**Midterm Exam**

**CSE528-Introduction to Blockchain and Crypto Currency (2021 Monsoon), IIIT Delhi**

**Max Points : 50**

**Total Time : 1 Hour, starting at 10am on Tuesday 28 September 2021 (IST)**

**Roll number: 2019061**

**Name: Mihir Chaturvedi**

*Instruction: The exam is open-book. It is strictly disallowed to copy&paste from the Internet, publicly available literature, or even from my lecture notes. Please rephrase your answers in your own language according to your understanding within the scope covered by the lecture notes. If you do not follow this instruction for a question, you will not get any points for that question. Within 1 hour and 5 minutes, where 5 mins are butter for uploading in the MS word format (doc or docx), you should successfully upload through the google classroom to avoid any penalty: 5 points will be deducted for every additional minute.*

**Question 1. [10 Points] Based on my lecture note, explain the process of P2PKH Script validation.**

**Answer:**

The script validations are done using a series of commands that are called and actioned on a stack. The P2PKH comprises the `scriptSig` and the `scriptPubKey` which are the unlocking and locking scripts. They look like the following:

* `scriptSig`: <sig> <public\_key>
* `scriptPubKey`: DUP HASH160 <public\_key\_hash> EQUALVERIFY CHECKSIG

The stack is evaluated in the following manner in accordance with the op commands.

1. <sig> is pushed to the stack
2. <public key> is pushed to the stack
3. DUP command duplicates the top of the stack, so <public key> is duplicated
4. HASH160 is a RIPEMD160(SHA256(P)) of the top of the stack, that is the duplicated public key. The hash is then pushed to the stack.
5. EQUALVERIFY verifies whether the hashed public key is the same as the supplied public key. These two are removed from the stack.
6. Finally, CHECKSIG verifies that the <sig> signature is created from the user of the public key only. If it is true, TRUE is pushed onto the stack. The checking is done using the ECDSA algorithm.

Validation succeeds if the end result is the stack containing TRUE, else the validation fails!

**Question 2. [10 Points] Based on my lecture note, explain how a bitcoin address is defined from a public key.**

**Answer:**

A bitcoin address is based on the public key of the user. The public key of the user can be resolved into two coordinates x and y, that represent a point on the standard bitcoin elliptic curve. When creating the bitcoin address, the following steps are involved. For this illustration we will be using the P2PKH type of bitcoin address:

1. The x coordinate is our first and main payload when creating the bitcoin address.
2. Perform a double hash on the payload, P, of the bitcoin wallet. SHA256(P) which produces a 256-bit string and RIPEMD160(SHA256(P)) which produces a shorter, 160-bit string.
3. The private key of the user indicates the compression type of the bitcoin address. In it, if it is encoded with 01 as a suffix flag, it means that the bitcoin address must be ‘compressed’, and otherwise ‘uncompressed’. ‘Uncompressed’ means that the entire value of ‘y’ is stored alongside the value of ‘x’, and ‘compressed’ only stores the parity (odd or even) of ‘y’ as a prefix to the payload.
   1. 0x02: Prefix if y is even
   2. 0x03: Prefix if y is odd
   3. 0x04: Prefix if the entire value of y is stored alongside x. This is in case of an uncompressed public key.
4. A checksum is appended to the end of this payload to quickly validate the correctness of the address. This is calculated by taking the first four bytes of the byte dump of the double-hashed payload, along with its prefixes.
5. Finally, the entire payload along with the prefix and checksum is base58 encoded. This has its particular benefits that it reduces the length of the shared address. Also, base58 encoded does not include ambiguous characters such as 'o' and '0', to prevent confusion and typos when sharing. The leading symbol of this produced address is always ‘1’.

**Question 3. [10 Points] Based on my lecture note, explain how we can protect assets with a co-signatory using a multi-signature address.**

**Answer:**

By using a co-signatory, we can protect a user’s bitcoin assets and mitigate fraud by involving the requirement of multi-party consent. This way, even if one user’s secret keys are stolen to create signatures, the assets remain secured as they require the signature of the other party(ies) involved too.

Let’s say Alice and Bob are co-signatories to a multi-signature address. When funds are required to be withdrawn from Alice’s wallet, we need both Alice and Bob to provide their signatures. This is ensured by creating transactions from UTXOs (unspent transaction outputs) that have a `scriptPubKey` that requires 2-of-2 validation (done either by using the P2SH or P2MS script type).

The steps are as follows:

1. Alice initiates a transaction on the multi-signature address.
2. Bob verifies that it is Alice only who initiated the transaction, through their own off-chain methods.
3. Once Bob sufficiently verifies this, Bob provides their signature on the transaction.
4. The transaction can now be posted to the blockchain since both Alice and Bob’s signatures are present.
5. If Bob were not sure of Alice’s identity, they would not supply their signature and the transaction would not take place.

**Question 4. [10 Points] Based on my lecture note, explain how a SPV node can check a person’s balance using a Bloom filter.**

**Answer:**

A bloom filter is a probabilistic mechanism that tells us whether a particular entry is either not present in the set, or that it *might* be present in the set. The ambiguity in whether it *might* be in the set allows for tools to quickly check for an entry’s presence.

The Bloom Filter is used by SPVs to retrieve transactions from a full node in a blockchain. The SPVs, due to space constraints, cannot store an entire copy of the blockchain but only store the much smaller copy of the block headers. SPVs then use the bloom filter to anonymously request for the transactions of the address that the SPV wallet is maintaining.

1. The SPV creates a bloom filter based on the address, keys and hashes of the user’s wallet
2. The SPV sends this bloom filter to the full-nodes requesting for all transactions in the blockchain that match this bloom filter.
3. Inside each transaction, the full node checks the public key hashes, scripts, OP\_RETURN values and public keys in signatures, and sends matching transactions (including their block headers, merkle tree roots and other messages related to the bloom filter) back to the SPV.
4. Once received by the SPV, it discards all information that does not match the user’s hashes or addresses.
5. Finally, the SPV updates its own copy of the UTXOs that are related to the user’s transactions.
6. The bloom filter is updated with the UTXOs that it received.

**Question 5. [10 Points] Based on my lecture note, explain how a bidirectional payment channel works.**

**Answer:**

Bidirectional payment channels are used when two parties require to pay each other on a frequent basis. The following is how a bidirectional payment channel works:

1. A multi-signature (2-of-2) address is created for the two parties (let's say A and B). Each sends an initial amount (say 2 BTC) to the address from their personal wallet. The multi-sig address balance is now 4BTC.
2. A and B create hashlocked secret keys. A1 = H(a1) and B1 = H(b1).
3. After this, they retrieve the contributed bitcoins back to their own personal addresses from the payment channel to get proof of each of their contributions.
4. Now, A can send B repeated transactions. After A sends one transaction to B, B signs another transaction and sends it to A. The transactions can be signed by anyone without the risk of bitcoin theft from each of the parties involved.
5. These transactions are continued.
6. A and B now exchange their 1st hashlocked secret keys.
7. Anyone of A or B can now broadcast all the transactions to the blockchain.