**CNT5412 – Network Security** **Spring 2025**

# Homework 2: Deadline Monday 3/31

*Instructor: Viet Tung Hoang*

**1.** (**Android Keystore Attack**) (60 points) Let $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a good blockcipher. Let CBC[$E$].Enc$(K, M)$ denote the encryption of a message $M$ under the CBC mode of blockcipher $E$ under key $K$, and define CBC[$E$].Dec$(K, C)$ for decryption similarly. For simplicity, assume that here we only deal with full-block messages. Let $H : \{0,1\}^* \to \{0,1\}^n$ be a hash function.

A recent implementation in Android Keystore uses the following authenticated encryption scheme: to encrypt a message $M$ under the key $K$, we output CBC[$E$].Enc$(K, H(M)\|M)$. For decryption, given the key $K$ and ciphertext $C$, we first run CBC[$E$].Dec$(K, C)$ and parse the result as $T\|M$, where $|T| = n$. We then output $M$ if $T = H(M)$, and output $\perp$ otherwise. In some sense, it's similar to the Encrypt-with-Redundancy paradigm that we studied and broke in class. However, the main difference here is that the redundancy $H(M)$ is put at the *beginning* of the message, instead of at the end.

Show that the Android encryption scheme is insecure by giving an authenticity attack.

**2.** (**S/key Identification Scheme**) (65 points) In the setting of one-time passwords, we have a client and a server that share a key, and the client has to prove his identity to the server. If an adversary can somehow steal the key from the server then all is lost. This is what happened to RSA's SecureID due to a hack at Lockheed Martin.

The S/Key identification scheme is an alternative to one-time passwords in which the client and server have *different* keys $K_c$ and $K_s$, and the server key is *public*.

To set up the scheme, we generate a random string $L \leftarrow\!\!\$\ \{0,1\}^k$ (where $k$ is sufficiently large, say $k \geq 128$). Let $H$ be a cryptographic hash function (e.g., HMAC-SHA-256). Let $H^n$ denote the $n$-th iteration of $H$, that is, $H^1(x) = H(x)$, and $H^i(x) = H(H^{i-1}(x))$ for every $i \geq 2$. Then, the client key is $K_c = (L, n)$, and the server key is $K_s = H^{n+1}(L)$ for some integer $n > 0$.

The scheme is then stateful (i.e., both the client and the server update $K_s$ and $K_c$ respectively). Concretely, upon running on input key $K_c = (L, i)$ for $1 \leq i \leq n$, the client sends $t = H^i(L)$ to the server, and sets $K_c = (L, i-1)$. If $i = 0$, the client does not do anything.

a) (20 points) Given a verifier key $K_s$ and a token $t$, how should the server validate it? How should it update the key $K_s$?

b) (25 points) Explain informally why – assuming $H$ is a good hash function – S/Key is a secure identification scheme against eavesdropping attackers.

c) (20 points) How would you choose the value $n$? Explain your choice.