Exercise 3: Authenticated transaction block (simplified Bitcoin)

CIS4930-6930 Spring 2018

Assigned: 3/15/18

Due: 3/29/2018

Associated Readings: Ch.4 and 6

Authenticated transaction block

Transaction – we already have a format for a basic transaction (ASCII encoded inputs, outputs, but not including the hash of the basic transaction). We will add signatures with public/private key pairs and validate the signature too. ASCII coded hexadecimal will be used for addresses for the keys. We will also restrict UTXOs that are inputs to the transaction to all be associated with the same account (that is, if Alice is the account name for one of the UTXOs used in the input, then Alice is the account name for all of the UTXOs used in the input).

Signing Transactions – we will add one form of authorization for spending a UTXO, namely, digital signatures. Since every UTXO in the input of a transaction belongs to the same account, the transaction only needs to have a single signature field that holds a digital signature by the account associated with the transaction inputs. The digital signature will cover only the basic transaction (i.e., number of inputs, inputs, number of outputs, outputs) and will use RSA and SHA256.

Transaction Block – We will include a field for the number of transactions, followed by that number of complete transactions (basic transaction + signature).

Exercise 3:

All (grad and undergrad):

Develop an interactive program called txblk that runs on storm.cise.ufl.edu.

It will build on the ledger program from exercise 2. Recall the commands for ledger:

[B]alance: Supply username: (e.g. Alice). This command prints the current balance of a user.

[D]ump: Supply filename:<outfilename>. Dump ledger to the named file. Print an error message to stderr if the output file named cannot be opened. The message shall be “Error: file <outfilename> cannot be opened for writing” on a single line, where <outfilename> is the name provided as additional command input.

[E]xit: Quit the program

[F]ile: Supply filename:<infilename>. Read in a file of transactions. Any invalid transaction shall be identified with an error message to stderr, but not stored. Print an error message to stderr if the input file named cannot be opened. The message shall be “Error: file <infilename> cannot be opened for reading” on a single line, where <infilename> is the name provided as additional command input.

[H]elp: Command Summary

[I]nteractive: Toggle interactive mode. Start in non-interactive mode, where no prompts are printed for commands or additional information. Print command prompts and prompts for additional input in interactive mode to stdout, starting immediately (i.e., print a command prompt following the I command).

[P]rint: Print current ledger (all transactions in the order they were added) to stdout in the transaction format given below, one transaction per line.

[T]ransaction: Supply Transaction:<see format below> Read in a single transaction in the format shown below. It shall be checked for validity against the ledger, and added if it is valid. If it is not valid, then do not add it to the ledger and print a message to stdout with the transaction ID as given followed by a colon and a space, and either “good” or “bad” followed by a newline. Also print an error message to stderr with the transaction number followed by a colon, a space, and the reason it is invalid on a single line.

[V]erbose: Toggle verbose mode. Start in non-verbose mode. In verbose mode, print additional diagnostic information as you wish to stdout. At all times, in verbose mode or non-verbose mode, output each transaction number as it is read in, followed by a colon, a space, and the result (“good” or “bad”) to stdout. In verbose mode, print additional diagnostic information as you wish to stdout. Additional error information may always be printed to stderr in either mode.

[W]ipe: Wipe the entire ledger to start fresh.

Additional commands are:

[O]utput transaction block: collect all correctly signed transactions that have not been output in a previous transaction block and output them as a transaction block. This outputs the current block only. This includes outputting a line with a single integer indicating the number of signed transactions that follow, followed by those transactions.

For example, if the ledger has 7 transactions, you would output:

7

Transaction1

Signature1

Transaction2

Signature2

. . .

Transaction7

Signature7

A signed transaction consists of one line with the basic transaction (as in Exercise 2, including the newline) immediately followed by another line containing the digital signature for that transaction in ASCII coded hexadecimal (followed by a newline). Any unsigned transactions shall remain in the ledger pending signing.

[C]heck transaction signature: supply <transactionID>. The signature of the signed transaction (in the two-line format given above) shall be checked. Output “OK” to stdout if good, else output “Bad” to stdout. If bad, output additional diagnostic information to stderr.

[R]ead key file: supply <account name> <keyfilename>. <account name is the name of the account associated with the key (e.g., Alice, Bob, Gopesh). <keyfilename> is the name of the file containing the public key associated with the account named. If the account already has a keyfile of the same type that has been read in associated with it, the new file contents shall supersede the old version. A warning may be printed to stderr. If the operation succeeds, then output “Public key <account name> <keyfilename> read” to stdout. If there is an error of any sort, output “Error: Public key <account name> <keyfilename> not read” to stdout. Additional diagnostic information may be printed to stderr explaining what the problem is. Problems include invalid account name (syntactic error), and various errors opening and reading the file that is supposed to contain the key.

Semantic changes to previous commands:

[D]ump shall only output transactions that have not been output in a transaction block already.

[P]rint shall only output transactions that remain in the ledger and have not been output in a transaction block already.

[W]ipe shall only remove transactions that have not been output in a transaction block already.

Grad only:

Implement another interactive program that runs on storm.cise.ufl.edu called wallet. Wallet also builds on ledger, with additional commands as follows:

[R]ead key file: supply <account name> <keyfilename>. <account name is the name of the account associated with the key (e.g., Alice, Bob, Gopesh). <keyfilename> is the name of the file containing the private key associated with the account. If the account already has a keyfile of the same type that has been read in associated with it, the new file contents shall supersede the old version. A warning may be printed to stderr. If the operation succeeds, then output “Key <type> <account name> <keyfilename> read” to stdout. If there is an error of any sort, output “Error: Private key <account name> <keyfilename> not read” to stdout. Additional diagnostic information may be printed to stderr explaining what the problem is. Problems include invalid account name (syntactic error), and various errors opening and reading the file that is supposed to contain the key.

[S]ign transaction: supply <transactionID>. Transaction (not including the transaction ID) from the integer m (specifying the number of inputs) through the newline (following the last output) shall be signed using SHA256 and RSA, with the private key associated with the account that owns the inputs. The signature shall be appended to the transaction on a line by itself, and the signed transaction shall be output to stdout. If there are any errors, an error message “Error: transaction not signed” shall be output to stdout. Additional diagnostic information may be output to stderr. Errors include invalid transaction ID (not in ledger), invalid transaction (transaction has multiple accounts that own inputs), and no private key found (program has not been given the private key file corresponding to the account that owns the inputs).

Submission: (use C, C++, or Java)

1) a makefile for all programs and support code that makes the executable; make all must do whatever is necessary to create an executable file called ledger that will run the program;

2) all source code (header files, etc.);

3) README.txt file that explains the environmental expectations of the code and lists any bugs and how to run it;

4) Brief textual report in text, doc, docx, or PDF format that reflects on how you approached the problem and how you solved any challenges, what you did to test the program, and what you learned from the assignment.

Compile errors will result in zero points as non-testable. Segmentation faults and core dumps will be judged depending on how much correct information is displayed. TEST YOUR MAKEFILE AND CODE ON STORM BEFORE SUBMITTING!

Rubric: (all) 10 points possible

Read transactions in and check for correctness

Check transaction has only one owner for all inputs

Read key file – catch errors

Correctly verify signature on transaction

Correctly output block of signed transactions

Other functions

Readme

Report

Rubric: (grad) 15 points possible

Read key file – catch errors

Check transaction has only one owner for all inputs

Correctly sign transaction and output it

Readme

Report