RSA implementation

RSA class is common for all Bank, Client and Merchant. To produce keys first we have generated all the prime numbers from 1 to 1000 the arbitrarily chosen two primes.

$$n = p1 * p2$$

After that calculated $phi = (p1-1)* (p2-1)$. Using this value generated multiple ***e*** and ***d*** pairs such as $e*d = 1 \mod phi$, but choose only one to work with.

RSA class encrypts and decrypts data, creates digital signature and extracts original signature from blind signature. Also, requests blinded from this class. To blind a request, we first generated multiple modular inverse pairs when ***n*** was calculated (on Bank server side) or received from the Bank server (on client side). If ***k'*** is modular inverse of ***k*** then,

$$k*k' = 1 \mod n$$

eCash class

eCash class is also common for all Bank, Client and Merchant. eCash class stores uniqueID, splitted shared secret and in cases (for Merchant and Bank) stores randomly generated bit sequence and their corresponding left and right parts. Also, digital signature is stored here.

Public Key and other request sharing

For public key sharing, Client have to request to the Bank Server. I used UDP P2P connection for all data transfer between Bank, Client and Merchant.

Commands

get pkey - Sends request to the Bank for public key pair. pkey stands for public key
mint X - Here X is numeric value, sends mint request to the BAnk for amount X
mint X cheat - Tries cheating while sending blind requests
show - Shows available eCash
transfer X - Transfers X amount to the Merchant
transfer X y - Transfers X amount to the Merchant, but keeps the cash to double spend
deposit X - Deposits X amount to the Merchant
deposit X y - Deposits X amount to the Merchant, but keeps the cash to double spend
stop - stops running the application

stop command is supported by all three. Other than stop, Merchant supports *deposit* and *show*. Client supports all the commands.

Design

Bank - Client

Bank server generates all it's keys at initialization. When, Client is initialized it requests the Bank to share the public key and the Bank sends a response with (n,e) pair.

Mint requests between client and bank shown in figure 1.

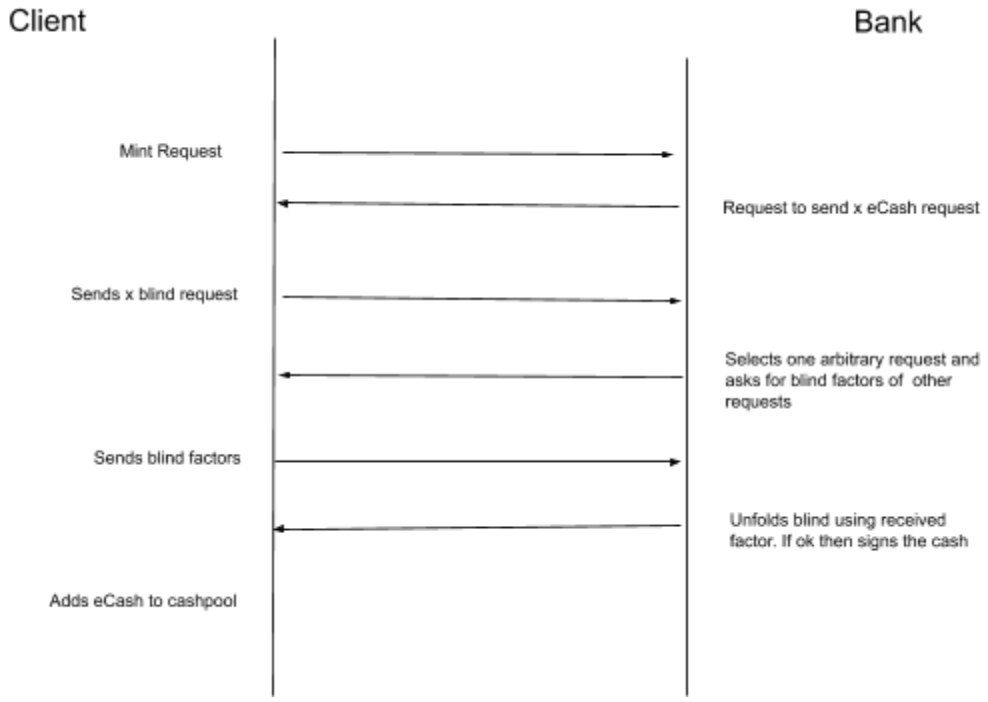

Fig. 1 mint request

To deposit eCash, the client sends the eCash to deposit to then bank, bank generates digital signature of the cash and matches it with the attached digital sign. If matches then the cash is valid. Now, the bank looks if the cash was deposited earlier or not, if no then stores it in the bank cashpool. But if otherwise then using bitcommitment pairs tries to find the cheater.

Bank - Merchant

Deposit process for the Merchant is the same as the Client.

Merchant - Client

Transfer request between the Client and Merchant is shown in figure 2.

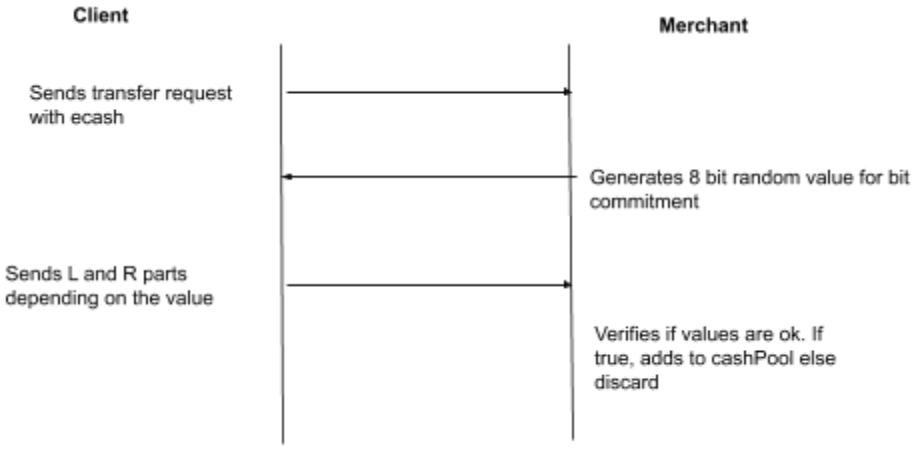

Figure. 2 transfer request