**Initialisation / Full Process steps**

Step 1 – Ensure input files (upd\_teal.teal, importi\_cedole, lista\_cedole) are the correct / accurate versions

Step 2 - Run the Core (Once only), ensuring the input files are stored correctly

* This generates / outputs the contracts, QR codes, and address/private key/mnemonic data files for bookshops and students
* Tests that rebuilding of logic signatures after QR reading is accurate
* Makes initial transfers of funding, minimum balance requirements and fees
* Makes a test logic sig transaction for each student to bookshop[0]

Step 3 – Store correct bookshop address in qrScan.py bookshop\_address variable, to ensure a correct receiver

Step 4 - Run qrScan.py, click “Scan QR code” button, and scan a QR code from mobile phone or printed paper

Step 5 – Enter a valid amount and a transaction note and click “Enter”

Step 6 – Wait for transaction confirmation information (assuming success). If failure, follow error message instructions and restart qrScan.py.

Step 7 – Verify transaction data by searching for the transaction id on the testnet block explorer

Step 8 – Assuming the bookshop address has been imported into a mobile wallet (i.e. Pera) using the mnemonic, the bookshop can check their wallet for proof that the transaction was successful

**CORE**

**Input files:**

-IMPORTI\_CEDOLE – contains 4 columns: ANNO\_SCOLASTICO, CLASSE, IMPORTO, SENZA\_RELIGIONE

-LISTA\_CEDOLE – also 4 columns: ANNO\_SCOLASTICO,ID\_MINORE,CLASSE,CODICE\_SCUOLA

-upd\_teal.teal – current working smart signature contract code

**Account Generation**

The core generates an account for each student id in LISTA\_CEDOLE, and outputs a file “stud\_acc\_pk\_mn.csv” with student data “ID, Address, Private Key and Mnemonic” for each student.

This is also done for 2 bookshops, without IDs (assuming bookshop addresses can be delegated to bookshops after-the-fact)

**Contract Generation**

Using the upd\_teal.teal skeleton contract, a smart signature contract is generated for each student, substituting the correct student address, 2 bookshop addresses and the correct EmFi funding amount based on the class. These contracts are all saved in a folder and named accordingly to the student’s address

**Initial Transfers**

Initial transfers of 100,000 mAlgo are made to the bookshops to satisfy minimum account balance requirements and ensure the accounts are active and a further 10,000 mAlgo for provide for some fees

A transfer of the correct funding amount is made to each student based on which class they are in, as well as a further 110,000 mAlgo for minimum balance requirement and fees provision.

**Delegated Logic Signature Generation**

The program generates a delegated signature for each student contract that was previously created. Each contract is compiled, a Logic Signature object is created and signed by the private key of the associated address.

**QR Encoding**

Each delegated signature is then encoded into a QR code, along with the student address associated with the signature. These QRs are also saved in a folder and named according to the address.

**Testing**

Every delegated signature created is stored in a variable for later testing.

First we test the rebuilding of each logic signature object. The QR code png image is read and decoded and the signature object rebuilt. It is then compared with the original signature (pre-QR encoding) and checked for equality.

Finally, once all the delegated signatures are rebuilt and validated to be equal to its original counterpart, we send a test transaction to bookshop[0] from each student. These are LogicSigTransactions. If sent successfully, we have a proven transaction being sent from the student account to the bookshop, without the signature or private key of the student.

**qrScan**

**Step 1**:

The bookshop address must be updated in the bookshop\_address variable. This address has been generated by the core and can be found in two places: in book\_acc\_pk\_mn.csv file, or in the student contracts. If this is not updated to a valid bookshop address, all transactions will fail due to the teal logic failing.

**Step 2:**

Run the program. The first window will open with a button “Scan QR code”. This button will open the laptop camera and scan every millisecond, looking for a QR code.

**Step 3:**

Scan a QR code previously generated by the core. The scanner will decode the QR and rebuild the Logic Sig Object / delegated signature. It will also save the student address for use in the transaction. The user will be prompted to enter a transaction amount and a note.

**Step 4:**

If no amount or note is entered, an error message will be shown and the user will need to close the window and restart. If a correct amount and a note are entered, a LogicSigTransaction will be created and sent, provided the logic is not violated by overspending or sending to an incorrect address, or other logic abuses. An error message related to its cause will be printed.

**Step 5:**

Finally, the transaction information will be printed in the final window when the transaction is confirmed. The Transaction ID can be copied and searched for on the block explorer. The process can be repeated by closing the window and re-running the program.