Fundamentos y Aplicaciones de Blockchains

Homework 2

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

Student: Juan DElia *Due: 14/10/25, 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>_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 4 questions, for a total of 60 points.

1. We saw in class the notion of *digital signatures* and their security properties, *existential unforgeability* being an important one.

   (a) (5 points) Describe the purpose of a *Public-Key Infrastructure* (PKI). Can the security properties of a digital signature be guaranteed without a PKI? Elaborate.

   Solution: El propósito de PKI es que haya una autoridad confiable, cuya public key es conocida, mediante la cual podemos asociar a participantes de la red con sus correspondientes public keys.

   Las propiedades de seguridad de las firmas digitales no se pueden conseguir sin PKI:

   1. Autenticación: Si no podemos asociar entidades con public keys no tenemos razon para creer que dicha entidad efectivamente firmo ese mensaje. Lo único que sabemos es que un mensaje fue enviado por quien posea dicha public key (que a priori es desconocido).

   2. Non-repudiation: La idea es que quien firmó, no pueda negar haberlo hecho. Si no hay una entidad que linkee a quien envió el mensaje y su public key, el recipiente del mismo no puede probar quien lo envió efectivamente (solo puede probar con que pk se firmó).

   3. Integridad: Esto si se mantiene, una vez firmado no se puede alterar al documento.

   (b) (5 points) Bitcoin transactions use the ECDSA signature scheme. Does Bitcoin assume a PKI? If not, reconcile with the above argument.

   Solution: Bitcoin no asume PKI. Esto no es un impedimento pues no es necesario vincular documentos y entidades reales, simplemente hay que asegurar que se guarden las transacciones entre nodos. Cuando se realiza una transacción quien la envía la firma con su clave privada y quien la recibe puede verificarla usando la clave pública de quién la envió. (ahí es donde se usa el esquema de firmas).

   Estas transacciones luego son organizadas por los mineros que solo deberían aceptar transacciones validas.

2. (10 points) Refer to Algorithm 4 (Main Loop) in [GKL15]. We saw in class that blockchains would start from a "genesis" block, which would provide an unpredictable string to start mining from. Yet, notice that in Algorithm 4 mining starts from the empty string ($\mathcal{C} \leftarrow \varepsilon$). How come? Explain why this works.

Solution: El algoritmo inicialmente inicializa la cadena en si, podemos pensarlo como la estructura de datos, por eso es vacía al igual que st. Inmediatamente después va a poner el flag init en falso y va a esperar a que aparezca una cadena para leer, y en caso de cumplir maxvalid() mediante pow definir C_{new} que puede ser la misma cadena o una que minó un bloque nuevo para finalmente broadcastearla.

En el momento que se crea la blockchain entonces va a estar vacía. Alguien va a escribir el primer bloque (genesis) y mediante ese mecanismo va a ser difundida la nueva cadena que lo contenga, pues va a ser la más larga.

Básicamente no se indica explícitamente el bloque génesis, pero mediante el algoritmo que va adoptando la cadena más larga, la primer cadena que se adopte se debería seguir exteniendo. Así la primer cadena de longitud uno contiene al bloque genésis y de ahí en adelante se extiende sobre la misma.

3. Refer to Algorithm 1 (validate) in [GKL15], which implements the *chain validation predicate*.

   (a) (5 points) Rewrite validate so that it starts checking from the *beginning* of the chain.

   Asumi que existe la funcion tail que devuelve la cadena sin el primer elemento y first que dada una cadena devuelve su primer elemento.

Algorithm 1 Validate chequeando desde el inicio de la cadena

```
function VALIDATE(C)
    b ← V(x_C)
    if b ∧ (C ≠ ε) then
        repeat
            ⟨s, x, ctr⟩ ← first(C)
            s' ← H(ctr, G(s, x))
            C ← tail(C)
            if C = ε ∧ validblock_q^T(⟨s, x, ctr⟩) then    ▷ Si es el ultimo solo chequea valid
                break
            end if
            ⟨s, x, ctr⟩ ← first(C)        ▷ hay que chequear que el siguiente tiene su hash
            if validblock_q^T(⟨s, x, ctr⟩) ∧ s == s' then
                continue
            else
                b ← False
            end if
        until (C = ε) ∨ (b = False)
    end if
    return b
end function
```

   (b) (5 points) Discuss pros and cons of this approach, compared to Algorithm 1's.

4. **Smart contract programming:** *Matching Pennies*. This assignment will focus on writing your own smart contract to implement the Matching Pennies game. The contract should allow two players (A, B) to play a game of Matching Pennies at any point in time. Each player picks a value of two options—for example, the options might be {0, 1}, {'a', 'b'}, {True, False}, etc. If both players pick the same value, the first player wins; if players pick different values, the second player wins. The winner gets 0.1 ETH as reward. After a game ends, two different players should be able to use your contract to play a new game.

   Example: Let A, B be two players who play the game, each with 0.5 ETH. A picks 0 and B picks 0, so A wins. After the game ends, A's balance is 0.6 ETH (perhaps minus some gas fees, if necessary).

   You should implement the smart contract and deploy it on the course's Sepolia Testnet. Your contract should be as *secure*, *gas efficient*, and *fair* as possible. After deploying your contract, you should engage with other student's contract and play a game on his/her contract. Before you engage with a fellow student's smart contract, you should evaluate their code and analyze its features in terms of security and fairness (refer to Lecture 10). You should provide:

   (a) (10 points) The code of your contract, together with a detailed description of the high-level decisions you made for the design of your contract, including:
      - Who pays for the reward of the winner?
      - How is the reward sent to the winner?
      - How is it guaranteed that a player cannot cheat?
      - What data type/structure did you use for the pick options and why?

   (b) (5 points) A detailed gas consumption evaluation of your implementation, including:
      - The cost of deploying and interacting with your contract.
      - Whether your contract is *fair* to both players, including whether one player has to pay more gas than the other and why.
      - Techniques to make your contract more cost efficient and/or fair.

   (c) (5 points) A thorough list of potential hazards and vulnerabilities that *may* occur in your contract; provide a detailed analysis of the security mechanisms you use to mitigate such hazards.

   (d) (5 points) A description of your analysis of your fellow student's contract (along with relative code snippets of their contract, where needed for readability), including:
      - Any vulnerabilities discovered?
      - How could a player exploit these vulnerabilities to win the game?

   (e) (5 points) The transaction history of an execution of a game on your contract.