Summary – `Create account`

Huajing Yu

I. The new concepts

I have learned three new concepts in this week.

1. The simplified payment verification (SPV) clients connect the full nodes to validate their transactions because SPV cannot determine whether the payee's transaction is valid for lacking full UTXO database. The SPV clients are usually resource-constraint devices that only store a part of blockchain like block headers to ensure it has the longest chain, and they leverage the bloom filter for receiving the required transactions filtered by the full nodes that they are connected to. The SPV clients confirm a transaction through how many blocks are referred to the block where the transaction is timestamped.

2. Ethereum doubles the gas limits size for including more transactions because the payee has to provide a higher gas fee for their transactions to be executed ahead of other transactions in a block, and that will cause more state change in one block.

The gas consumption of operating contracts is based on the time needed to process the opcodes. However, storaging-access opcodes (SLOAD, CALL, BALANCE, and EXP) have been underpriced, which provides attackers a chance to perform a denial-of-service (DoS) attack where attackers are simply sent transactions that access or call lots of accounts and make the block verification time up to 80s, and the DoS attack could be easier in a double-sized block.

In the Ethereum improvement proposals 2929, the gas consumption of storaging-access opcodes is increased when they are executed for the first time, but it is low in subsequent access. For minimizing the gas costs in normal transactions, in the Ethereum improvement proposals 3030, the transaction maintains a set 'accessed_addresses: Set[Address]' and 'accessed_storage_keys: Set[Tuple[Address, Bytes32]]'. The accessed_addresses is initialized to include the 'tx.sender', 'tx.to' (or the address being created if it is a contract creation transaction), the initialized address would eliminate a part of effects caused by the Ethereum improvement proposals 2929.

3. In the Ethereum improvement proposals 3607, transactions will be rejected from senders with deployed code. Ethereum address has currently 160-bit long, and this means that it could be a collision between contract-owned accounts (SOA) and externally owned accounts (EOA) that are generated by formula as follows:

Ethereum EOA address is computed as 160-bit hashes:

$$A_{pk} = Keccak_{256}(K)[0..160] \tag{1}$$

where $K = [k]P$ is the public key derived from secret key k and $[0..160]$ indicated that we take only 160 bits of the output.

Ethereum SOA address is computed as 160-bit hashes in two ways based on 'CREATE' and 'CREATE2'. First, 'CREATE' create a contract address using a hash of a nonce and the creator address:

$$A_{ct} = Keccak_{256}(A, N)[0..160] \tag{2}$$

where A is the 'msg.sender' address and N is a nonce starting from 1.

'CREATE2' allows interactions to be made with addresses that do not exist yet on-chain, but can be created by a particular piece of init code. Important for state-channel use cases that involve counterfactual interactions with contracts.

$$\begin{aligned} A_{cr} = Keccak256(0xff, A, salt, \\ Keccak256(init\_code))[0..160] \end{aligned} \tag{3}$$

where $0xff$ is a single byte and $salt$ denotes the third item on the machine's stack and $init\_code$ is the runtime opcodes of contracts.

There is a hypothetical case that the SOA accounts' value is aimed to be spent by triggering the contracts, but which can then be spent using the EOA signature for the same address. The Ethereum improvement proposals 3607 fixes these potential attacks, which never allow transactions where 'msg.sender' has a 'CODEHASH != EMPTYCODEHASH' and 'EMPTYCODEHASH = crypto.Keccak256Hash(nil)'.