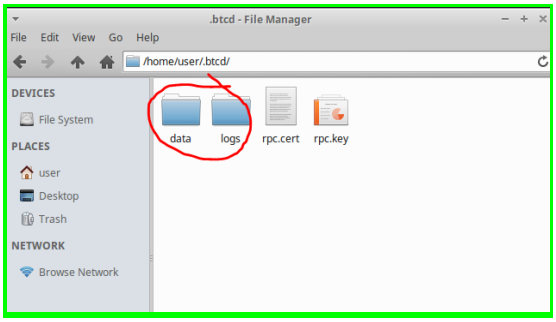
Okay so next time i go through the instructions

-make sure the extra log file is deleted (hidden file)



-run the crossed out command instead of the other one

-have everyone mine blocks for funds before you actually create the channels

check to see if this is in the whitepaper that even if 1 person funds it, there has to be at least twice the amount in the channel

general instructions

steps of stuff you have to do

commands to copy and paste into terminals

the addresses and keys change each time you restart the network

copy and paste into files

keyboard instructions (to stop a process: ctrl + C or ctrl + z)

which directory to be in

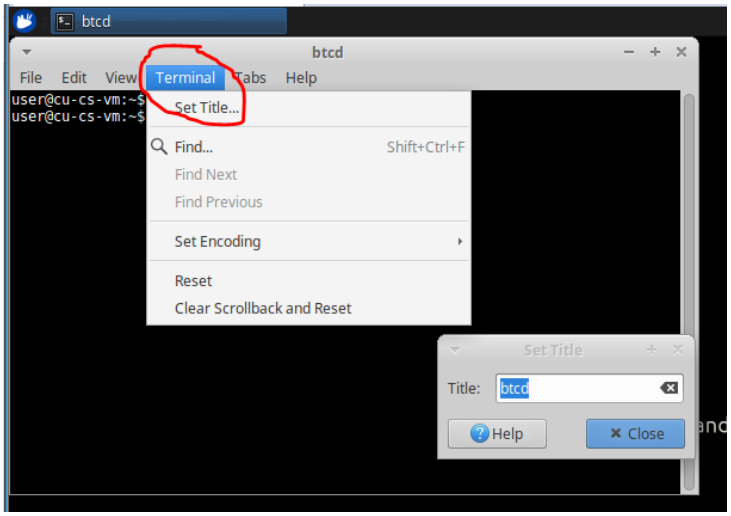
LN white paper application/ explanations

If you accidentally close the btcd , just close the whole window and enter

cd $GOPATH

```
btcd --txindex --simnet --rpcuser=kek --rpcpass=kek
```

to name terminals:



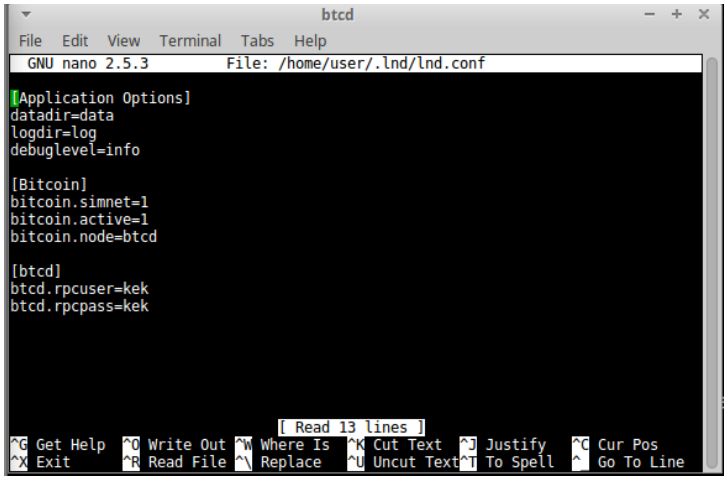
From <https://dev.lightning.community/tutorial/01-lncli/index.html>

STARTING HERE ONLY DO THIS THE FIRST TIME

add flags to the lnd.conf file, type this from any directory

nano ~/.lnd/lnd.conf

cd



to save the file then exit it

ctrl +O

enter

ctrl + X

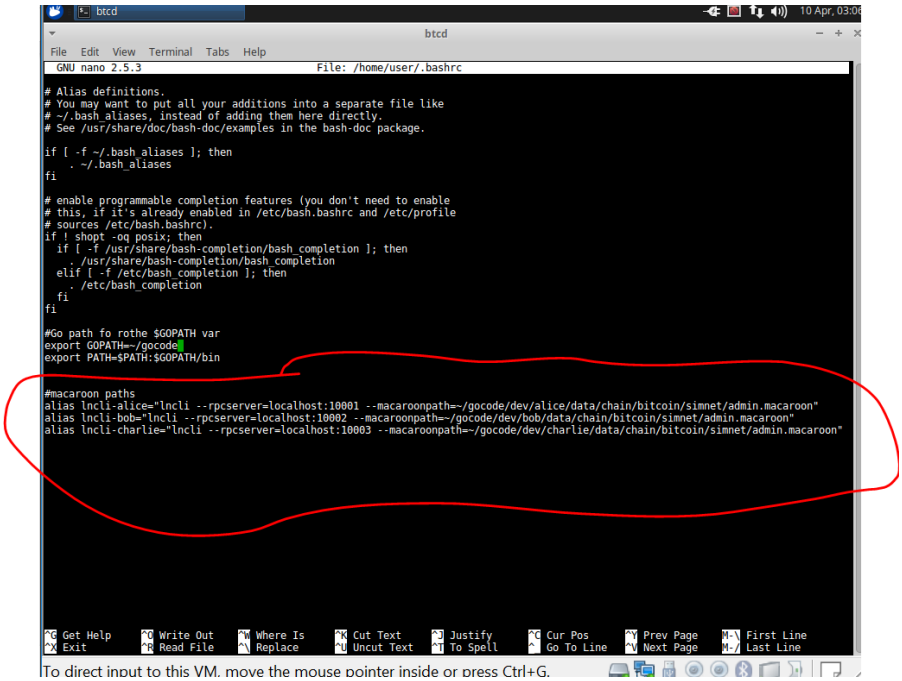
create macaroon paths in bashrc file at the end:

open the bashrc file from any terminal

sudo nano ~/.bashrc

and put this in the bashrc file at the end of the file:

```
#macaroon paths
alias lncli-alice="lncli --rpcserver=localhost:10001
--macaroonpath=~/gocode/dev/alice/data/chain/bitcoin/simnet/admin.macaroon"
alias lncli-bob="lncli --rpcserver=localhost:10002
--macaroonpath=~/gocode/dev/bob/data/chain/bitcoin/simnet/admin.macaroon"
alias lncli-charlie="lncli --rpcserver=localhost:10003
--macaroonpath=~/gocode/dev/charlie/data/chain/bitcoin/simnet/admin.macaroon"
```



to save the file then exit it

ctrl +O

enter

ctrl + X

TO EXECUTE THE BASHRC FILE:

source ~/.bashrc

END OF FIRST TIME INITIALIZATION OF FILES

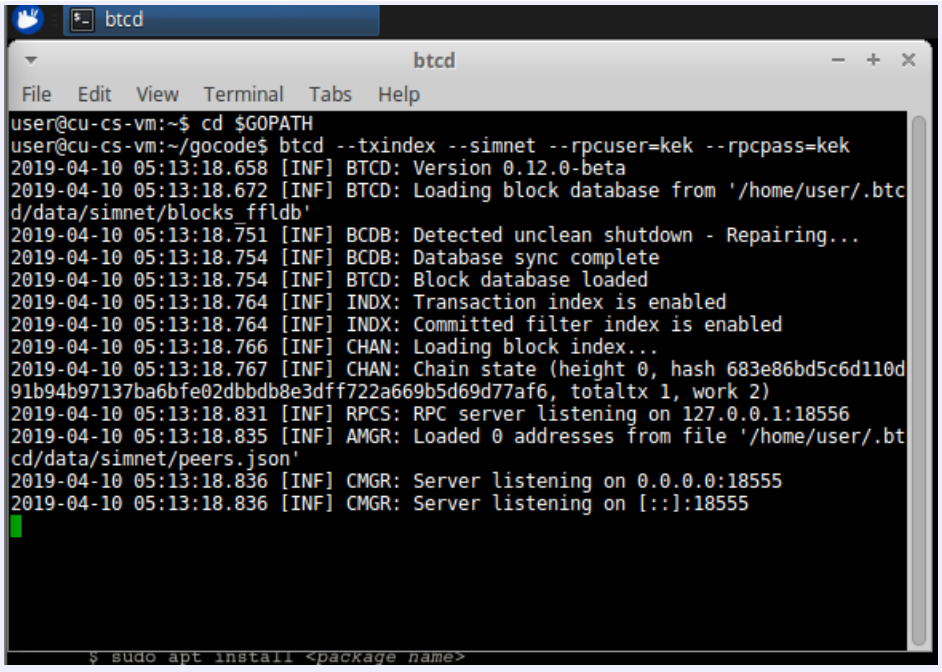
TO START OR RESTART THE NETWORK

(assuming not using seed phrases, but starting with new wallets)

there will be 1 bitcoin daemon (btcd) terminal + 2 terminals per node opened, 1 as the lnd and 1 as the wallet

the btcd

```
cd $GOPATH
btcd --txindex --simnet --rpcuser=kek --rpcpass=kek
```



START THE NODES

each one in their own terminal: (these can be tabs or new windows, doesn't matter)

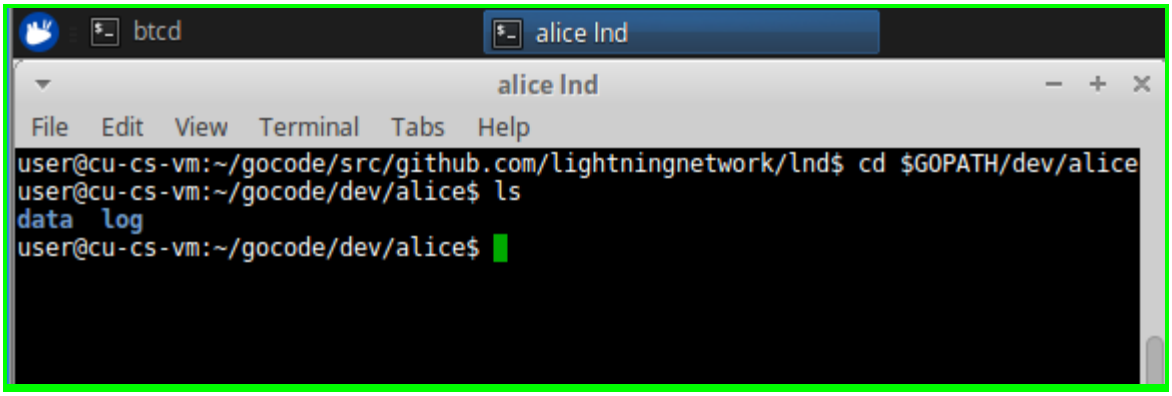
for alice

l

(if you are starting the wallets over) you need to **delete the log and data files for each node**

check if you have old logs or data if you don't want to use seed phrases(want to start over with new wallets)

ls

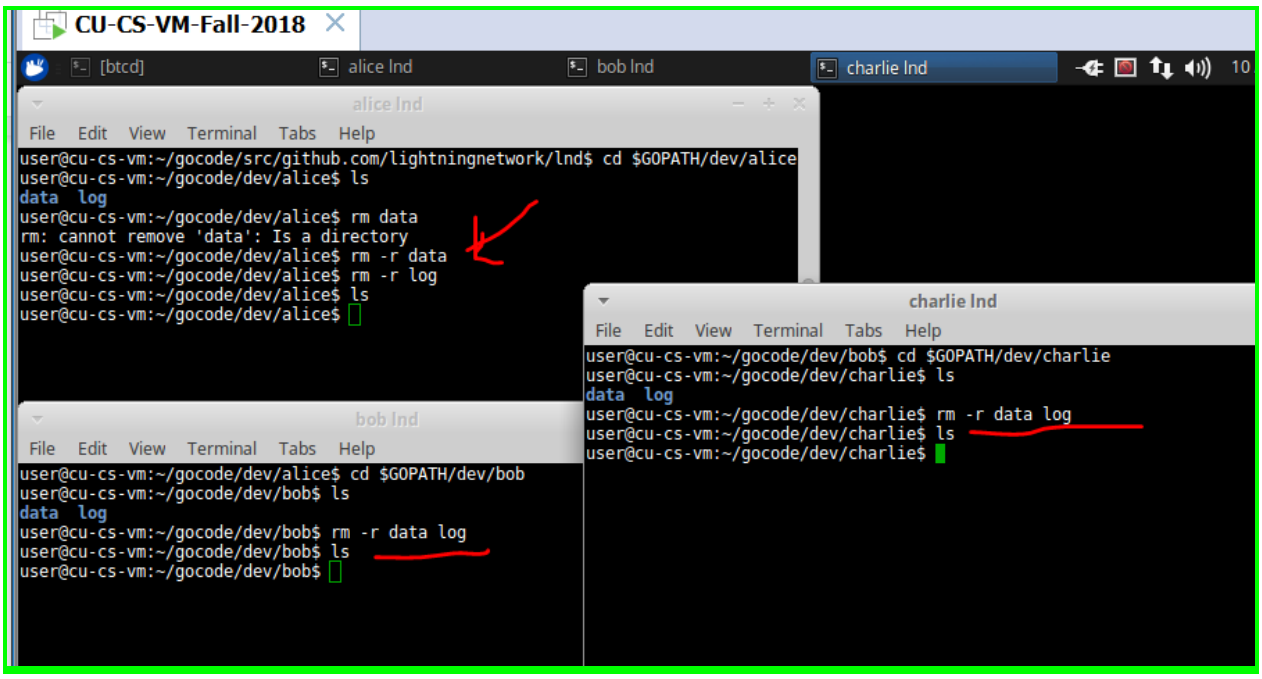


if you have data and log directories in the dev folder, delete both data and log

DO THIS CHECK FOR EVERY NODE from there dev/<node> location

rm -r data log

ls



all dev/<node> folders should be empty or else will get listening error on the ports if starting new wallets

SET UP LISTENING PORTS (NODES)

each in their own, different terminal (this will be the 2nd terminal for each node)

for alice

in a new terminal (alice lnd)

cd $GOPATH/dev/alice

~~****use this one~~

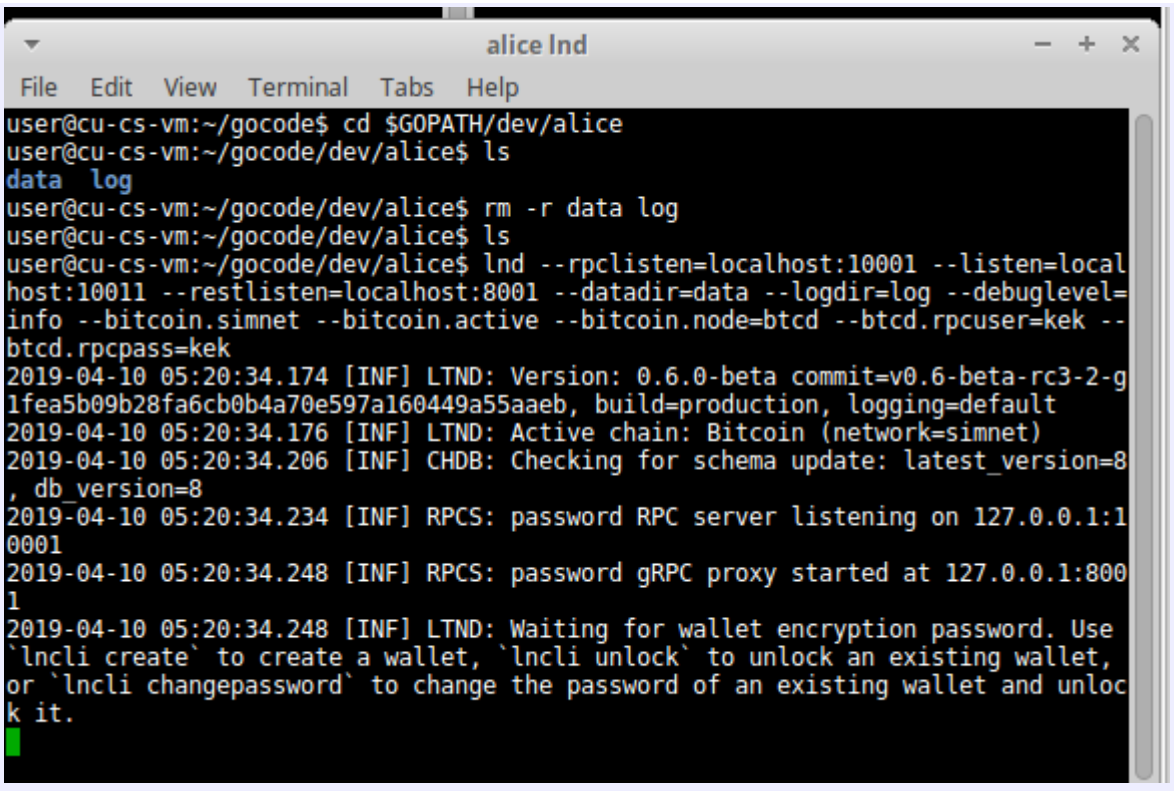
~~lnd --rpclisten=localhost:10001 --listen=localhost:10011 --restlisten=localhost:8001~~

that is not working for me

lncli-alice getinfo

***dont use this oneXXXXX

lnd --rpclisten=localhost:10001 --listen=localhost:10011 --restlisten=localhost:8001 --datadir=data --logdir=log --debuglevel=info --bitcoin.simnet --bitcoin.active --bitcoin.node=btcd --btcd.rpcuser=kek --btcd.rpcpass=kek



leave it running like in the picture above, waiting for wallet encryption

SET UP WALLETS:

each one in their own terminal: (these can be tabs or new windows, doesn't matter)

I used password for the password for all of them

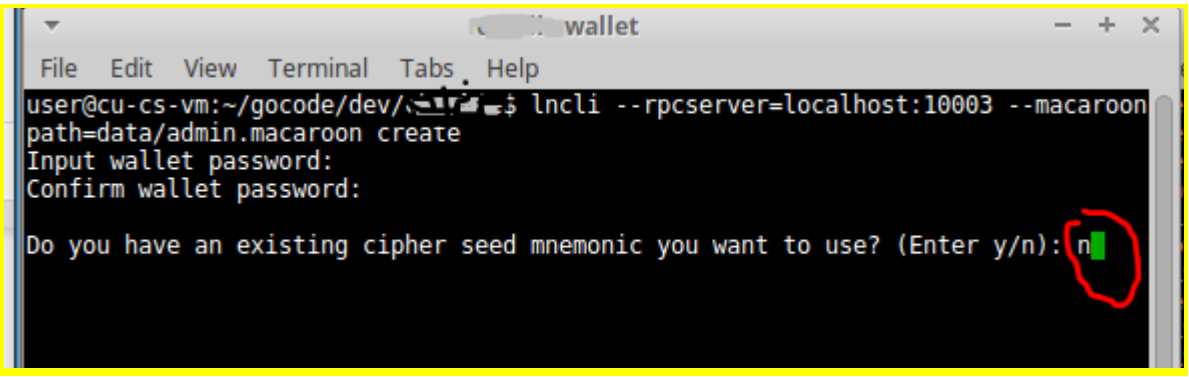
for alice

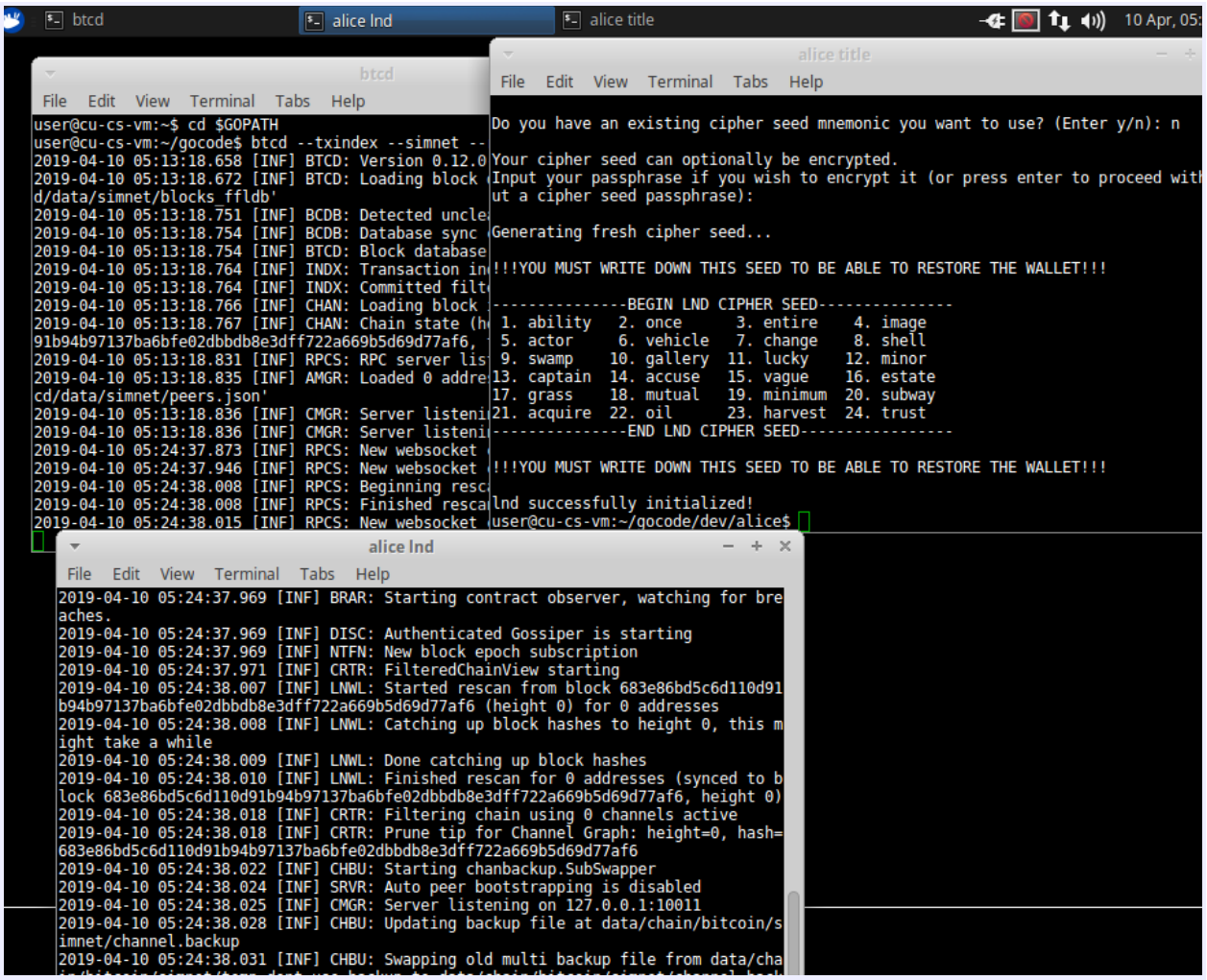
in a new terminal (alice wallet)

cd $GOPATH/dev/alice

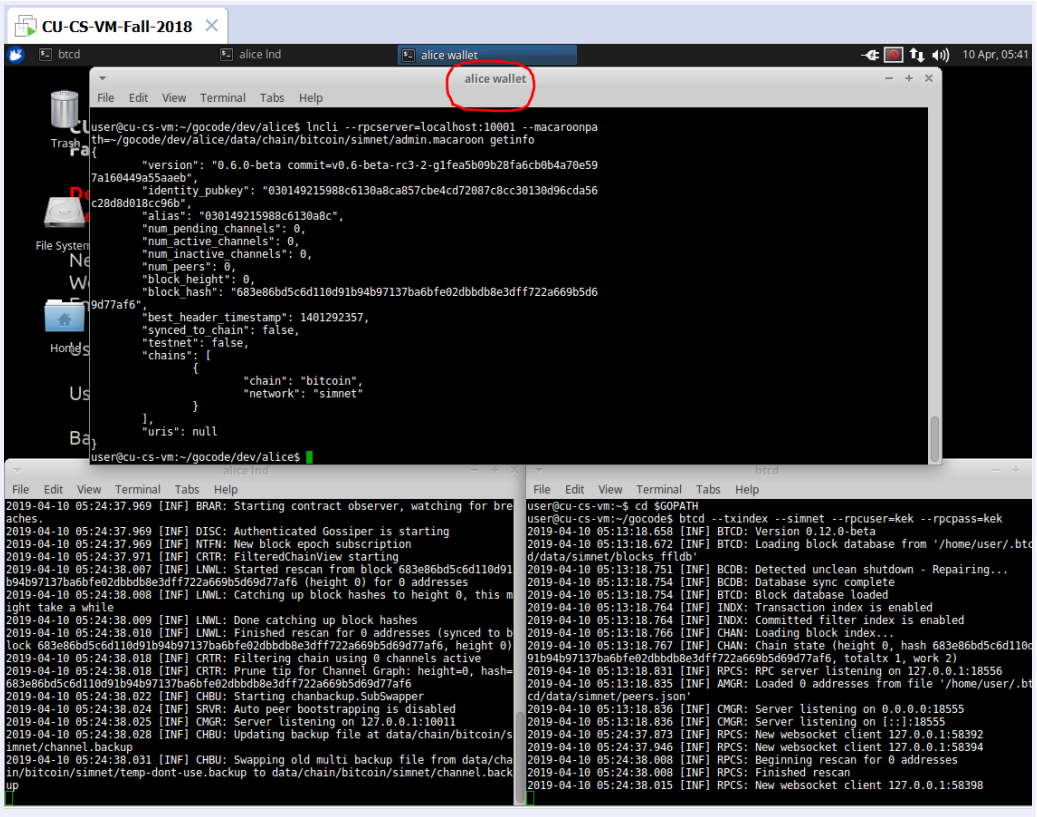
lncli --rpcserver=localhost:10001 --macaroonpath=data/admin.macaroon create

when it asks for a password you can use password, then say n, then enter



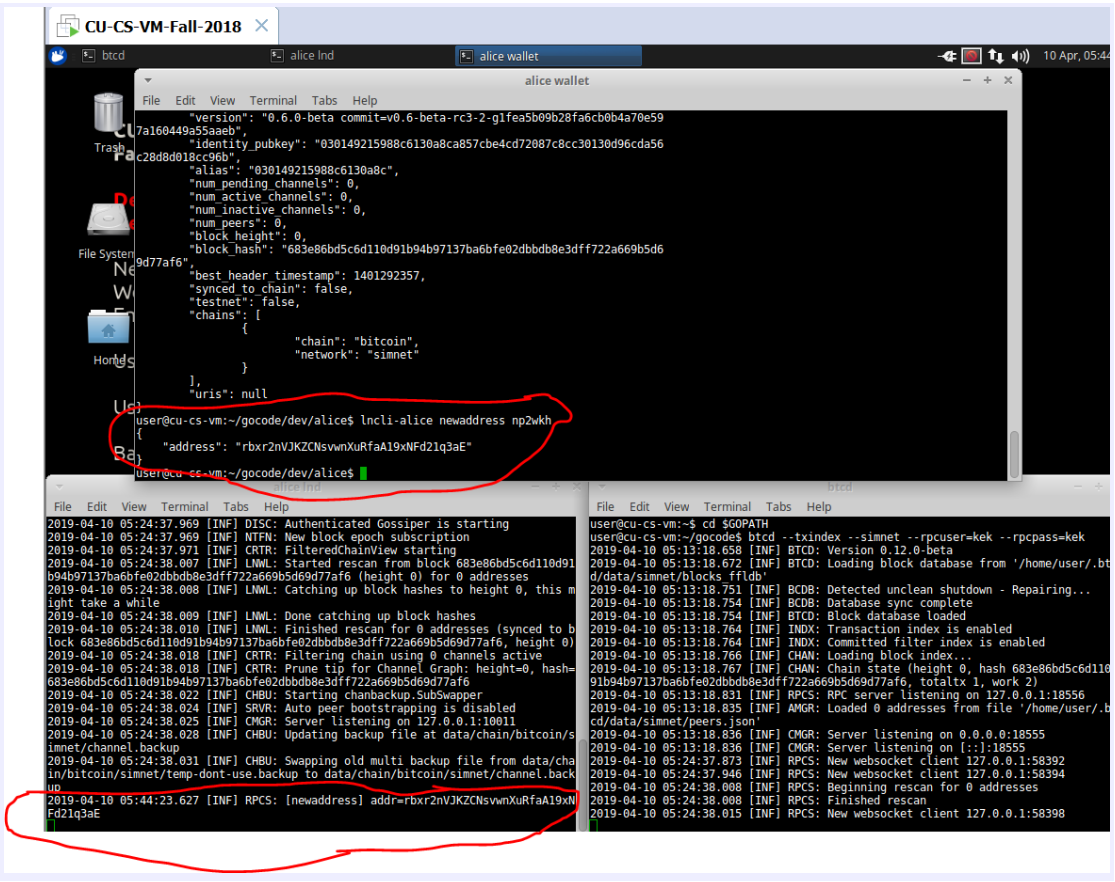


lncli --rpcserver=localhost:10001
--macaroonpath=~/gocode/dev/alice/data/chain/bitcoin/simnet/admin.macaroon getinfo



GET ALICE'S ADDRESS

Incli-alice newaddress np2wkh

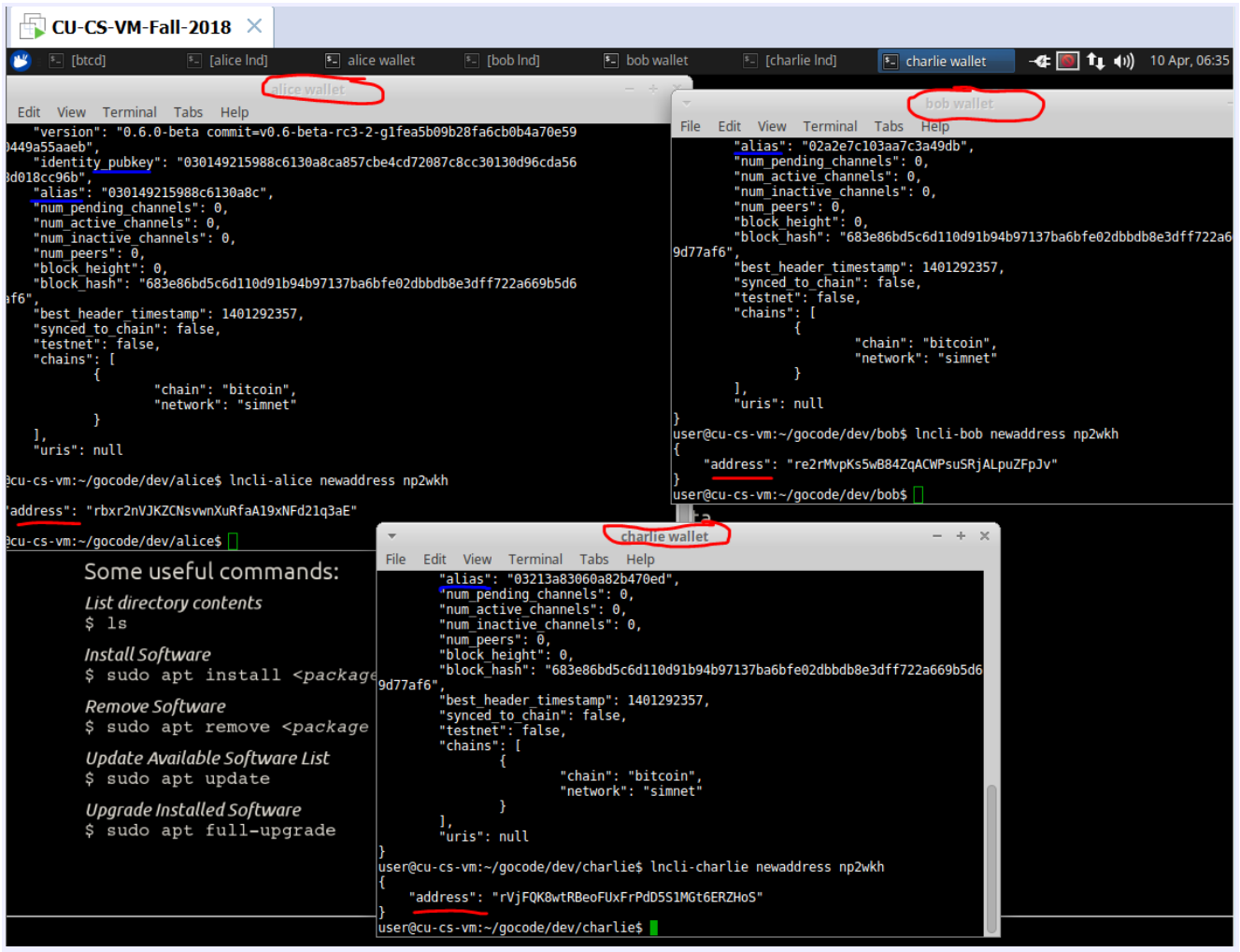


~~if this is first time ***ever*** opening wallets, do source bashrc for each lnd terminal, otherwise skip this step~~
~~source ~/.bashrc~~

SET UP THE PEERS (OTHER NODES) IN THE SIMNET

COMMAND LINE TERMINAL COMMANDS PER NODE		
after having made **the btcd** **`cd $GOPATH`** **`btcd --txindex --simnet --rpcuser=kek --rpcpass=kek`**		
for alice alice lnd terminal in a new terminal (alice lnd) **SET UP LISTENING PORT**	for bob bob lnd terminal in a new terminal (bob lnd) **SET UP LISTENING PORT**	for charlie charlie lnd terminal in a new terminal (charlie lnd) **SET UP LISTENING PORT**
cd $GOPATH/dev/alice ls rm -r data log ls lnd --rpclisten=localhost:10001 --listen=localhost:10011 --restlisten=localhost:8001 --datadir=data --logdir=log --debuglevel=info --bitcoin.simnet --bitcoin.active --bitcoin.node=btcd --btcd.rpcuser=kek --btcd.rpcpass=kek	cd $GOPATH/dev/bob ls rm -r data log ls lnd --rpclisten=localhost:10002 --listen=localhost:10012 --restlisten=localhost:8002 --datadir=data --logdir=log --debuglevel=info --bitcoin.simnet --bitcoin.active --bitcoin.node=btcd --btcd.rpcuser=kek --btcd.rpcpass=kek	cd $GOPATH/dev/charlie ls rm -r data log ls lnd --rpclisten=localhost:10003 --listen=localhost:10013 --restlisten=localhost:8003 --datadir=data --logdir=log --debuglevel=info --bitcoin.simnet --bitcoin.active --bitcoin.node=btcd --btcd.rpcuser=kek --btcd.rpcpass=kek
for alice alice wallet terminal in a new terminal (alice wallet) **SET UP WALLET** used password for the password	for bob bob wallet terminal in a new terminal (bob wallet) **SET UP WALLET** used password for the password	for charlie charlie wallet terminal in a new terminal (charlie wallet) **SET UP WALLET** used password for the password
cd $GOPATH/dev/alice	cd $GOPATH/dev/bob	cd $GOPATH/dev/charlie

lncli --rpcserver=localhost:10001 --macaroonpath=data/admin.macaroon create (if you have seed phrase enter it here and press return on next 2 screens) lncli --rpcserver=localhost:10001 --macaroonpath=~/gocode/dev/alice/data/chain/bitcoin/simnet/admin.macaroon getinfo lncli-alice newaddress np2wkh	lncli --rpcserver=localhost:10002 --macaroonpath=data/admin.macaroon create (if you have seed phrase enter it here and press return on next 2 screens) lncli --rpcserver=localhost:10002 --macaroonpath=~/gocode/dev/bob/data/chain/bitcoin/simnet/admin.macaroon getinfo lncli-bob newaddress np2wkh	lncli --rpcserver=localhost:10003 --macaroonpath=data/admin.macaroon create (if you have seed phrase enter it here and press return on next 2 screens) lncli --rpcserver=localhost:10003 --macaroonpath=~/gocode/dev/charlie/data/chain/bitcoin/simnet/admin.macaroon getinfo lncli-charlie newaddress np2wkh



TO GIVE ALICE MONEY RESTART BTCD

for all the lnd terminals, do ctrl+z or ctrl+c

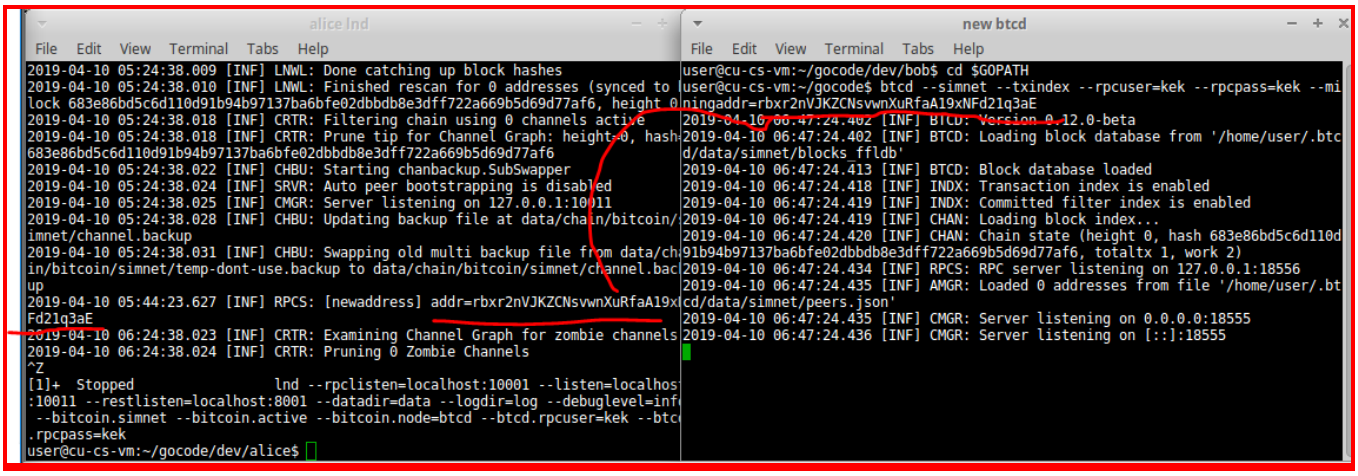
close the btcd terminal window completely and open a new one

from new btcd terminal:

THE ADDRESSES WILL CHANGE EACH TIME YOU RESTART THE NETWORK IF YOU DONT USE THE SEED PHRASE!! IF YOU USE THE SEED PHRASE, DON'T USE THE newaddress np2wkh COMMAND

```
cd $GOPATH
```

btcd --simnet --txindex --rpcuser=kek --rpcpass=kek --miningaddr=rgBgmPryVzATHihNZTShnEiwAArLt8izpb

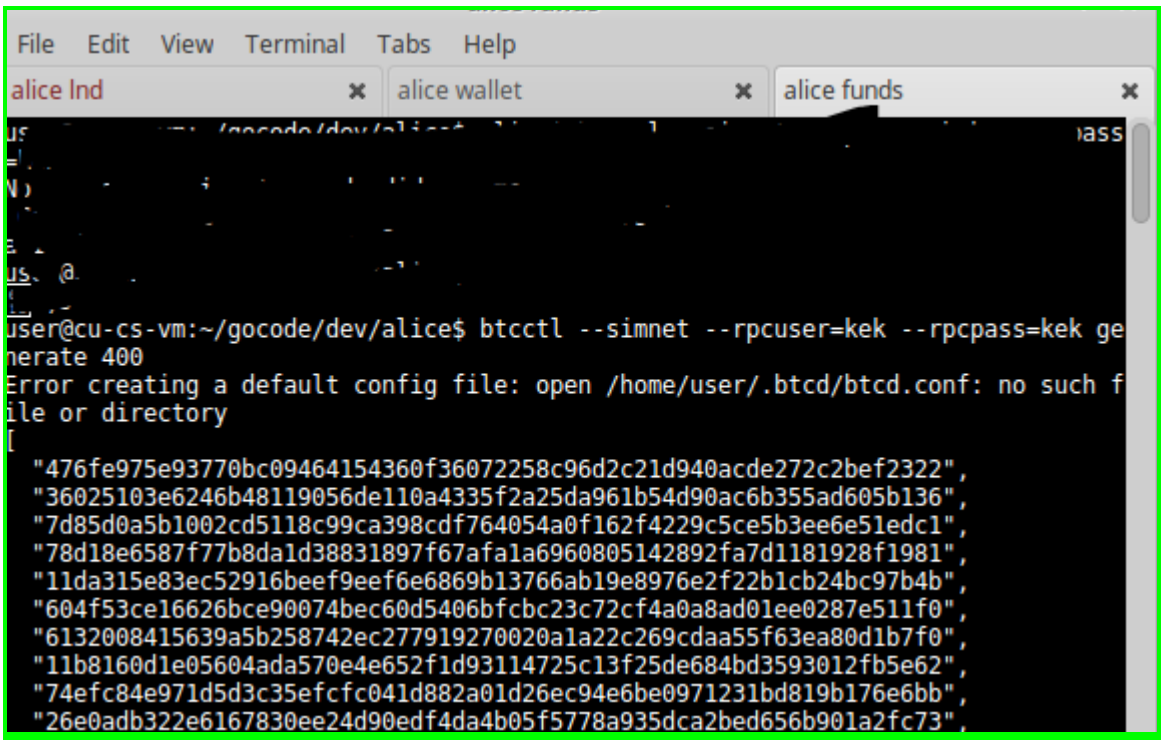


open a NEW window, alice miner:

mine 400 blocks, restart the new btcd window like so:

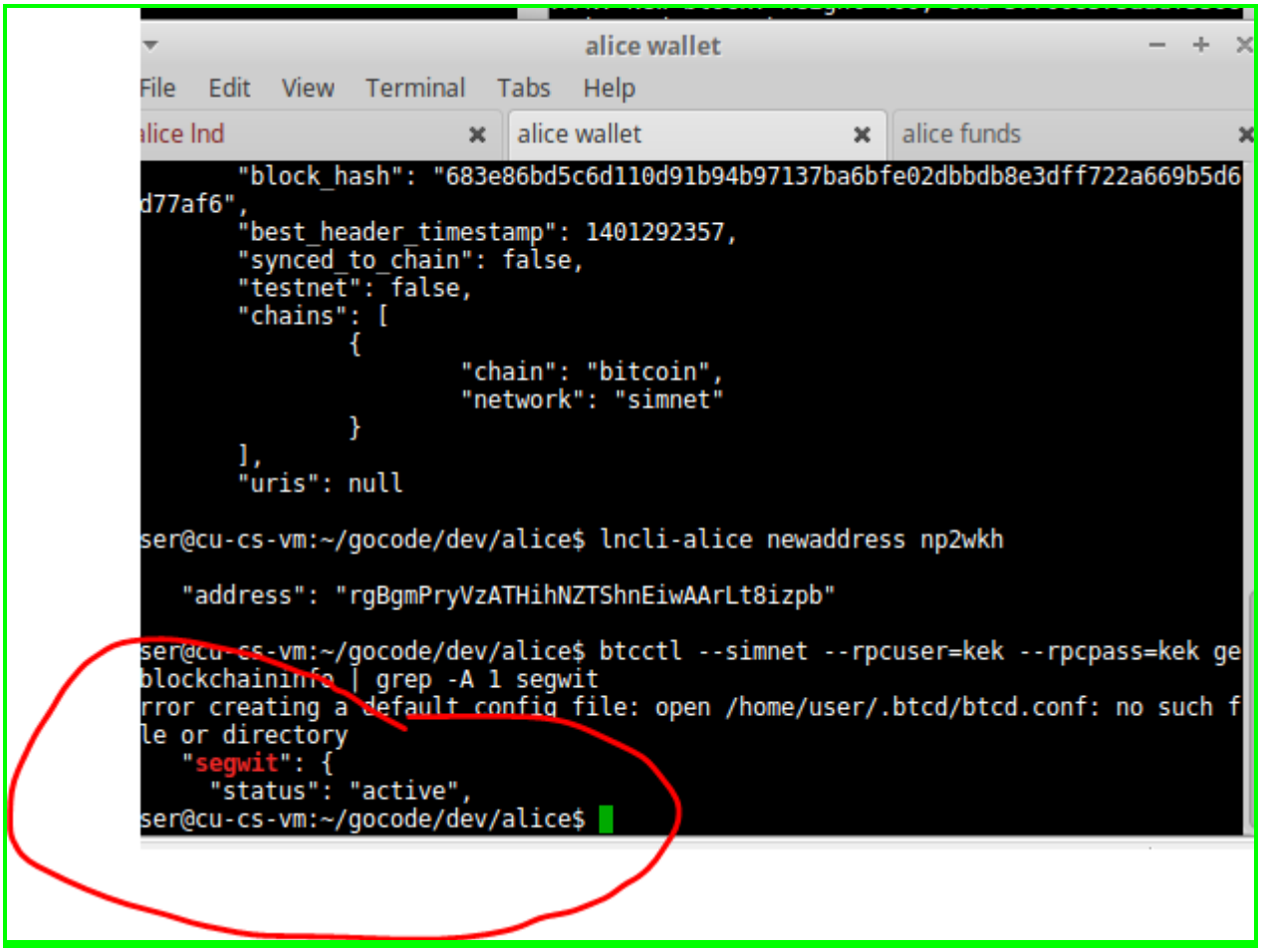
btcctl --simnet --rpcuser=kek --rpcpass=kek generate 400

(it will say error creating directory then the blocks will get mined like so:)



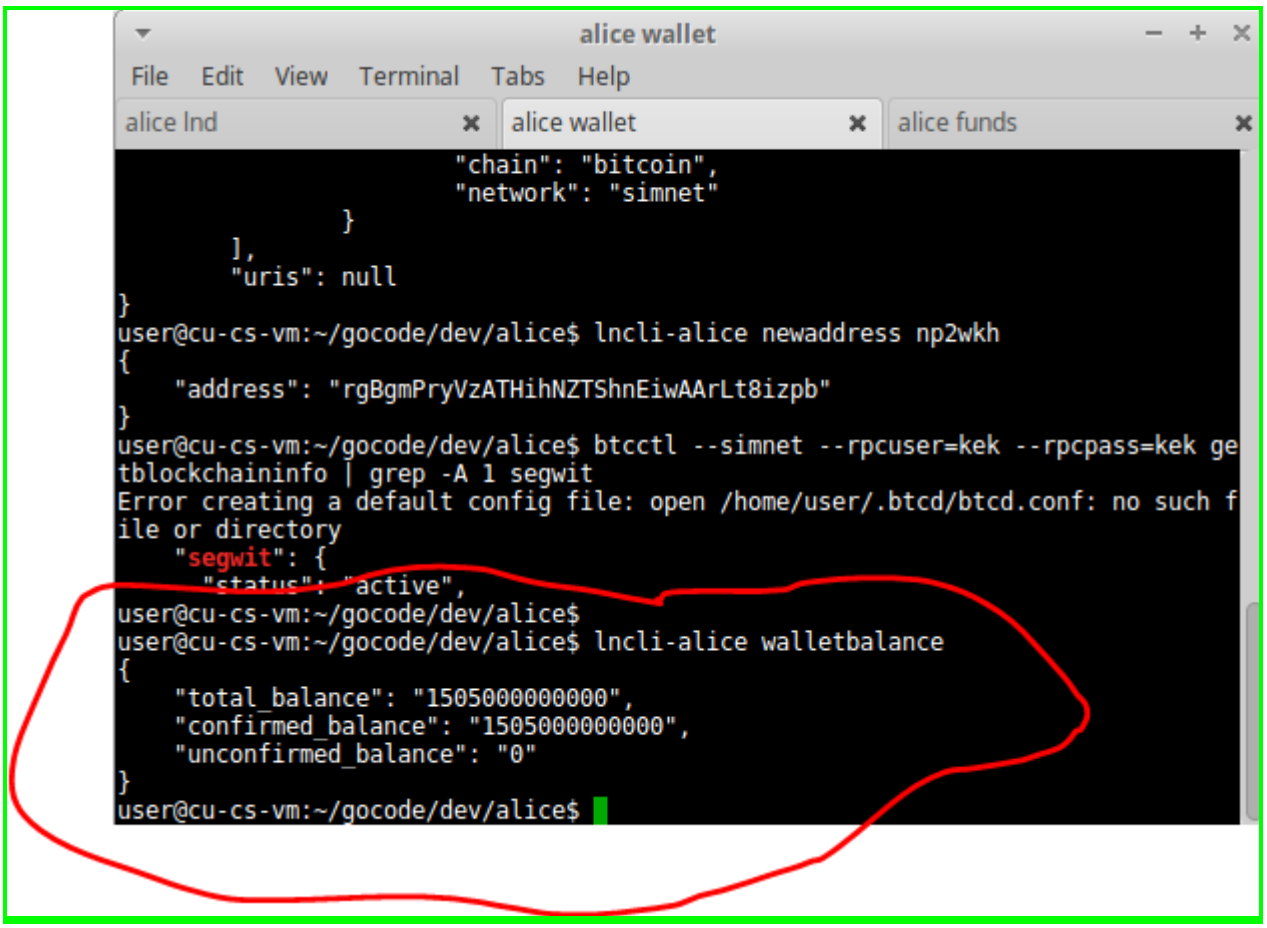
check segwit from alice’s wallet

```
btcctl --simnet --rpcuser=kek --rpcpass=kek getblockchaininfo | grep -A 1 segwit
```



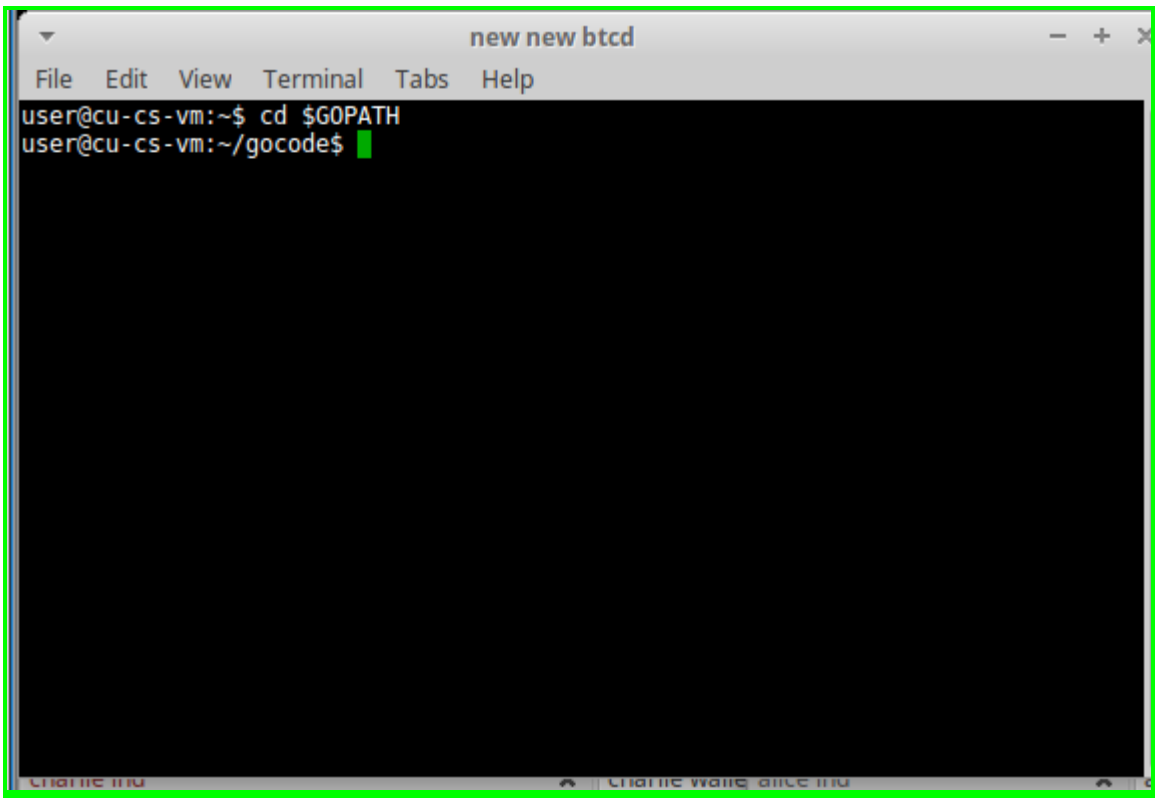
check alice's wallet balance from alice wallet

lncli-alice walletbalance



give charlie some money

close the new btcd terminal completely and open a new one, the new new btcd

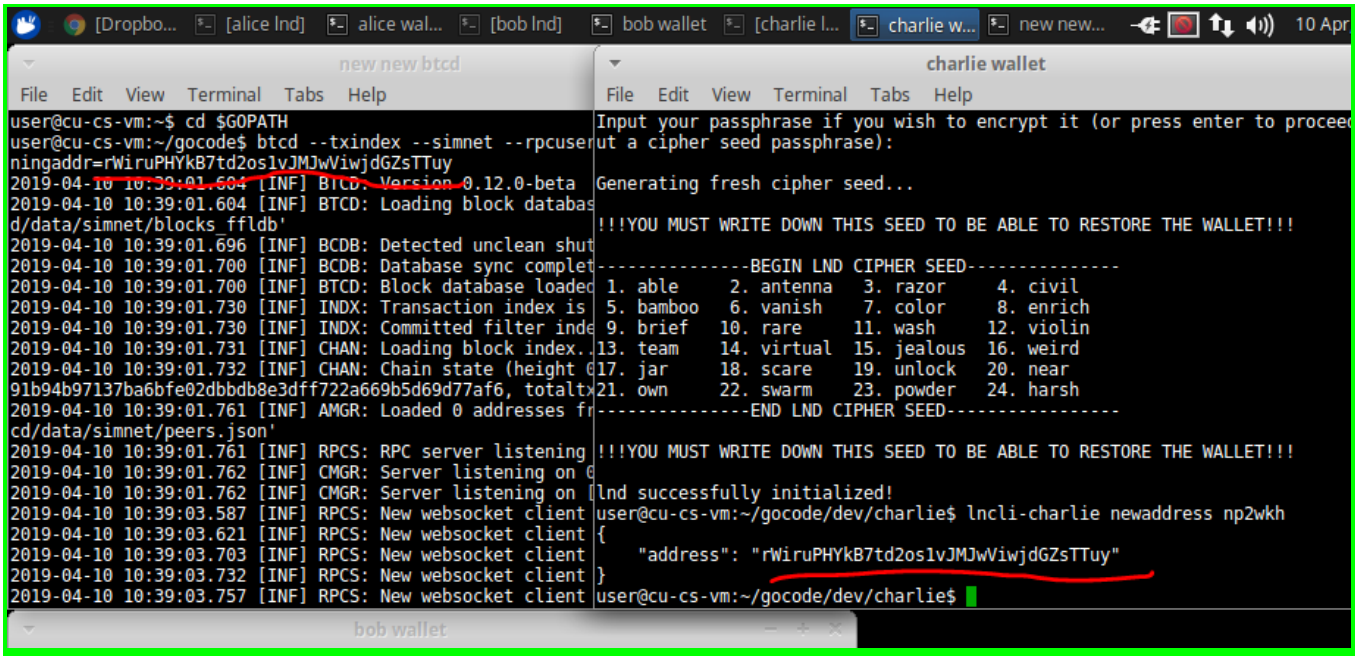


from new btcd terminal:

THE ADDRESSES WILL CHANGE EACH TIME YOU RESTART THE NETWORK!!

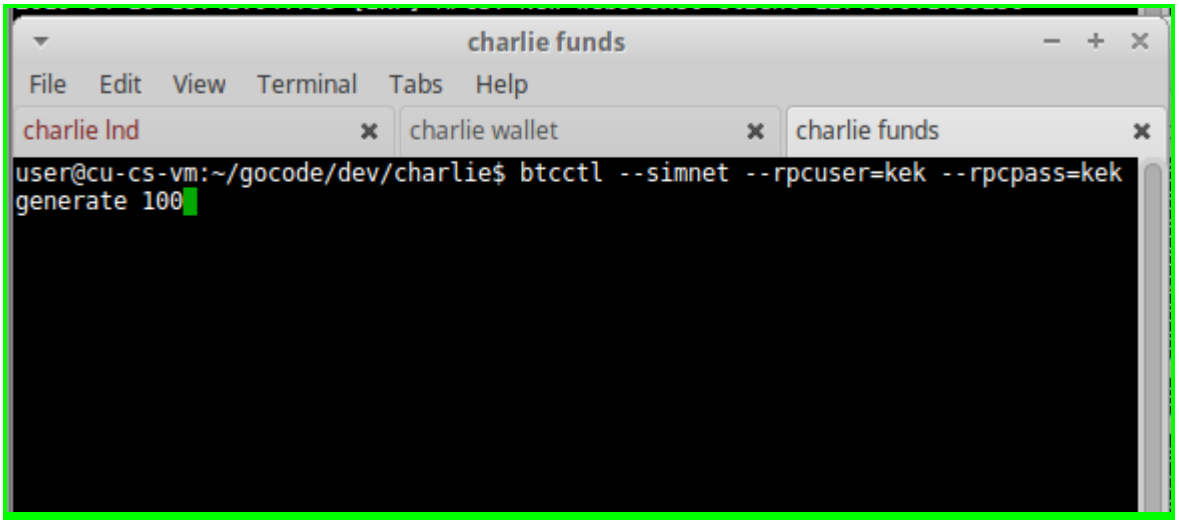
```
cd $GOPATH
```

btcd --txindex --simnet --rpcuser=kek --rpcpass=kek --miningaddr=put charlie's address here like you did alice from step above



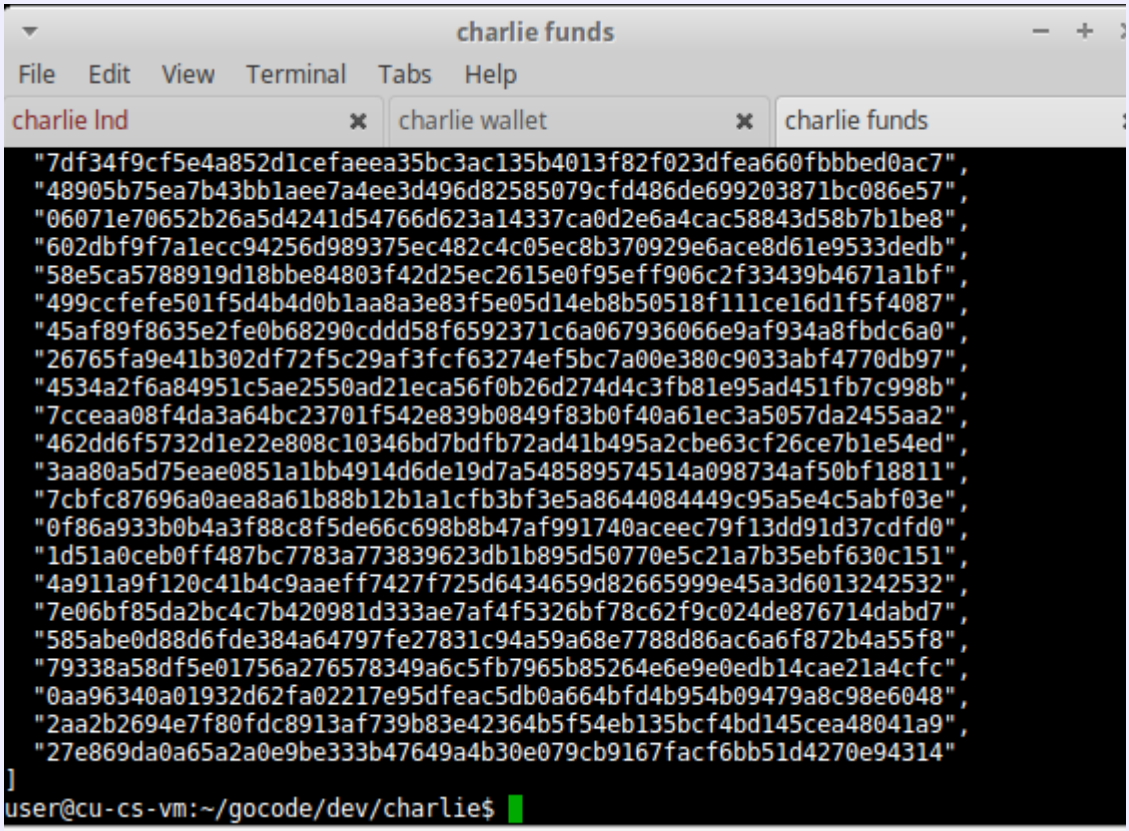
mine blocks for charlie

create new terminal: charlie funds terminal



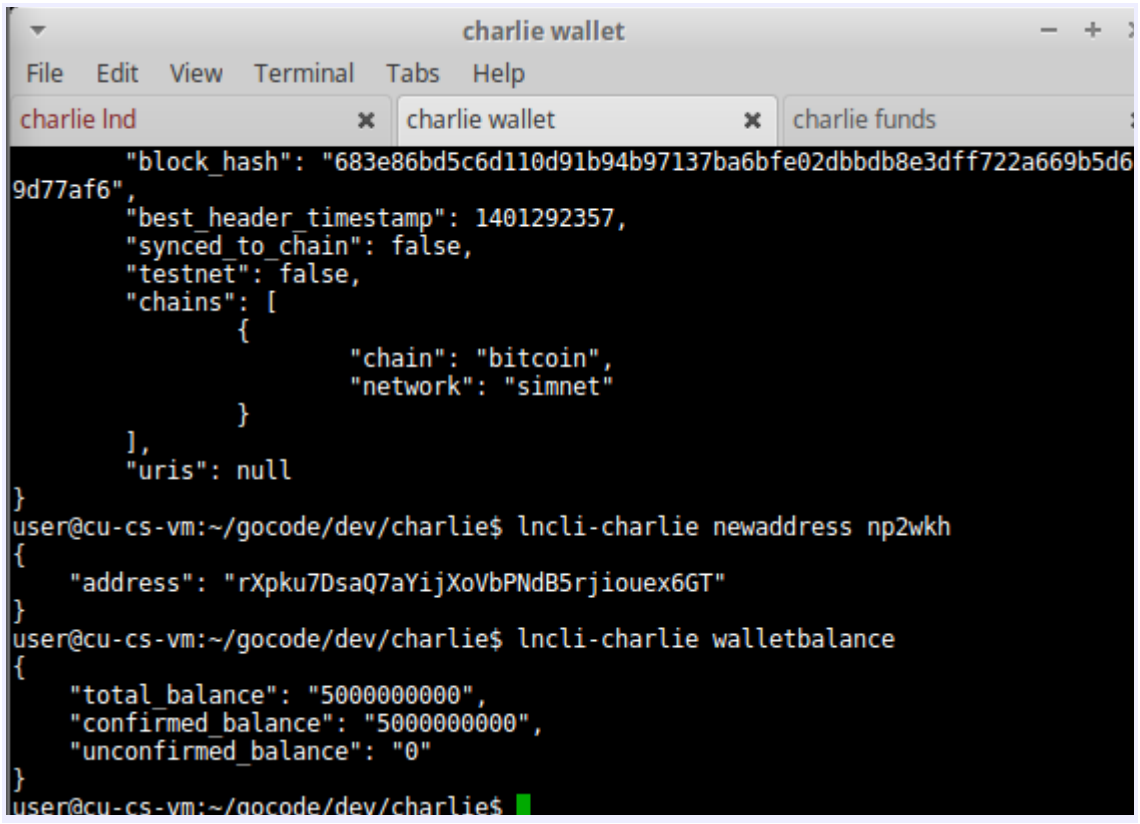
cd ~/gocode/dev/charlie

btcctl --simnet --rpcuser=kek --rpcpass=kek generate 100



check his balance from charlie wallet

lncli-charlie walletbalance



CONNECTING THE PEER TO PEER NETWORK

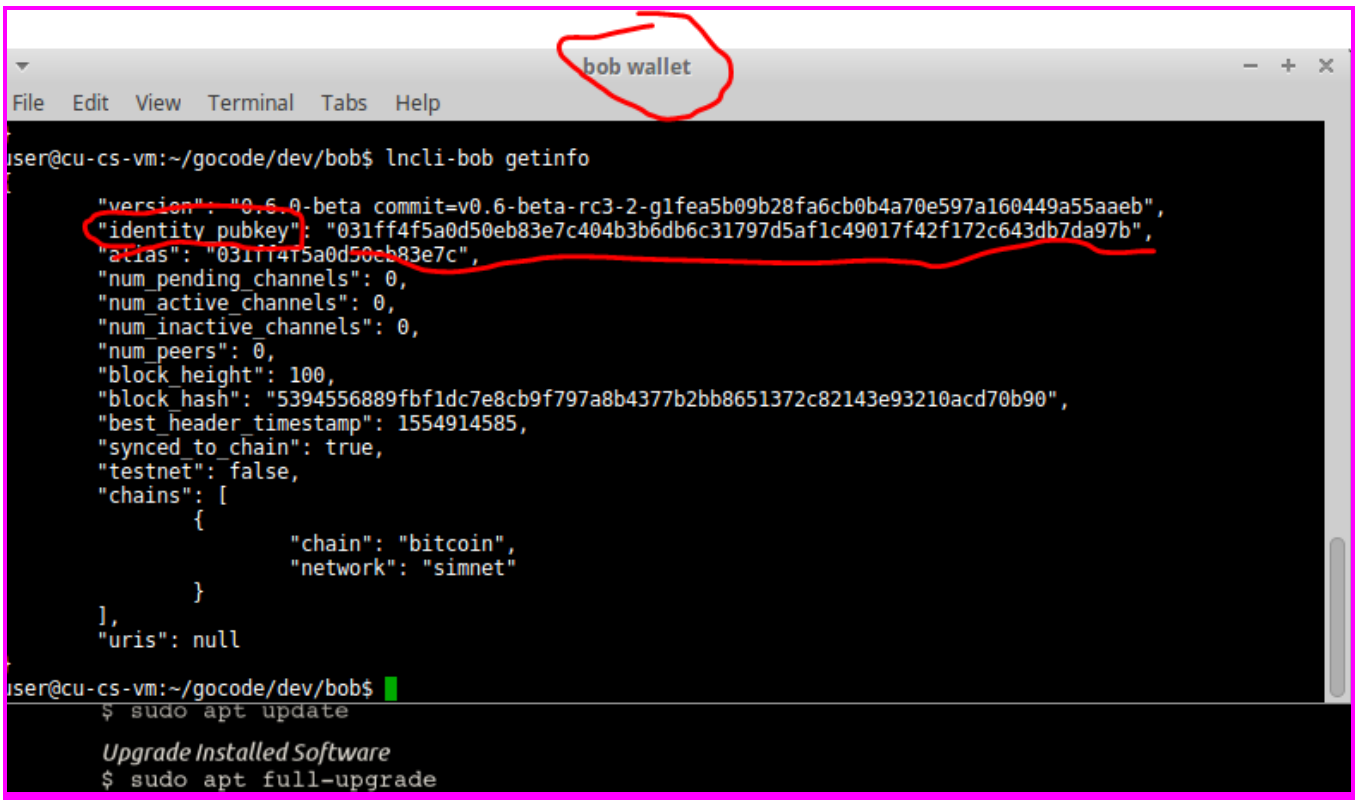
Alice and Charlie now have money
Alice will connect to Charlie by going through Bob as a channel

Alice --------> Bob ----------> Charlie

get Bob's identity pubkey
from bob wallet

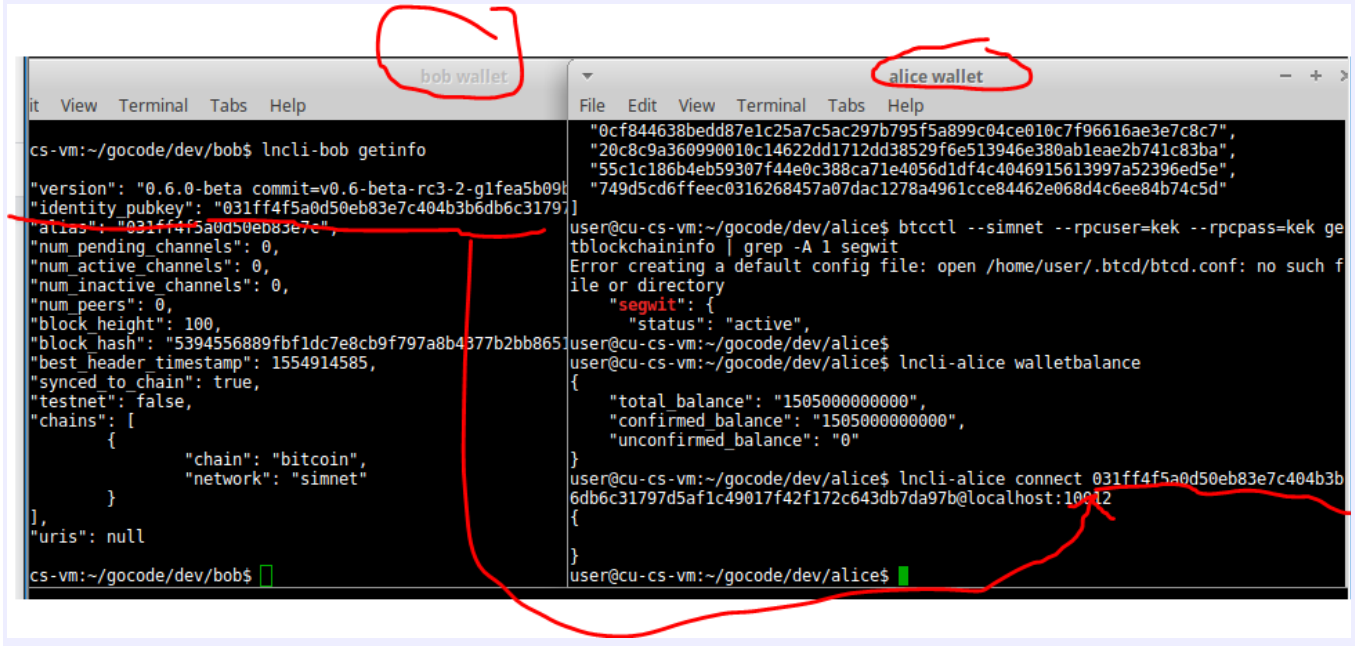
cd ~/gocode/dev/bob

lncli-bob getinfo



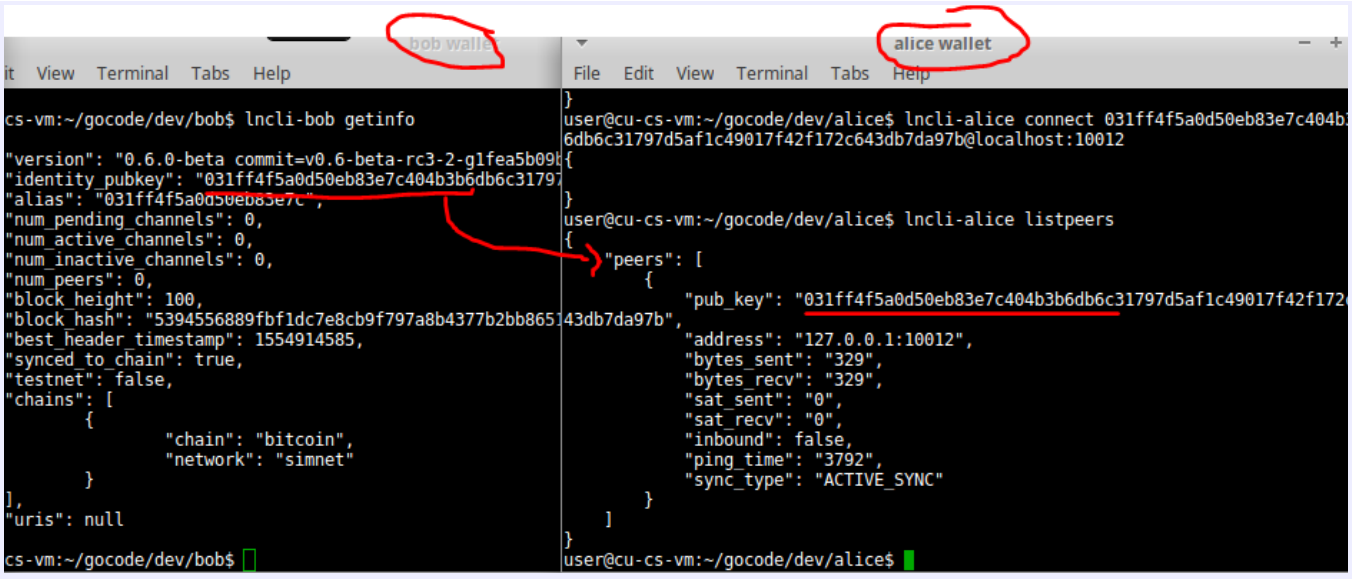
from alice wallet connect Alice --------> Bob
(the pubkey will change each time you restart the network)
from alice wallet

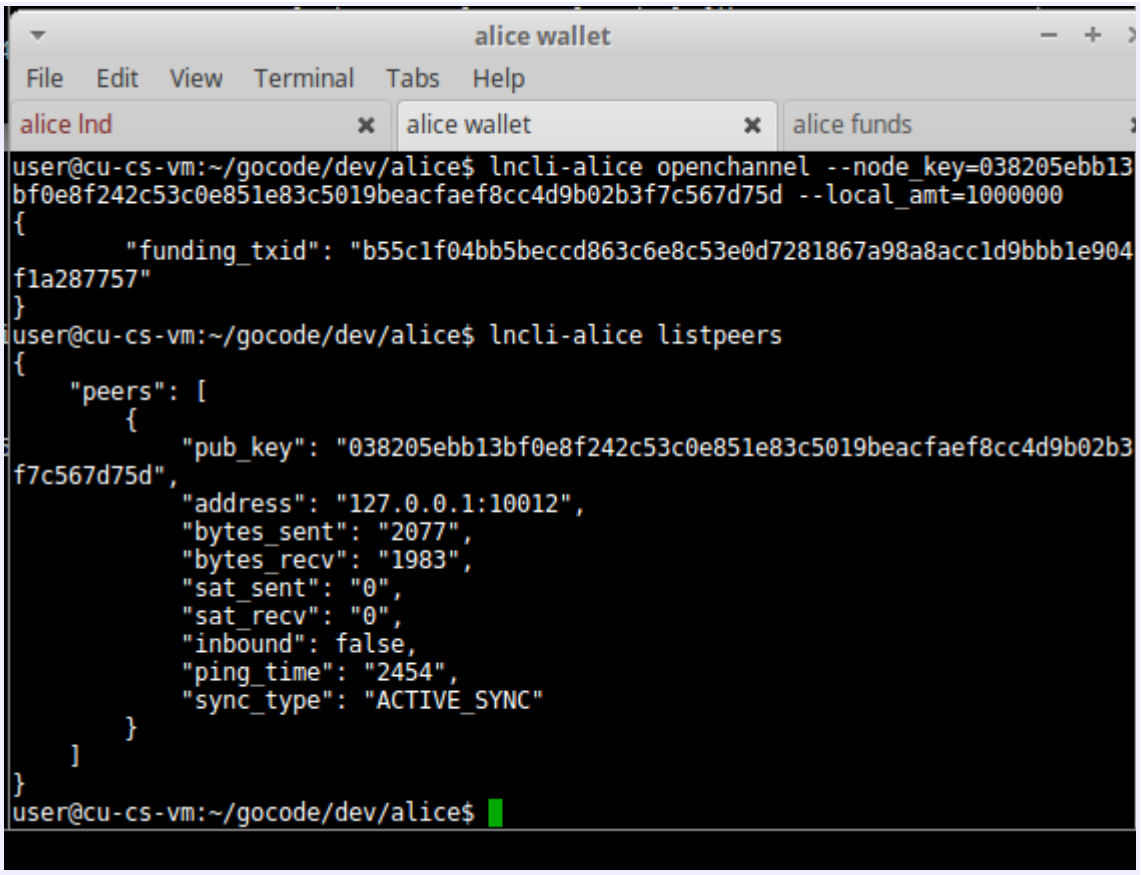
lncli-alice connect
038205ebb13bf0e8f242c53c0e851e83c5019beacfaef8cc4d9b02b3f7c567d75d@localhost:10012



check that Bob and Alice are connected as peers
from alice wallet

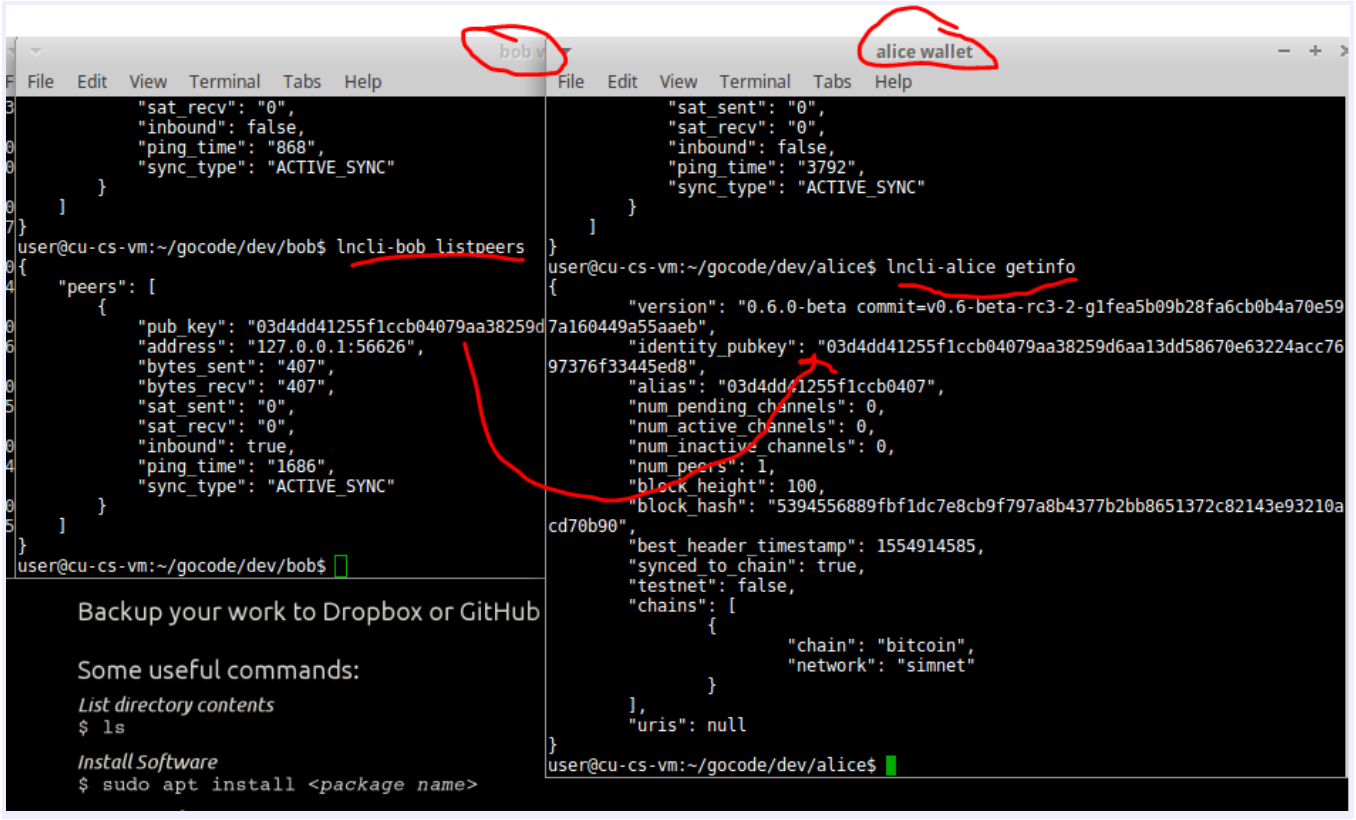
lncli-alice listpeers





from bob wallet

lncli-bob listpeers



in the same way, connect Bob ----------> Charlie

from charlie's wallet get his info

lncli-charlie getinfo

from bob wallet

lncli-bob connect 0376d11f24a3f581ad050fee5e51138edfdb8fb2bcf82fe4f97f994d75a8d87bd6@localhost:10013

check that they are connected

from charlie wallet

lncli-charlie listpeers

from bobwallet

lncli-bob listpeers

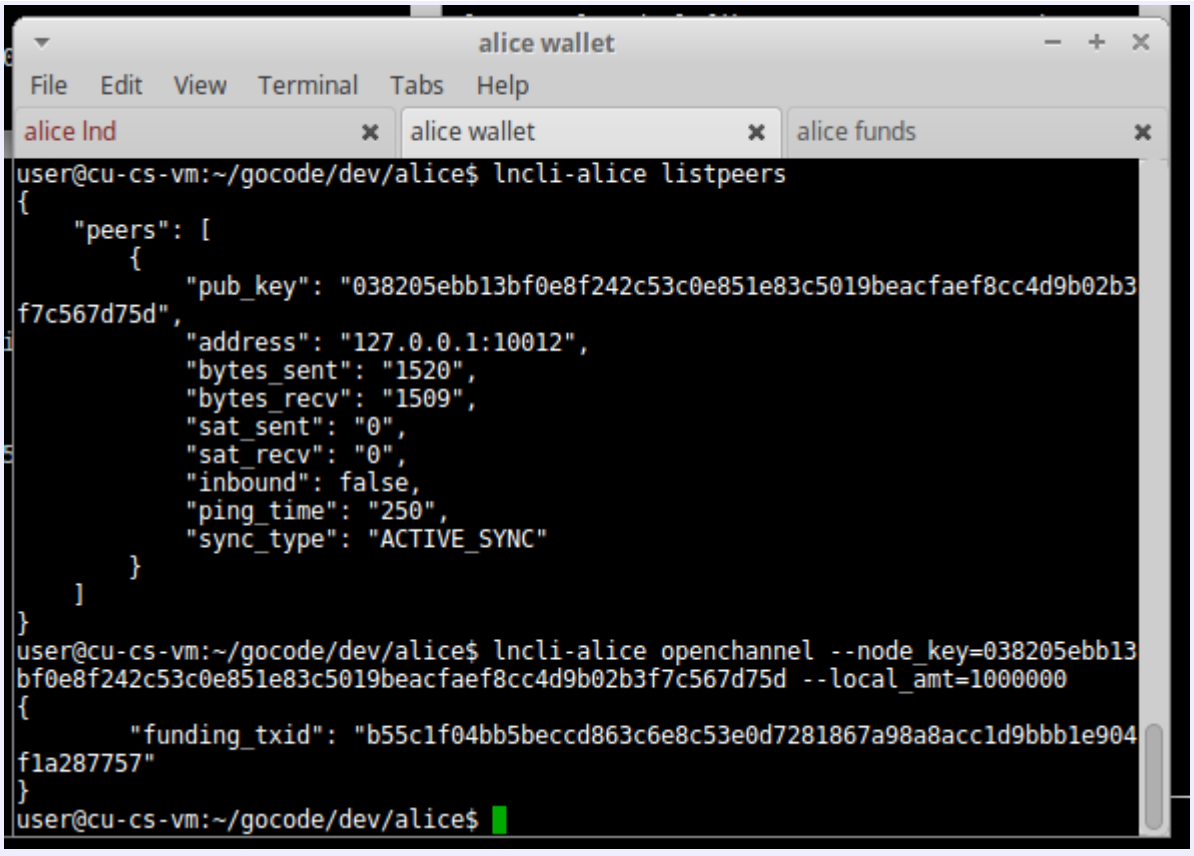
SENDING PAYMENTS

open a channel from Alice to Bob (alice is funding it)

from alice wallet

lncli-alice openchannel
--node_key=038205ebb13bf0e8f242c53c0e851e83c5019beacfaef8cc4d9b02b3f7c567d75d
--local_amt=1000000

^^^use bobs pubkey here



six blocks have to be mined for it to be valid

from anyone’s funding channel

btcctl --simnet --rpcuser=kek --rpcpass=kek generate 6

make sure they are connected in a channel

from alice wallet

lncli-alice listchannels

:(

to set specific requests such as how many blocks before mining

OPTIONS:

--node_key value the identity public key of the target node/peer serialized in compressed format

--connect value (optional) the host:port of the target node

--local_amt value the number of satoshis the wallet should commit to the channel (default: 0)

--push_amt value the number of satoshis to give the remote side as part of the initial commitment state, this is equivalent to first opening a channel and sending the remote party funds, but done all in one step (default: 0)

--block block and wait until the channel is fully open

--conf_target value (optional) the number of blocks that the transaction *should* confirm in, will be used for fee estimation (default: 0)

--sat_per_byte value (optional) a manual fee expressed in sat/byte that should be used when crafting the transaction (default: 0)

--private make the channel private, such that it won't be announced to the greater network, and nodes other than the two channel endpoints must be explicitly told about it to be able to route through it

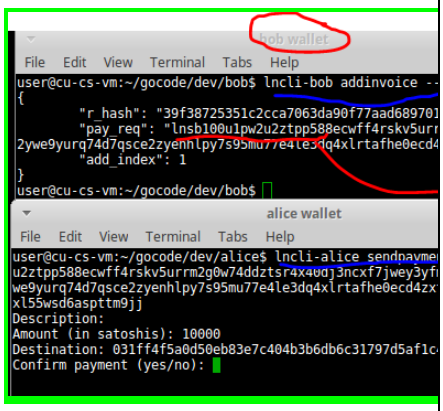
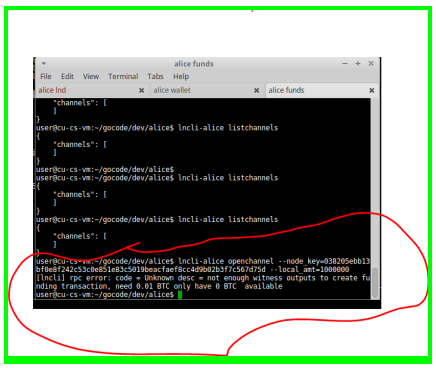
--min_htlc_msat value (optional) the minimum value we will require for incoming HTLCs on the channel (default: 0)

--remote_csv_delay value (optional) the number of blocks we will require our channel counterparty to wait before accessing its funds in case of unilateral close. If this is not set, we will scale the value according to the channel size (default: 0)

--min_confs value (optional) the minimum number of confirmations each one of your outputs used for the funding transaction must satisfy (default: 1)

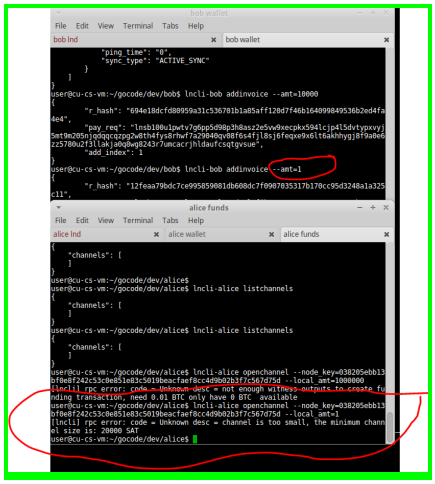
bob sends an invoice

single hop 2-party Unidirectional multiSig Payment Channel explanation of outcomes	multi hop 2-Party Bidirectional multiSig Payment Channel explanation of outcomes	

Alice --------> Bob	Bob <--------> Charlie	
from bob wallet lncli-bob addinvoice --amt=10000 from alice wallet lncli-alice sendpayment --pay_req=038205ebb13bf0e8f242c53c0e851e83c5019beacfaef8cc4d9b02b3f7c567d75d  if alice tries to open a channel without having enough bitcoin:  		

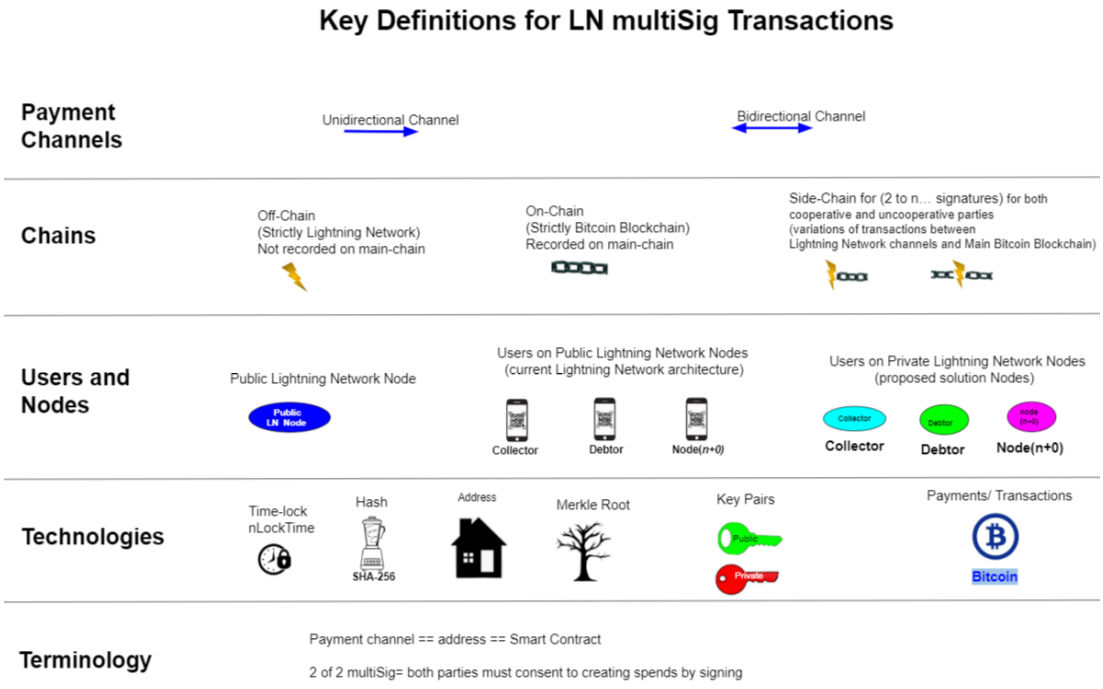
not enough witness outputs to create funding transaction, need

if they try to open a channel that's got too little transaction amount



channel is too small,

EXPLANATION OF OUTCOMES



2-party Unidirectional multiSig Payment Channel Explanation of Outcomes *link back to SENDING PAYMENTS*

LN Unidirectional multiSig

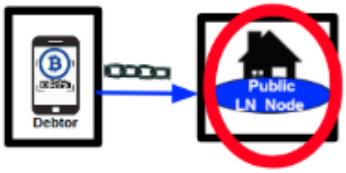
For debtor to pay collector:

Step 1.
Create a refund transaction.

One or more of the parties create a 2 of 2 multiSig address together on the main-chain where both put in funds.
Collector puts in the amount of Bitcoin that will be used to refund, i.e. make change for, the Debtor (can be 0 Bitcoin). Collector signs 1 of 2 multiSig, and sets a Time-lock.
Collector's deposit is the safety measure for Debtor. In case Collector is uncooperative, Debtor can sign the 2 of 2 multiSig at the end of the timelock and get all his bitcoin back by keeping Collector's deposit.
The debtor puts sufficient funds into the address.

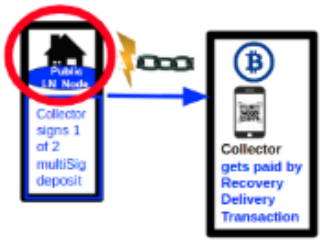
Step 2.
Debtor makes payment(s).

Off-chain, Debtor can send Collector all the payments they want to the address they created before the timelock expires.
The only valid balance is the most recent one
There are no main-chain fees.
The payments are subtracted from the Debtor's deposit balance in the smart contract.

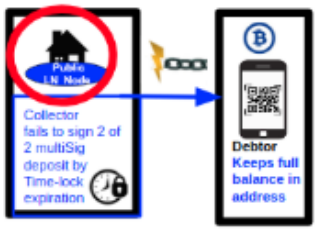


Step 3.
Channel closes by either signed 2 of 2 multiSig or expired Time-lock.

FULL COOPERATION BY ALL
When Collector signs the 2 of 2 multiSig for *any* of Debtor's payments, the channel closes and the transaction is committed to the main-chain as a UTXO from the multiSig address and the channel closes.



UNCOOPERATIVE COLLECTOR
If Collector fails to sign the 2 of 2 multiSig by Time-lock, the Debtor keeps the entire Bitcoin amount in their payment channel address and the Collector forfeits their Collector's deposit and the channel closes.



UNCOOPERATIVE DEBTOR
If Debtor fails to broadcast payment updates by the Time-Lock expiration, Collector keeps the entire Bitcoin amount in their payment channel address and the Collector forfeits their refund and the channel closes.

LN Unidirectional multiSig

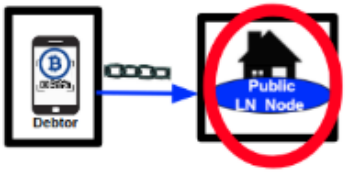
For debtor to pay collector:

Step 1.
Create a refund transaction.

One or more of the parties create a 2 of 2 multiSig address together on the main-chain where both put in funds.
Collector puts in the amount of Bitcoin that will be used to refund, i.e. make change for, the Debtor (can be 0 Bitcoin). Collector signs 1 of 2 multiSig, and sets a Time-lock.
Collector's deposit is the safety measure for Debtor. In case Collector is uncooperative, Debtor can sign the 2 of 2 multiSig at the end of the timelock and get all his bitcoin back by keeping Collector's deposit.
The debtor puts sufficient funds into the address.

Step 2.
Debtor makes payment(s).

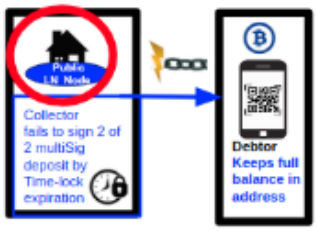
Off-chain, Debtor can send Collector all the payments they want to the address they created before the timelock expires.
The only valid balance is the most recent one
There are no main-chain fees.
The payments are subtracted from the Debtor's deposit balance in the smart contract.



Step 3.
Channel closes by either signed 2 of 2 multiSig or expired Time-lock.

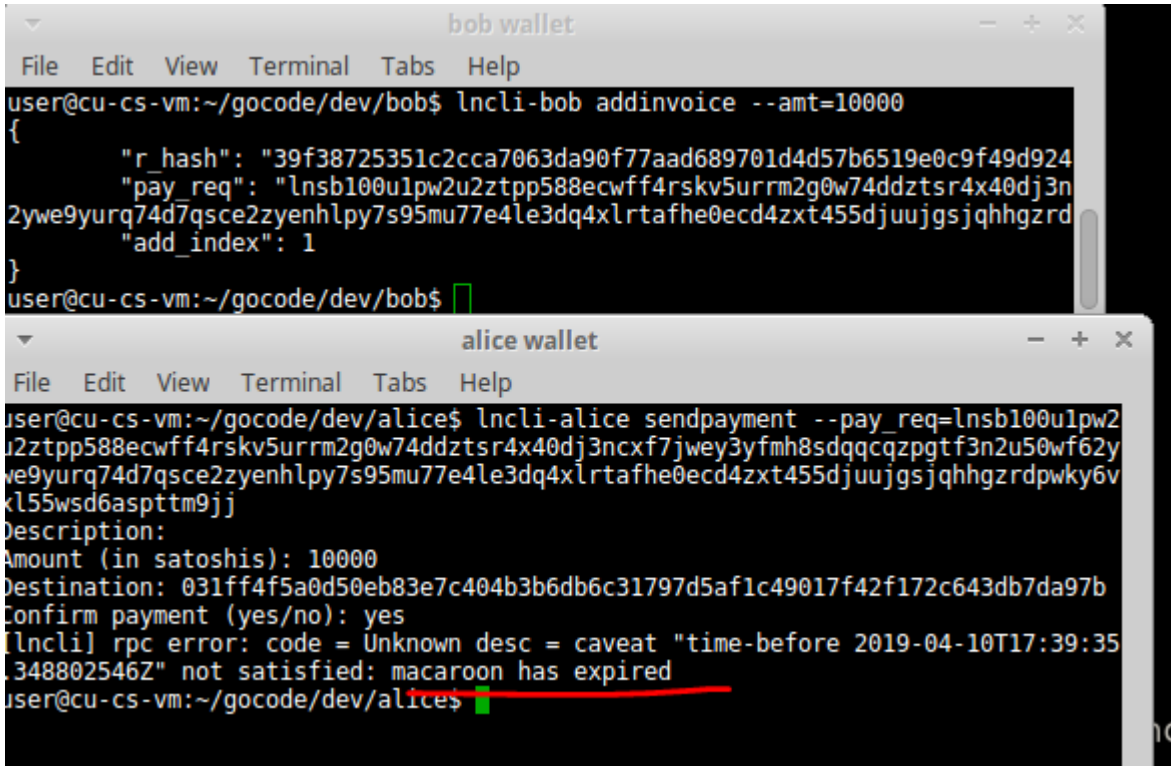
FULL COOPERATION BY ALL
When Collector signs the 2 of 2 multiSig for *any* of Debtor's payments, the channel closes and the transaction is committed to the main-chain as a UTXO from the multiSig address and the channel closes.

UNCOOPERATIVE COLLECTOR
If Collector fails to sign the 2 of 2 multiSig by Time-lock, the Debtor keeps the entire Bitcoin amount in their payment channel address and the Collector forfeits their Collector's deposit and the channel closes.



UNCOOPERATIVE DEBTOR
If Debtor fails to broadcast payment updates by the Time-Lock expiration, Collector keeps the entire Bitcoin amount in their payment channel address and the Collector forfeits their refund and the channel closes.

example of uncooperative debtor, expired timelock:



2-Party Bidirectional multiSig Payment Channel explanation of outcomes

link back to SENDING PAYMENTS

LN Bidirectional multiSig between 2 parties

LN Multidirectional multiSig as Micropayment Channels Using Cryptographic Signatures

(Section 3.3.5 of LN Whitepaper)

For debtor to pay collector and collector to update change as each spending transaction occurs (similar to an open tab on an escrow):

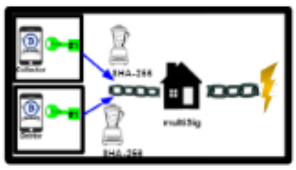
Step 1.

Create an address that requires both party's signatures

Script:
SIGHASH_NOINPUT

- Both parties agree to create a Commitment Transaction
- Together they create 1 multiSig
- Each party puts in their Public Key of their asymmetric key pair
- The funds are like an escrow to fund the channel and can't be spent
- The collector must match the debtor's UTXO amount (if the debtor skips out, the main-chain is still correctly balanced by the collector's escrow)

Commitment Transaction #1 = multiSig(Collector's Public Key #1, Debtor's Public Key #1)



Step 2.

A new Commitment Transaction is created on the side-chain each time the debtor's tab changes

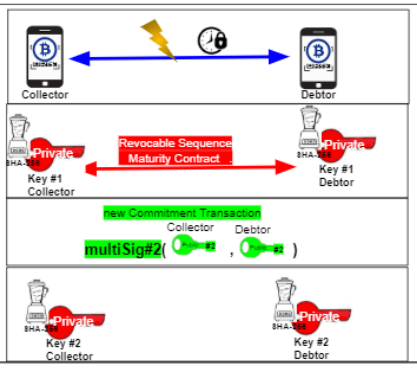
Both parties must sign the same Revocable Sequence Maturity Contract (RSMC) with their private keys

Script:
OP RELATIVECHECKLOCKTIMEVERIFY

- The tab changes so the UTXO changes
- A new UTXO requires a new Commitment Transaction
- For security reasons (to avoid replay attacks based on a merkle trees property) they have to use new key pairs
- Using their private keys both parties sign the same RSMC
- New key pairs are generated for each party by adding +1 to each party's public key
- The current and verifiable key pairs are the new keys: Public Keys #2

Commitment Transaction#2 = multiSig(Collector's Public Key #2, Debtor's Public Key #2)

- new private keys are generated as a result of the new public keys (a features of asymmetric encryption),
- Both this new tab balance and the previous tab balance (the UTXO's) are broadcast on the sidechain channel only (the Lightning network)



Off-chain, the tab can change infinite number of times before the timelock expires.

There are no main-chain fees for any of these changes while on the sidechain.

The updates happen only on the sidechain.

IF BOTH PARTIES ARE HONEST

- Repeat Step 2 each time there is an update to the tab, i.e. a change in the UTXO amounts
- When both parties want to close the tab (close the sidechain channel) go to Step 3

IF THERE IS A PROBLEM

- The channel is closed and published on the mainchain
- *The penalty*: the bad party loses ALL their funds to the victim party
- The victim has to wait some set number of main-chain published blocks to get those funds

PUBLISHING THE WRONG BALANCE- BAD ACTOR

- If anything older than the current and immediate previous UTXO is broadcast, it is considered cheating

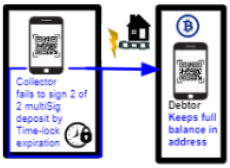
Counterparty gets ALL the bitcoin

NOT PUBLISHING AN UPDATE- UNCOOPERATIVE DEBTOR

- If a party doesn't broadcast the latest Commitment Transaction by the timelock expiration, it is considered cheating

COLLECTOR FAILS TO SIGN- UNCOOPERATIVE COLLECTOR

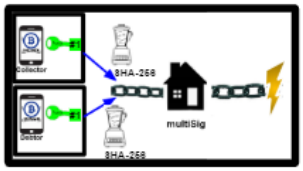
- If the collector doesn't signthe 2 of 2 MultiSig to close out by the timelock, they are considered in default



Step 3.
Close the channel.

Script:

REDEEMING THE FUNDS



COOPERATIVE COUNTERPARTIES

- Either party can redeem the funds

1) A Commitment Transaction is published by either party
2) The counterparty can spend their funds off the mainchain immediately, thus acknowledging this was the correct final ledger
3) The original party must wait out the timelock, and can then broadcast a Revocable Delivery Transaction (RDT)
4) The original party then waits another round of the same timelock, and the channel is then closed and the funds are redeemable by the original party

***if the original party attempts to broadcast early, the 2nd round timelock is restarted from that point*

IF THEY CHANGE THEIR MINDS ABOUT THE FINAL BALANCES

- A Breach Remedy Transaction (BRT) is signed mutually by both parties
- The BRT overrides the BRT
- The same penalty as before applies if there is an attempt to cheat

PUBLISHING THE WRONG BALANCE- BAD ACTOR

- If anything older than the current and immediate previous UTXO is broadcast, it is considered cheating

Counterparty gets ALL the bitcoin

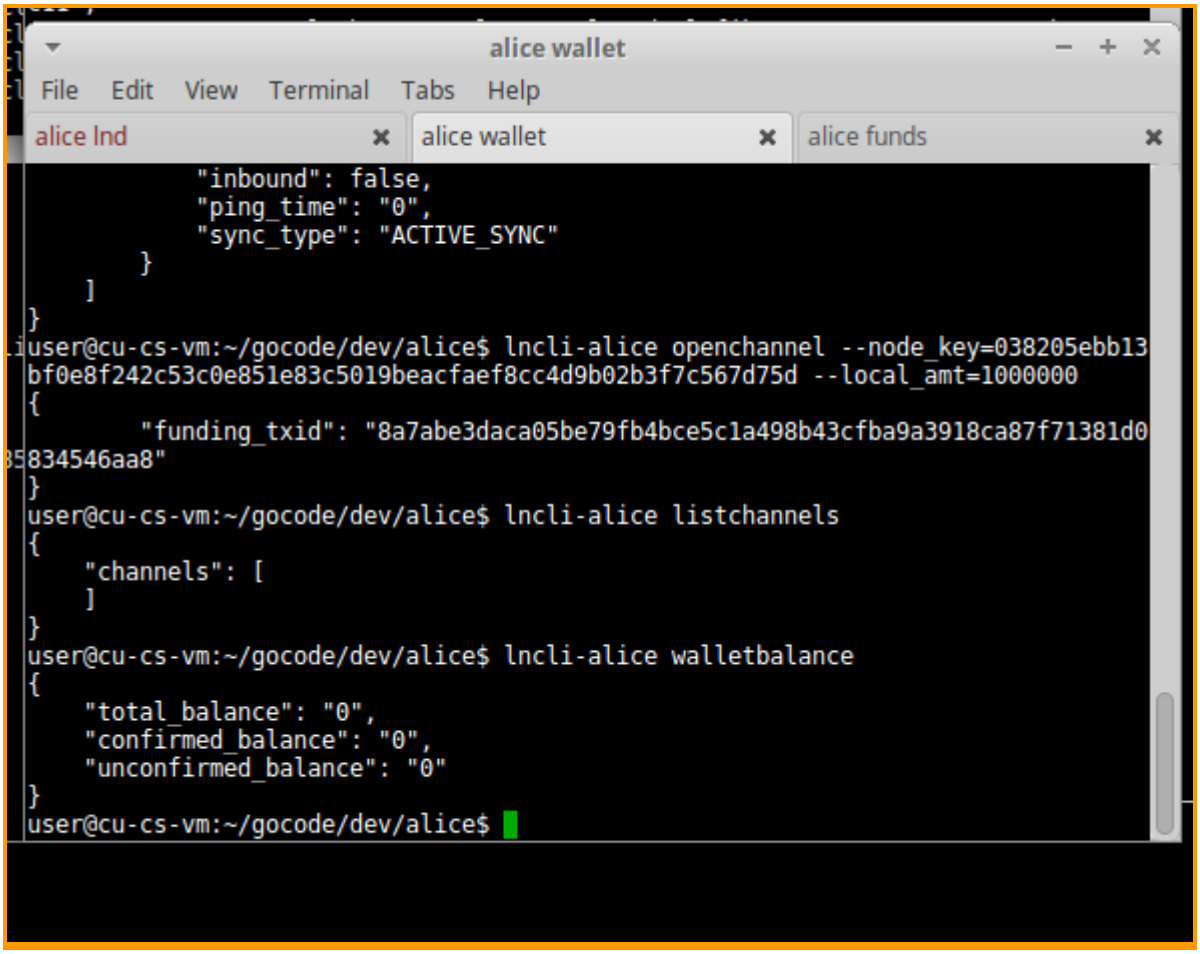
- To receive payment, the victim must broadcast the correct BRT within the timelock cycle
- If the victim fails to publish the correct BRT within the timelock cycle, the bad actor is able to take all the funds because the channel is open

n-Party Bidirectional multiSig Payment Channel explanation of outcomes

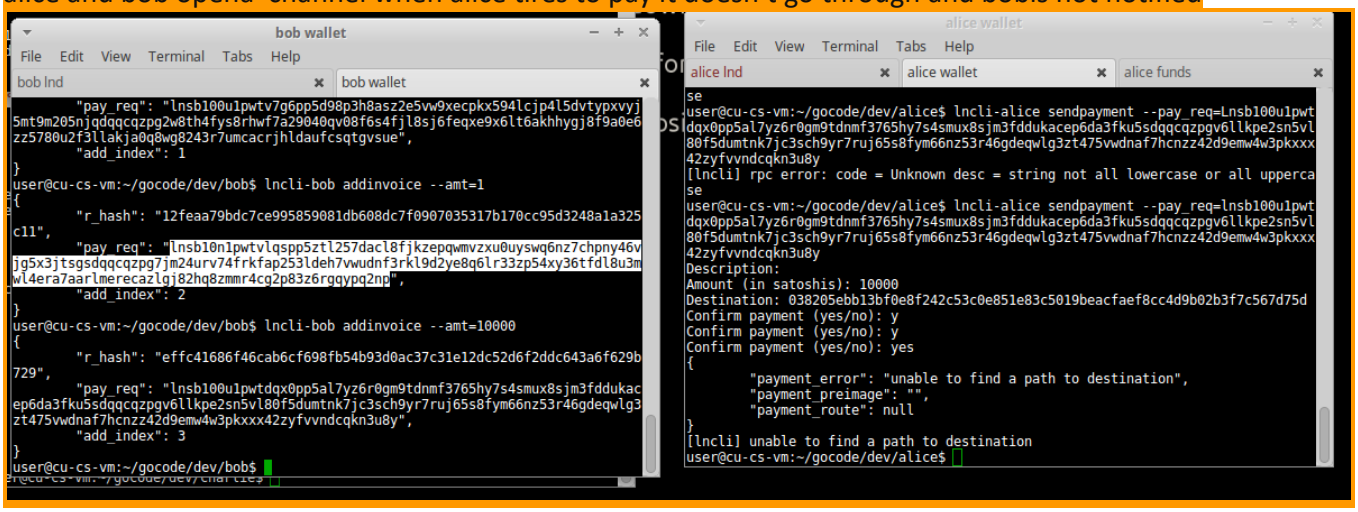
link back to SENDING PAYMENTS

ERRORS WITH THE CODE

ALICE had funds before, then she signed back in with her seed phrase and had lost all her funds



alice and bob opena channel when alice tires to pay it doesn't go through and bobis not notified



```
cd $GOPATH

 btcd --txindex --simnet --rpcuser=kek --rpcpass=kek
```

I think To include watchtower flags start up btcd with reindex flag but idk how to do that ,or use this command or something similar

getmempoolentry "txid"

:

https://medium.com/chlunetwork/bitcoin-lightning-network-tutorial-how-to-integrate-lightning-to-a-crypto-project-f48743c8644a

```
cd $GOPATH

 btcd --txindex --simnet --reindex --rpcuser=kek --rpcpass=kek
```

From <https://medium.com/chlunetwork/bitcoin-lightning-network-tutorial-how-to-integrate-lightning-to-a-crypto-project-f48743c8644a>

cd $GOPATH/dev/alice rm -r data log lnd --rpclisten=localhost:10001 --listen=localhost:10011 --restlisten=localhost:8001 --datadir=data --logdir=log --debuglevel=info --bitcoin.simnet --bitcoin.active --bitcoin.node=btcd --btcd.rpcuser=kek --btcd.rpcpass=kek NEW WINDOW lncli --rpcserver=localhost:10001 --macaroonpath=data/admin.mac aroon create lncli --rpcserver=localhost:10001 --macaroonpath=~/gocode/dev/al ice/data/chain/bitcoin/simnet/ad min.macaroon create lncli-alice newaddress np2wkh rhV6X4fnR7HpTpyEeZunhX2t1HRFg Yvt32	cd $GOPATH/dev/bob rm -r data log lnd --rpclisten=localhost:10002 --listen=localhost:10012 --restlisten=localhost:8002 --datadir=data --logdir=log --debuglevel=info --bitcoin.simnet --bitcoin.active --bitcoin.node=btcd --btcd.rpcuser=kek --btcd.rpcpass=kek NEW WINDOW lncli --rpcserver=localhost:10002 --macaroonpath=data/admin.mac aroon create # This is the alias in the ~/.bashrc file lncli --rpcserver=localhost:10002 --macaroonpath=~/gocode/dev/b ob/data/chain/bitcoin/simnet/ad min.macaroon getinfo #then to control bob we type Lncli-bob $command lncli-bob newaddress np2wkh rpcte1cKpvyoGZPU35bbq8ZjPTWsB Cmufv	cd $GOPATH/dev/charlie rm -r data log nd --rpclisten=localhost:10003 --listen=localhost:10013 --restlisten=localhost:8003 --datadir=data --logdir=log --debuglevel=info --bitcoin.simnet --bitcoin.active --bitcoin.node=btcd --btcd.rpcuser=kek --btcd.rpcpass=kek NEW WINDOW lncli --rpcserver=localhost:10003 --macaroonpath=data/admin.maca roon create Lncli-charlie getinfo lncli --rpcserver=localhost:10003 --macaroonpath=~/gocode/dev/ch arlie/data/chain/bitcoin/simnet/ad min.macaroon getinfo lncli-charlie newaddress np2wkh rYGLpFTiJ4vTKYW1PWUsUExWZ9yM CJz8Qh

TO GIVE them MONEY RESTART BTCD

MONEY for Alice		
Btcd- close it down completely **Open a new window** **cd $GOPATH** btcd --simnet --txindex --rpcuser=kek --rpcpass=kek --miningaddr=rhV6X4fnR7HpTpyEeZunhX2t1HRFgYvt32		
Open a new tab in alice- alice miner mine 400 blocks, in her miner window like so: btcctl --simnet --rpcuser=kek --rpcpass=kek generate 400 check segwit from alice's wallet btcctl --simnet --rpcuser=kek --rpcpass=kek getblockchaininfo | grep -A 1 segwit check alice's wallet balance from alice wallet lncli-alice walletbalance		

MONEY for Charlie
Btcd- close it down completely **Open a new window** **cd $GOPATH** btcd --simnet --txindex --rpcuser=kek --rpcpass=kek --miningaddr=rYGLpFTiJ4vTKYW1PWUsUExWZ9yMCJz8Qh

Open a new tab in charlie- charlie miner
mine 400 blocks, restart her window like so:

btcctl --simnet --rpcuser=kek --rpcpass=kek generate 400

check segwit from charlie's wallet

btcctl --simnet --rpcuser=kek --rpcpass=kek getblockchaininfo | grep -A 1 segwit

check charlie's wallet balance from alice wallet

lncli-charlie walletbalance

MONEY FOR BOB

btcd --txindex --simnet --rpcuser=kek --rpcpass=kek --miningaddr=raL2cCCUBEMVx71uwmsXojcCtgM59XezPG

CONNECTING THE PEER TO PEER NETWORK

Open a channel from alice to bob

User	In wallet you are connecting to	In wallet you are connecting from			
alice wallet	lncli-bob getinfo	lncli-alice connect8e62fca34e52963d7cd663fb4fbd192f1bcdd7091654a3a0182c7bca30ae7bf2@localhost:10012			
bob wallet					

Check connection

User	In wallet you are connecting to	In wallet you are connecting from		
bob		lncli-bob listpeers		
alice	lncli-alice listpeers			

Open a channel from bob to charlie

User	In wallet you are connecting to	In wallet you are connecting from			
bob wallet	lncli-charlie getinfo	lncli-bob connect 0263009c90eca398e2664525baf8b43c5f31655ddafb43a3dd50842b4c136f23d6@localhost:10013			

Check connection (bob should have 2 peers)

User		In wallet you are connecting from		
charlie		lncli-charlie listpeers		

Check connection

User	In wallet you are connecting to	In wallet you are connecting from		
bob		lncli-bob listpeers		
charlie	lncli-charlie listpeers			

Check connection (bob should have 2 peers)

User		In wallet you are connecting from		
bob wallet		lncli-bob listpeers		

SENDING PAYMENTS

open a channel from Alice to Bob (alice is funding it)

Alice is paying Bob

btcd --testnet --rpcuser=kek --rpcpass=kek

From <https://dev.lightning.community/guides/installation/>

Open a channel

User	Who will be in the channel	Person who is funding	Person you are paying	Mine 6 blocks	
alice	Alice and bob	alice	bob		
Alice wallet		lncli-alice openchannel --node_key=028e62fca34e52963d7cd663fb4fbd192f1bcdd7091654a3a0182c7bca30ae7bf2 --local_amt=1000000 This is bob's pubkey			
Anyone's miner window				btcctl --simnet --rpcuser=kek --rpcpass=kek generate 6	
Alice wallet		lncli-alice listchannels			

charlie Opens a channel to bob

User	Who will be in the channel	Person who is funding	Person you are paying	Mine 6 blocks	
alice	Alice and bob	alice	bob		
Alice wallet		lncli-charlie openchannel --node_key=035fdf4a10104a695fdddbdab0f67c0c07b447af05c62d6ff84d279224353127b1 --local_amt=1000000 This is bob's pubkey			
Anyone's miner window				btcctl --simnet --rpcuser=kek --rpcpass=kek generate 6	
Alice wallet		lncli-alice listchannels			

Bob makes an invoice

Make an invoice

User					
Bob wallet	lncli-bob addinvoice --amt=10000				

Mine 6 blocks so the invoice posts

User					
Any miner terminal	btcctl --simnet --rpcuser=kek --rpcpass=kek generate 6				

Alice pays Bob on that channel

Alice sends payment

User						
Alice wallet	lncli-alice sendpayment --pay_req=lnsb100u1pwvykuxpp5payec6w2a2vrh5qe8pr5mlxrz443pr0cndgqey443zreymksg9nsdqqcqzpgnjxy8yyyfq45sknzklg5258kasrjdppeuvwyrq9eelcy3q6cmqp4cxnmfaqy085l0upse637yjahguj5ks5qcme4zcz838g8xt7jl8qprr2ala					

Alice closes the channel with Bob

lncli-alice listchannels
lncli-alice closechannel
--funding_txid=bb6ec0889ef37312bb9ca2741c34178bd94113ccd126f05ebb587d6ee3eb21c4

lncli-alice listchannels

THE MISSING PICS ARE IN THE OneNote FILE, WE CAN ADD THEM LATER IF NEED BE

1st Fraud Attempt
Alice and Bob make a few transactions and Alice tries to send an earlier indexed transaction to cheat Bob

Step 1-- Alice gets Bob's pubkey
035fdf4a10104a695fdddbdab0f67c0c07b447af05c62d6ff84d279224353127b1

Step 2-- She uses his info to fund and open a channel with Bob
lncli-alice openchannel
--node_key=035fdf4a10104a695fdddbdab0f67c0c07b447af05c62d6ff84d279224353127b1

--local_amt=1000000

Verify Alice and Bob have a channel between each other

Bob makes an invoice, note the "add_index" field , it's keeping track of which transaction has happened between them

Alice pays Bob For THIS #2 invoice on that channel
lncli-alice sendpayment
--pay_req=lnsb100u1pwvykuxpp5payec6w2a2vrh5qe8pr5mlxrz443pr0cndgqey443zreymksg9nsdqqcqzpgnjxy8yyyfq45sknzklg5258kasrjdppeuvwyrq9eelcy3q6cmqp4cxnmfaqy085l0upse637yjahguj5ks5qcme4zcz838g8xt7jl8qprr2ala
Proof it's already paid:

Alice's walletbalance at this point:

Bob's walletbalance at this point:

Bob runs ANOTHER invoice, invoice #3
lnsb100u1pwvyhg8pp5ga7e2en45asw6fg5s23ckp4zapvtqe35lftg3d49xv2r9p77melqdqqcqzpgvz9j4n45zgs0j443uca99t0dhdrrv3pmjvq5h3kd8m7w0m3vt8jze9uu7sgvr22kwv6ges9fmk5x4px7jy43zlnjuc5mjgmv8tg6p5cp7r6tnt

Alice tries to resend an old payment to Bob For invoice #3 to fool him, on that same channel
lncli-alice sendpayment
--pay_req=lnsb100u1pwvykuxpp5payec6w2a2vrh5qe8pr5mlxrz443pr0cndgqey443zreymksg9nsdqqcqzpgnjxy8yyyfq45sknzklg5258kasrjdppeuvwyrq9eelcy3q6cmqp4cxnmfaqy085l0upse637yjahguj5ks5qcme4zcz838g8xt7jl8qprr2ala

Alice wallet balance at this point:

lncli-alice listchannels
lncli-alice closechannel
--funding_txid=bb6ec0889ef37312bb9ca2741c34178bd94113ccd126f05ebb587d6ee3eb21c4
lncli-alice listchannels

She is not able to close without paying and is punished by haing to give Bob all the $ in the channel

Bob's new balance: