Fundamentos y Aplicaciones de Blockchains

Homework 1

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

Student: Gaspar Joel Zuker *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.

   Solution: Los Merkle Trees son utilizados con el objetivo de poder demostrar la inclusión (o no inclusión) de un elemento en el árbol (en tiempo logarítmico en la cantidad de elementos del árbol, es decir, de forma eficiente) Además, por la naturaleza criptográfica de la blockchain, es importante que tener estructuras no puedan ser modificadas una vez generadas, para que no se pueda modificar el contenido de un bloque anterior.

   Para lograr estos objetivos, los merkle tree usan funciones de hash entre sus elementos y combinan los resultados de las mismas de forma ascendente hasta la raíz. Esto genera que la raíz contenga un valor de su función de hash que depende de todos y cada uno de los elementos del árbol. Luego, por propiedades de las funciones de hash, cualquier cambio en un elemento del árbol generaría un gran cambio en el valor de la raíz. Esto nos garantiza que no se va a poder vulnerar el contenido del árbol.

   Luego, para comprobar la inclusión de un elemento, solo hay que comparar el hash del elemento con los hashes presentes en el árbol en el camino desde la ubicación del elemento hasta la raíz, esto requiere hacer una cantidad logarítmica de comparaciones y, ya que los elementos pueden guardarse de forma ordenada, se puede verificar la no inclusión de un elemento comprobando la inclusión de un elemento menor y uno mayor que sean contiguos.

   (b) (3 points) Describe how a (complete, binary) MT is constructed for the following five chunks of data: ABC, DEF, GHI, KLM, OPQ.

   Solution:

   Para poder completar este merkle tree, tenemos que repetir algunos nodos, ya que la cantidad de elementos no es potencia de dos. Para llevarlo a cabo, seguí los consejos dados en esta página.

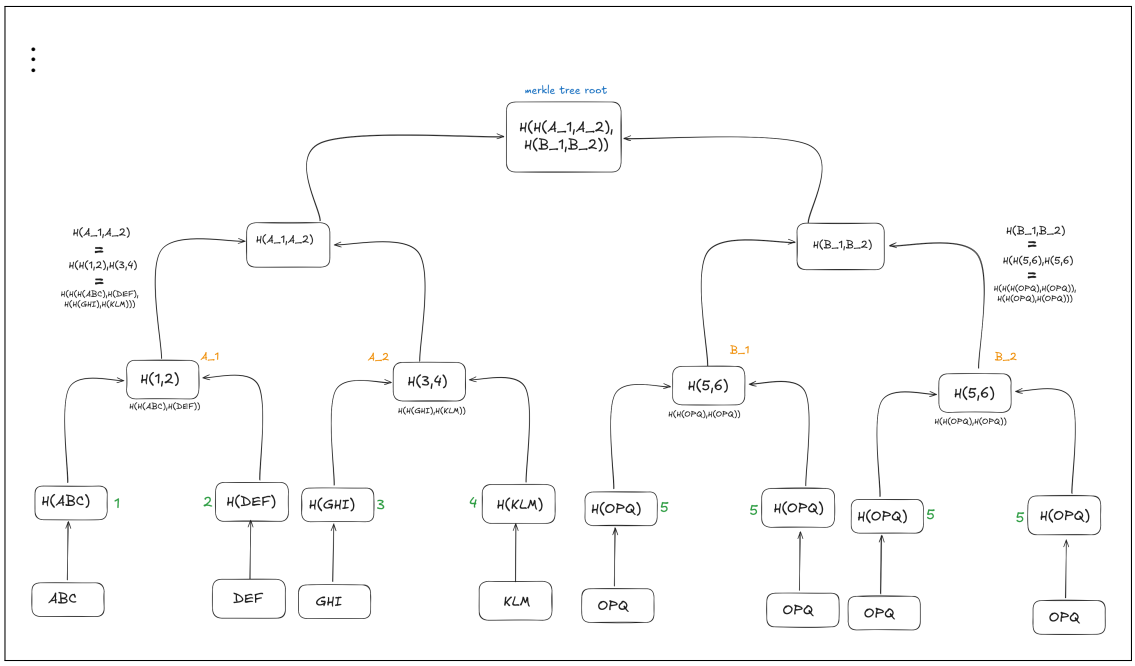
   Para hacerlo completo como pide la consigna, decidí duplicar el último elemento 3 veces.

   No calculé los hashes de los elementos de ejemplo, ya que me parece visualmente más fácil de entender poniendo $H(ABC)$ o $H(1,2)$ para mostrar el hash del elemento "ABC" o el hash de los nodos 1 y 2. Sin embargo, si fuera necesario calcularla realmente, solo debería utilizar sha256 para calcular estos valores. Por ejemplo:

   H(ABC)=b5d4045c3f466fa91fe2cc6abe79232a1a57cdf104f7a26e716e0a1e2789df78

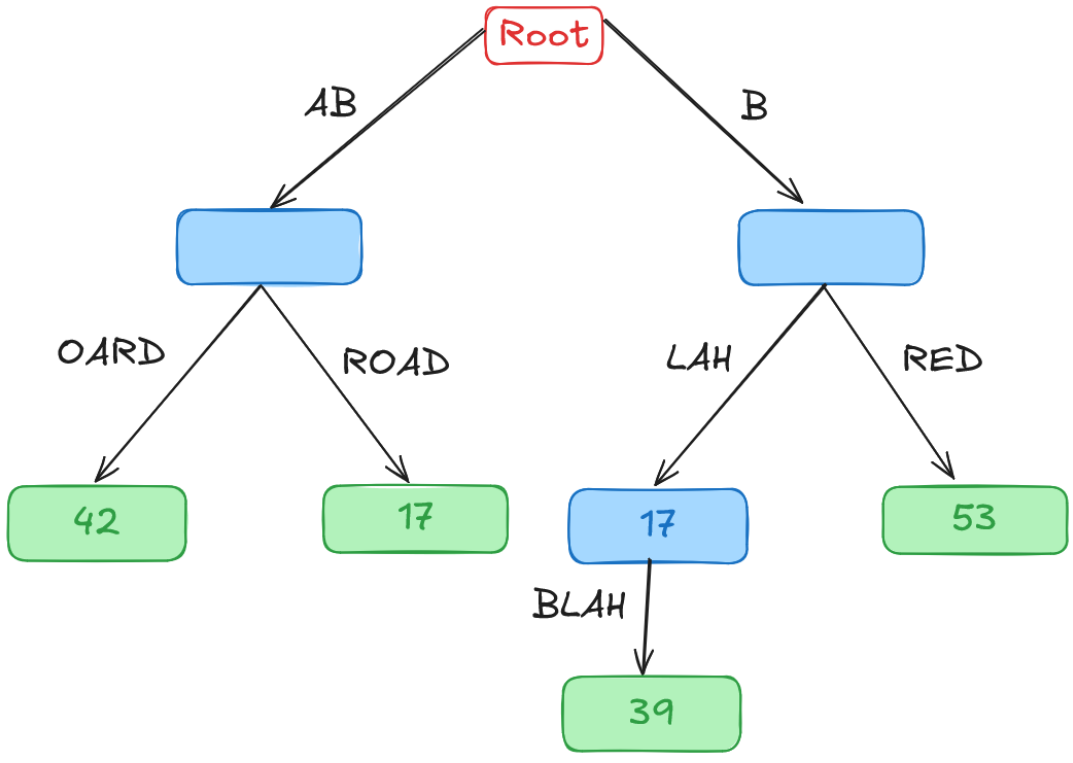
   H(DEF)=967c5a5b7e2fbbe3080a0c5cefea7c279570b16ae8465525538bc3b115267a45

   H(H(ABC),H(DEF))= 15f6e88b1a3ffba1f48ae0b34dcbfdff5c00f13a33d48fd44c103217308c580b



(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}.

Solution:



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%.

Solution: La probabilidad de que una colisión se realice en un sistema que distribuye de manera uniforme los valores posibles se puede calcular usando sus dos factores:

n = La cantidad de distintos valores posibles que puede tomar cada elemento (en el ejemplo del cumpleaños sería 365) k = La cantidad de elementos a los que asignarles un valor

Con estos dos factores podemos determinar $P(C)$ (la probabilidad de que ocurra una colisión), siendo la forma más fácil calcular $P(\neg C)$ (la probabilidad de que todos los valores sean distintos) y luego despejar $P(C) = 1 - P(\neg C)$.

La probabilidad de que no haya una colisión en caso de que k sea mayor a n es de 1, en otro caso, viene dada por la fórmula:

$$\prod_{i=1}^{k-1} 1 - \frac{l}{n}$$

Ejemplo: Si tengo 3 elementos en 5 valores posibles que son asignados de forma uniforme, la posibilidad de no encontrar 2 elementos con el mismo valor asignado es de:

$$5/5 \cdot 4/5 \cdot 3/5 = 12/25$$

Esto es así porque el primer elemento no tiene riesgo de colisión, el segundo tiene 4 de 5 valores posibles en los que no colisiona y el tercero sólo 3 de 5 posibilidades.

Para números grandes, la fórmula previamente enunciada puede ser difícil de utilizar, ya que contiene una productoria. Por suerte, podemos aproximar (acotar) la fórmula con una exponencial transformando la productoria en una sumatoria:

$$\prod_{i=1}^{k-1} 1 - \frac{i}{n} \leq exp(-\frac{1}{n} \sum_{i=1}^{k-1} i)$$

Finalmente, como tenemos una suma de Gauss, podemos simplificar la sumatoria remplazandola por su fórmula cerrada:

$$exp(-\frac{1}{n} \cdot (k \cdot (k-1)/2)) = exp(-\frac{k \cdot (k-1)}{2n})$$

Ahora aplicando $P(C) = 1 - P(\neg C)$ nos quedaría:

$$P(C) \approx 1 - exp(-\frac{k \cdot (k-1)}{2n})$$

Reemplazando P(C) por 0.5 y n por la cantidad de elementos distintos, podemos despejar un valor de k que nos genere más de 50% de probabilidad de colisión para ese problema específico.

(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.

Solution:

Para hacer este cálculo, tenemos que calcular nuestro n y despejar k tal que la fórmula enunciada en el punto anterior sea mayor a 0.5.

Calcular n: La seed de bitcoin es una selección de 12 palabras de una lista de 2048 palabras. Este es un número realmente grande, que puede calcularse haciendo el combinatorio $\binom{2048}{12}$.

Reemplazando en la fórmula, nos queda:

$$P(C) \approx 1 - exp(-\frac{k \cdot (k-1)}{2 \cdot \binom{2048}{12}})$$

Luego, ayudandome con wolfram obtuve el resultado:

$$k = 3,905961117474372 \times 10^{15}$$

Esto significa que debería crearse esa cantidad de cuentas para que la probabilidad de tener una colisión sea mayor al 50%.

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.

Solution: Una función de hash criptográfica necesita cumplir las siguientes propiedades:

• Collision resistance. It is infeasible to find any x, x' for which $H(x) = H(x')$
• Pre-image resistance: Given $y = H(x)$ (but not x itself), for a randomly chosen x, it is infeasible to find a value x' s.t. $H(x') = y$
• Second pre-image resistance: For any x, it is infeasible to find $x' \neq x$ such that $H(x) = H(x')$

H_g Rompe con todas estas propiedades. Collision resistance: Dado cualquier x es $H_g(x)$ es h, entonces $H_g(h)$ también es h. Pre-image resistance: Dado un y, si hago $H_g(y)$ vuelvo a obtener y, por ende, y es preimagen de si mismo. Second pre-image resistance: Dado cualquier x es $H_g(x)$ es h, entonces $H_g(h)$ también es h. (idem a collision resistance, dado que es menos fuerte)

Por ende, aunque se evite colisiones en elementos de n bits, esta no es una buena decisión de diseño.

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).

Solution: El atacante podría elegir dos mensajes $m1$ y $m2$ estratégicamente, tales que $m = m1.m2$ y, de esta manera, generar el par $(m, \sigma_1\sigma_2)$ que es válido.

Esto se puede ver fácilmente utilizando la tesis de la demostración de RSA. Dado que el protocolo se basa en comprobar que $x \equiv x^{ed} \pmod{N}$, si tomamos ahora $x = a \cdot b$ entonces es fácil ver que

$$(a \cdot b)^{ed} \pmod{N} = (a^{ed} \pmod{N}) \cdot (b^{ed} \pmod{N})$$

por propiedades de aritmética modular.

De esta manera, si un atacante obtuvo los pares (m_1, σ_1) y (m_2, σ_2), puede generar $(m, \sigma_1\sigma_2)$ que sería válido al verificarse utilizando la clave de verificación pública (e, n), sin necesidad de conocer d para poder generar esa firma.

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.

Solution:

En el caso hipotético de la consigna, al cambiar la dirección α por α' en el bloque minado B, va cambiar el valor de la raíz del merkle tree que representa a B, ya que su contenido cambió.

Entonces, es extremadamente poco probable que el bloque B' (que es igual a B cambiando α por α') tenga un hash que cumpla con el puzzle utilizado para generar dificultad en el minado de los nuevos bloques.

Por esta razón, no se aceptaría el bloque B', ya que no es un bloque válido.

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.)

Solution: Para transferir programé un smart contract que haga uso del método send, como fue indicado por los docentes.

Datos de llamada al contrato en Etherscan

Mi adress es:

0xBdCaCA8d3DD1c33d7c42BB9146DF34fA5287307d

Transaction hash:

0xc2ad94b207f78cf7d6878896b0383b5d8ded7718b6e89275a3c8e9fcce3bf4bc

Código del contrato:

```
pragma solidity ^0.8.20;

contract SendEther {
    address public owner;

    constructor() {    238319 gas 213800 gas
        owner = msg.sender;
    }

    function getBalance() external view returns(uint) {    312 gas
        return address(this).balance;
    }

    receive() external payable { }    undefined gas

    function sendEth(address payable _to, uint256 _amount) external payable onlyOwner {    undefined gas
        bool sent = _to.send(_amount);
        require(sent, "Send failed");
    }

    // Remix ASSISTANT recommended me to add this in order to add security to the contract
        modifier onlyOwner() {
        require(msg.sender == owner, "Not owner");
        _;
    }
}
```

Para hacerlo utilicé un par de tutoriales de youtube

How to Send Ether in contract using Remix

Send ETH — Solidity 0.8

Y un poco de ayuda de la IA de Remix para corregir un bug, que además me dió unos consejos de seguridad.

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).

Solution: Este contrato, como su nombre indica, funciona como una especie de ”banco”. Se puede llamar a la función *deposit* indicando una cantidad a ser depositada. Luego, al utilizar *withdraw* se puede ”retirar” lo depositado, que será devuelta a la cuenta que depositó originalmente.

Para implementar estas funciones se utiliza una lista de customers (a la que se le hace push cada vez que un nuevo customer deposita) y un diccionario que mapea direcciones con sus balances.

La función *empty* vacía el contrato, y su implementación conlleva un peligro de seguridad en mi opinión ya que es pública y solo comprueba que el llamador sea $customers[0]$. Sin embargo, ese customer puede no ser el creador del contrato (es simplemente el primero que depositó). En caso de que una vez creado el contrato, un usuario malicioso deposite antes que el creador, tendría acceso a retirar todo lo que se deposite posteriormente.

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

Solution:

Mi adress es: 0xBdCaCA8d3DD1c33d7c42BB9146DF34fA5287307d

Deposit

Transaction hash:

0xe10ce4409923ea049f008f2188a9b52e2ccd3b6e26e7730c5165b883e20ddc7f

Withdraw

Transaction hash:

0xace0074f6feacdfb7b9b2861516149b6133840bbfb192801a35dee0f44eebeda