Discreet Log Contracts

Invisible bitcoin smart contracts

Thaddeus Dryja <tdryja@media.mit.edu>
MIT DCI

Dev++
2017-11-03

Intro

- I'm Tadge, I work at the Media Lab Digital Currency Initiative (nearby!)
- Working on lightning network software

github.com/mit-dci/lit

- Also other fun bitcoin stuff. Like this!

Intro

- Discreet Log contracts are pretty new. This whole thing might not work! (I think it does though)
- Smart contracts using bitcoin, similar to LN
- Also nobody can see the contracts

Discreet: unobtrusive, unnoticeable

Discrete: consisting of distinct or unconnected elements

Discrete log problem: math bitcoin signatures are based on

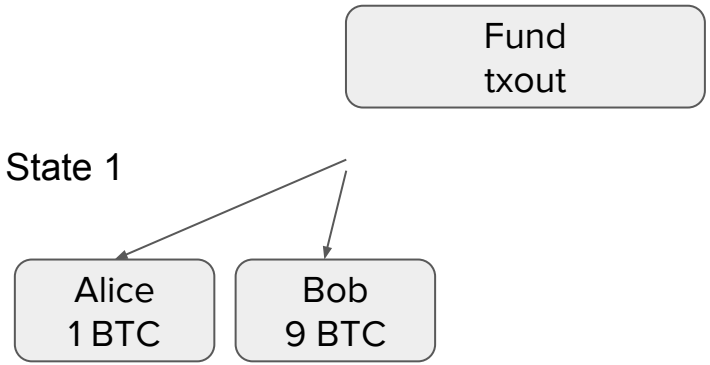
Outline

- LN recap
- Elliptic curves and combining keys
- Smart contracts and oracles
- Schnorr signatures, anticipated signatures
- Discreet log contracts
- Scalability
- Privacy
- Uses, questions &c

Recap of Lightning Network

- Set up a channel, 2 of 2 multisig
- Interactively make lots of transactions spending that output
- Only the most recent transaction is valid
- Global network has no idea which is most recent, but channel participants do

Lightning Network Payment Channel

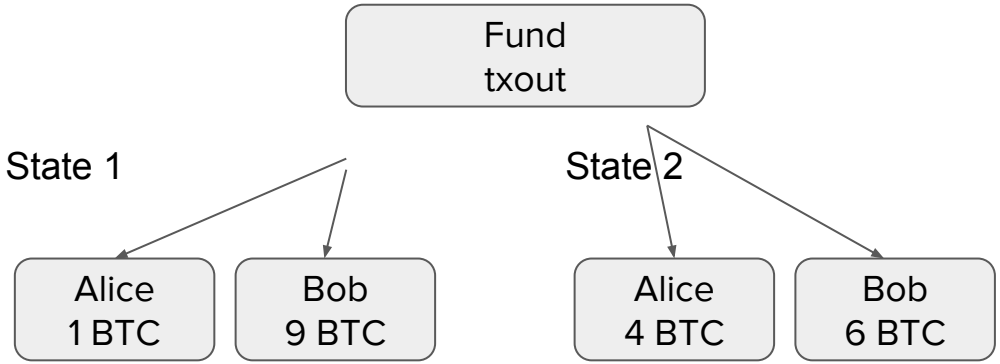


Bob funds a channel and broadcasts the fund tx to the blockchain

Alice and Bob together create transaction 1, which sends 1 coin to Alice and 9 coins to Bob.

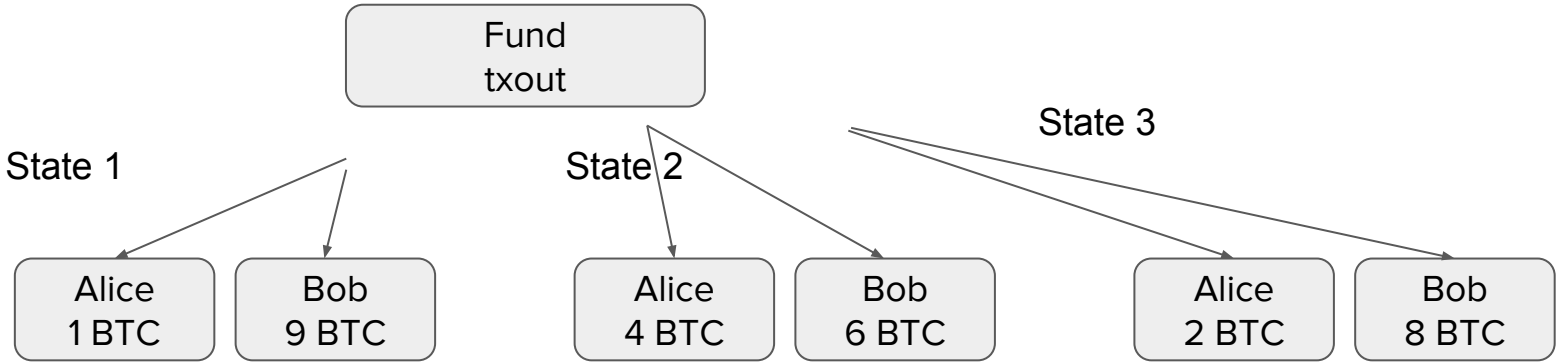
This transaction is *not* broadcast

Lightning Network Payment Channel



They can make new states with different amounts; here Alice gets 4 and Bob gets 6

Lightning Network Payment Channel



Or Alice:2 and Bob:8

Output script

```
pubX OR (pubY AND Time)
```

In Lightning, PubY is “correct”, and pubX is only used in case of fraud

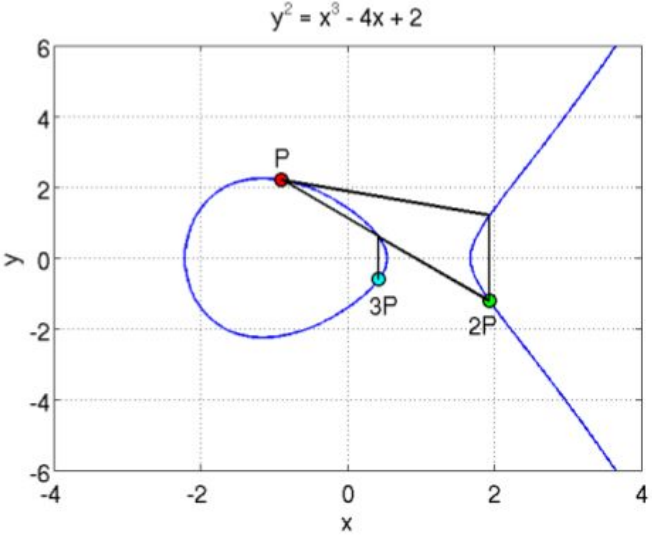
pubX is the combination of both participants keys

Elliptic curve usage

Points on a curve

You can add them!

You can’t multiply them!

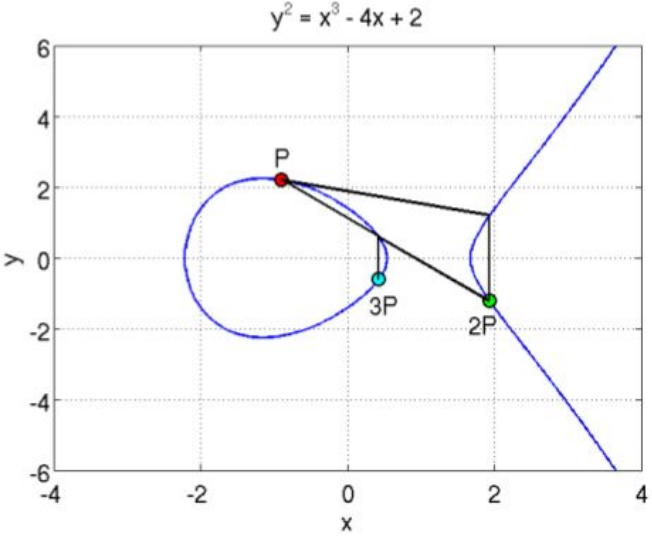


You can multiply them by a natural coefficient (by adding them a bunch)

Elliptic curve usage

a (scalar)

A (point)



a+b a-b a*b a/b (everything OK)

A+A A-B A*B A/B (add/sub OK, no mult)

A+b A-b A*b A/b (can mult scalar&point)

Elliptic curve usage

point G "Generator"

just some point we agree on

private keys are random a

public keys are a*G = A

Elliptic curve homomorphism

(aG) + (bG) = (a+b)G

sum of private keys gives sum of public keys! fun stuff ensues

Revocable key

aG = A, bG = B

A+B = C = (a+b)G

Alice knows a, Bob knows b. Neither can sign with C.

Bob can give b to Alice, then Alice can sign with C.

Output script

```
     pubX OR (pubY AND Time)
pubX = aG + bG
```

Alice gives her part of the private key to Bob, to revoke her claim on the tx

Conditional payments

- This is a smart contract: payment conditional on some external data
- In this example, Alice and Bob bet on tomorrow's weather. If it rains, Alice gets 1 BTC. If it's sunny, Bob gets 1 BTC.
- One problem: The bitcoin blockchain is not aware of the weather. (OP_WEATHER has not yet been soft-forked in)

"Smart contracts" and oracles

- LN is a simple script, enforcing the most recent tx
- Made of smart contracts, but has no external state. Everything comes from Alice & Bob
- If we want external state, need some way to get it, usually called an "oracle"
- Simple oracle: 2 of 3 multisig

Why oracles?

- 2 of 2 multisig means conflict freezes funds
- Rich players at an advantage (lower time value of money)
- Works great with friends, but bitcoin is the currency of enemies :)
- A 3rd party can decide in case of conflict
- 2of3multisig oracle

2 of 3 multisig oracle

- 3 keys: Alice, Bob, Olivia
- If Alice and Bob are chill, they can both sign without contacting Olivia
- If Alice and Bob fight or are unresponsive, one of them can ask Olivia to sign
- Problem: It's sunny. Alice tells Olivia, "Hey, Alice. Say it's raining and I'll give you 0.8"

Interactive oracle

- 2 of 3 multisig oracles are **interactive**
- Not only do they **see** every contract, they **decide** the outcome of every contract, individually. (Can equivocate)
- It'd be better if the oracle couldn't equivocate, and even better if they never saw the contracts. But how?

Schnorr signatures

- a (scalar) A (point)
- make a keypair: a <- $ (random)
- A = aG
- h() is a hash function
- m is some message

Schnorr signature

```
aG = A public key
k <- $; R = kG (nonce for signature)
to sign, compute s = k - h(m, R)a
signature is (R, s)
To verify  sG =? kG - h(m, R)aG
              =? R  - h(m, R)A
```

Fixed-R Schnorr signature

Pubkey A signature: (R, s)

DLC:

Pubkey (A, R) signature: s

Same thing right? But can only sign once!

k-collision

Signature 1 $s_1 = k - h(m_1,R)a$

Signature 2 $s_2 = k - h(m_2,R)a$

$s_1 - s_2 = k - h(m_1,R)a - k + h(m_2,R)a$

$= h(m_2,R)a - h(m_1,R)a$

$= (h(m_2,R) - h(m_1,R))a$

$a = (s_1 - s_2) / (h(m_2,R) - h(m_1,R))$

Fun fact: this is what brought down Playstation 3 code signing

Anticipated Signature

Given 'pubkey' (A, R) and a message m, you can't compute s.

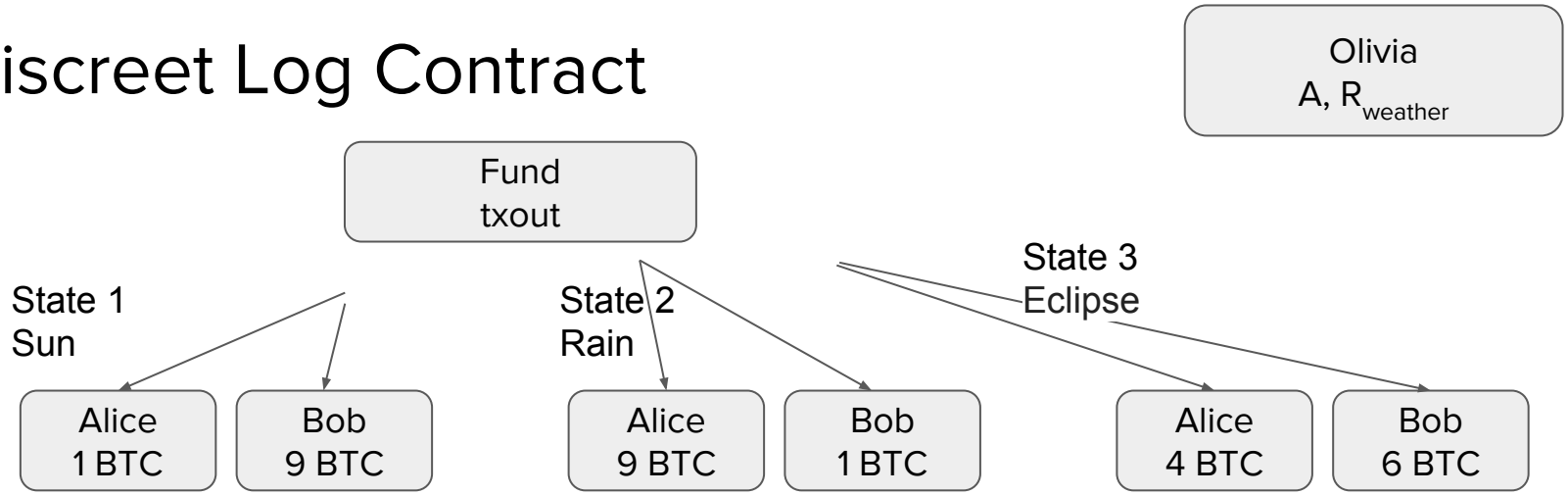
but you **can** compute sG = R - h(m,R)A

sG is computable for any message!

But you can't get s.

(EC Discrete log problem)

Discreet Log Contract



Alice & Bob build a contract
Looks like LN, but instead of making outputs sequentially, they make them all at once.
Instead of 'most recent' determining validity, Olivia's signature determines validity.
Olivia can't see the contract, (it's unbroadcast) and wouldn't recognize her part of the keys even if she could.

Signatures as private keys

- It's an unknown scalar, but you know what it is times the generator point. Hmm!
- Seems a lot like a keypair!
- Use Olivia's signature s as the private key
- sG is the public key

Signatures as private keys

Olivia's s as private key

sG as public key

Mix with Alice and Bob's public keys

$$pub_{alice} + sG = pub_{contract}$$
$$priv_{alice} + s = priv_{contract}$$

Example

3 possibilities: m_{sun} m_{rain} m_{ecl}

3 sigKeys: $s_{sun}G = R - h(m_{sun},R)A$

$AlicePub_{sun} = AlicePub + s_{sun}G$

$BobPub_{sun} = BobPub + s_{sun}G$

$AlicePub_{rain} = AlicePub + s_{rain}G$

$BobPub_{rain} = BobPub + s_{rain}G$

Same script as LN

PubR OR (PubT AND time)

In lightning, The "correct" use is the timeout, op_csv

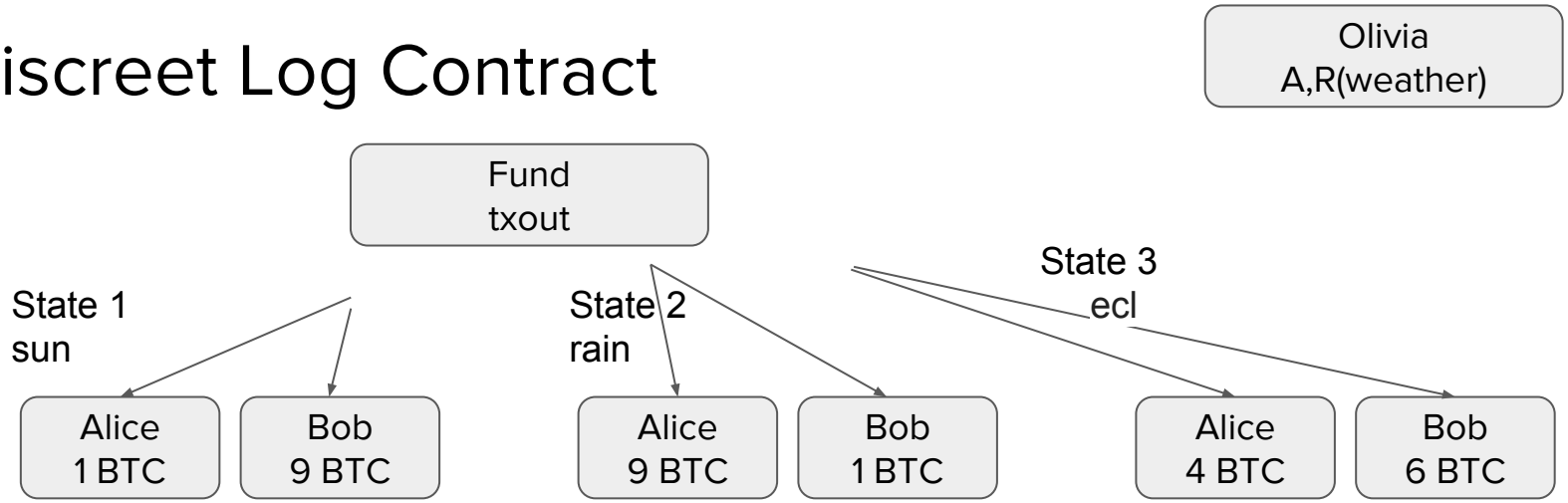
In cases of fraud, the revocable key can be used (half the key revealed)

OP_IF PubR OP_ELSE delay OP_CSV OP_DROP PubT OP_ENDIF OP_CHECKSIG

2 keys in Lightning and DLC

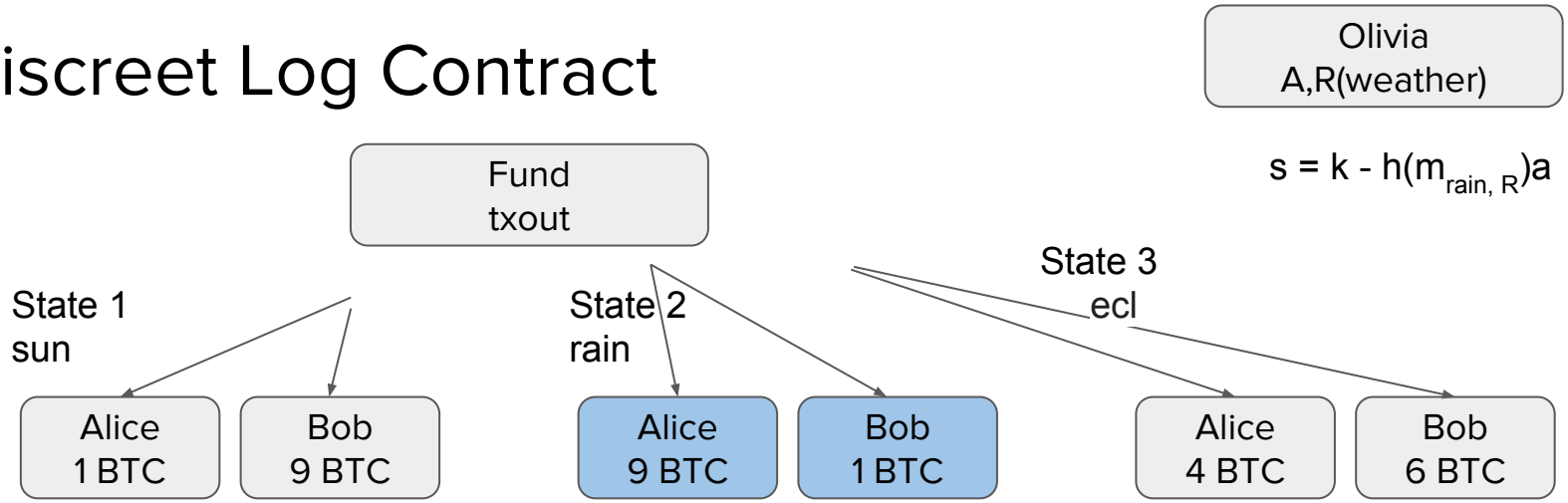
```
PubX OR (PubY AND time)
PubY is "pure",PubX is sum of 2 parts

Lightning: PubY correct PubX fraud
DLC:       PubY fraud   PubX correct
```

Discreet Log Contract



It rained. Olivia signs the message "rain"

Discreet Log Contract

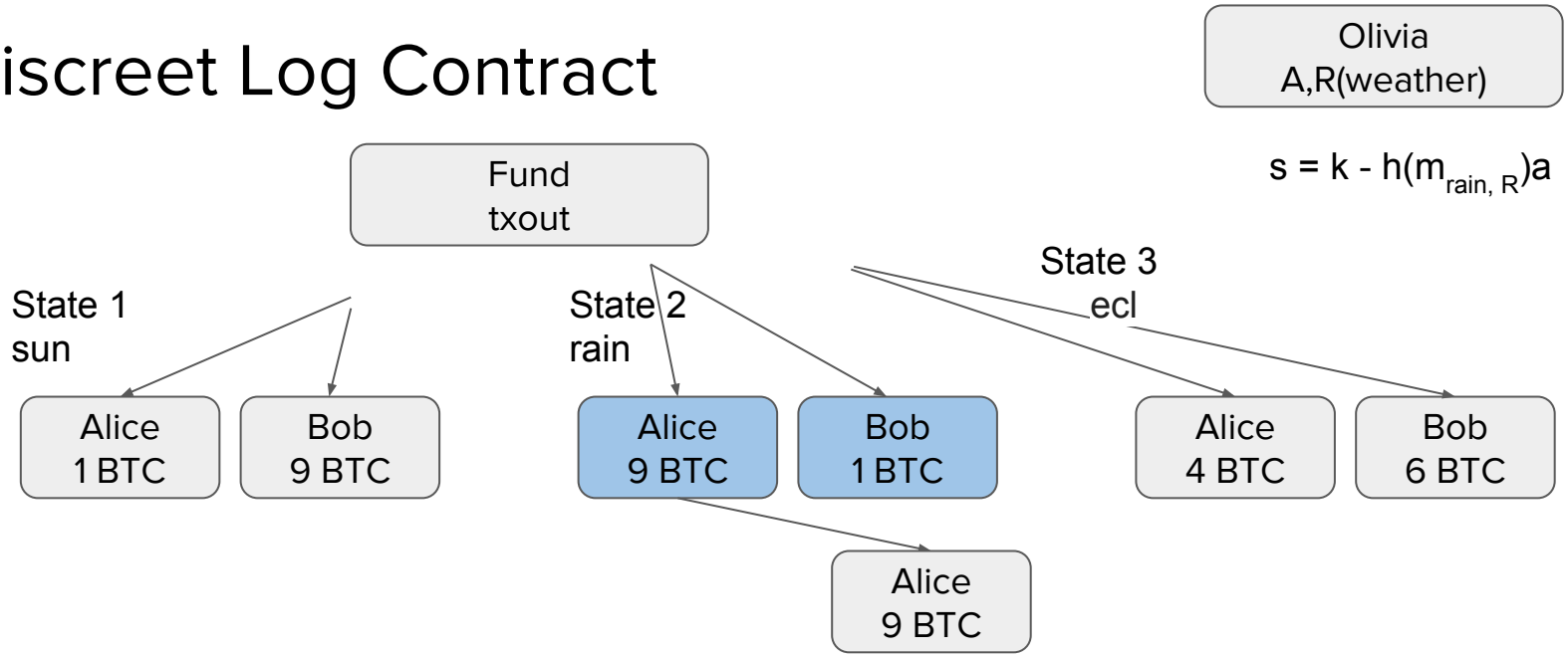


Olivia's signature is s_{rain} which is a partial private key

State 2 is the correct state

Alice (or Bob) should broadcast state 2

Discreet Log Contract



Alice knows the private key to spend her blue output.
It's the sum Alice's own private key, plus s_{rain}.

Alice makes a transaction sending the 9 coins to herself immediately after broadcasting state 2.
If she doesn't Bob could grab those 9 coins after the time has passed

Time and DLCs

In LN, you need to always watch for fraud, as old states could be broadcast. Gotta grab that output.

In DLC, you sweep the output as soon as you make it. Easier, and have the software broadcast both txs at the same time. No surprises.

Evil Olivia

A bad Oracle **can** cause contracts to execute the wrong way!

But all contracts must execute the same way; Olivia can't sign both sun and rain.

An incorrect signature is public.

Olivia doesn't know about the contract

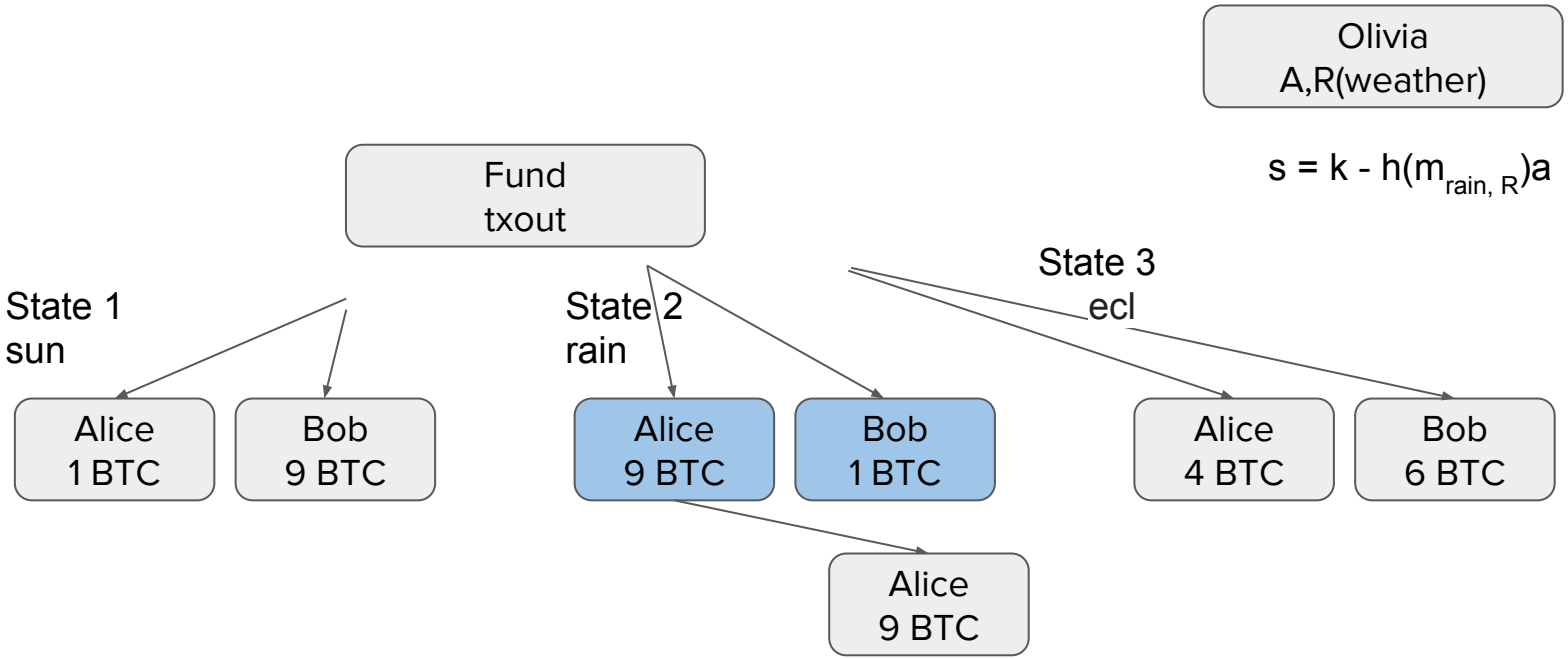
Scalability of DLC

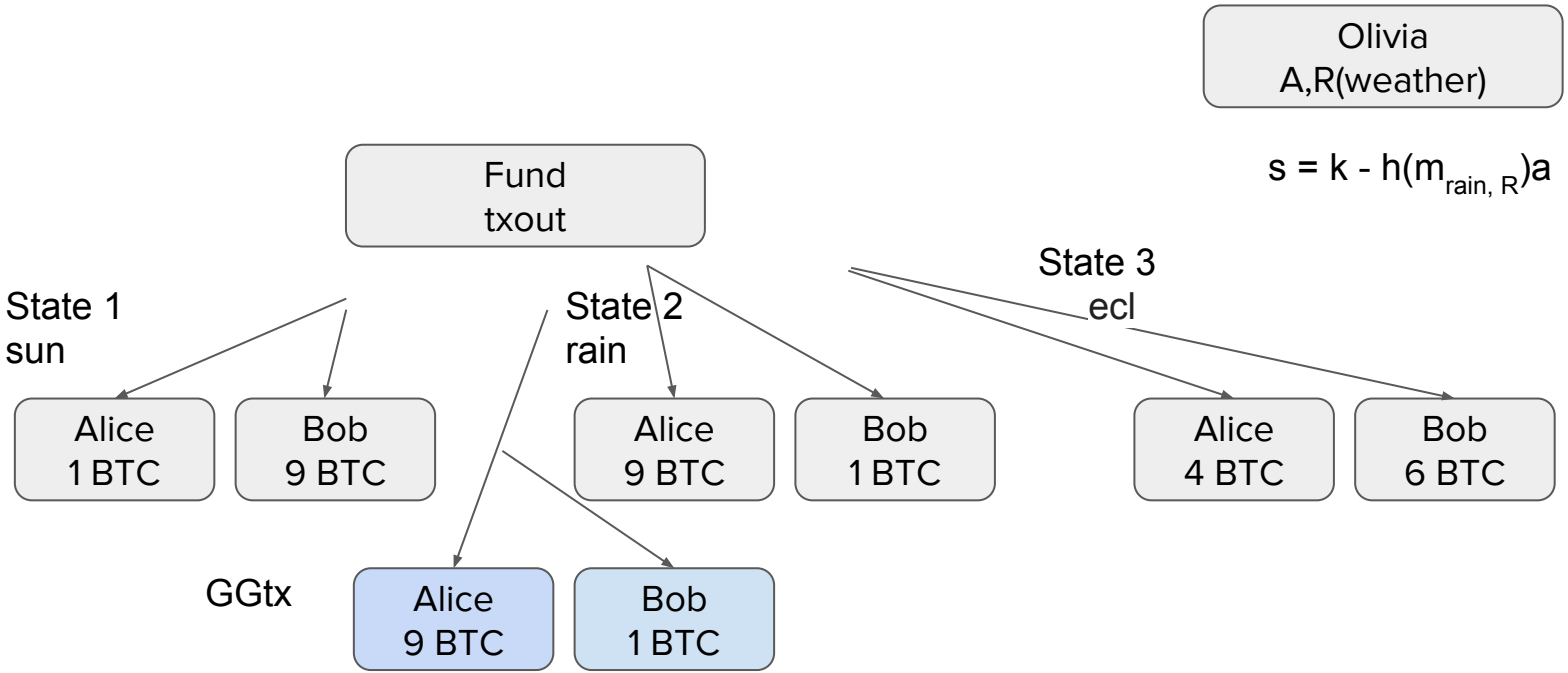
Whole process is 3 txs:

fund, close, sweep

if the parties are chill, can reduce to 2 txs:

fund, gg


Olivia
A,R(weather)
s = k - h(m rain, R)a
Fund
txout
State 1
sun
State 2
rain
State 3
ecl
Alice
1 BTC
Bob
9 BTC
Alice
9 BTC
Bob
1 BTC
Alice
4 BTC
Bob
6 BTC
Alice
9 BTC

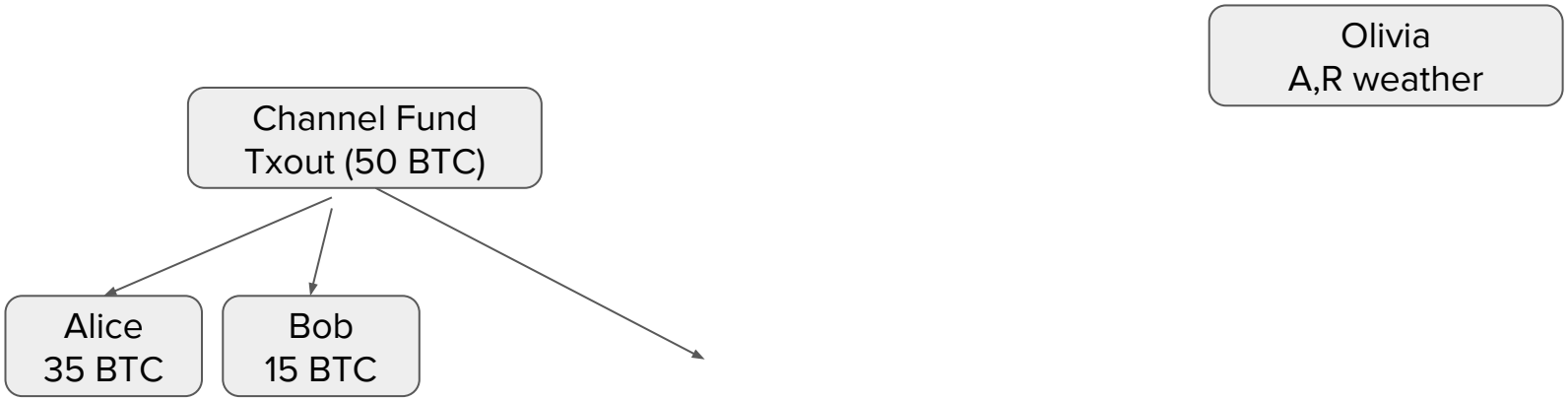


GG tx: if everyone agrees, create a new transaction at closing time which sends to unencumbered outputs

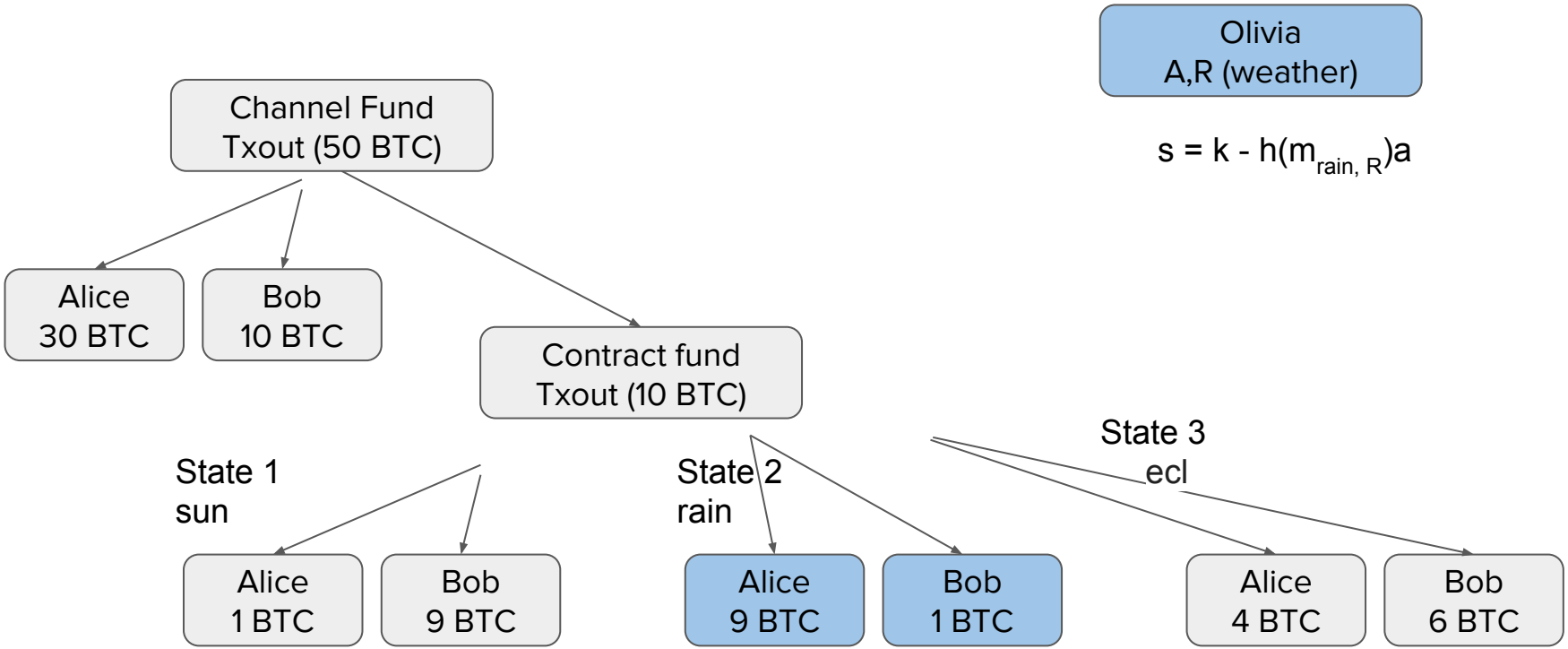
DLCs within channels

Make a DLC output from an LN channel

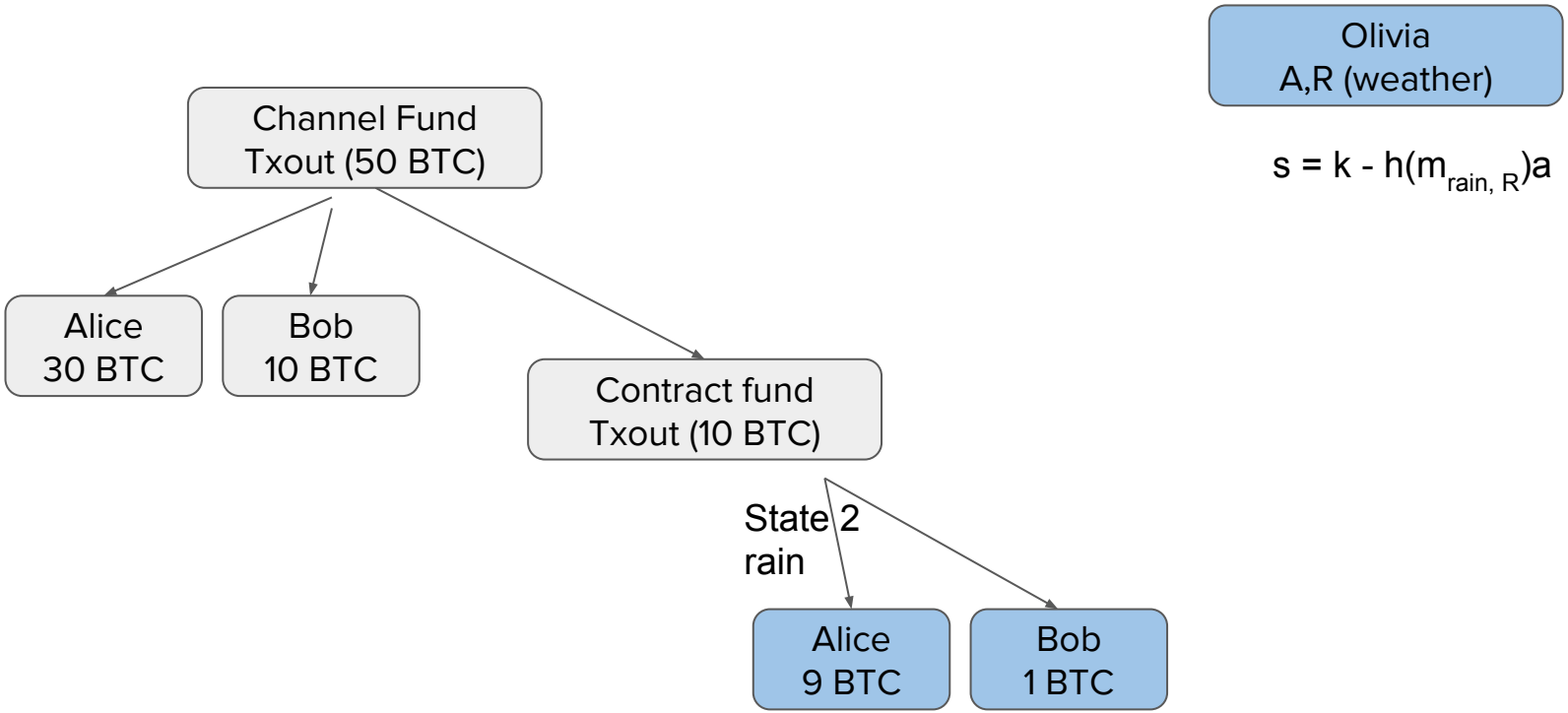
If parties cooperate, 0 txs get broadcast to the blockchain



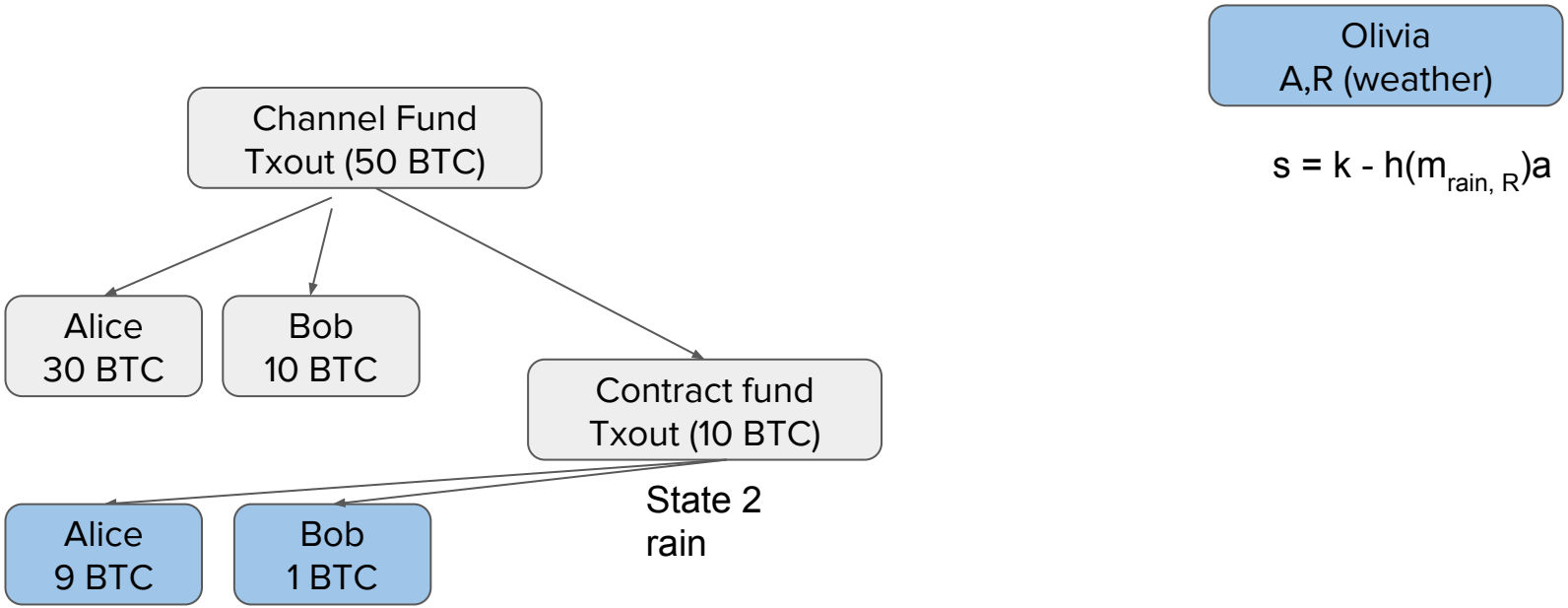
Alice & Bob have a normal LN channel



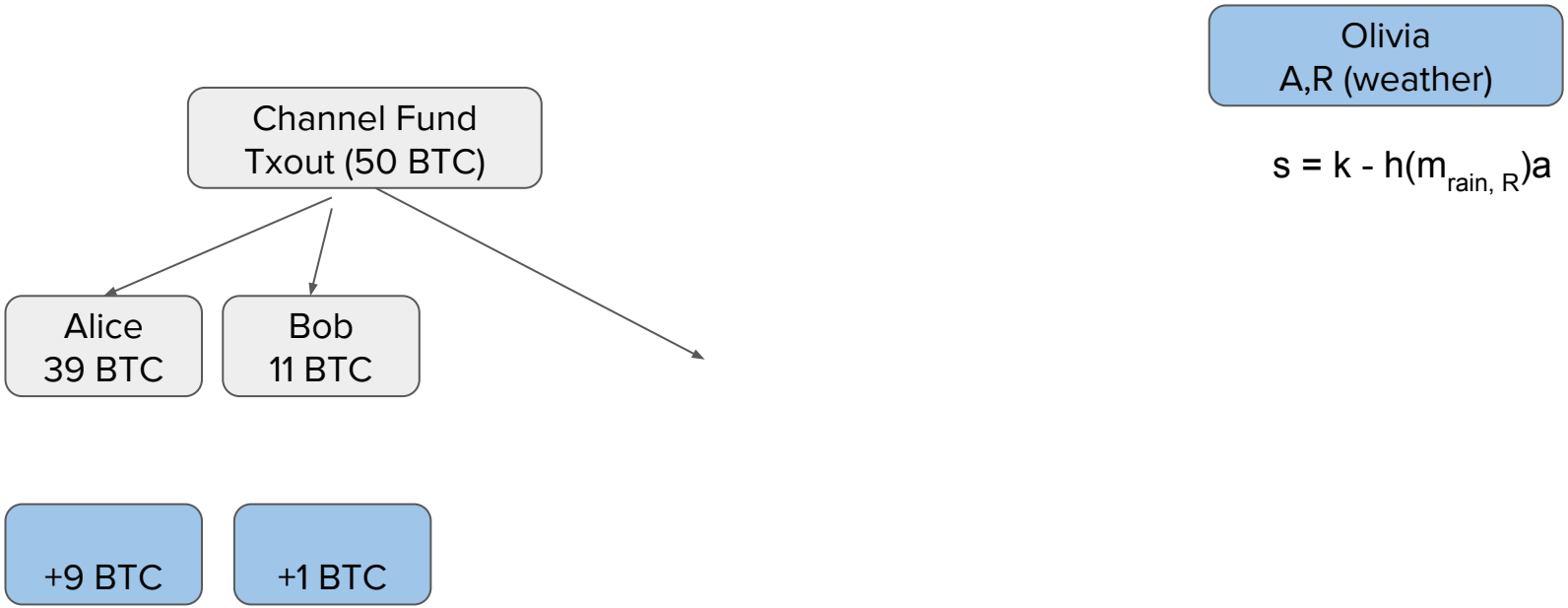
With Olivia's s_{rain} Alice can close both the channel, and the contract.
(some delays are required)



With Olivia's s_{rain} Alice can close both the channel, and the contract.
(some delays are required)



With Olivia's s_{rain} Alice can close both the channel, and the contract.
(some delays are required)



If they cooperate, they can update the channel balances to reflect the difference from the contract execution, and remove the contract output.
The channel can keep going and 0 txs go on the blockchain

How discreet are the contracts

For in-channel contracts, nobody sees it but the counterparties.

If broadcast to the network, it's still not clear that it was a contract. The oracle's sG pubkey is not detectable or decidable.

Weather is great and all but...

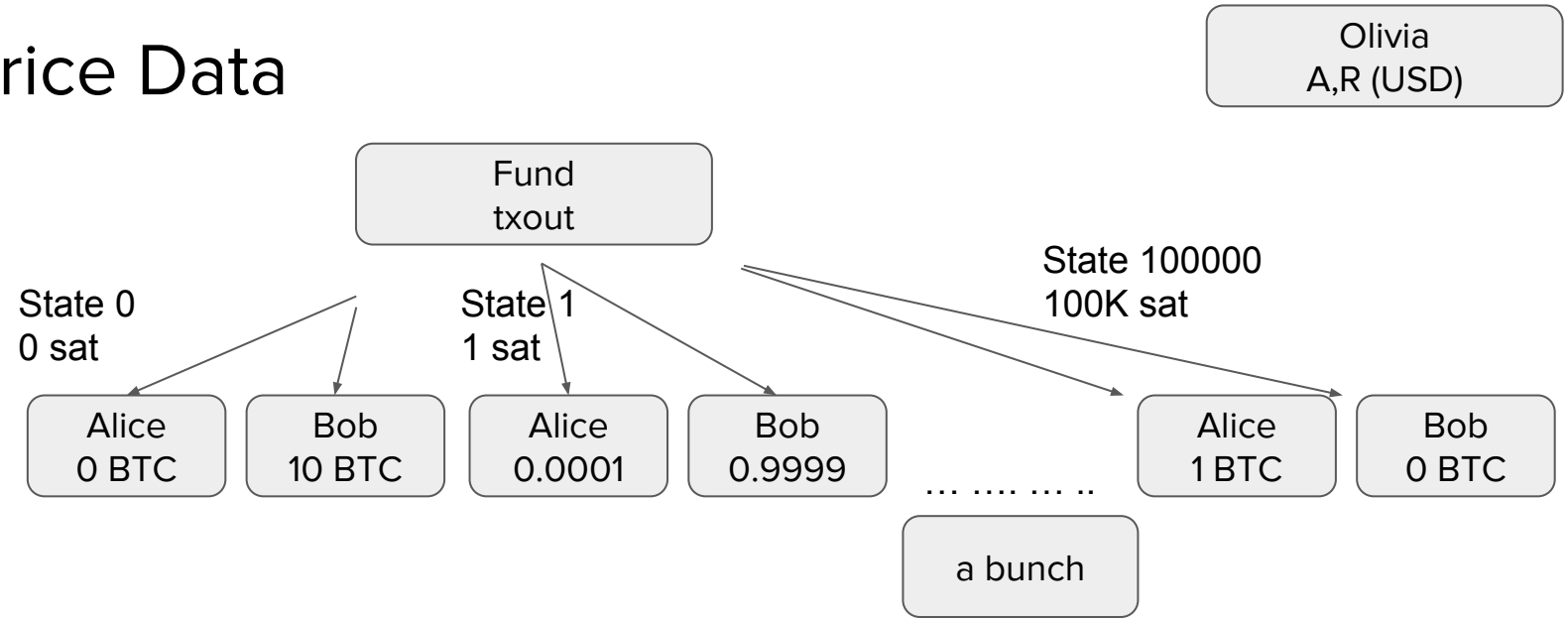
There are contracts with more than 2 or 3 possible outcomes. Like prices.

use m = price(in satoshis)

1 USD = 25K sat

make thousands of txs

Price Data



Make thousands of txs for all the possible prices

1 tx is around 100 bytes
100K transactions would be around 10MB

Off-chain scalability

Can split the R value (and message) in to a R-exponent and R-mantissa

Helps cut down the off-chain transactions needed in ranges which don't lead to different allocations

MultiOracle

Maybe Alice and Bob want to use 2 oracles. No problem.

$s_aG + s_bG = s_cG$

Just add the sG points. n of n, no size increase. (n of m, size blowup)

Novation

Alice is in a DLC with Bob.

Contract ends next week.

Alice wants out now.

If Bob is offline, we're stuck.

If Bob's online, some options

Novation

Alice: Hey Bob I want to [take profit / stop losses].

Bob: Sure, I'm out too.

problem: interactive, unlikely

Novation

Alice: Hey Bob I changed my pubkey

Bob: Ok whatever.

problem: interactive in the computer sense, not in the human sense.

Alice needs to find Carol.

Novation

Alice & Bob build new contract with Carols keys as payout.

Alice shows contract (tx set) to Carol, she signs all of them.

All 3 need to sign, but Bob's can be automated.

DLC use cases

Weather? Currency futures? Stocks? Commodities? Sports? Insurance?

Pretty general; conditional payments based on any number or element from predetermined set.

No token needed. No ICO. Sorry. Not sorry.

Disctreet log contracts

Questions

Thanks for coming!