Lecture 20: Message Authentication Codes from PRF

### Outline

- In the previous lecture, we defined MACs and their security and constructed them using random functions
- In today's lecture, we shall construct MACs using pseudo-random functions

#### Scheme.

- Secret-key Generation. Sample sk uniformly at random from  $\{0,1\}^{n/100}$  and provide sk to both the sender and the verifier
- Tagging a message  $m \in \{0,1\}^n$ . The sender computes tag  $\tau = g_{\rm sk}(m)$  (evaluate using the GGM construction, where we consider functions  $\{0,1\}^n \to \{0,1\}^{n/100}$  and id in  $\{0,1\}^{n/100}$ )
- Verifying a message-tag pair  $(\widetilde{m}, \widetilde{\tau})$ . Check whether  $\widetilde{\tau}$  is same as  $g_{\rm sk}(\widetilde{m})$  or not

#### Security

• An adversary cannot forge if it sees t message-tag pairs, where  $t = \operatorname{poly}(n)$  and the adversary is computationally bounded If an adversary can forge a signature in this case, we can distinguish the random functions from pseudo-random functions. Because, in the former case, forgeability was impossible for any adversary. However, in the latter case, this adversary makes forgeability possible. s

# The scheme mentioned above is secure ONLY for messages in $\{0,1\}^n$ and NOT $\{0,1\}^*$

What does it mean?

- The set  $\{0,1\}^n$  represents n-bit messages, and  $\{0,1\}^*$  represents arbitrary-length messages. This scheme is secure only when an adversary sees message-tag pairs for messages  $m_1, m_2, \ldots, m_t$  such that all of them have identical length n. Moreover, the adversary has to forge by producing  $(m', \tau')$  pair such that the length of the message m' is exactly n.
- The scheme is <u>not</u> secure if the adversary can produce a message of a different length. The attack is explained in the next slide

## Adversarial strategy to forge a message-tag pair of different lengths.

- Suppose the adversary has seen a message-tag pair  $(m, \tau)$  such that  $\tau = F_{\rm sk}(m)$
- The adversary creates m'=m0 (i.e., the message m concatenated at the end with 0). The adversary computes  $\tau'$  as the first half of  $G(\tau)$ .
- Verify that  $F_{\mathsf{sk}}(m') = au'$
- In fact, the adversary can successfully tag any m' such that m is the prefix of m'

## Lesson Learned (Very Important)

- The sender and the verifier should establish one secret key sk for EACH length of the message they want to sign. For example
  - They establish a secret-key sk  $\in \{0,1\}^k$  for 1024-bit messages and use  $F_{sk}(m)$  as the tag for 1024-bit messages m
  - If they want to tag 2048-bit messages, then they establish a new secret-key  $\operatorname{sk}' \in \{0,1\}^k$  and use  $F_{\operatorname{sk}'}(m)$  as the tag for 2048-bit messages m
  - The verifier should <u>only</u> check the validity of the tags corresponding to 2048-bit messages using the secret key associated with message length 2048 (in our case, it is the secret key sk')

### Food for thought

 Suppose we want to construct a MAC so that if t-parties among a set of n-parties decide to endorse a message m, they can add a tag that the verifier can verify. How to construct such a scheme?