MÁSTER INTERUNIVERSITARIO EN CIBERSEGURIDAD

Information Security
Lab assignment II

1 Problem Description: bit commitment

In this exercise you will implement a secure commitment protocol between two parties, using a secure pseudo-random generator (PRG) as a building block. First, let us describe the problem of distributed commitment between `Alice` and `Bob`.

Suppose they have to decide between two choices —say, holidays at a beach resort or a week trekking in Vietnam. To that end, one of the parties flips a coin and if the outcome happens to be heads, Alice's choice wins. The problem is that Alice and Bob are at different locations, they cannot see each other, so she has no reason to believe that Bob is telling the truth after he flips the coin.

A solution to this problem uses the idea of **bit commitment**. Bob picks a bit $b \in \{0,1\}$ and commits to it. Later, Bob can open the commitment and prove to Alice that b was the value he committed to. Committing to a bit b results in a commitment string c, that Bob send to Alice, and an opening string s that Bob uses for opening the commitment later. The system is secure if it satisfies the following two properties:

- **Hiding**: the commitment string c should not reveal information about b.
- **Binding**: if c is Bob's commitment string, then Bob must be able to open the commitment as some $b \in \{0,1\}$ but not as $\overline{b}$.

2 Coin flipping protocol

Assuming that a commitment scheme is in use, let us show how Alice and Bob can generate a random bit without a bias to their particular preference.

1. Bob generates a random bit $b_0 \leftarrow \{0,1\}$. Then both parties, Alice and Bob execute the commitment protocol; Alice obtains a commitment c to b_0, and Bob obtains an opening string s.
2. Alice chooses a random bit $b_1 \leftarrow \{0,1\}$ and sends b_1 to Bob.
3. Bob opens the commitment by revealing b_0 and s to Alice. Alice verifies that c is really a commitment to b_0 and aborts if the verification fails.
4. The decision bit is $b = b_0 \oplus b_1$.

This coin flipping protocol can be proven to be correct, in that no one of the parties can drive the result to its own interest. This is due to the two properties above, hiding and binding.

The remaining piece for completing the protocol is to explain the commitment strategy.

3 Bit commitment protocol

Here, we specify how Bob can produce a secure bit commitment. To that end, let $G : \mathcal{S} \longrightarrow \mathcal{R}$ be a secure PRG where $|\mathcal{R}| \geq |\mathcal{S}|^3$ and $\mathcal{R} = \{0,1\}^n$ for some n. In order to commit to the bit b_0, Alice and Bob execute this sequence of actions: for Bob to commit to $b_0 \in \{0,1\}$

1. Alice chooses a random $r \in \mathcal{R}$ and send r to Bob.

2. Bob chooses a random $s \in \mathcal{S}$ and computes $c \leftarrow \text{commit}(s, r, b_0)$ where commit is the function

$$c = \text{commit}(s, r, b_0) = \begin{cases} G(s) & \text{if } b_0 = 0, \\ G(s) \oplus r & \text{if } b_0 = 1. \end{cases}$$

3. Bob outputs c as the commitment string and uses s as the opening string. When the commitment has to be verified, Bob sends (b_0, s) to Alice, and she accepts the opening if $c = \text{commit}(s, r, b_0)$.

4 Vector committment

Let us generalize the elementary bit committment protocol to vectors. One straightforward way of committment of a string $b_0, b_1, \ldots, b_{n-1}$ of n bits is just to create a committment for each bit b_i. However, this is clearly inefficient, since we can do better if we reuse the same pseudorandom sequence $G(s)$ for committing more than a single bit. Here is the prescription.

Fix n, $q = 3m$. The protocol for committing the bit sequence $\mathbf{b} = (b_1, b_2, \ldots, b_m)$ is the following:

- Commit stage

  1. Bob generates a random vector $\mathbf{r} = (r_1, r_2, \ldots, r_{2q})$ of bits where exactly q of its bits are equal to 1 and sends $\mathbf{r}$ to Alice.
  2. Alice forms $\mathbf{c} = (\mathbf{b}, \mathbf{b}, \mathbf{b})$ —this is a repetition code on $\mathbf{b}$—, selects a seed $s \in \{0,1\}^n$ and sends to Bob $\mathbf{e} = \mathbf{c} \oplus G_{\mathbf{r}}(s)$, where the XOR is calculated bitwise. Here, $G_{\mathbf{r}}(s)$ is the subsequence of $G(s)$ formed by the bits of $G(s)$ at the indices j where $r_j = 1$. Also, for each $1 \leq i \leq 2q$ such that $t_i = 0$, Alice sends the i-th bit of $G(s)$.

- Prove and verify stage.

  1. Alice sends $(s, b_1, \ldots, b_m)$ to Bob.
  2. Bob verifies for each $1 \leq i \leq 2q$ with $r_i = 0$ that Alice sent the correct sequence. Bob computes $\mathbf{c}' = (\mathbf{b}, \mathbf{b}, \mathbf{b})$ and verifies that $\mathbf{e} = \mathbf{c}' + G_{\mathbf{r}}(s)$.

5 Task

Your task is to implement the above vector bit commitment protocol in Python (or another language of your preference). Do the following:

- Write two programs for the computations of Alice and Bob.
- The programs for Alice and Bob should communicate over the MQTT server used in Lab Assignment I. You are free to design the format of the messages exchanged between the parties and to use (or not) nested encryption for sending and receiving the messages.
- Test your program and submit the code to the instructors.

References

[1] M. Naor. "Bit commitment using psueod-randomness". In CRYPTO, pp. 128.136, 1989.