

## Tarea 3: Transacciones de Bitcoin

Entrega: Junio 19, 2020

### 1. Cosas administrativas

Cada tarea en este curso equivale a 5 % de la nota final. En total tendremos 5 tareas, pero la peor tarea que solucionan no cuenta para la nota final. Quiere decir que pueden botar una tarea.

La solución se entrega usando la plataforma Canvas en un archivo de texto.

El uso de materiales (o soluciones) encontrados en Internet está permitido. Solo se les pide citar la fuente que utilizaron. No se aplica ninguna penalidad para su uso. Si entienden como usar una billetera electrónica comercial para resolver la tarea tampoco se aplica una penalidad.

### 2. La tarea

En esta tarea vamos a crear varias direcciones de tipo P2PKH y generar transacciones desde y hacia estas direcciones en el testnet de Bitcoin. Efectivamente la tarea es simular una billetera de Bitcoin que soporta direcciones de tipo P2PKH (y P2PK), y puede enviar dinero a una dirección de tipo P2PKH, y P2SH.

Junto con este enunciado se adjunta una implementación mínima que nos permitirá crear y validar transacciones con scripts de tipo P2PKH (y P2PK, pero estas no usaremos para disminuir el uso de RAM en nodos manteniendo el UTXO pool). La implementación es una simplificación de lo visto en la clase (incluye solo el soporte de P2PKH, y no P2SH), y les ayudará realizar la tarea. La implementación también incluye unos ejemplos mostrando cómo enviar bitcoin de una dirección P2PKH a otra, o incluso de una dirección P2PKH a una dirección de tipo P2SH.

Las tareas específicas para realizar son las siguientes:

1. **Conseguir testcoins [1 punto]**. Primero lo que se les pide es generar dos direcciones de Bitcoin **Testnet** de tipo P2PKH. Las llaves secretas de sus direcciones deberían ser las siguientes:

- `hash256(b'IIC3272SuNombre1')`

■ `hash256(b'IIC3272SuNombre2')`,

dónde el substring "SuNombre" debería ser reemplazado por su nombre completo (no compliquen la cosa por favor). Desde aquí en adelante, nos referiríamos a estas dos direcciones como **dirección 1** y **dirección 2**.

Las direcciones pueden generar utilizando su solución de Tarea 2. Para los que no la tienen, junto con el código está incluido el archivo "tarea2\_solution.py" que les permitirá generar la dirección.

Luego de generar las dos direcciones, deberían enviar testcoins de un faucet de testnet en **dos** transacciones distintas a la **dirección 1**. Un faucet que pueden ocupar es <https://bitcoinafaucet.uo1.net/send.php>.

Entrega de esta parte de la tarea serán: sus llaves secretas, y las direcciones generadas. En un block explorer como, por ejemplo, <https://live.blockcypher.com/> verificaremos que consiguieron los testcoins en dos transacciones.

2. **Transacción P2PKH a P2PKH [2 puntos]**. Ahora que tienen las dos direcciones y dos transacciones recibidas por la **dirección 1**, deberían generar **una transacción** que gasta estos dos outputs recibidos por la **dirección 1**, y los envía a la **dirección 2**. Los dos outputs deberían ser gastados en una transacción! La transacción pueden lanzar en la testnet usando <https://live.blockcypher.com/btc/pushtx/> (ocupando la red Bitcoin testnet). No se olviden del transaction fee!

Entrega de esta parte será la versión hexadecimal de la transacción (entera) **aceptada** por la testnet. En nuestros ejemplos en las clases este código se genera con el comando `newTx.serialize().hex()`, después de firmar la transacción. Para conseguir los puntos en esta parte de la tarea, hay que asegurarse que los fondos llegaron a la **dirección 2**. De nuevo, para esto pueden ocupar un block explorer. Nosotros nos encargaremos de revisar el estatus de su transacción usando su hex, así que asegúrense que lo podemos revisar de manera correcta. Las últimas dos líneas del archivo "txP2PKH.py" apunta cómo buscaremos su transacción en el block explorer.

3. **Transacción P2PKH a P2PKH y P2SH [3 puntos]**. Ahora que tienen fondos controlados por la **dirección 2**, deberían enviarlos a dos direcciones en una transacción. En particular, deberían generar una transacción que gasta el output generado en la parte 2 de la tarea, y envía 50 % de este output a **dirección 1**, y 50 % a la dirección `2NGZrVvZG92qGYqzTLjCAewvPZ7JE8S8VxE` (el faucet). Los 50 % son modulo transaction fee. Es importante notar que `2NGZrVvZG92qGYqzTLjCAewvPZ7JE8S8VxE` no es una dirección de tipo P2PKH, sino de tipo P2SH, así que su script debería ser distinto. La transacción pueden lanzar en la testnet usando <https://live.blockcypher.com/btc/pushtx/> (ocupando la red Bitcoin testnet).

La entrega de esta parte es la mismo cómo en la parte dos de la tarea: el hex de la transacción.

4. **Transacción P2PKH a P2SH [1 punto]**. Finalmente, deberían enviar los fondos recibidos por la **dirección 1** en la parte 3 de la tarea a `2NGZrVvZG92qGYqzTLjCAewvPZ7JE8S8VxE`. Esto es una transacción con un input y un output.

La entrega de esta parte es la mismo cómo en las partes dos y tres de la tarea: el hex de la transacción.

**Bonus 1. [5 % de la nota final]** Usando el código que tienen para verificar y formar las transacciones, desarrollen una billetera sencilla que puede manejar las direcciones de tipo P2PKH, enviar los fondos a direcciones de tipo P2PKH y P2SH. La billetera no debería mostrar al usuario la transacción, sino solo entrar las direcciones y el monto por transferir. La billetera también debería mantener un registro de cuales outputs todavía no tiene gastados para poder decidir cómo construir una nueva transacción que quizás debería utilizar más de un output para sacar suficiente fondos. Los outputs todavía no gastados pueden conseguir de una API de algún blockexplorer. Igualmente, la billetera debería tener un “change address” dónde puede enviar los fondos del output que sobran. Go nuts here. Este bonus reemplaza una tarea entera!

**Bonus 2. [+10 Karma]** Devuelvan los testcoins conseguidos en la tarea 2 al faucet. El enunciado de la tarea 2 contiene la llave secreta que necesitan utilizar para firmar las transacciones.