$

Therefore the scheme is only secure if no 2 users collude.

Proof extensible to general l keys and n users/ subsets, and k number of colluders.

6 Finding Many Collisions

6.1

Let there be N messages to hash. As the hash values are uniformly distributed, the chance of collision is $\frac{1}{N}$.
Let i be the number of evaluations needed. With i evaluations, there will be $\binom{i}{2} = \frac{i(i-1)}{2}$ pairs of hash values. Let X represent the random variable for number of collisions in these hash values. $E(X) = \frac{i(i-1)}{2} \times \frac{1}{N} = \frac{i(i-1)}{2N}$.

Thus if s pairs of collisions are wanted,
$E(X) = s = \frac{i(i-1)}{2N} \rightarrow i \approx \sqrt{2sN}$

So $O(\sqrt{sN})$ evaluations of H are sufficient to find s collisions.

6.2

While I am unable to prove the given expression, I found a research paper potentially suggesting otherwise:
Suzuki, K., Tonien, D., Kurosawa, K., & Toyota, K. (2006). Birthday Paradox for Multi-collisions. Lecture Notes in Computer Science, 29–40. doi:10.1007/11927587_5

In their conclusion, the writers mentioned that:

> "By hashing about $n^{(s-1)/s}$ times, an s-collision is found with probability at most $\frac{1}{s!}$."

If this is so, for $\frac{1}{s!} \geq \frac{1}{2}$, $s! \leq 2 \rightarrow s < 2$, which means $s = 1$ in the context of this question. However, with $s = 1$, it would be trivially true that $H(m_1) = H(m_1)$ with probability at least $\frac{1}{2}$.

7 Logjam Attack

7.1

Yes, checking the integrity of "ServerHello" message in step 3 can prevent this attack. The necessary condition for this attack is the compromise on integrity between the server and the client. If the client is able to verify the "ServerHello" message, he can tell when the message is tampered with, and can thus discard it.

7.2

Assumption: "avoid step 4" means the attacker can discard the message sent by the server to the client, and still be able to decrypt the conversation between the client and the server.

The attacker can do this by employing replay attack. He could have captured a previously valid message sent from server to client containing g, g^b, signed with the server's private key.

8 Understanding Security Principles

8.1 A1: NopCommerce - Integration of PayPal Standard

8.1.1 Principle 3

This protocol seems to violate Principle 3 as the identity of the principals shopper and merchant are both important, yet their names are not always explicitly mentioned in the messages. This claim is made based on the workflow provided in the paper as well as its description, which made no mention of the receiver name in RT3.b, RT3.a.b.

8.1.2 Principle 11

This protocol violated Principle 11 as none of the messages are signed. The protocol depends on trust between the shopper and merchant, and between the shopper and CaaS. Yet, with no way to verify authenticity of any message sent, it is not acceptable for the protocol to assume that authenticity can be ensured. This violation leaves the scheme vulnerable to tampering of messages.

8.2 B3: Interspire - Integration of Google Checkout

8.2.1 Principle 11

This protocol violated Principle 11 as the processing of shopper's payment with CaaS and invoking merchant handler for checking out of goods are separate operations and not atomic. When the merchant handler is invoked for check out, it also loads the shopper's cart from the shopper session, rather than from CaaS. This can result in inconsistency between what the shopper paid for and what he actually checked out, as he can pay for a cheaper product and then add more expensive products to cart. The protocol depends on trust between the shopper's payment and the corresponding goods. However, it assumed that the payment amount and goods checked out will correspond, which is not acceptable given that there is no procedure to guarantee so.