Since the official private data example <https://hyperledger-fabric.readthedocs.io/en/release-2.0/private_data_tutorial.html?highlight=using%20private%20data> does not use the ‘transient’ data field in the function *readMarblePrivateDetails(),* we develop another chaincode project called “saccPrivate” and I have attached it to this comment.

(If you want repeat my experiment, you need deploy the chaincode “saccPrivate” in a channel that contains three organizations and each organization has one peer. If you have any questions, you can contact me by shanwang1994@gmail.com)

**Experiment setup:**

In my experiment, there are 3 organizations in the hyperledger network. Each organization has one peer. All three peers join the one channel.

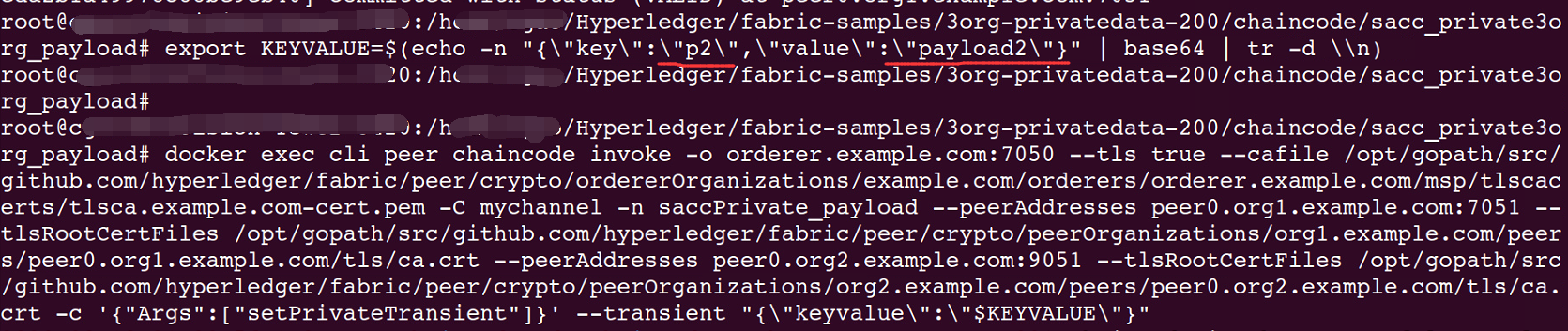
I deploy the “saccPrivate” chaincode on the channel. This chaincode defines the private data using the “org1-2.json” (contained in the attached too).

In the “sacc.go” chaincode, there is a function *getPrivateTransient()* which is used to read the private data. The private data collection is “privateall” in the org1-2.json, which contains org1 and org2, but not contain org3.

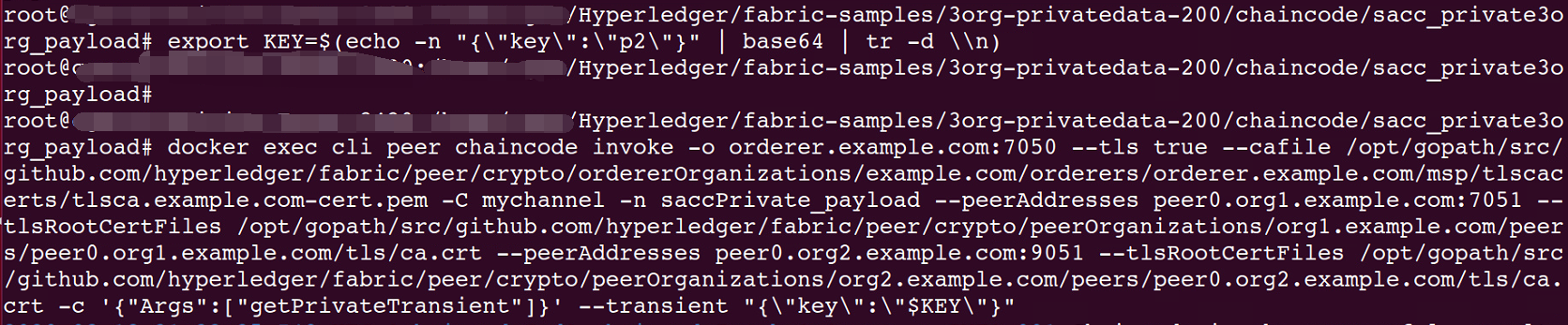
Next, I show the process how private data non-member peer0.org3 gets the actual private data.

**Experiment process and results:**

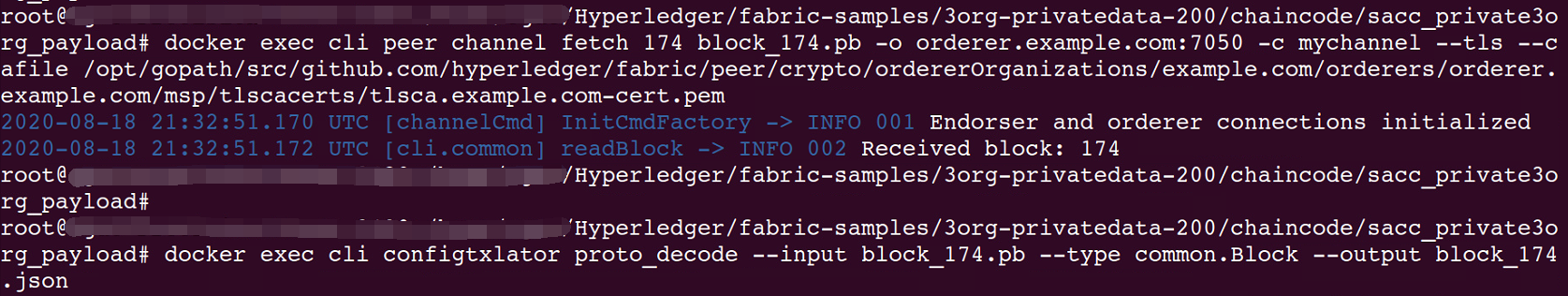
Step1: peer0.org1 invokes the *setPrivateTransient()* to write the private data <p2,payload2> to the world state.



Step2: peer0.org1 invokes the *getPrivateTransient()* to read the value of private data p2. This generates a transaction that is stored in block 174.



Step3: Peer0.org3 fetches the block 174 and decode it to a .json file.



Step4: I use a JSON viewer ( <https://countwordsfree.com/jsonviewer>) to read the block\_174.json file and find the corresponding transaction. The “payload” field in the transaction is encoded by Base64.



Step5: I use a tool (<https://www.base64decode.net/>) to decode the ‘payload’ and get the actual private data.

