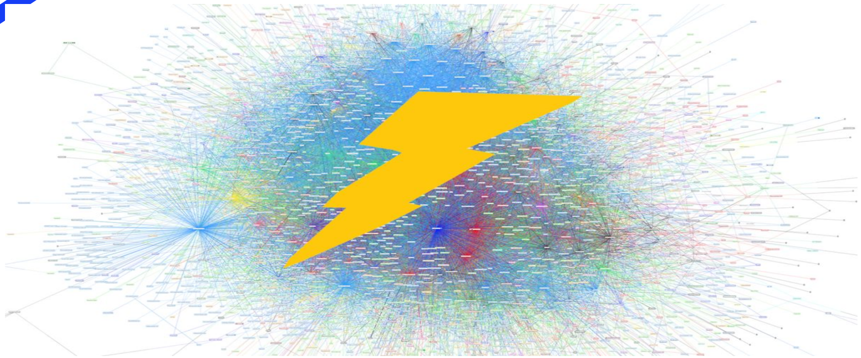
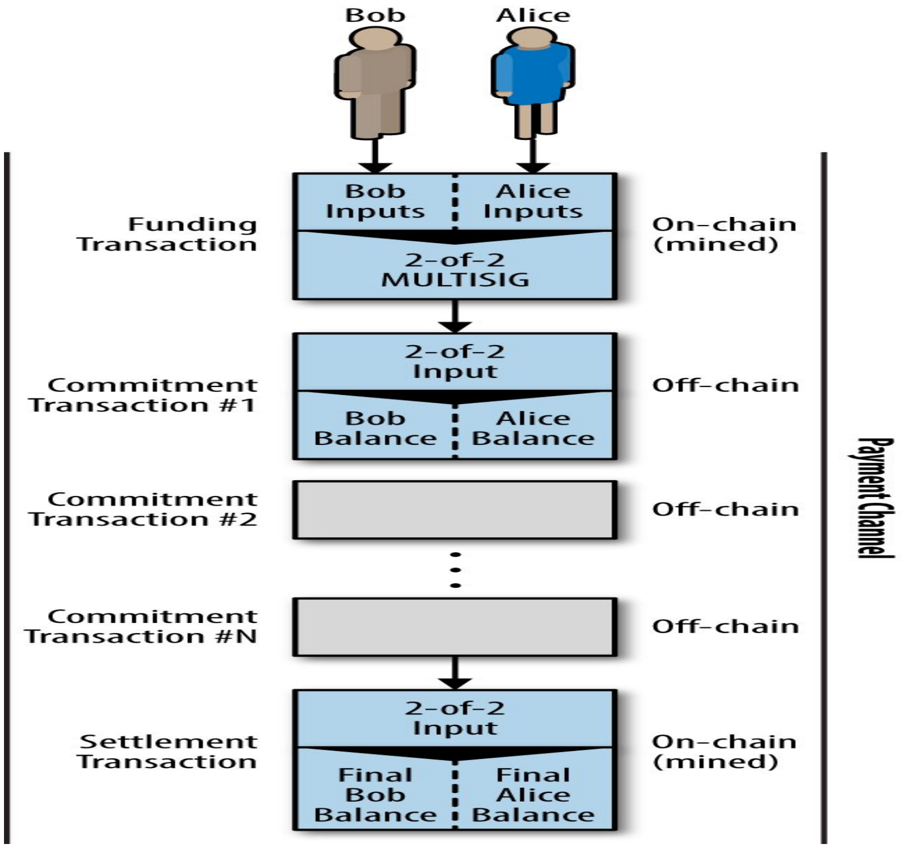


Lightning Network: Payments and Security

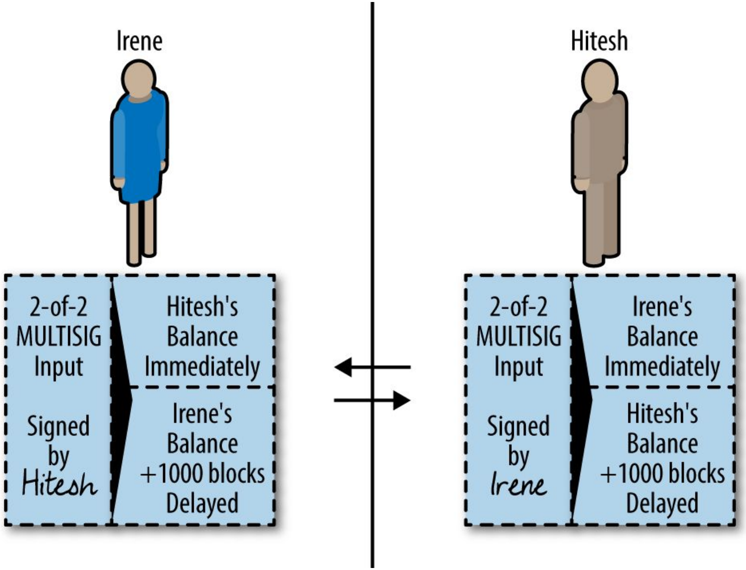
Joel Davidson, Tanner Lillich, Elsa Velazquez

Bi-Directional Payment Channel on LN

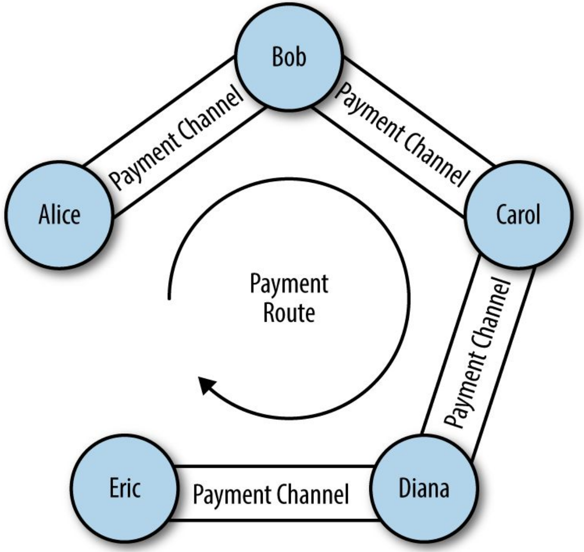


Payments are Secured with HTLC's

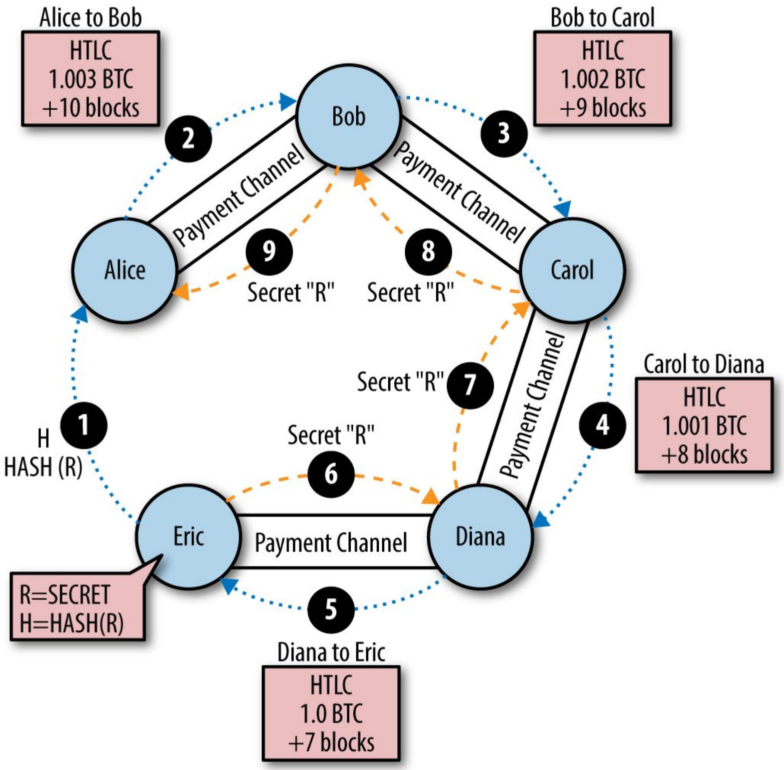
HTLC: Hashed Time Locked Contract (Special type of smart contract)


Irene
2-of-2 MULTISIG Input
Signed by Hitesh
Hitesh's Balance Immediately
Irene's Balance +1000 blocks Delayed
Hitesh
2-of-2 MULTISIG Input
Signed by Irene
Irene's Balance Immediately
Hitesh's Balance +1000 blocks Delayed

Lightning Nodes Create a Network With Their Peers

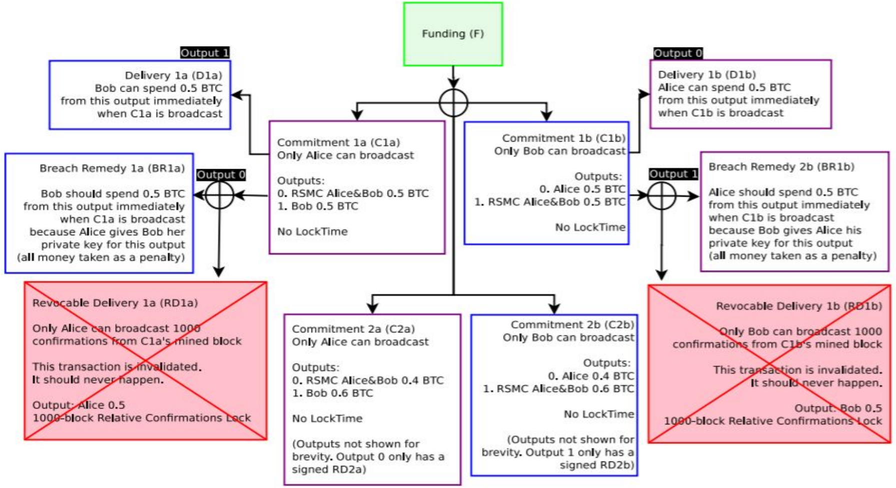


Multi-hop Payments



Lightning Channel Attack

Lightning Network White paper- Attack Description



Funding (F)

Output 1

Delivery 1a (D1a)
Bob can spend 0.5 BTC
from this output immediately
when C1a is broadcast

Commitment 1a (C1a)
Only Alice can broadcast

Outputs:
0. RSMC Alice&Bob 0.5 BTC
1. Bob 0.5 BTC

No LockTime

Commitment 1b (C1b)
Only Bob can broadcast

Outputs:
0. Alice 0.5 BTC
1. RSMC Alice&Bob 0.5 BTC

No LockTime

Output 0

Delivery 1b (D1b)
Alice can spend 0.5 BTC
from this output immediately
when C1b is broadcast

Breach Remedy 1a (BR1a)

Bob should spend 0.5 BTC
from this output immediately
when C1a is broadcast
because Alice gives Bob her
private key for this output
(all money taken as a penalty)

Output 0

Output 1

Breach Remedy 2b (BR1b)

Alice should spend 0.5 BTC
from this output immediately
when C1b is broadcast
because Bob gives Alice his
private key for this output
(all money taken as a penalty)

Revocable Delivery 1a (RD1a)

Only Alice can broadcast 1000
confirmations from C1a's mined block

This transaction is invalidated.
It should never happen.

Output: Alice 0.5
1000-block Relative Confirmations Lock

Commitment 2a (C2a)
Only Alice can broadcast

Outputs:
0. RSMC Alice&Bob 0.4 BTC
1. Bob 0.6 BTC

No LockTime

(Outputs not shown for
brevity. Output 0 only has a
signed RD2a)

Commitment 2b (C2b)
Only Bob can broadcast

Outputs:
0. Alice 0.4 BTC
1. RSMC Alice&Bob 0.6 BTC

No LockTime

(Outputs not shown for
brevity. Output 1 only has a
signed RD2b)

Revocable Delivery 1b (RD1b)

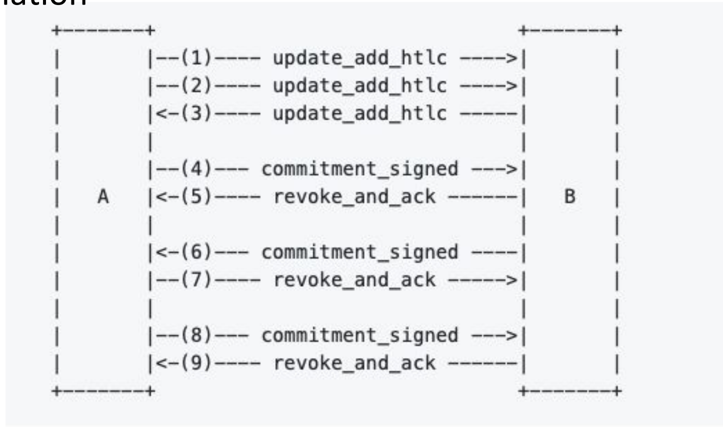
Only Bob can broadcast 1000
confirmations from C1b's mined block

This transaction is invalidated.
It should never happen.

Output: Bob 0.5
1000-block Relative Confirmations Lock

Figure 8: When C2a and C2b exist, both parties exchange Breach Remedy transactions. Both parties now have explicit economic incentive to avoid broadcasting old Commitment Transactions (C1a/C1b). If either party wishes to close out the channel, they will only use C2a (Alice) or C2b (Bob). If Alice broadcasts C1a, all her money will go to Bob. If Bob broadcasts C1b, all his money will go to Alice. See previous figure for C2a/C2b outputs.

BOLT(Basics of Lightning Technology)-HTLC explanation



```
+-------+                               +-------+
|       |--(1)---- update_add_htlc ---->|       |
|       |--(2)---- update_add_htlc ---->|       |
|       |<-(3)---- update_add_htlc -----|       |
|       |                               |       |
|       |--(4)--- commitment_signed --->|       |
|   A   |<-(5)---- revoke_and_ack ------|   B   |
|       |                               |       |
|       |<-(6)--- commitment_signed ----|       |
|       |--(7)---- revoke_and_ack ----->|       |
|       |                               |       |
|       |--(8)--- commitment_signed --->|       |
|       |<-(9)---- revoke_and_ack ------|       |
+-------+                               +-------+
```

Lightning Channel Attack

Submitting an earlier commitment transaction is a way to try to steal money from the other party.

After a lightning transaction, both parties have a commitment transaction, signed by both parties that spends the funds from the funding transaction and sends it to the parties in the current state. For example:

Alice is going to fund a channel to buy a bike from Bob and is then going to attack Bob

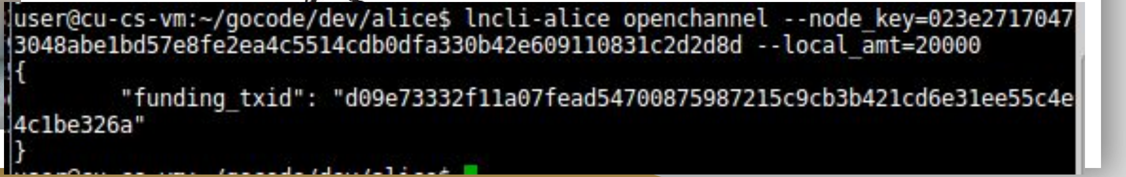
Lightning Channel Attack

Alice wants to buy a bike from Craigslist that Bob listed for 10,000 Satoshis.

Alice and Bob agree to meet.

Alice funds a lightning channel with Bob for 20,000 Satoshis.

$20,000



```
user@cu-cs-vm:~/gocode/dev/alice$ lncli-alice openchannel --node_key=023e2717047
3048abe1bd57e8fe2ea4c5514cdb0dfa330b42e609110831c2d2d8d --local_amt=20000
{
        "funding_txid": "d09e73332f11a07fead54700875987215c9cb3b421cd6e31ee55c4e
4c1be326a"
}
user@cu-cs-vm:~/gocode/dev/alice$
```

Lightning Channel Attack

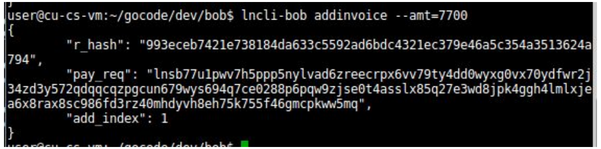
After haggling, Alice and Bob agree Alice will pay Bob $8800 Satoshis for the bike.

$8800 Satoshis

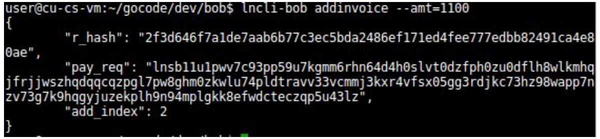
Lightning Channel Attack

Bob writes up invoices totaling 8800 Satoshis.

Invoice:
$7700 +
1100
Satoshis



```
user@cu-cs-vm:~/gocode/dev/bob$ lncli-bob addinvoice --amt=7700
{
        "r_hash": "993eceb7421e738184da633c5592ad6bdc4321ec379e46a5c354a3513624a
794",
        "pay_req": "lnsb77u1pwv7h5ppp5nylvad6zreecrpx6vv79ty4dd0wyxg0vx70ydfwr2j
34zd3y572qdqqcqzpgcun679wys694q7ce0288p6pqw9zjse0t4asslx85q27e3wd8jpk4ggh4lmlxje
a6x8rax8sc986fd3rz40mhdyvh8eh75k755f46gmcpkww5mq",
        "add_index": 1
}
```



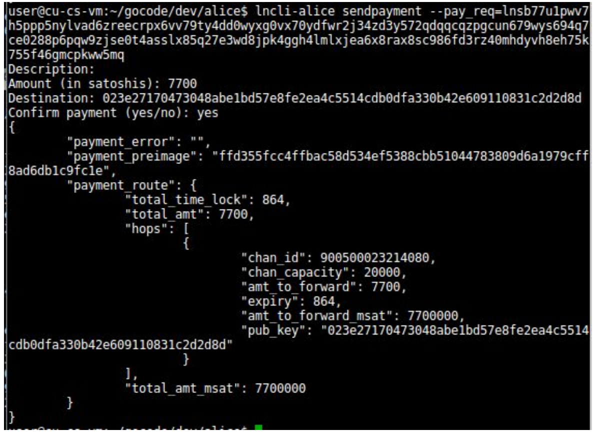
```
user@cu-cs-vm:~/gocode/dev/bob$ lncli-bob addinvoice --amt=1100
{
        "r_hash": "2f3d646f7a1de7aab6b77c3ec5bda2486ef171ed4fee777edbb82491ca4e8
0ae",
        "pay_req": "lnsb11u1pwv7c93pp59u7kgmm6rhn64d4h0slvt0dzfph0zu0dflh8wlkmhq
jfrjjwszhqdqqcqzpgl7pw8ghm0zkwlu74pldtravv33vcmmj3kxr4vfsx05gg3rdjkc73hz98wapp7n
zv73g7k9hqgyjuzekplh9n94mplgkk8efwdcteczqp5u43lz",
        "add_index": 2
}
```

Alice sends the payments

and Bob accepts them.

Both have signed.

$8800 Satoshis



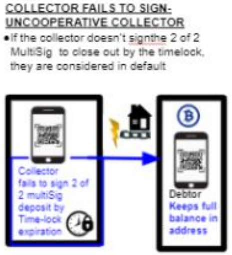
```
user@cu-cs-vm:~/gocode/dev/alice$ lncli-alice sendpayment --pay_req=lnsb77u1pwv7
h5ppp5nylvad6zreecrpx6vv79ty4dd0wyxg0vx70ydfwr2j34zd3y572qdqqcqzpgcun679wys694q7
ce0288p6pqw9zjse0t4asslx85q27e3wd8jpk4ggh4lmlxjea6x8rax8sc986fd3rz40mhdyvh8eh75k
755f46gmcpkww5mq
Description:
Amount (in satoshis): 7700
Destination: 023e27170473048abe1bd57e8fe2ea4c5514cdb0dfa330b42e609110831c2d2d8d
Confirm payment (yes/no): yes
{
        "payment_error": "",
        "payment_preimage": "ffd355fcc4ffbac58d534ef5388cbb51044783809d6a1979cff
8ad6db1c9fc1e",
        "payment_route": {
                "total_time_lock": 864,
                "total_amt": 7700,
                "hops": [
                        {
                                "chan_id": 900500023214080,
                                "chan_capacity": 20000,
                                "amt_to_forward": 7700,
                                "expiry": 864,
                                "amt_to_forward_msat": 7700000,
                                "pub_key": "023e27170473048abe1bd57e8fe2ea4c5514
cdb0dfa330b42e609110831c2d2d8d"
                        }
                ],
                "total_amt_msat": 7700000
        }
}
```

Bob does not notice Alice does not close the channel...

Lightning Channel Attack

...because she is waiting for Bob to go on vacation so while he is away he won’t notice his LN node is off-line while the next 864 blocks elapse...

Alice broadcasts the revoked commitment to the earlier state of the channel to the blockchain, to before when she had paid $7,700, so there’s more $ in her wallet.



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

Collector fails to sign 2 of 2 multiSig deposit by Time-lock expiration
B
Debtor
Keeps full balance in address

Lightning Channel Attack

Alice makes off with the bike and $20,000 Satoshis because she knows Bob is not watching for her fraudulent transaction attempt.

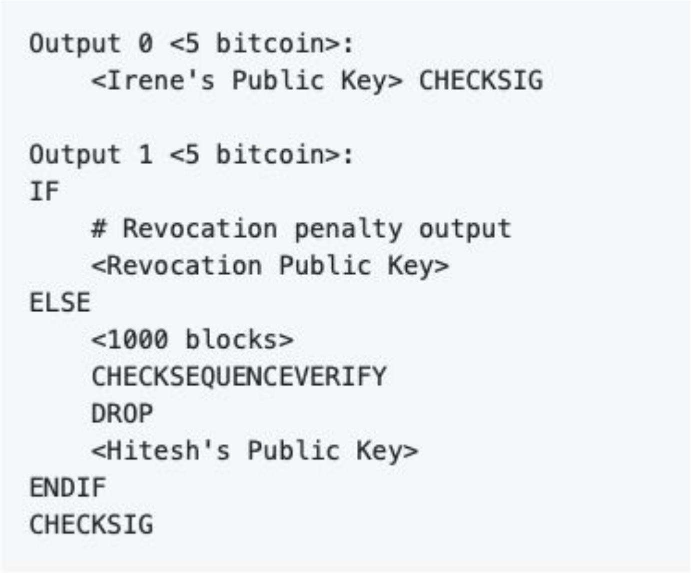
Because the HTLC lapses while Bob’s node is down, he loses the chance to broadcast the latest revocation transaction, where they had both signed as Alice paying $7,700 Satoshis.

Bob could have used some help watching out for this.

Watchtowers

- A watchtower is a program that could be a server you run or a third party which automates revocation of fraudulent transactions
- Being implemented as a part of LND lightning implementation
- Future plans are for third parties on the lightning network that you could pay a small fee to watch for you

Commitment Transaction



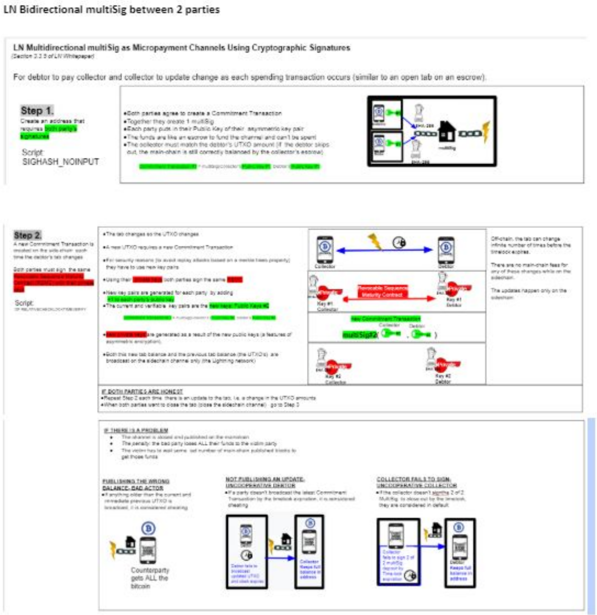
```
Output 0 <5 bitcoin>:
    <Irene's Public Key> CHECKSIG

Output 1 <5 bitcoin>:
IF
    # Revocation penalty output
    <Revocation Public Key>
ELSE
    <1000 blocks>
    CHECKSEQUENCEVERIFY
    DROP
    <Hitesh's Public Key>
ENDIF
CHECKSIG
```

Mastering Bitcoin:

https://github.com/bitcoinbook/bitcoinbook/blob/develop/ch12.asciidoc

The Alice on Bob attack in detail



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 [illegible]

Script:
[illegible]

- The tab changes as 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 [illegible] both parties sign the same [illegible]
- New key pairs are generated for each party by adding [illegible] to each party's public key
- The current and verifiable key pairs are the new keys: Public Keys #2

[illegible] = multisig(collector's [illegible], debtor's [illegible])

- [illegible] 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

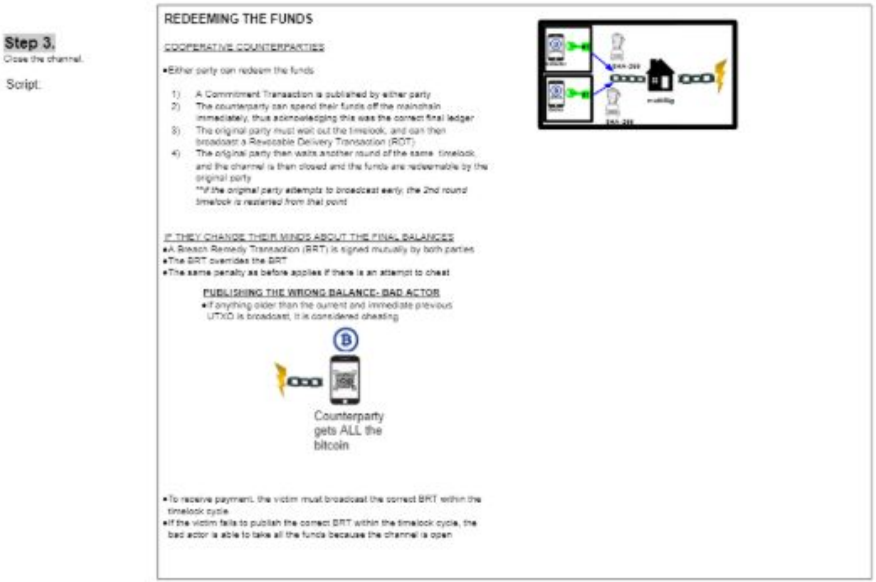
Counterparty gets ALL the bitcoin

NOT PUBLISHING AN UPDATE- UNCOOPERATIVE DEBTOR
- If a party doesn't broadcast the latest Commitment Transaction by the timelock expiration, it is considered cheating

Debtor fails to broadcast updated UTXO and clock expires
Collector Keeps full balance in address

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

Collector fails to sign 2 of 2 multiSig deposit by Time lock expiration
Debtor Keeps full balance in address



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