NUS CS-3235: Computer Security September 4, 2020

Assignment 1

Lecturer: Reza Shokri *Student: Name Number*

Instructions

- All submissions should be in PDF and should be made via LumiNUS. You are strongly encouraged to use LaTeX (use the provided template).
- Indicate your name and your student number on the top of your file.
- Your file name should be your student number (e.g., A01xxx.pdf).
- Use font size 12. Your submission needs to have clear sectioning (similar to this document), one for each solution tag in this document (i.e., one section for each of the following: Solution 1.1, Solution 2.1, Solution 2.2, ...)
- Your can use this LaTeX template, compile it to PDF, and submit the PDF file.
- We will ignore all the submissions that violate any of the above.
- You have exactly 2 weeks to submit your solutions (no credit is given to late submissions).
- Be concise; excessively long answers will be penalized.
- **No collaboration** is permitted on this assignment. Any cheating (e.g., copying from any source, or permitting your work to be copied) will result in a failing grade. The assignments are provided to help you learning by solving problems. What's the point of obtaining the solutions from elsewhere? Your grade won't help you securing your code.

1 Think like an Adversary (15 pts)

You are sitting in Starbucks, enjoying the free Wi-Fi provided by the cafe. You go online and buy tickets for a movie, and enter your credit card details into the ticket-booking portal. Put on an attacker's hat, and list out at least seven different ways in which an attacker can obtain your credit card details. Mention the attacker's capabilities and other assumptions that you make for each suggested approach.

2 Perfectly Secret or Not (10+10 = 20pts)

For each of the following schemes, state whether the scheme is perfectly secret or not. Justify your answer in each case.

1. Message space $M = \{0, ..., 4\}$. Algorithm Gen chooses a uniform key from key-space $K = \{0, ..., 5\}$, and $Enc_k(m)$ returns $[(k + m) \mod 5]$, and $Dec_k(c)$ returns $[(c - k) \mod 5]$.

2. Consider a variant of the one time pad with message space $\{0, 1\}^l$ where the key space K is restricted to all l-bit strings with an even number of 1's.

3 Permutation Cipher (13 pts)

Consider the following cipher (E, D) defined over (K, M, C) where $C = M = \{0,1\}^l$ and K is the set of all $l!$ permutations of the set $\{0, \ldots, l-1\}$. For a key $k \in K$ and message $m \in M$, define $E(k, m)$ to be the result of permuting the bits of m using the permutation k, namely $E(k, m) = m[k(0)] \ldots m[k(l-1)]$. Show that this cipher is not semantically secure by showing an adversary that achieves advantage 1.

4 Collision Resistant Hash Functions (5 pts)

We want to build a Collision Resistant Hash Function(CHRF) H with two CHRFs H_1 and H_2, so that if at some future time one of H_1 or H_2 is broken (but not both) then H is still secure.

- Suppose H_1 and H_2 are defined over $(\mathcal{M},)$. Let $H(m) := (H_1(m), H_2(m))$. Show that H is a secure CHRF if either H_1 or H_2 is secure.
- Show that $H'(x) = H_1(H_2(x))$ need not be secure CHRF even if one of H_1 or H_2 is secure.

5 Multicast MACs (5 + 6 + 4 = 15 pts)

Consider a scenario in which Alice wants to broadcast the same message to n users, $U_1, \cdots, U_n$. She wants the users to be able to authenticate that the message came from her, but she is not concerned about message secrecy. More generally, Alice may wish to broadcast a series of messages, but for this exercise, let us focus on just a single message.

- In the most trivial solution, Alice shares a MAC key k_i with each user U_i. When she broadcasts a message m, she appends tags $t_1, \cdots, t_n$ to the message, where t_i is a valid tag for m under key k_i. Using its shared key k_i, every user U_i can verify m's authenticity by verifying that t_i is a valid tag for m under k_i.

  Assuming the MAC is secure, show that the broadcast authentication scheme is secure even if *users collude*. For example, users $U_1, \cdots, U_{n-1}$ may collude, sharing their keys $k_1, \cdots, k_{n-1}$ among each other, to try to make user U_n accept a message that is not authentic.

- While the above broadcast authentication scheme is secure, even in the presence of collusions, it is not very efficient; the number of keys and tags grows linearly in n.

  Here is a more efficient scheme, but with a weaker security guarantee. We illustrate it with $n = 6$. The goal is to get by with $l \leq 6$ keys and tags. We will use just $l = 4$ keys, $k_1, \cdots, k_4$. Alice stores all four of these keys. There are $6 = \binom{4}{2}$ subsets of $\{1, \cdots, 4\}$ of size 2. Let us number these subsets $J_1, \cdots, J_6$. For each user U_i, if $J_i = \{v, w\}$, then this user stores keys k_v and k_w.

  Assuming the MAC is secure, show that this broadcast authentication scheme is secure *provided no two users collude*. For example, using the keys that he has, user U_1 may attempt to trick user U_6 into accepting an inauthentic message, but users U_1 and U_2 may not collude and share their keys in such an attempt.

- Show that the scheme presented in part (2) is completely insecure if two users are allowed to collude.

6 Finding Many Collisions (5 + 7 = 12 pts)

Let H be a hash function defined over $(\mathcal{M}, \mathcal{T})$ where $N := |\mathcal{T}|$ and $|\mathcal{M}| \gg N$. From Birthday Paradox, we know that $O(\sqrt{N})$ evaluations of H are sufficient to find a collision for H with probability 1/2. Based on similar ideas,

- Show that $O(\sqrt{sN})$ evaluations of H are sufficient to find s collisions $(x_0^{(1)}, x_1^{(1)})$, $\cdots, (x_0^{(s)}, x_1^{(s)})$ for H with probability at least 1/2. Therefore, finding a million collisions is only about a thousand times harder than finding a single collision.

- We say that an s-collision for H is a set of s distinct points $x_1, \cdots, x_s$ in $\mathcal{M}$ such that $H(x_1) = \cdots = H(x_s)$. Show that for each constant value of s, $O(N^{(s-1)/s})$ evaluations of H are sufficient to find an s-collision for H, with probability at least 1/2.

7 Logjam Attack (5+5 = 10pts)

The Logjam attack was a famous vulnerability that affected older versions of SSL / TLS. Recall that TLS allows a server and client to negotiate a ciphersuite which they will use for a session as a first step. The Logjam attack is an example of a "re-negotiation" vulnerability, wherein a man-in-the-middle (MitM) attacker forces the server and client to use a weaker ciphersuite than what the client requested. It targets TLS implementations which support "export-grade" (based on 512-bit groups) Diffie-Hellmann ciphersuite, which is known to be insecure. Recall that, in Diffie-Hellman key exchange, Alice and Bob agree on a prime number p and a generator g of a multiplicative subgroup modulo p. Alice then sends $g^a \mod p$ to Bob and he replies with $g^b \mod p$. Now, both of them compute g^{ab} and use it as their shared secret key. In our situation (figure 1), the client C wants to connect to the server S and starts a TLS session. An attacker highjacks the session by performing the Logjam attack as follows -

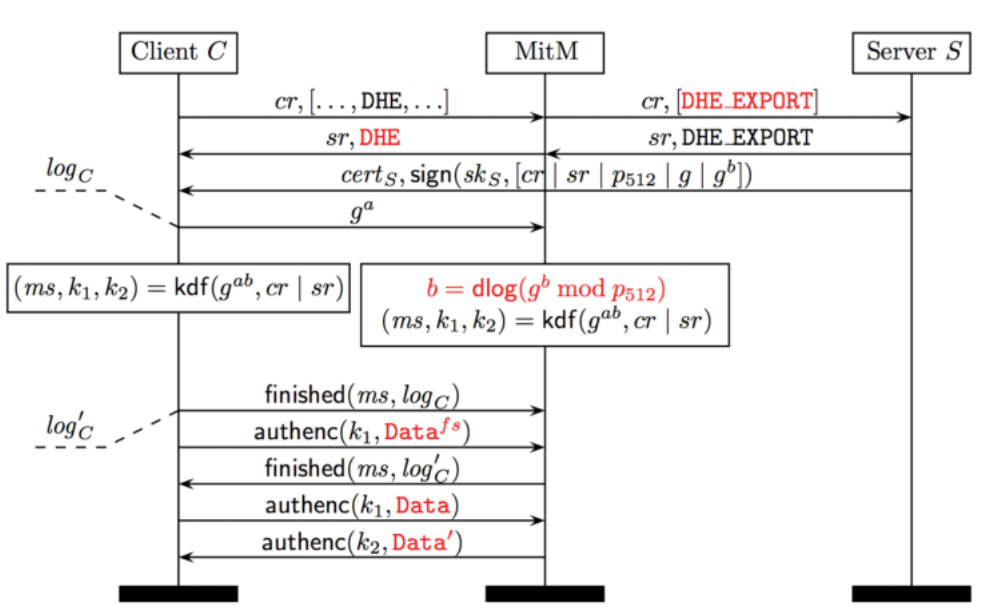

Figure 1: The Logjam attack

1. A client starts a TLS session with a server by sending a ClientHello message along with all the ciphersuites it can offer. The token DHE is specified in the client message to indicate the usage of 1024 bit groups for Diffie Hellman key exchange protocol (which is a secure cipher suite).

2. The attacker armed as a Man in the middle(MitM), intercepts this message and removes all other ciphersuites except DHE_EXPORT from the message. This leaves no choice to the server apart from choosing DHE_EXPORT ciphersuite.

3. The server replies with a ServerHello message indicating the chosen ciphersuite with format $(sr, \text{DHE\_EXPORT})$. The attacker intercepts this message and changes DHE_EXPORT to DHE before forwarding the message to client.

4. The server then chooses a prime number p_{512} and a generator g of a multiplicative subgroup modulo p_{512}. It computes g^b and sends a signed message with format
$(cert_S, sign(sk_S, [cr|sr|p_{512}|g|g^b]))$. Here, $cert_S$ is the public certificate of the server.

5. The client replies with g^a which can be used by the server to compute the shared secret key g^{ab}.

6. However, the attacker intercepts the message and uses a pre-computed value of the discrete log of $(g^b \mod p)$ to get the value of b and hence, the shared secret key, g^{ab}. For the 512 bit groups discrete log can be pre-computed with modest effort.

Here cr and sr are auxiliary constants which can be ignored.

In the given context, explain your answers for the following questions -

(a) Will checking the integrity of "ServerHello" message in step 3 prevent this attack?

(b) Suppose the server S uses same g, b for all future connections with client C and the attacker knows g and b. Can the attacker avoid step 4 by sending g and g^b to C himself?

8 Understanding Security Principles (5+5 = 10pts)

In their "How to Shop for Free Online" paper, Wang et al. describe vulnerabilities in several cashier-as-a-service protocols. For 2 of these protocols (which you choose from the ones listed in Section III), explain how its design violates one or more of the principles listed by Abadi and Needham in their paper on "Prudent Engineering Practice for Cryptographic Protocols." Both these papers are uploaded to LumiNUS.