Fundamentos y Aplicaciones de Blockchains

Homework 1

Depto. de Computación, UBA, 2do. Cuatrimestre 2025 *4/9/25*

Student: *Due: 18/9/25, 17:00 hs*

Instructions

- Upload your solution to Campus; make sure it's only one file, and clearly write your name on the first page. Name the file '<your last name>_HW1.pdf.'

  If you are proficient with LaTeX, you may also typeset your submission and submit in PDF format. To do so, uncomment the "%\begin{solution}" and "%\end{solution}" lines and write your solution between those two command lines.

- Your solutions will be graded on *correctness* and *clarity*. You should only submit work that you believe to be correct.

- You may collaborate with others on this problem set. However, you must **write up your own solutions** and **list your collaborators and any external sources (including ChatGPT and similar generative AI chatbots)** for each problem. Be ready to explain your solutions orally to a member of the course staff if asked.

This homework contains 7 questions, for a total of 80 points.

1. This question is about Merkle Trees (MTs).
   (a) (3 points) Describe how (cryptographic) hash functions are used in a MT.
   (b) (3 points) Describe how a (complete, binary) MT is constructed for the following five chunks of data: ABC, DEF, GHI, KLM, OPQ.
   (c) (4 points) Describe how a Patricia Trie is constructed for the following key/value store: {blah: 17, blahblah: 28, bored: 53, board: 39, aboard: 42, abroad: 17}.

2. Cryptographic hash functions:
   (a) (5 points) Derive the formula for the *birthday paradox* we saw in class (show your work, explaining every step) and calculate the number of elements needed to find a collision with at least 50%.
   (b) (5 points) Apply the above result to find out how many Bitcoin users are needed to initialize their wallet's seed, which is based on a random selection of 12 random words from the list in `https://github.com/bitcoin/bips/blob/master/bip-0039/english.txt`, to have the event that, with probability at least 50%, at least two users end up with exactly the same seed.

3. (10 points) Prof. Gray has designed a cryptographic hash function $H_G : \{0,1\}^* \to \{0,1\}^n$. One of his brilliant ideas is to make $H_G(x) = x$ if x is an n-bit string (assume the behavior of H_G is much more complicated on inputs of other lengths). That way, we know with certainty that there are no collisions among n-bit strings. Has Prof. Gray made a good design decision? Justify your answer, in particular listing the properties that may or may not be satisfied by the design.

4. (10 points) As we saw in class, the main security notion for digital signatures is called *existential unforgeability*, which prevents an attacker from producing a signature on a message that has not been produced by the legitimate signer. I.e., the attacker should not be able to produce the pair (m, σ), where σ was not produced by the legitimate signer.

   Show an attack on the plain RSA signature scheme we saw in class in which an attacker forges a signature on an arbitrary message m by asking the signer to sign two other different messages (not necessarily unrelated to m).

5. (10 points) A Bitcoin miner creates a block B which contains address α, on which it wants to receive its rewards. An attacker changes the contents of B, such that instead of α it defines a new address α', which is controlled by the attacker. Will the attacker receive the rewards that the miner tries to claim? Why or why not? Give a detailed explanation of your answer.

6. (10 points) Using the course's Sepolia Testnet chain, send 0.05 ETH to the address of a fellow student. Describe how you conducted the payment, including the transaction's id and addresses which you used. (Refer to the Solidity tutorial material "How to Connect to a Public Testnet" document shared on Campus, as well as the Solidity tutorial material and pointers.)

7. The following smart contract has been deployed on the Sepolia Testnet chain:

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

contract Bank {
    mapping(address => uint256) balance;
    address[] public customers;

    event Deposit(address customer, string message);
    event Withdrawal(address customer);

    function deposit(string memory message) public payable {
        require(msg.value > 10);
        balance[msg.sender] += msg.value - 10;

        customers.push(msg.sender);

        emit Deposit(msg.sender, message);
    }

    function withdraw() public {
        uint256 b = balance[msg.sender];
        balance[msg.sender] = 0;
        payable(msg.sender).transfer(b);

        emit Withdrawal(msg.sender);
    }

    function getBalance() public view returns (uint256) {
        return balance[msg.sender];
    }

    function empty() public {
        require(msg.sender == customers[0]);

        payable(msg.sender).transfer(address(this).balance);
    }
}
```

Its deployed address is: 0x992434dCe1423e03548e0DF1189B2EE21a316494. You can compile it and interact with it using Remix. You should successfully create a transaction that interacts with the contract, either depositing or withdrawing from it some coins.

(a) (5 points) Describe the contract's functionality (that is, the purpose of each variable, function, and event).

(b) (15 points) Provide the id of the transaction you performed and the address you used.