CNT4406 – Network Security **Spring 2024**

Homework 2: Deadline Tuesday 2/27

Instructor: Viet Tung Hoang

Recall that your solution must be typed via Latex.

1. (**Database authenticity**) (60 points) Let $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a good MAC. Suppose that a database contains records $M_1, \ldots, M_q$. To provide authenticity for the records, the admin generates a random secret key $K \in \{0,1\}^k$ and stores $T_i \leftarrow F_K(M_i)$ alongside record M_i for every $i = 1, \ldots, q$. This does not ensure authenticity because an attacker can remove a record or duplicate a record without being detected. To deal with this, the admin generates another secret key $K' \in \{0,1\}^k$ and computes an additional tag T'. She stores (K, K', T') in her machine, away from the database.

   How should the admin compute T' so that if we update a single record M_i, the cost to update (T_i, T') is cheap, meaning we need to run the MAC to sign messages of total size $O(|M_i| + qn)$? Briefly explain why your solution can detect if an adversary modified the database.

2. (**TCP SYN flood**) (70 points) Recall that in a TCP handshake, a client first sends a SYN message to a server. The server then sends a SYN/ACK message back to the client, and waits for the latter's ACK. During this wait, the server has to *remember* the connection information (client/server IP and ports), as well as the *maximum segment size* (MSS), the maximal size of the data a TCP packet may contain, which is communicated by the client in its initial SYN's message. (We assume here that there are only 8 possible choices for MSS.) In other words, every TCP connection makes the server allocate some memory during the wait for the client's final ACK message. This can be exploited for a denial-of-service attack as follows: send lots of TCP SYN packets to the server, but never send the final ACK messages. The server will then quickly run out of memory.

   Consider now the following choice of the server's sequence number for a connection. The server chooses the 32-bit sequence number s defined as follows:

   - The first 5-bit are a time-counter t, increased every minute, and reduced modulo 32 to fit in 5 bits.
   - The middle 3 bits (denote them as m) encode 8 possible values for the MSS field.
   - The last 24 bits are the output σ of a PRF (using a secret key only known to the server) applied to the concatenation of the server IP address and port number, the client IP address and port number, and the values t and m.

   In other words, $s = t \| m \| \sigma$, where $\|$ denotes concatenation.

   Explain how this choice of s can be used by the server to prevent SYN flooding.

3. (**TCP SYN flood, second take**) (70 points) The following countermeasure against SYN flood attacks is widely adopted, and is known as a "SYN cache". For simplicity, we assume that the server only responds on port 80, and only needs to remember the client's sequence number C and its own sequence number S between the second and third message in the TCP handshake. The server maintains a table with exactly B cells (e.g., $B = 2^{16}$), numbered $1, \ldots, B$, and each cell in the table can store an entry with form

$$(\text{ClientIP}, \text{ClientPort}, S, C) .$$

   In particular, the handshake is now modified as follows:

- When a client with IP ClientIP initiates the handshake with the server, sending a SYN message from port ClientPort to the server's port 80 with sequence number C, the server generates its initial sequence number S, and stores $T = (\text{ClientIP}, \text{ClientPort}, S, C)$ in the table at location i_T, where i_T is a deterministic function of T. (E.g., one applies a PRF to T, and then maps the PRF output to a number between 1 and B.) **If cell i_T already stores something, its contents are replaced by T.** The server then sends back a SYN/ACK message with sequence number S and ACK number $C + 1$. *The server does not remember anything else in its memory.*
- When the server receives an ACK message from port ClientPort of a client with IP address ClientIP with sequence number $C + 1$ and ACK number $S + 1$, the server checks whether $T = (\text{ClientIP}, \text{ClientPort}, S, C)$ is at location i_T (note that i_T can be recomputed as above). If yes, it continues the TCP connection as usual. If not, it simply aborts the handshake.

a) (35 points) Why does a SYN cache mitigate the effects of SYN flood attacks?

b) (35 points) What is a possible disadvantage of using a SYN cache?