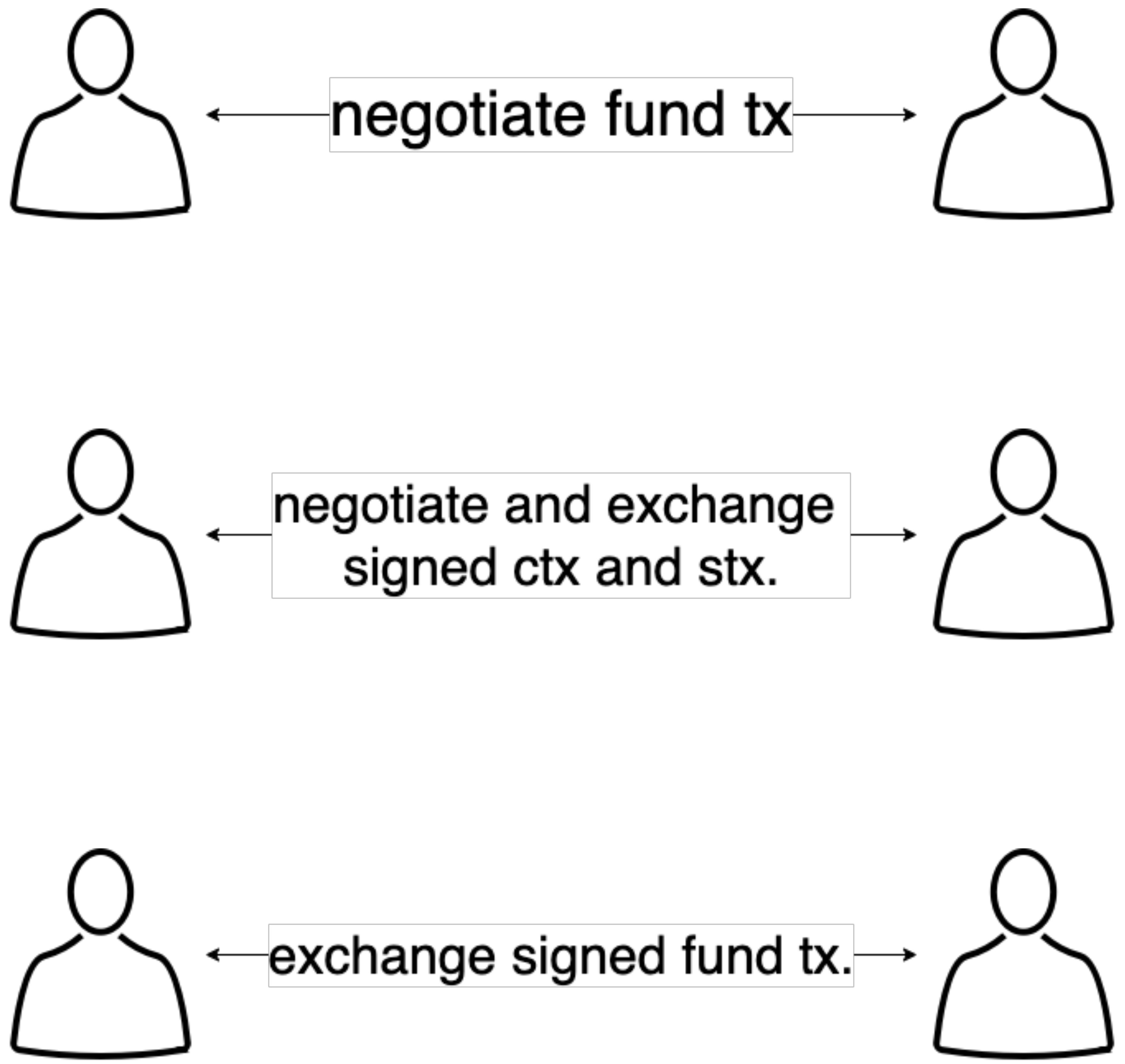
The update of GPC

Zhichun Lu

Channel establishment



Zhichun Lu

Communication model

Let us begin with the document structure in database and the msg composition.

Msg: {id, type}, Doc: {id, status, msg_cache}

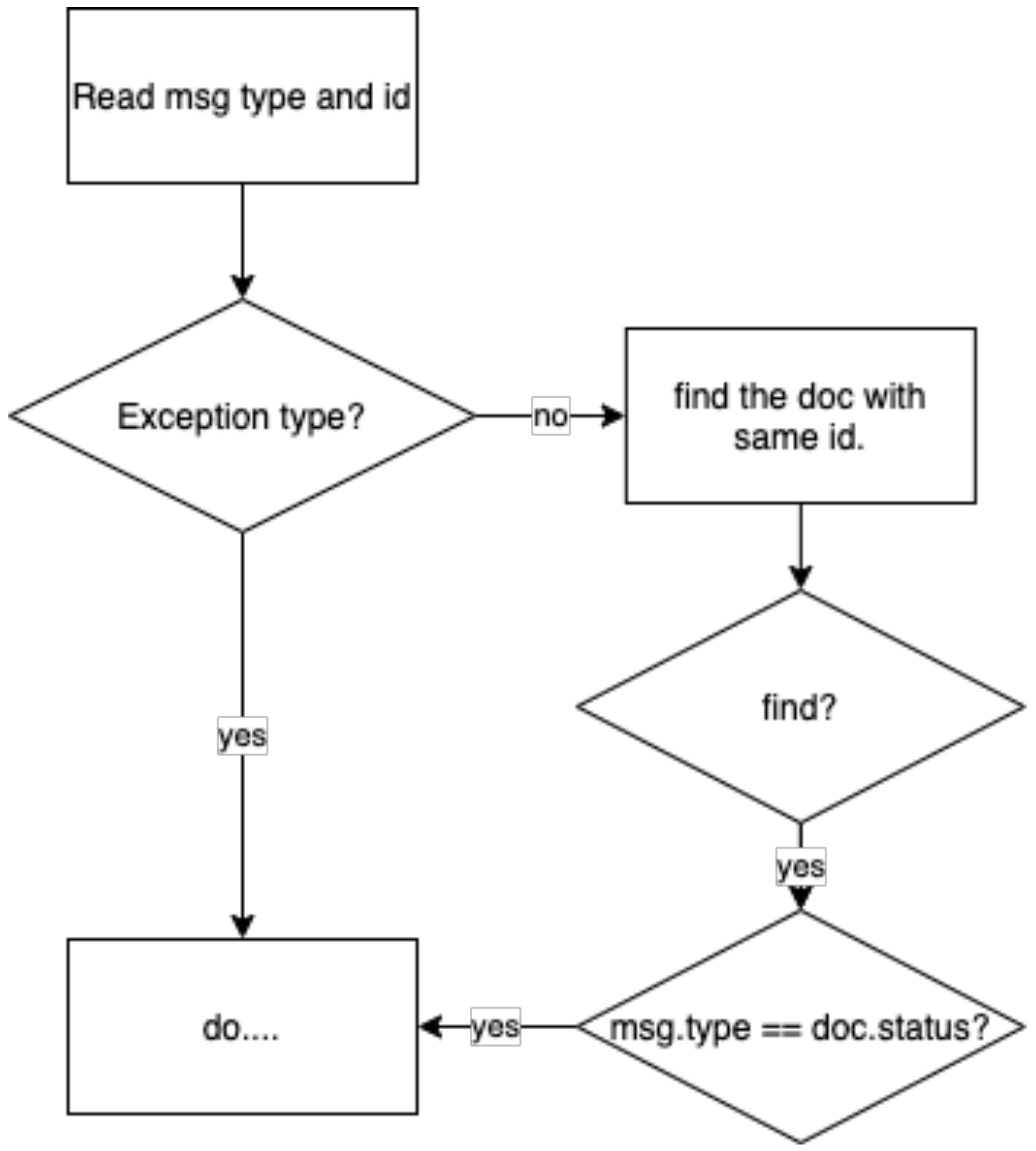
- -2: If the msg type is -2, it means the msg is the “reset” msg. More specifically, let us assume there is an interaction with five steps, and we have researched the 4th. However, we just want to change some args negotiated in previous steps. In this case, we just adopt the “reset” msg. Note: It can only reset the status to the latest steady state. I.e., if we have reached the stage of making payments, reseting the local status to stage of establishing channel is forbidden.

- -1: It is the “re-request” msg. If A misses the msg sent by B, he can send the “re-request” msg. Then B will response with the msg in msg_cache.

- 0: It is the plain text, user will parse the msg to get the “text” and print it.

- 1: It is the first step, so we do not require the type is consistent with the local status, but it demands there is no doc with same id in local database.

Zhichun Lu



Msg detail

Msg_send: 1

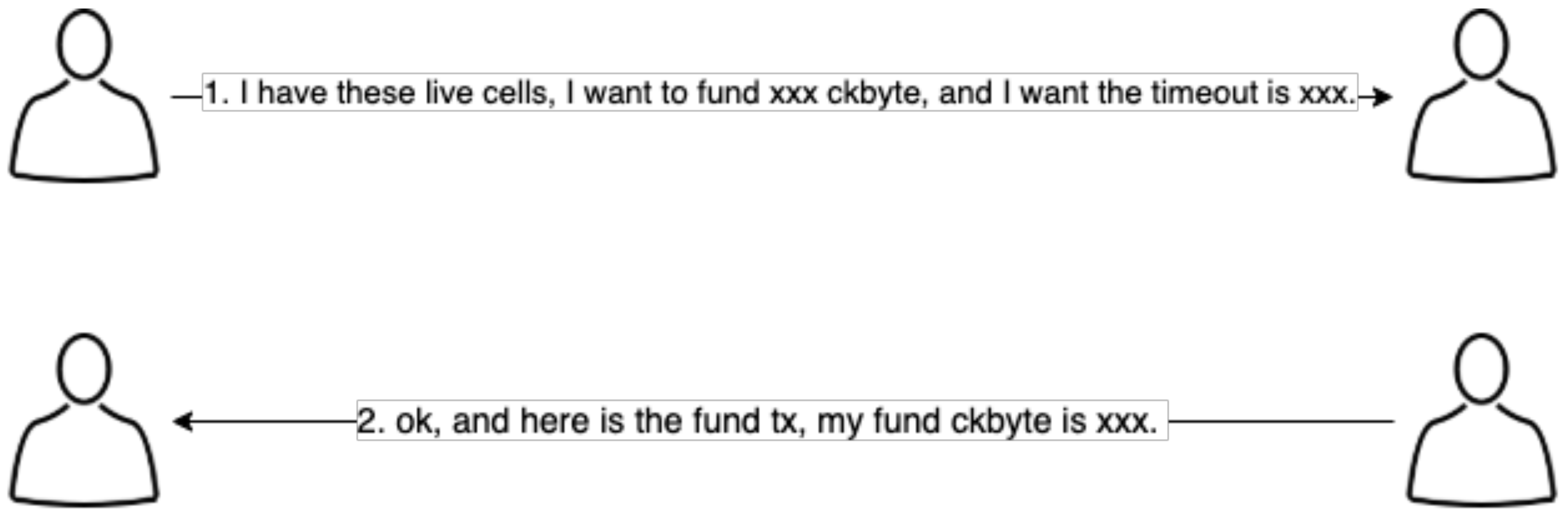
Require: remote_ip, remote_port, local_capacity, local_fee, timeout.

do:

- local_fund_inputs = Gather_input (local_capacity, local_fee)
- Id = hash (local_fund_inputs)
- Pubkey = blake160(key.pubkey)

Ensure:

- msg = { id, 1 || local_fund_input, local_capacity, local_fee, timeout}
- doc = { id, key, pubkey, status:2, nounce: 0, ctx: 0, stx: 0, gpc_script_hash: 0, local_fund_cells: local_fund_input, }


1. I have these live cells, I want to fund xxx ckbyte, and I want the timeout is xxx.
2. ok, and here is the fund tx, my fund ckbyte is xxx.

Zhichun Lu

Msg detail

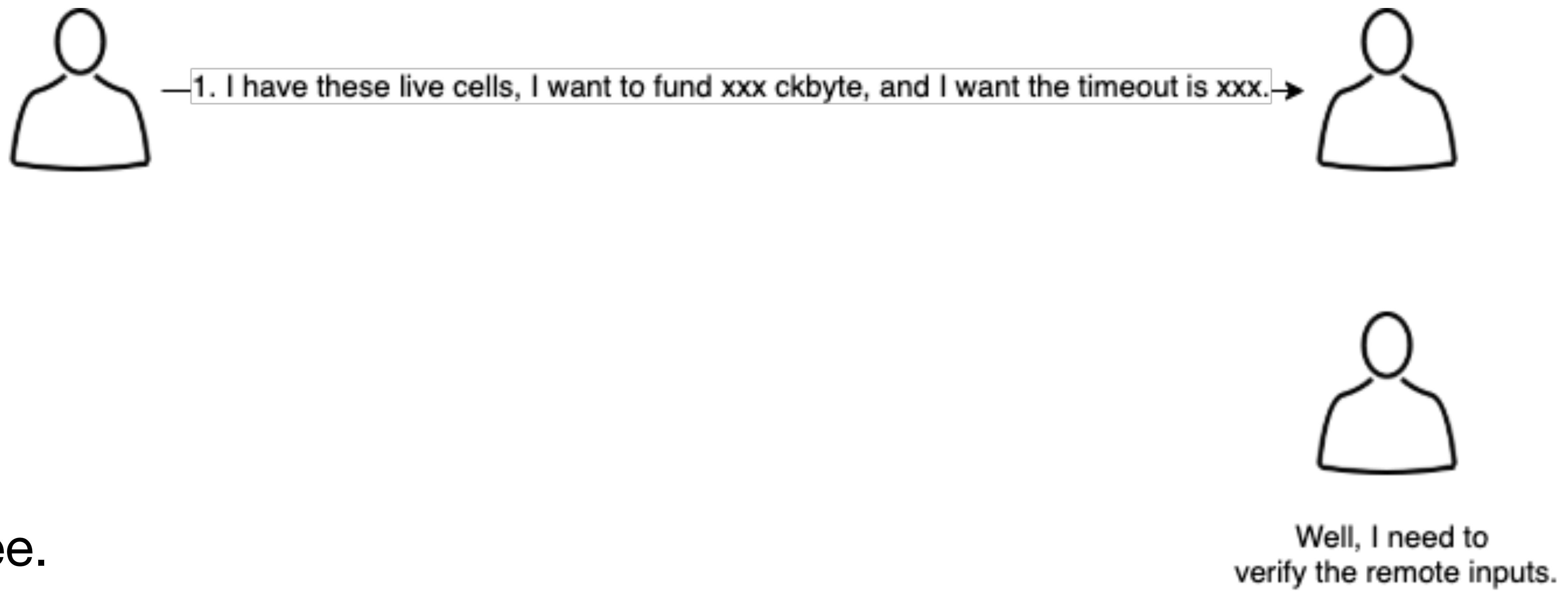
Msg_recv: 1

Require: remote_inputs

do:

- The cells are alive.
- The cell’s capacity is enough for remote_capacity and remote_fee.
- The capacity is satisfactory.

Ensure: Whether the inputs are valid (live and rich enough). If not, just report errors (using msg type 0).


1. I have these live cells, I want to fund xxx ckbyte, and I want the timeout is xxx.
Well, I need to verify the remote inputs.

Zhichun Lu

Msg detail

Well. now, I need to construct the fund tx,
1. input, 2. output, 3. output_data,
4. witness, 5. version, 6. cell_deps

Inputs: Just gather inputs as A did.

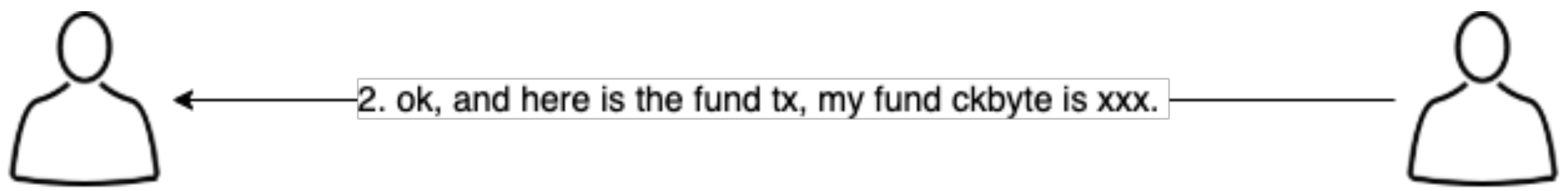
outputs: 1. GPC, 2. remote_change (just calculate it !), 3.local_change. More specifically. 1.capacity 2.lock 3.type

Do I have enough information to generate the GPC lock?
(lock_status:0, nounce: 0, timelock: timelock, remote_pubkey: remote_pubkey, local_pubkey: lcoal key in database). Yes!

ouput_data: just set it to “”. Future for UDT or other assets, the user should calculate it by himself.

Zhichun Lu

Msg detail



Msg_send: 2

Require: local_capacity, local_fee.

do:

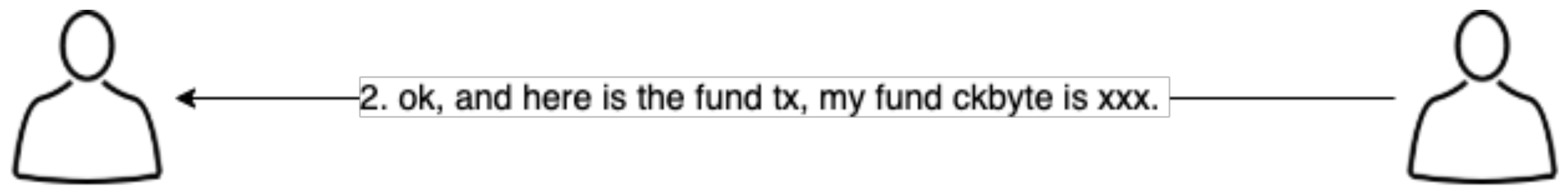
- local_fund_inputs = Gather_input (local_capacity, local_fee)
- GPC_lock = assemble_gpc_lock (0, 0, timeout, remote_pubkey, local_pubkey)
- fund_tx = (local_fund_inputs, remote_capa, local_capa, GPC_lock)
- gpc_script_hash = GPC_lock_script

Ensure:

- msg = { id, 2 || fund_tx, local_capacity, local_fee, timeout}
- doc = { id, key, pubkey, status:3, nounce: 0, ctx: 0, stx: 0, gpc_script_hash, fund_tx, }

Zhichun Lu

Msg detail



Msg_recv: 2

Require: remote_fund_tx

do:

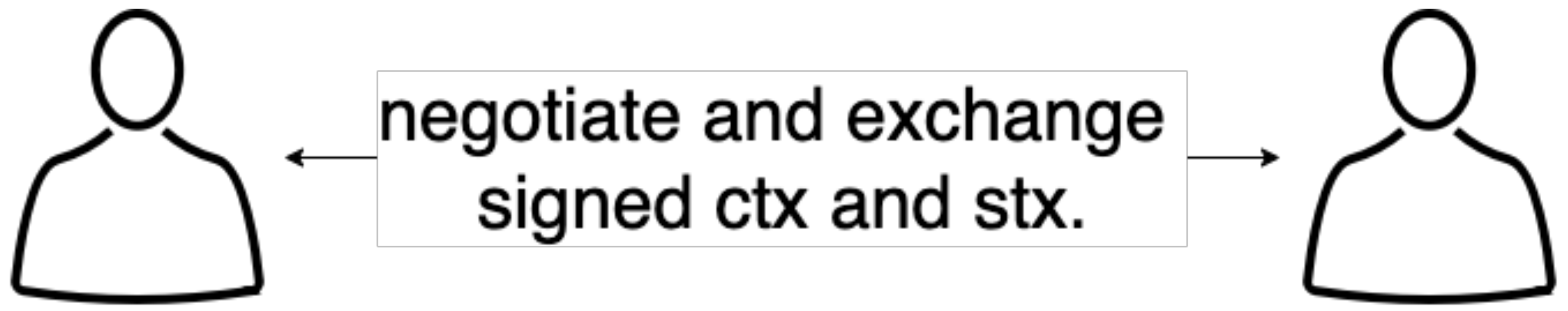
Will he modify my input? Does the capacity he fund is satisfactory? I need to verify it!

- The cells are alive, the capacity is enough and the fund capacity is satisfactory.
- The GPC lock is right. (just construct and compute the hash by myself, and compare).
- My inputs is not modified. (there is a duplicate in my database).
- All the condition that the tx can be accepted by blockchain. (except the witness)

Ensure: Whether the fund tx is valid. If not, just report errors (using msg type 0).

Zhichun Lu

Msg detail



Firstly, lets discuss the no-input design applied in ctx and stx. In fact, the two users exchange the ctx_info but not ctx, since the inputs of ctx are still pending. So we need to specify the information for users to generate the ctx, i.e., ctx_info.

Tx: version, cell_dep, inputs, outputs, ouputs_data, witnesses.

It is obvious that inputs should not included in ctx_info. Version (?) and cell_dep also, since if user change these two fields, the tx will not be accepted by the blockchain or by the pointed smart contract. So we can conclude that (outputs, outputs_data, witness) is the ctx_info tuple.

Zhichun Lu

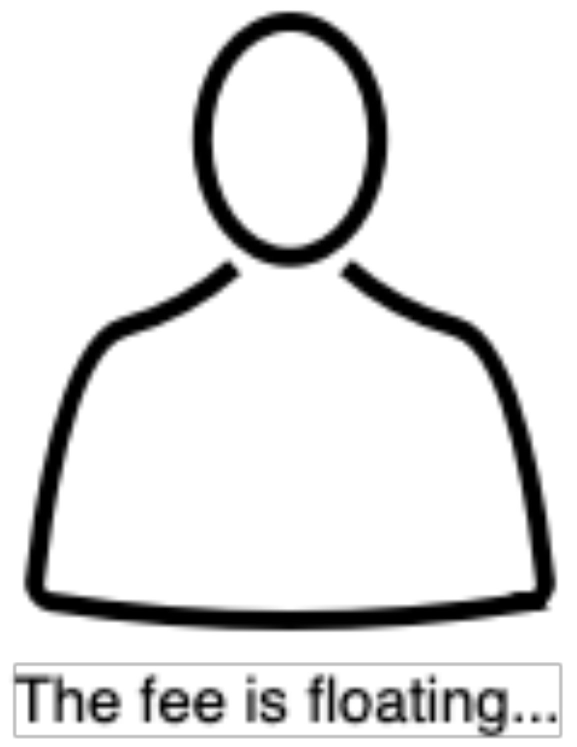
Msg detail

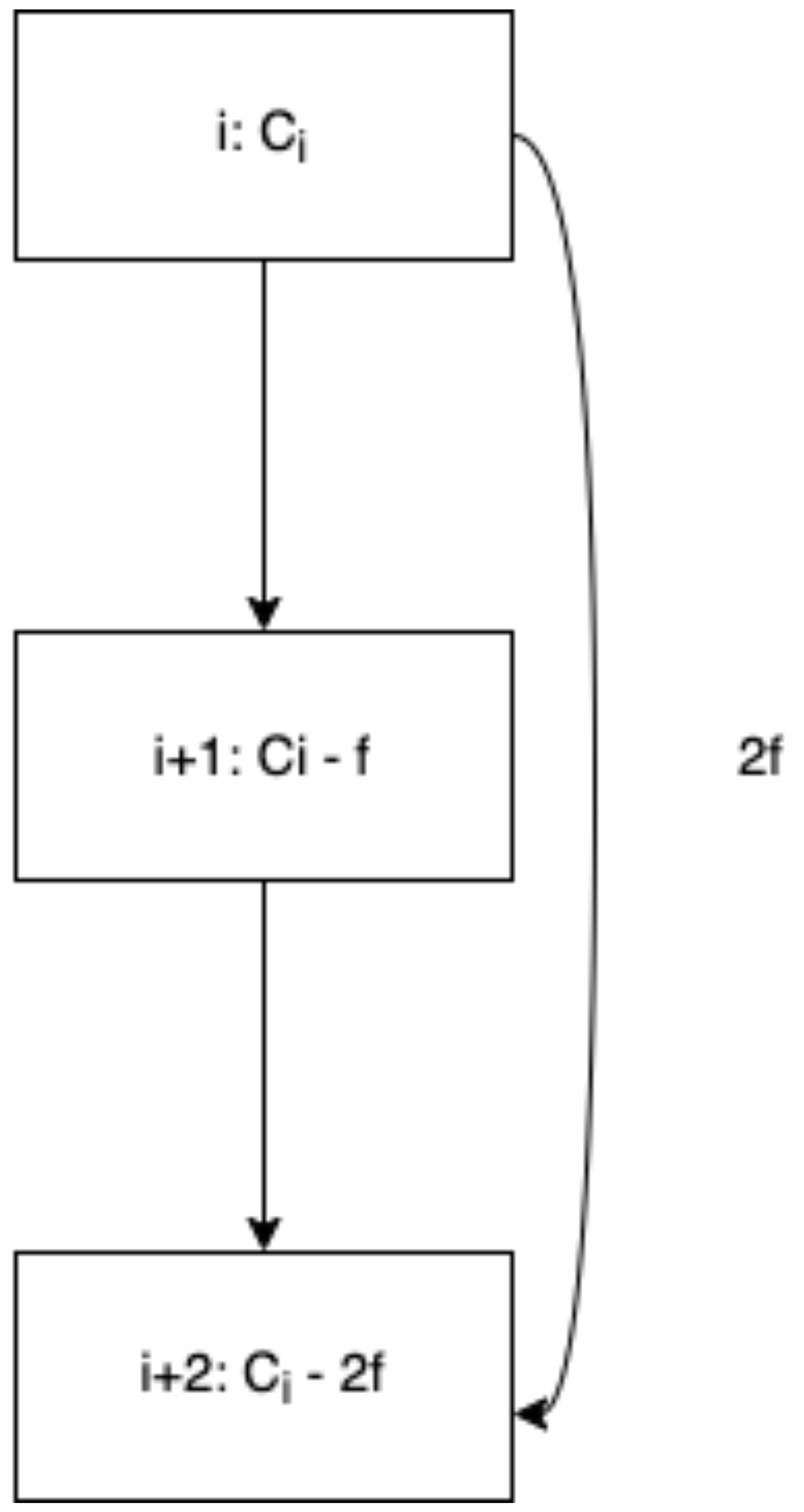
Q: what should we sign? A: The ctx_info.

Q: what should ctx_info.outputs have?

Lets begin with just one output, I.e., the output with GPC lock. Obviously, it is necessary to determine the capacity of it. Lets say the capacity of output of nounce i is C_i.

Design 1: Fee is f. $C_i - f = C_{i+1}$; $C_{i+1} - f = C_{i+2}$;

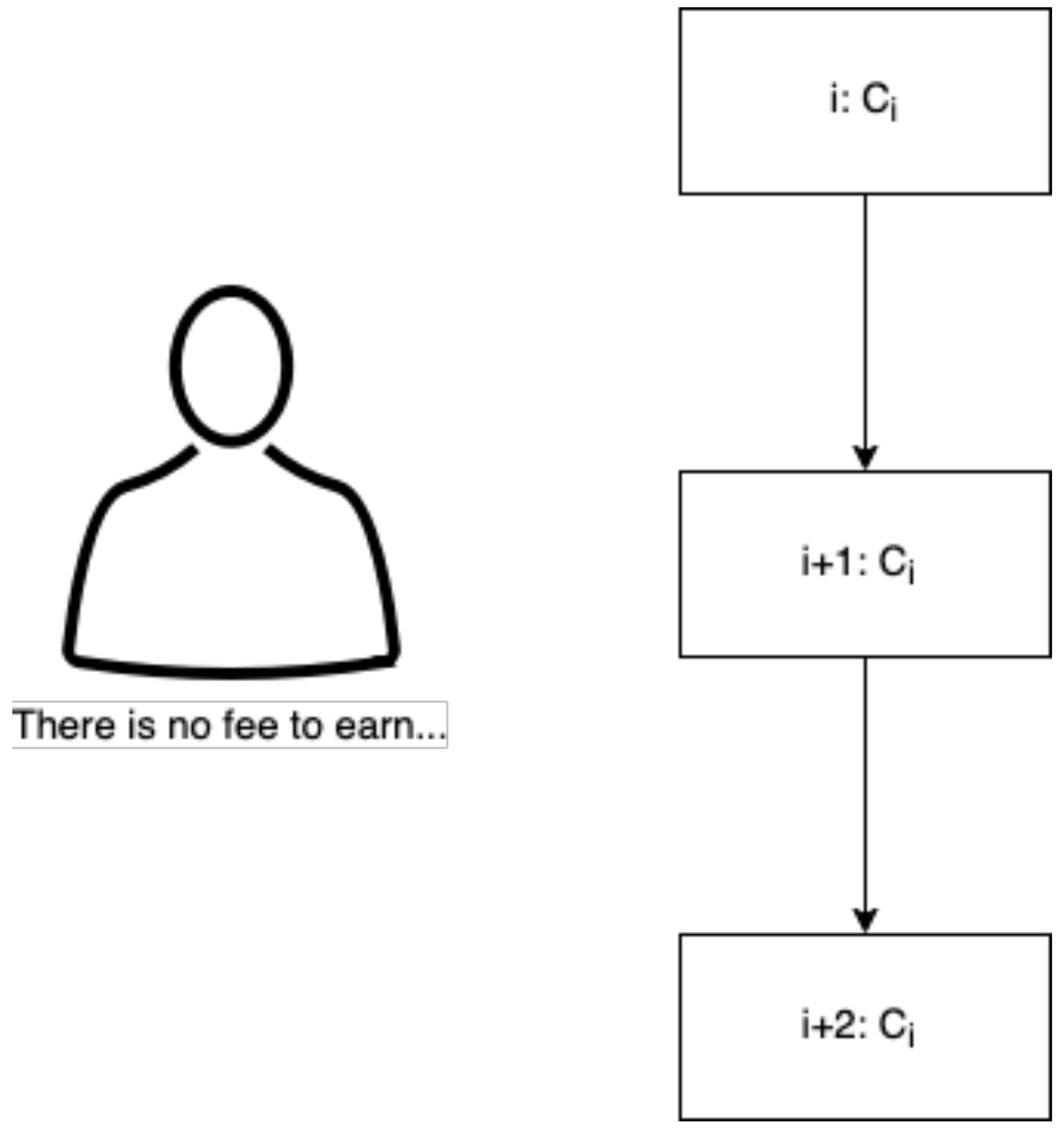




Zhichun Lu

Msg detail

Design 1: Fee is f. $C_{i+1} = C_i$; $C_{i+2} = C_i$;



Zhichun Lu

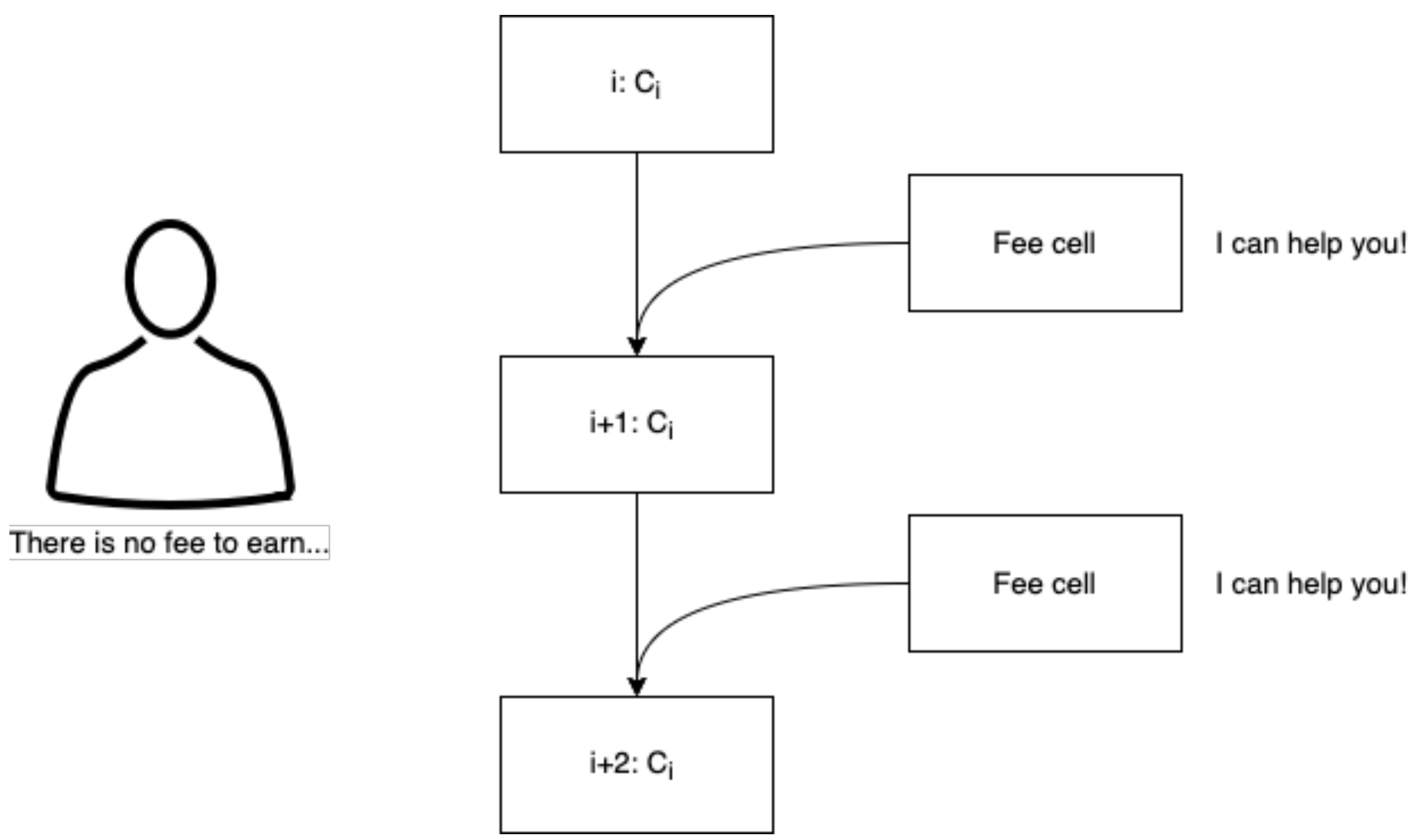
Msg detail

Design 2: Fee is f. $C_{i+1} = C_i$; $C_{i+2} = C_i$;

So, we can know that the inputs must include the fee cell, as a result, there are a change output, which is unpredictable. So, the no-input signature and verification only check the first output in ctx and first two outputs in stx. Output_data and witness are the same.

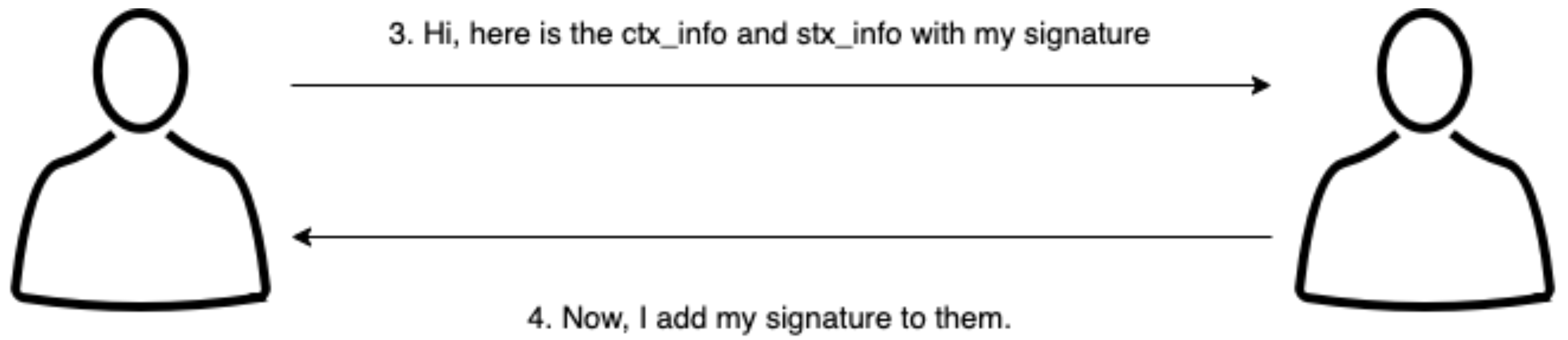
ctx_info: The first output, outputs_data and witness.

stx_info: The first two outputs, outputs_data and witnesses.



Zhichun Lu

Msg detail


3. Hi, here is the ctx_info and stx_info with my signature
4. Now, I add my signature to them.

Msg_send: 3

Require: fee

do:

- Closing_GPC_lock = assemble_gpc_lock (1, 1, timeout, remote_pubkey, local_pubkey)
- Closing capacity = fund_tx[0].capacity - fee. Closing_witnesses = generate_empty_witness (flag: 1, noucne: 1, sig_index).
- ctx_info = (Closing_GPC_lock, closing capacity, closing_output_data, closing_witness (unsign), sig_index: 0)
- stx_info is almost the same as ctx…

Ensure:

- msg = { id, 3 || ctx, stx, fee}
- doc = { id, key, fund_tx, status:4, nounce: 1 (it maybe confusing, since I just merge fund_closing and closing-closing.), ctx, stx. }

Zhichun Lu

Msg detail

Msg_recv: 3

Require: ctx_info, stx_info and the remote_pubkey.

do:

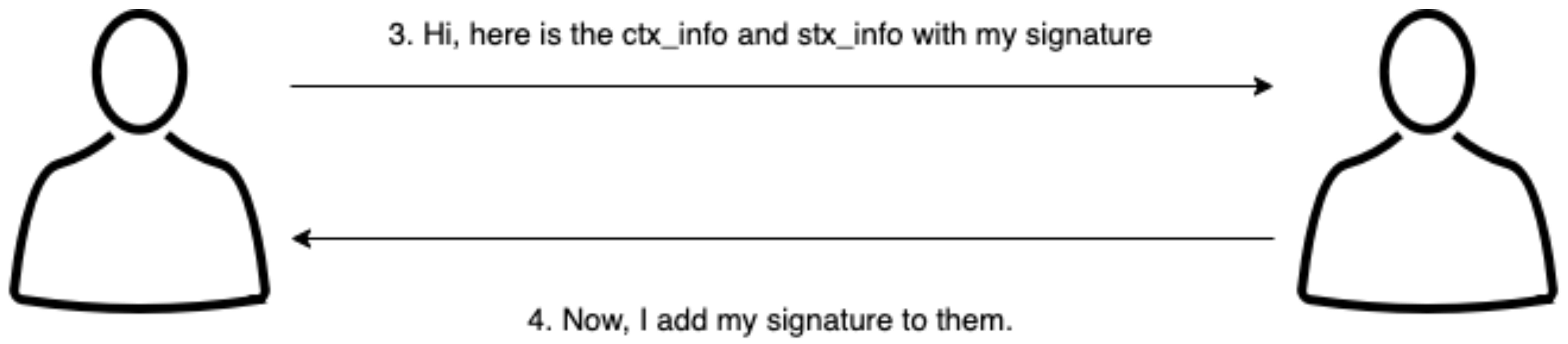
- The signature is valid.
- The information are right. (just create the info by myself and check).

Ensure:

- Whether the stx and ctx are valid. If not, just report errors (using msg type 0).

Zhichun Lu

Msg detail


3. Hi, here is the ctx_info and stx_info with my signature
4. Now, I add my signature to them.

Msg_send: 4

Require:

do:

- Just like the msg 3, but the sig_index set to 1. (0 in msg 3).

Ensure:

- msg = { id, 3 || ctx, stx, fee}
- doc = { id, key, fund_tx, status:5, nounce: 1, ctx, stx. }

Zhichun Lu

Msg detail

Msg_recv: 4

Require: ctx_info, stx_info, remote_pubkey and local signature.

do:

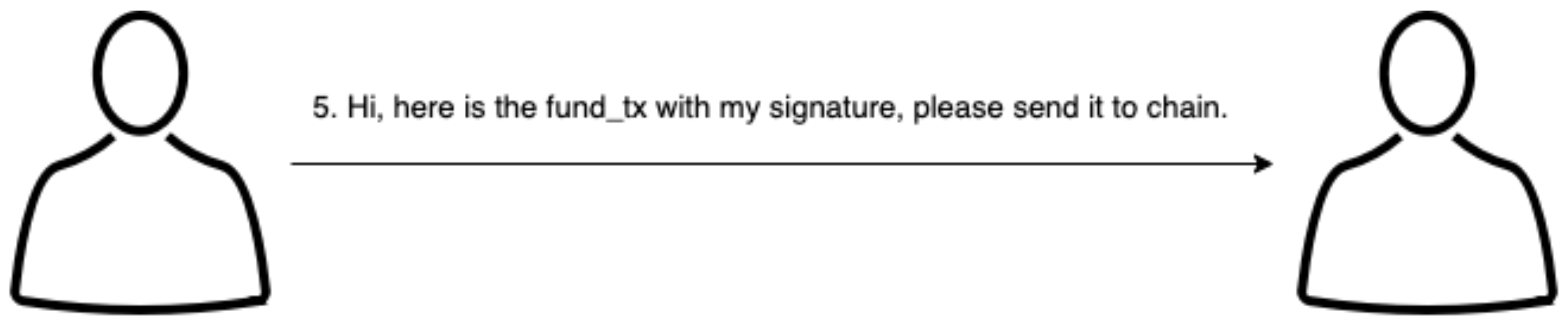
- The remote signature is valid.
- The info is not modified, just using the local signature to verify msg.

Ensure:

- Whether the stx and ctx are valid. If not, just report errors (using msg type 0).

Zhichun Lu

Msg detail


5. Hi, here is the fund_tx with my signature, please send it to chain.

Msg_send: 5

Require: fund_tx

do:

- fund_tx = Sign_tx (fund_tx)

Ensure:

- msg = { id, 5 || fund_tx}
- doc = { id, key, fund_tx, status:6. (steady status)}

Zhichun Lu

Msg detail

Msg_recv: 5

Require: fund_tx, local_fund_tx.

do:

- The fund_tx are same.
- Sign the fund_tx.

Ensure:

- Whether the stx and ctx are valid. If not, just report errors (using msg type 0).
- doc = { id, key, fund_tx, status:6. (steady status)}

Zhichun Lu

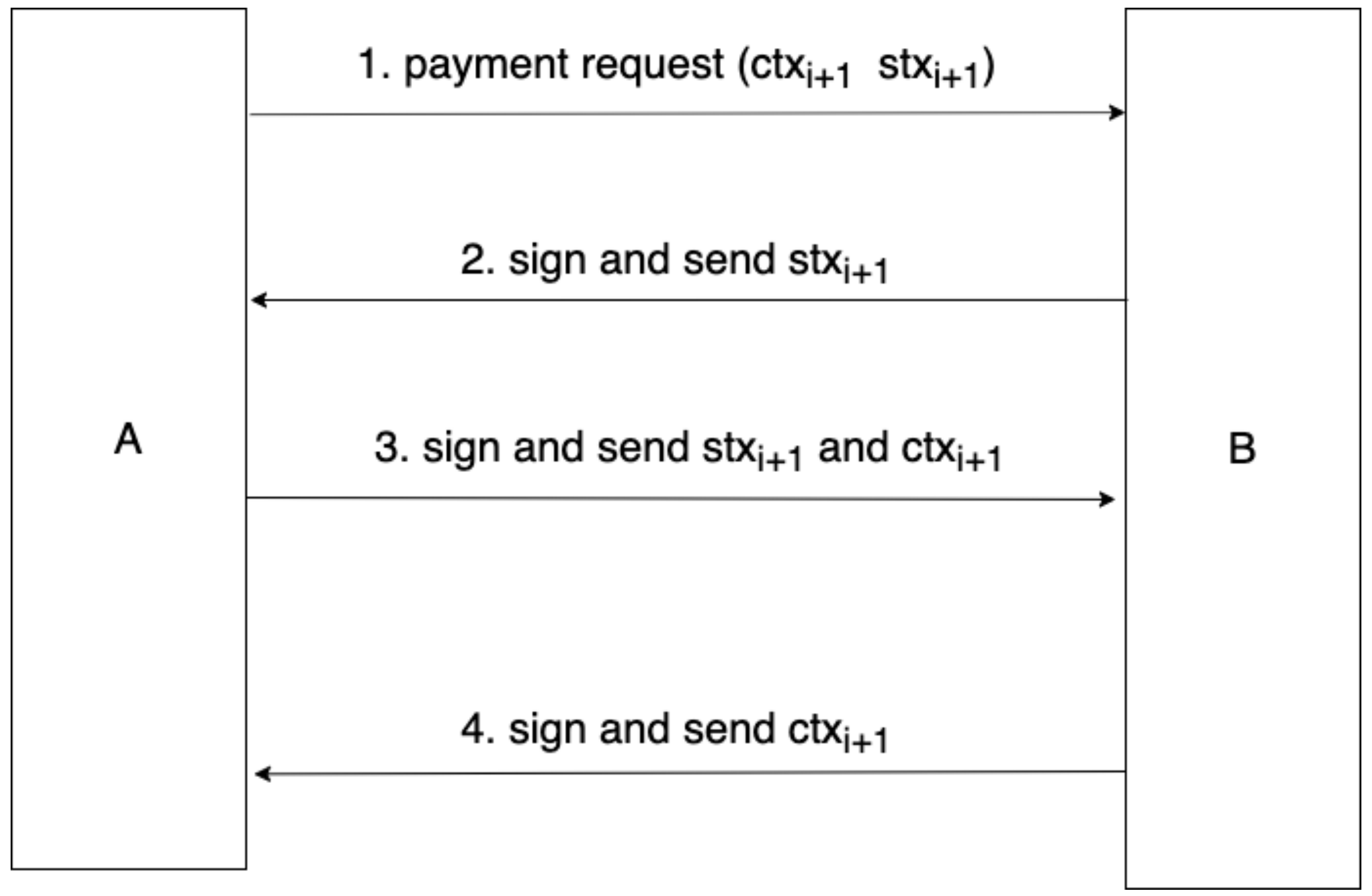
Msg detail

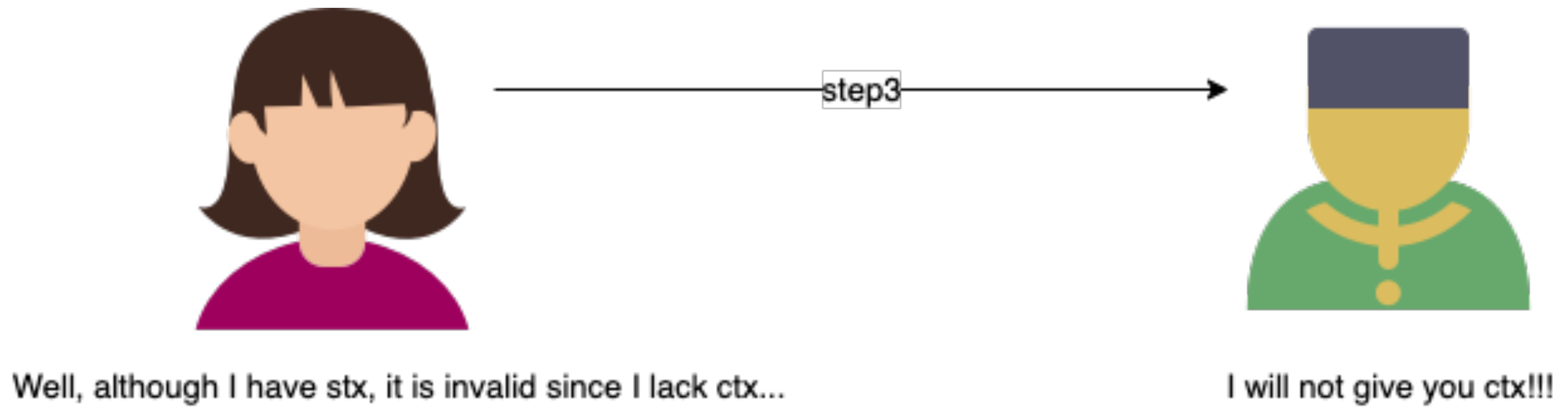
Well, I will spend the fund input.

I will not upload the fund_tx!!!!

Zhichun Lu

Make payment

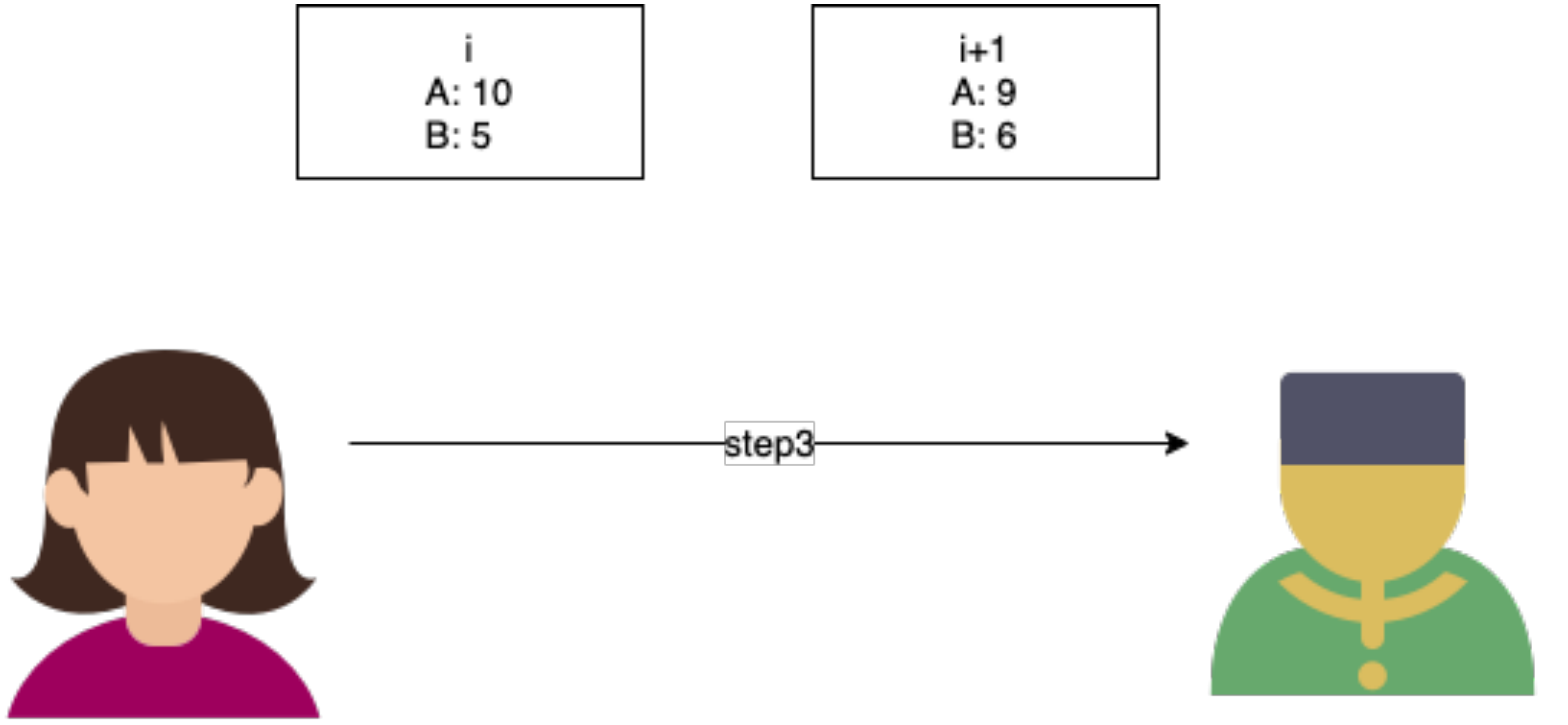

1. payment request (ctx_{i+1} stx_{i+1})
2. sign and send stx_{i+1}
A
3. sign and send stx_{i+1} and ctx_{i+1}
B
4. sign and send ctx_{i+1}


step3
Well, although I have stx, it is invalid since I lack ctx...
I will not give you ctx!!!

- If Alice is the payee in the payment, she loses the money. Of course, they can regard the payment is invalid until the step 4 is done.

Zhichun Lu

Make payment

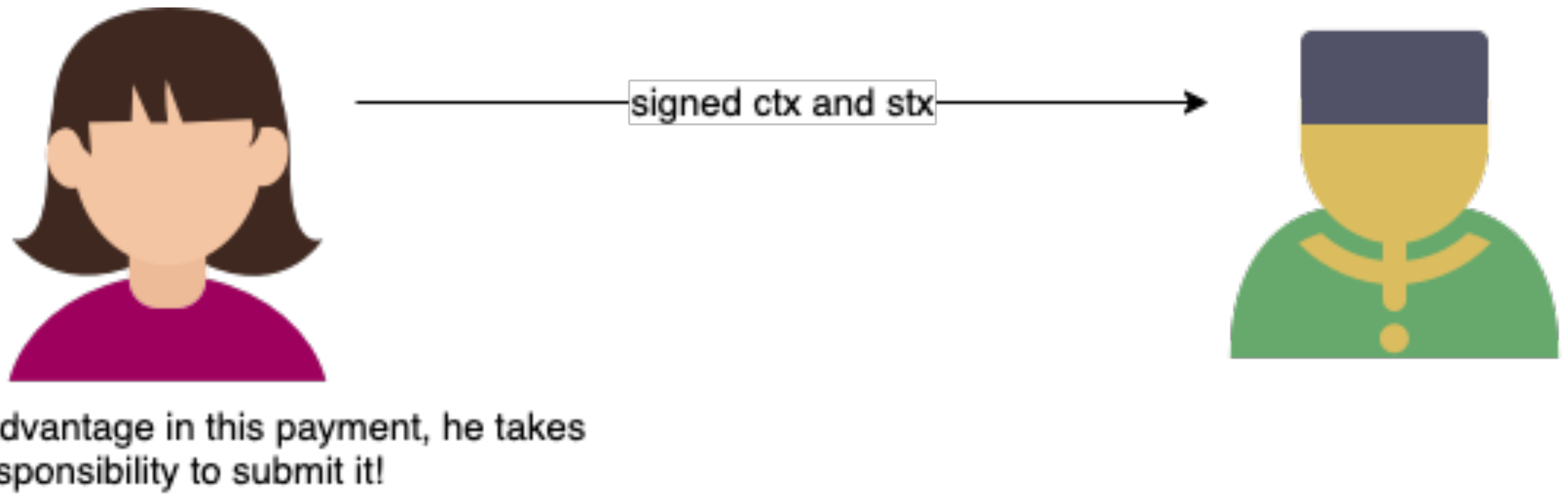


It is fine! I just submit c_i, if you refuse to submit c_{i+1}, you can only get 5 btc.

I will not give you ctx!!!

Zhichun Lu

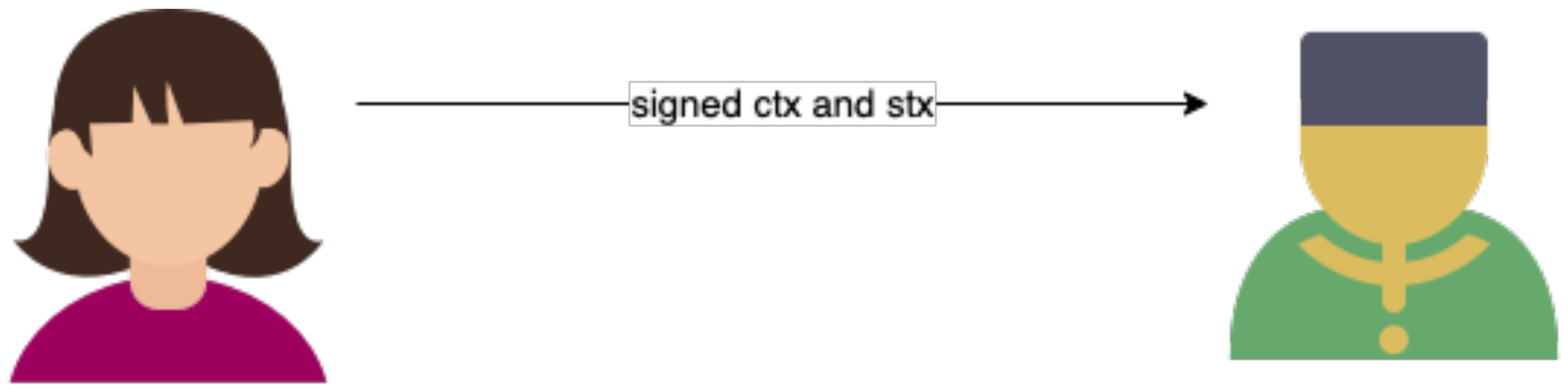
Make payment


signed ctx and stx
Since he has advantage in this payment, he takes responsibility to submit it!

Who is the payer of the payment, who give the signature of ctx and stx. It only needs one step to make the payments!

Zhichun Lu

Make payment



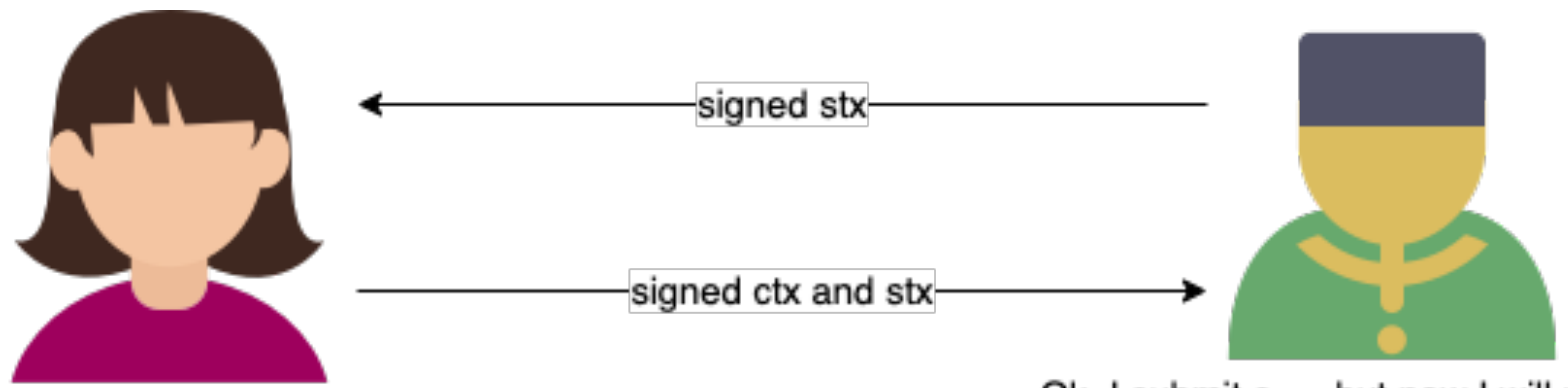
I submit the c_i, you should submit the c_{i+1}! Otherwise, you will lose money!

Ok, I submit c_{i+1}, but now I will not submit stx_{i+1}, hahaha, now you money is locked forever!

Zhichun Lu

Make payment

Solution 1

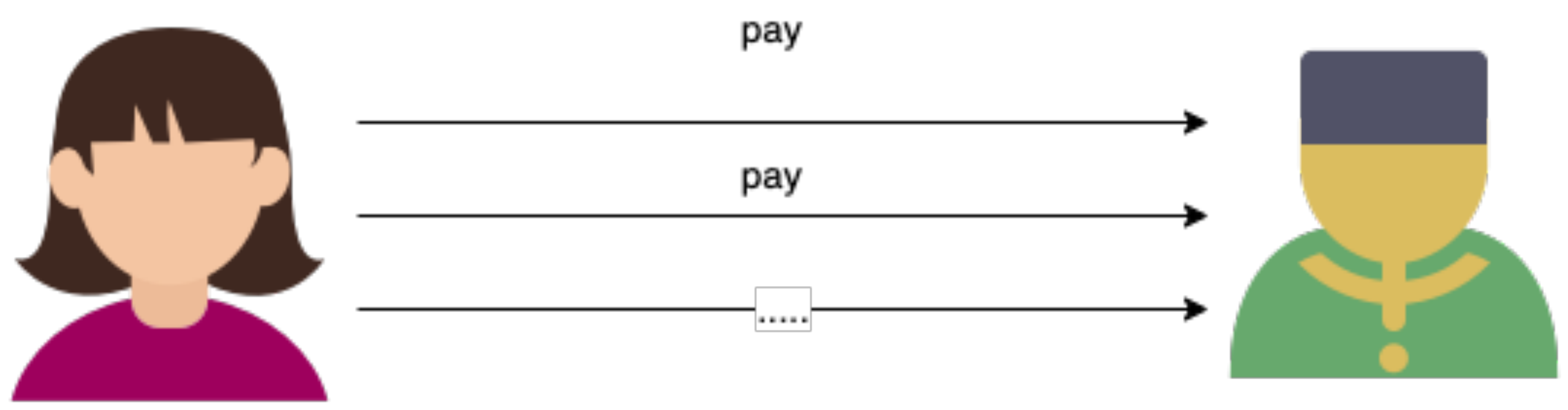


I submit the c_i, you should submit the c_{i+1}!
Otherwise, you will lose money!

Ok, I submit c_{i+1}, but now I will not submit stx_{i+1}, hahaha, now you money is locked forever... oops, you also have stx_{i+1}...

Zhichun Lu

Make payment

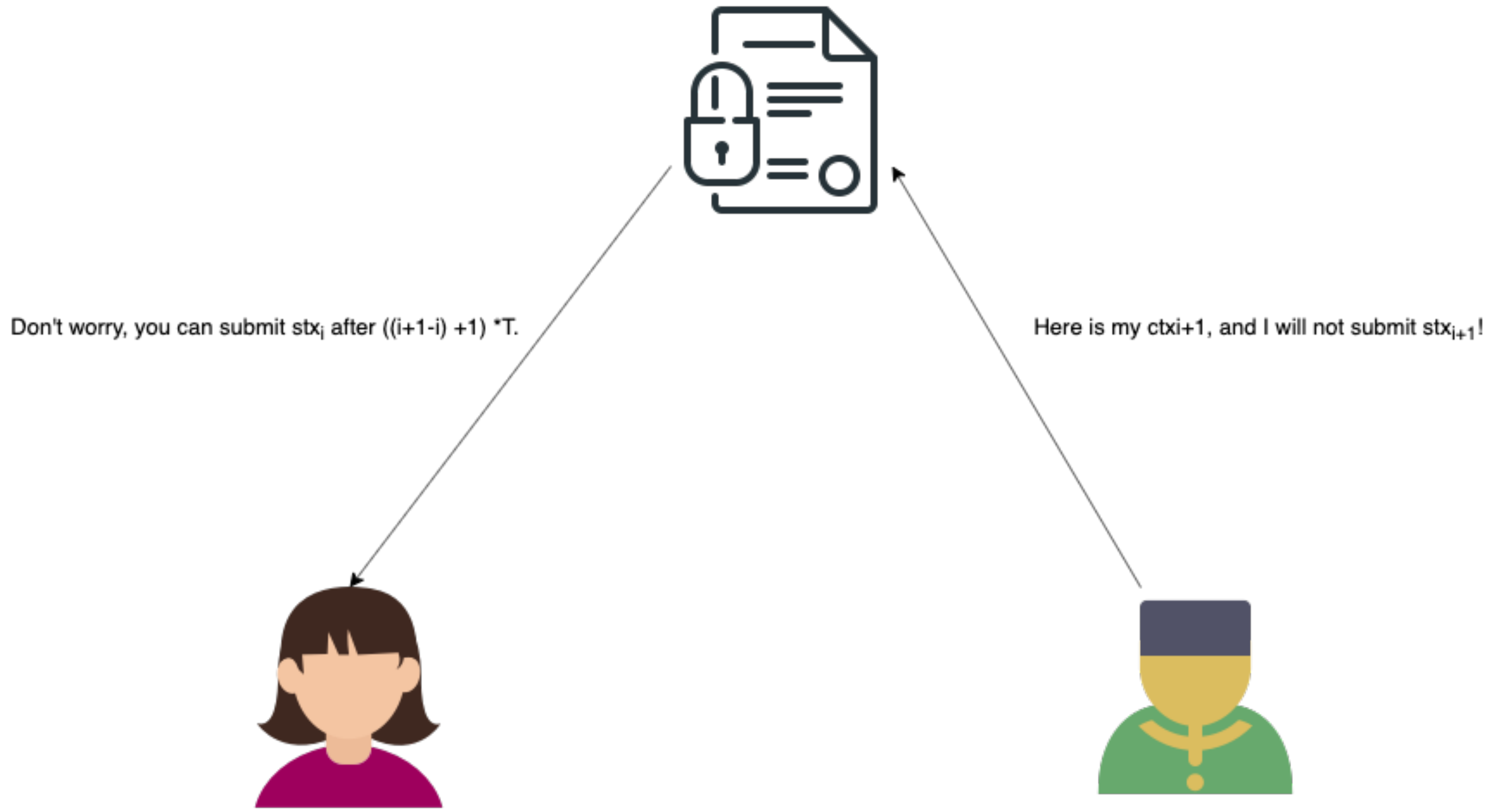


Oh, I need to store all stx, since I
do not know he will submit which one. It is frustrating...

Zhichun Lu

Make payment

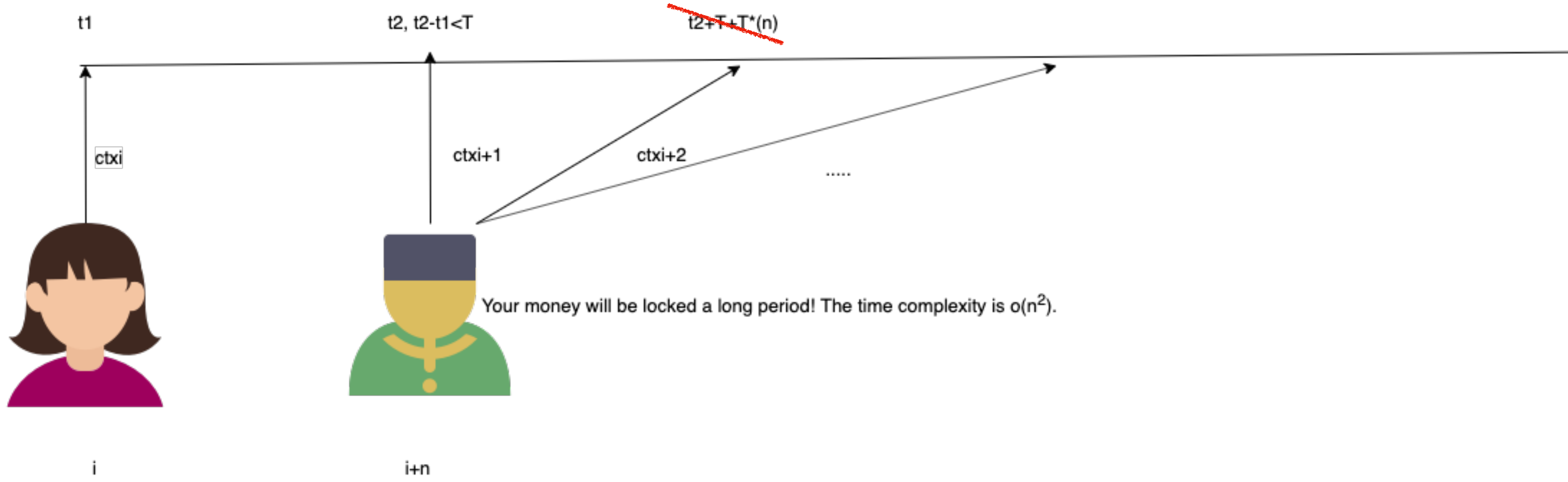
Solution 2


Don't worry, you can submit stx_i after $((i+1-i)+1)*T$.
Here is my ctxi+1, and I will not submit stx_{i+1}!

Zhichun Lu

Make payment

It should be t2+T

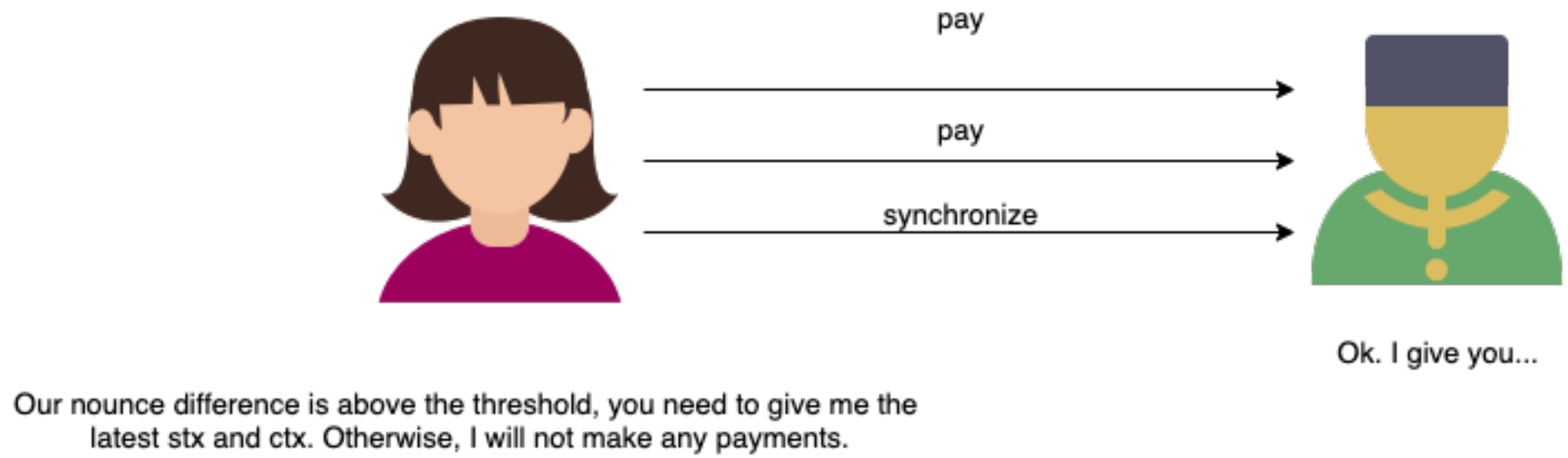


Your money will be locked a long period! The time complexity is o(n^2).

Zhichun Lu

Make payment

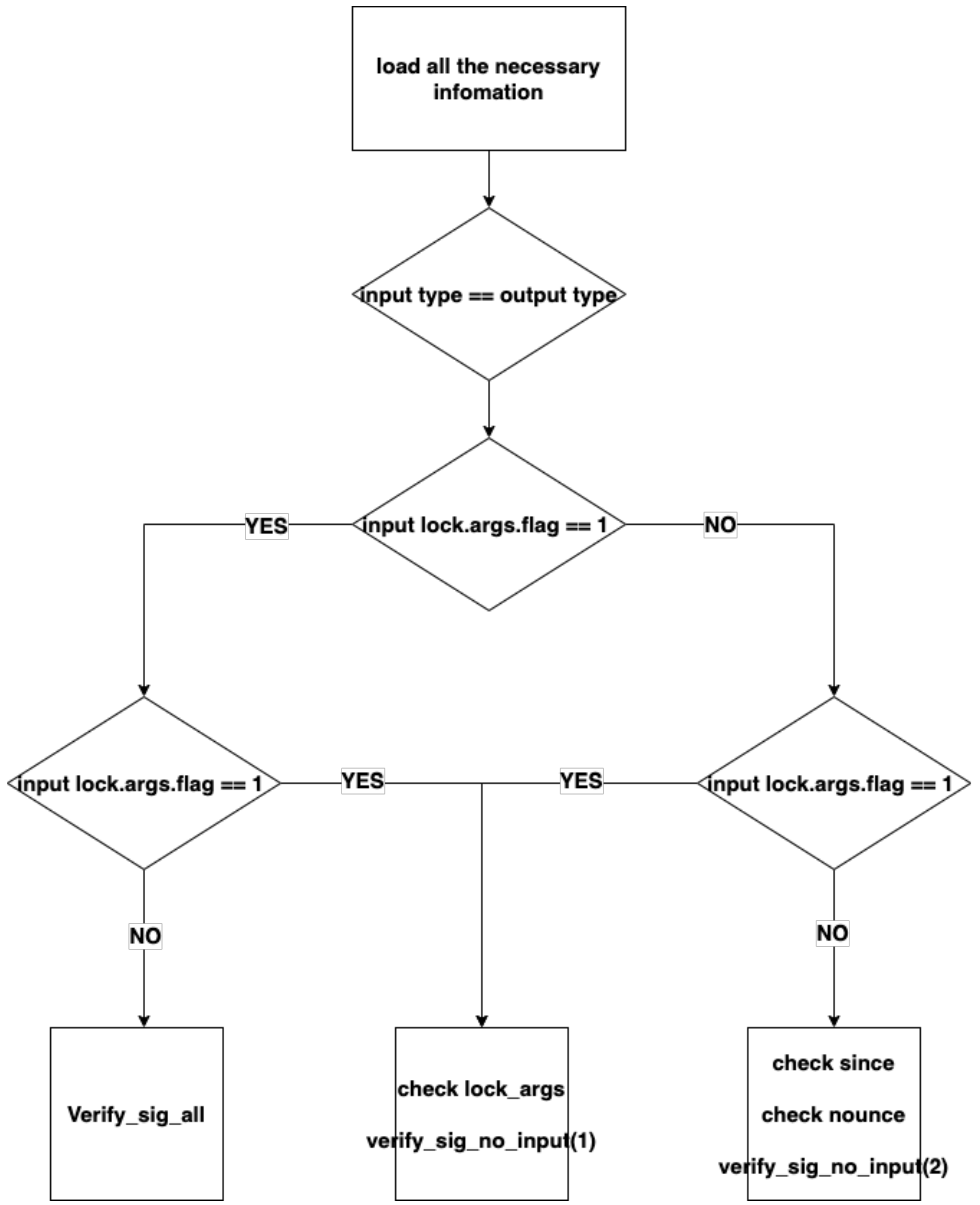
A simple but ugly solution.


pay
pay
synchronize
Ok. I give you...
Our nounce difference is above the threshold, you need to give me the latest stx and ctx. Otherwise, I will not make any payments.

Note: The time complexity is still o(n^2), but n is limited below threshold, if you want to let the complexity to o(n), you need to come up with a design that after time T, the contract will not accept any closing tx.

Zhichun Lu

The contract


load all the necessary infomation
input type == output type
input lock.args.flag == 1
YES
NO
input lock.args.flag == 1
YES
YES
input lock.args.flag == 1
NO
NO
Verify_sig_all
check lock_args
verify_sig_no_input(1)
check since
check nounce
verify_sig_no_input(2)

Zhichun Lu

Discussion

- The lock args and witness need channel id. If there are more than one channel between two users. One can use closing tx in channel 1 to closing channel 2. (state, timeout, pubkey_A, pubkey_B, nounce) (flag, nounce, sig_A, sig_B).
- CKByte has natural advantage being the asset in layer 2. As long as you have ckbyte, you can create channel with anyone with any assets.
- Concurrency: Just using the binary exponential back off alg in tcp.
- About the contract, it will not check the output lock script in bilateral closing and settlement.
- About the testing example….

Zhichun Lu

TODO

- Finish all the red content in the slides.
- The monitor module, and the process of making payments.
- multi_thread.
- UDT version.
- GUI.
- Payment channel network.

Zhichun Lu