COSC 4765
Computer Security

Name: Jacob Tuttle

Homework 03

September 14, 2020

1. Nonces

1. The technical requirements for a sequence number based nonce are relatively simple. Having a transmitter simply increment and send the next number in sequence is quite simple. Generating a random number would require slightly more processing and the ability to actually produce a truly random number. It would also require the evaluator to keep track of a wide range of random numbers in order for them to truly be a nonce. The primary technical problem with using a timestamp nonce is properly synchronizing the time across the two devices so that they recognize each other's times as valid.

2. The sequence number and timestamp nonces become completely useless without the secret of the key KT. If a malicious agent is able to intercept a transmission under the sequence nonce paradigm, they can simply increment that nonce and send their newly encrypted message to fool the evaluator. In the case of the timestamped nonce, assuming they're able to properly synch their time with the evaluator, they could send a message easily under the KT key. The random number nonce is slightly more difficult to be used by an attacker because they have no idea what random numbers have been previously used by the system; however, this is easily circumvented by simply spamming multiple random numbers as eventually one will eventually have never been used. Clearly, keeping the key KT secret is fundamental to the security if any of these nonce systems.

3. It seems as though the idea of a random nonce maintains the most value or the most possibility to be salvaged. Unless an attacker is able to view the system for its entire lifetime, there is no way of knowing what nonces have already been used, which seems like a valuable feature in a security system. Beyond that, all three of the systems seem as though they could have value if the key can be kept truly secret. Low security environments could also find a use for the timestamp or sequential nonces; if the secret being protected isn't incredibly important, a lower security system (or if the nonce acted in the capacity of a secondary security system) might be appropriate.

2. Protocol - Prepayment Meters

The system for evaluating prepayment meters input for a valid code can be described as the following protocol. A customer, C, inputs a token, T, encrypted with a key, K, into a meter, M. The meter then checks if the token is different from the previous token, T_{prev}. If it is, it enables the power output and outputs the purchased amount of electricity, E.[1]

$$
\begin{aligned}
&C \to M : \{T\}_K \\
&M \to C : \{T\}_K \neq \{T_{prev}\}_K \ ?\ E :\ 0
\end{aligned}
$$

Using this description of the system's protocol, we can then formally describe the attack from the text as:

$$
\begin{aligned}
&C \to M : \{T_1\}_K \\
&M \to C : \{T_1\}_K \neq \{T_{prev}\}_K \ ?\ E :\ 0 \\
&C \to M : \{T_2\}_K \\
&M \to C : \{T_2\}_K \neq \{T_1\}_K \ ?\ 2E :\ 0 \\
&C \to M : \{T_1\}_K \\
&M \to C : \{T_1\}_K \neq \{T_2\}_K \ ?\ 3E :\ 0 \\
&\qquad\vdots \\
&C \to M : \{T_2\}_K \\
&M \to C : \{T_2\}_K \neq \{T_1\}_K \ ?\ E' :\ 0
\end{aligned}
$$

This attack exploits the fact that the protocol only remembers the previously input token to produce an infinite output of electricity ($2E$ is used to denote twice the amount of expected energy, $3E$ triple the amount, and E' to denote some excessive amount achieved through repeated exploitation). After the first token is input, T_1, you can alternate between tokens T_2 and T_1 to fool the protocol into outputting as much electricity as you have time to input tokens for.

[1] This if notation mirrors the ternary operator of ?: from various programming languages