Fundamentos y Aplicaciones de Blockchains

Homework 3

Depto. de Computación, UBA, 2do. Cuatrimestre 2025 *9/10/25*

Student: *Due: 21/10/25, 15:00 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>_HW3.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 4 questions, for a total of 65 points.

1. **Bitcoin transactions:**

   (a) (7 points) Describe the mechanism used in the actual Bitcoin application that the miners follow in order to insert transactions into a block. Would the Consistency (aka Persistence) and Liveness properties we saw in class, as well as the V, I, R functions have to be modified to capture the actual mechanism? Elaborate.

   Solution: Para llevar a cabo el procedimiento de verificación, se verifica qu el output de la transacción "A" sea igual al input de la transacción "B". Este procedmiento se lleva a cabo con un lenguaje que no es Turing completo para mayor seguridad. Las transacciones también tienen que pasar un test llamado IsStandard() para ser aceptadas, donde se verifica que cumplan condiciones como tener el numero de versión correcto, para ser aceptadas si son broadcasteadas a la red. Esto no impide que transacciones distintas aparezcan en un bloque minado por un usuario que utiliza un protocolo no estándar, pero impide que se propaguen transacciones dañinas por la red.

   Las transacciones son guardadas en formato binario crudo de transacciones, y sus hashes se guardan en el merkle tree del nuevo bloque, al que se le calculará la PoW antes de ser agregado a la blockchain.

   Fuente: bitcoin.org transactions y block-chain

   (b) (3 points) Describe the purpose of a *coinbase* transaction.

   Solution: Las Coinbase Transactions son un tipo especial de transacciones creadas por los mineros al minar bloques, y son utilizadas para generar la emisión de nuevos bitcoins (la recompensa por minar el bloque), además de para cobrar los transaction fees. Los UTXOs generados por estas transacciones tienen una restricción de esperar 100 bloques antes de poder ser utilizada para una nueva transacción, con el objetivo de tener un margen suficiente como para que el bloque sea totalmente aceptado.

   Fuente: bitcoin.org

2. **Proofs of work:**

(a) (5 points) Describe the purpose and implementation of the **2x1 PoW** technique.

> **Solution:** La técnica de 2x1 PoW consiste en generar una doble proof of work al tener que encontrar un hash específico que contenga a un par de datos h y h'. Para esto tenemos que calcular $H(h, h', ctrl) < T$ y $[H(h, h', ctrl')]^R < T$. Esto es equivalente a decir que queremos una cierta cantidad de 0s al principio para el hash de h y una cantidad de 0s al final para el hash de h', utilizando al otro valor como "label" en cada ocasión.
>
> Este método nos asegura que si se busca una PoW válida para h, necesitamos también encontrar una PoW para h', por lo que estaríamos dividiendo el poder de cómputo equivalentemente entre ambas.
>
> Para implementar este modelo se calculan en parelalo ambas PoW. El algoritmo puede devolver como resultado una solución para ambas, para ninguna o para una sola.

(b) (7 points) Let T_1 and T_2 be the target values in 2x1 PoW, and κ the size of the hash function's output. What relation should they satisfy for the technique to work? Elaborate.

> **Solution:** T_1 y T_2 no deberían tener correlación respecto a sus valores de hash. La técnica de 2x1 PoW tiene el objetivo de dividir proporcionalmente la cantidad de capacidad computacional entre ambas pruebas (para poder garantizar los requisitos del protocolo de concenso 1/3). Si no tienen la misma dificultad ambos puzzles (T != T') o tienen correlación las soluciones, no se cumplirá este objetivo.

(c) (8 points) Design and argue correctness of an ℓx1 PoW scheme. Note that such a scheme would enable an ℓ-parallel blockchain. What properties of the blockchain or ledger application would, if any, benefit from a parallel blockchain? Elaborate.

3. **Strong consensus:** We saw how the Bitcoin backbone protocol can be used to solve the *consensus* problem (aka *Byzantine agreement*). In the *strong consensus* problem, the *Validity* condition is strengthened to require that the output value be one of the honest parties' inputs—this property is called *Strong Validity*. (Note that this distinction is relevant only in the case of non-binary inputs.)

   (a) (5 points) What should be the assumption on the adversarial computational power (similar to the "Honest Majority Assumption") for Strong Validity to hold?

   Solution: Strong Concensus es una versión más restrictiva del concenso. Además de requerir cumplir con las propiedades de terminación y acuerdo (todos los participantes honestos deciden un valor y si dos participantes honestos ya decidieron un valor entonces decidieron el mismo valor), requiere que se cumpla el Strong Validity Lemma.

   Este lema requiere que el valor de salida del concenso tiene que haber sido un valor elegido por un participante honesto. Para cumplir esto necesitamos un $n > max(3, m) \cdot t$ siendo:

   - n: Número de participantes
   - m: Cantidad de opciones a elegir
   - t: Cantidad de participantes maliciosos

   Esto es para poder distinguir, incluso en el caso donde los participantes toman elecciones diversas, al menos un grupo de más de m/t participantes honestos que hayan elegido el mismo valor.

   Si no ocurre esto, y todos los adversarios eligen el mismo valor, ese valor no sería distinguible de un valor elegido por participantes honestos.

   Por último, puede pasar que el valor más elegido sea el que eligen los t participantes maliciosos, pero bajo la asunción propuesta arriba, es fácil ver que al menos un participante honsto también eligió ese valor.

   (b) (5 points) State and prove the Strong Validity lemma.

   Solution: Strong Validity Lemma: Si los participantes honestos deciden en un valor v, entonces, el valor v fue propuesto originalmente por algún participante honesto.

   Para la demostración podemos usar como base la demostración de validity utilizada previamente y modificar lo necesario para que cumpla la versión fuerte.

   Para probar validity, dijimos que dado un número de rounds, la propiedad de

Chain Quality garantiza que la cadena contiene un bloque honesto. Con suficientes rounds, la cantidad de PoW generadas por participantes honestos va a ser abrumadoramente mayor a la cantidad de PoW correctas del adversario.

Luego, lo único que tenemos que modificar de la prueba anterior es el método de selección, pasando de tomar la opción mayoritaria (entre opciones binarias) a tomar la opción con más votos. Para esto necesitamos la propiedad enunciada en el punto anterior (para más de 2 valores): $n > m \cdot t$.

Con esta propiedad podemos observar que cualquiera sea la distribución de votos de los participantes honestos, como tenemos $n/m > t$, alguna de las m opciones van a tener más de t votos. Si elegimos la opción más elegida siempre va a tener estrictamente más de t votos, por ende, siempre va a haber sido votada por al menos un participante honesto.

4. **Smart contract programming:** In this assignment you will create your **own custom token**. Your contract should implement the public API described below:

   - **owner:** a public payable address that defines the contract's "owner," that is, the user that deploys the contract
   - ***Transfer(address indexed from, address indexed to, uint256 value):*** an event that contains two addresses and a uint256
   - ***Mint(address indexed to, uint256 value):*** an event that contains an address and a uint256
   - ***Sell(address indexed from, uint256 value):*** an event that contains an address and a uint256
   - **totalSupply():** a view function that returns a uint256 of the total amount of minted tokens
   - **balanceOf(address_account):** a view function returns a uint256 of the amount of tokens an address owns
   - **getName():** a view function that returns a string with the token's name
   - **getSymbol():** a view function that returns a string with the token's symbol
   - **getPrice():** a view function that returns a uint128 with the token's price (at which users can redeem their tokens)
   - **transfer(address to, uint256 value):** a function that transfers *value* amount of tokens between the caller's address and the address to; if the transfer completes successfully, the function emits an event *Transfer* with the sender's and receiver's addresses and the amount of transferred tokens and returns a boolean value (*true*)
   - **mint(address to, uint256 value):** a function that enables *only the owner* to create value new tokens and give them to address to; if the operation completes successfully, the function emits an event *Mint* with the receiver's address and the amount of minted tokens and returns a boolean value (*true*)
   - **sell(uint256 value):** a function that enables a user to sell tokens for wei at a price of *600 wei per token*; if the operation completes successfully, the sold tokens are removed from the circulating supply, and the function emits an event *Sell* with the seller's address and the amount of sold tokens and returns a boolean value (*true*)
   - **close():** a function that enables *only the owner* to destroy the contract; the contract's balance in wei, at the moment of destruction, should be transferred to the owner's address
   - **fallback** functions that enable anyone to send Ether to the contract's account
   - **constructor** function that initializes the contract as needed

You should implement the smart contract and deploy it on the course's Sepolia Testnet. Your contract should be as secure and gas efficient as possible. After deploying your contract, you should buy, transfer, and sell a token in the contract.

Your contract should implement the above API **exactly as specified**. *Do not* omit implementing one of the above variables/functions/events, do not change their name or parameters, and do not add other public variables/functions. You can define other private/internal functions/variables, if necessary.

You should provide:

(a) (5 points) A detailed description of your high-level design decisions, including (but not limited to):

- What internal variables did you use?
- What is the process of buying/selling tokens and changing the price?
- How can users access their token balance?

Solution: Traté de hacer una implementación de la API lo más directa y minimalista posible, aunque dejé algunas variables privadas como ”precio” por si más adelante se quiere agregar funcionalidad al contrato.

Para comprar y vender, parte de la transacción debe hacerse por fuera del contrato. Por ejemplo, no existe una función que permita hacer una compra de nuevos tokens directamente al contrat usando wei, para emular esto, se debe hacer un pago al contrato y luego el owner debe emitir el token poniendo como destinatario a quien hizo el pago. (Haciendo esto se puede mantener la equivalencia que se usa en ”Sell”) Luego, se pueden transferir libremente los tokens con transfer, intercambiandolos por otra cosa en otro medio.

Para acceder a su balance, los usuarios solo deben llamar a la vista pública ”balanceOf” con su address como parámetro.

Por último, una importante decisión de diseño es usar una funcion

(b) (5 points) A detailed gas evaluation of your implementation, including:

- The cost of deploying and interacting with your contract.
- Techniques to make your contract more cost effective.

(c) (5 points) A thorough listing of potential hazards and vulnerabilities that can occur in the smart contract and a detailed analysis of the security mechanisms that can mitigate these hazards.

Solution: Desde el punto de vista del usuario de los tokens, el contrato es muy peligroso, ya que el owner del mismo puede destruirlo y llevarse todo el balance en wei del contrato en cualquier momento, ademas de que puede

emitir cualquier cantidad de tokens. Además, no pueden comprarse tokens ni intercambiarse utilizando metodos públicos del contrato (solo pueden venderse o transferirse). Esto hace que, en caso de querer hacerte con tokens, una de las partes dependa de la buena fe de una la otra parte (que te envíe los tokens a cambio de algún elemento externo, o arriesgarse a enviarlos antes).

(d) (5 points) The transaction history of the deployment of and interaction with your contract.

Solution:
Deploy
Mint
Transact
Sell
Transfer

(e) (5 points) The code of your contract.

Solution: Código del contrato