Lecture 23: Pseudo-random Functions

Motivation I

- In the following slides, we will construct a MAC using Random Functions
- Understand its properties and its shortcomings

Motivation II

Goal.

- Suppose we have n-bit messages, i.e., the message space is $\{0,1\}^n$
- We will generate $n/100$-bit tags, i.e., the space of tags is $\{0,1\}^{n/100}$

Motivation III

Scheme.

- Secret-key Generation Algorithm.
  - Let F be a random function from the domain $\{0,1\}^n$ to the range $\{0,1\}^{n/100}$
  - Let the secret key sk be the function table of F
  - Both the sender and the verifier will share the secret-key $\mathsf{sk} = F$
- Tagging Algorithm.
  - The tag $\tau\{0,1\}^{n/100}$ for a message $m\{0,1\}^n$ using the secret key $\mathsf{sk} = F$ is computed by: $\tau = F(m)$
  - To endorse the message m, the sender will send the pair (m, τ)
- Verification Algorithm.
  - The verifier will receive a pair $(\widetilde{m}, \widetilde{\tau})$
  - The verifier will check whether $\widetilde{\tau} = F(\widetilde{m})$, where the secret-key $\mathsf{sk} = F$

Motivation IV

Analysis of Adversarial Attack.

- Suppose the adversary sees a pair (m, τ)
- The adversary does not know the secret-key $\mathsf{sk} = F$, but it knows that $F(m) = \tau$
- Now, the adversary has to generate a different message $m' \in \{0,1\}^n$ and a tag τ' such that the pair (m', τ') verifies
- The adversarial pair (m', τ') will verify if an only if $F(m') = \tau'$
- Let us look at this probability

$$\mathbb{P}\left[F(m') = \tau' | F(m) = \tau\right]$$

  - Let us parse this mathematical expression. The adversary already knows the fact that "$F(m) = \tau$." So, we are conditioning on that fact in the probability expression. And, conditioned on this fact, we are interested in finding the probability that $F(m') = \tau'$.

Motivation V

- First observation. Given the fact that $F(m) = \tau$ (i.e., evaluation of a function at one input) the evaluation of $F(m')$ is uniformly random over the range. Because, for a random function, given the evaluation of a function at one input, the evaluation of the function F at any other input is uniformly random over the range.
- So, conditioned on the knowledge of the adversary that $F(m) = \tau$, the probability that $F(m') = \tau'$, where $m' \neq m$, is "1 divided by the size of the range." In our case, that is

$$\frac{1}{2^{n/100}}$$

- Therefore, we conclude

$$\mathbb{P}\left[F(m') = \tau' | F(m) = \tau\right] = \frac{1}{2^{n/100}}$$

Conclusion.

- It is highly unlikely that an adversary will be able to forge a tag given one (m, τ) pair

Extension.

- In fact, this scheme has an even more interesting property
- Suppose the sender has sent several message-tag pairs. That is, the sender has sent (m_1, τ_1), (m_2, τ_2), ..., (m_t, τ_t). Note that they satisfy the following relation $\tau_1 = F(m_1)$, $\tau_2 = F(m_2)$, ..., $\tau_t = F(m_t)$.
- The adversary has seen all these message-tag pairs. Can the adversary forge a new message-tag pair? Let us see.

Analysis of the Probability of Forging in the Extension.

- Let us write down what the adversary has seen. The adversary knows that

$$F(m_1) = \tau_1, F(m_2) = \tau_2, \ldots, F(m_t) = \tau_t$$

- Conditioned on this information, we are interested in the probability that $F(m') = \tau'$, where m' is different from all the messages $m_1, m_2, \ldots, m_t$
- So, we are interested in the probability

$$\mathbb{P}\left[F(m') = \tau' | F(m_1) = \tau_1, F(m_2) = \tau_2, \ldots, F(m_t) = \tau_t\right]$$

- Main Observation. Even if we know the evaluation of the function F at inputs m_1, m_2, ..., m_t, the evaluation of F at a new input m' is uniformly random over the range. So, we can conclude that the probability of forging is

$$\mathbb{P}\left[F(m') = \tau' | F(m_1) = \tau_1, F(m_2) = \tau_2, \dots, F(m_t) = \tau_t\right] = \frac{1}{2^{n/100}}$$

Conclusion.

- The MAC using random function to generate tags is secure even when the adversary see t message-tag pairs (for any value of t less than the size of the range, i.e., $t < 2^n$)

Positive Features.

- Even if the adversary has unbounded computational power, the probability arguments bounding its probability to forge still holds
- Recall that if we use 2-wise independent hash functions or universal hash functions instead of the random function, our MAC is secure only when $t = 1$. This new scheme is secure even for larger values of t
- Recall that, in general, if we used k-wise independent hash functions instead of the random function, our MAC is secure only when $t \leqslant (k-1)$. This new scheme is secure even for large values of t

Primary Shortcoming.

- Let us compute the size of the function-table for the function F. Recall that F is from the domain $\{0,1\}^n$ to the range $\{0,1\}^{n/100}$. So, there are a total of $\left(2^{n/100}\right)^{2^n} = 2^{(n/100)2^n}$ different functions. This implies that we need $(n/100)2^n$ (exponential in n) bits to represent this function! Even for $n = 512$, this number is larger than the number of atoms ($< 2^{273}$) in the entire universe.

What Next?

To fix the shortcoming mentioned above, we set forth the following goals for ourselves

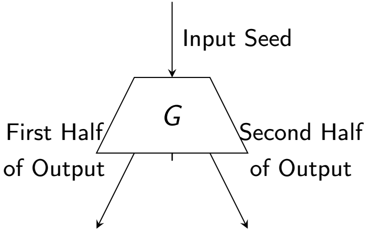
- We will construct functions that use smaller key, i.e., length is polynomial in n

However, our security will hold only for computationally bounded adversaries (instead of adversaries with unbounded computational power)

- Solution: We replace "random functions" with "pseudo-random functions" (PRF) (i.e., functions that "look" like random functions for computationally bounded adversaries)

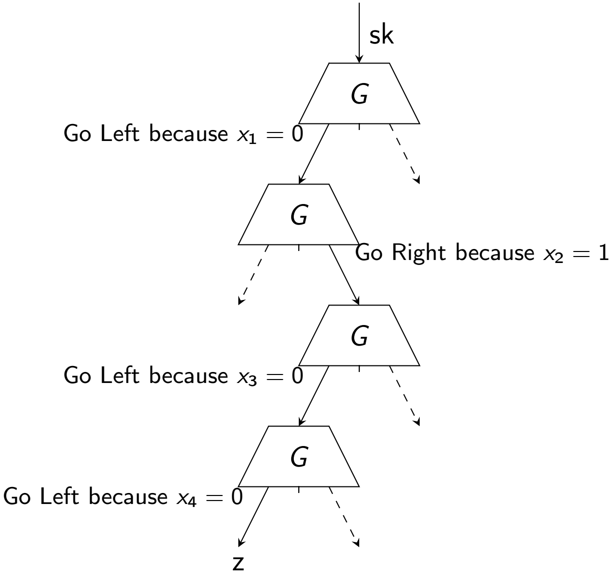
Notation.

- We will construct pseudorandom functions from the domain $\{0,1\}^n$ to the range $\{0,1\}^k$
- A length-doubling PRG $G\colon \{0,1\}^k \to \{0,1\}^{2k}$. We will pictorially represent as follows



The "GGM" stands for Goldreich-Goldwasser-Micali (the name of the founders)

- Let us understand the function evaluation with an example
- Let $n = 4$ and the input be $x = x_1x_2x_3x_4 = 0100$ For $\mathsf{sk} \in \{0, 1\}^k$, the evaluation of the function $F_{\mathsf{sk}}(x)$ is defined to be z computed as follows (see the next slide for the figure)


sk
G
Go Left because $x_1 = 0$
G
Go Right because $x_2 = 1$
G
Go Left because $x_3 = 0$
G
Go Left because $x_4 = 0$
z

Comments.

- For each value of sk $\in \{0,1\}^k$, we have a function F_{sk}. Instead of storing the entire function table of F_{sk}, we can now only store the sk (a k-bit long string). To compute $F_{\mathsf{sk}}(x)$, we compute the function on the fly, as described in the previous slide.
- If the input x is n-bit long, then the tree is evaluates till depth n
- Think: How to make a dedicated hardware to implement the GGM construction

MAC using Pseudorandom Functions I

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 $\tau = F_{\mathsf{sk}}(m)$ (evaluate using the GGM construction)
- Verifying a message-tag pair $(\widetilde{m}, \widetilde{\tau})$. Check whether $\widetilde{\tau}$ is same as $F_{\mathsf{sk}}(\widetilde{m})$ or not

Security

- An adversary cannot forge if it sees t message-tag pairs, where $t = \mathrm{poly}(n)$ and the adversary is computationally bounded

Subtlety I

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 see 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', τ') pair such that the length of the message m' is exactly n.
- The scheme is not secure if the adversary can produce a message of a different length. The attack is explained in the next slide

Subtlety II

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

- Suppose the adversary has seen a message-tag pair (m, τ) such that $\tau = F_{\mathsf{sk}}(m)$
- The adversary creates $m' = m0$ (i.e., the message m concatenated at the end with 0). The adversary computes τ' as the first half of $G(\tau)$.
- Verify that $F_{\mathsf{sk}}(m') = \tau'$
- 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 that they want to sign. For example
  - They establish a secret-key $\mathsf{sk} \in \{0,1\}^k$ for 1024-bit messages and use $F_{\mathsf{sk}}(m)$ as the tag for 1024-bit messages m
  - If they want to tag 2048-bit messages, then they establish a new secret-key $\mathsf{sk}' \in \{0,1\}^k$ and use $F_{sk'}(m)$ as the tag for 2048-bit messages m
  - The verifier should only 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')