Discreet Log Contracts

Invisible bitcoin smart contracts

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

Scaling Bitcoin Stanford
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

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

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 do we need 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

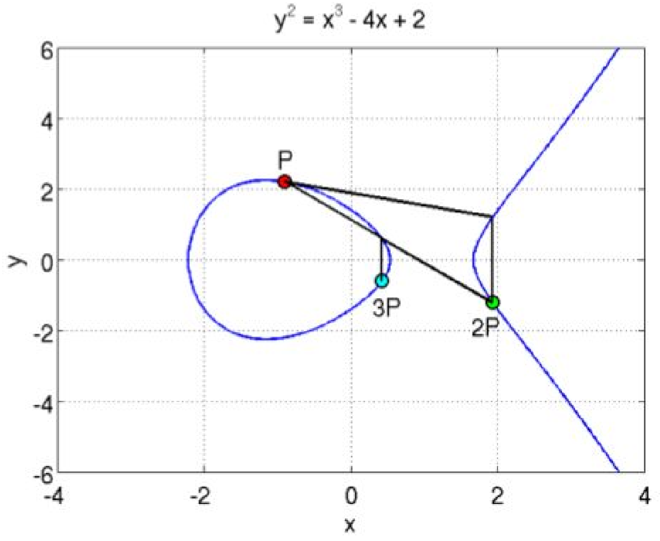
2 of 3 multisig oracle

- 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?

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 homomorphism

```
(aG) + (bG) = (a+b)G
sum of private keys gives sum of
public keys! fun stuff ensues
```

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)
```

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!
- Think of Olivia's signature s as a private key
- sG is a public key

Signatures as private keys

```
Olivia's s as private key
sG as public key
Add to Alice and Bob's public keys
```

$$\mathrm{pub}_{\mathrm{alice}} + sG = \mathrm{pub}_{\mathrm{contract}}$$

$$\mathrm{priv}_{\mathrm{alice}} + s = \mathrm{priv}_{\mathrm{contract}}$$

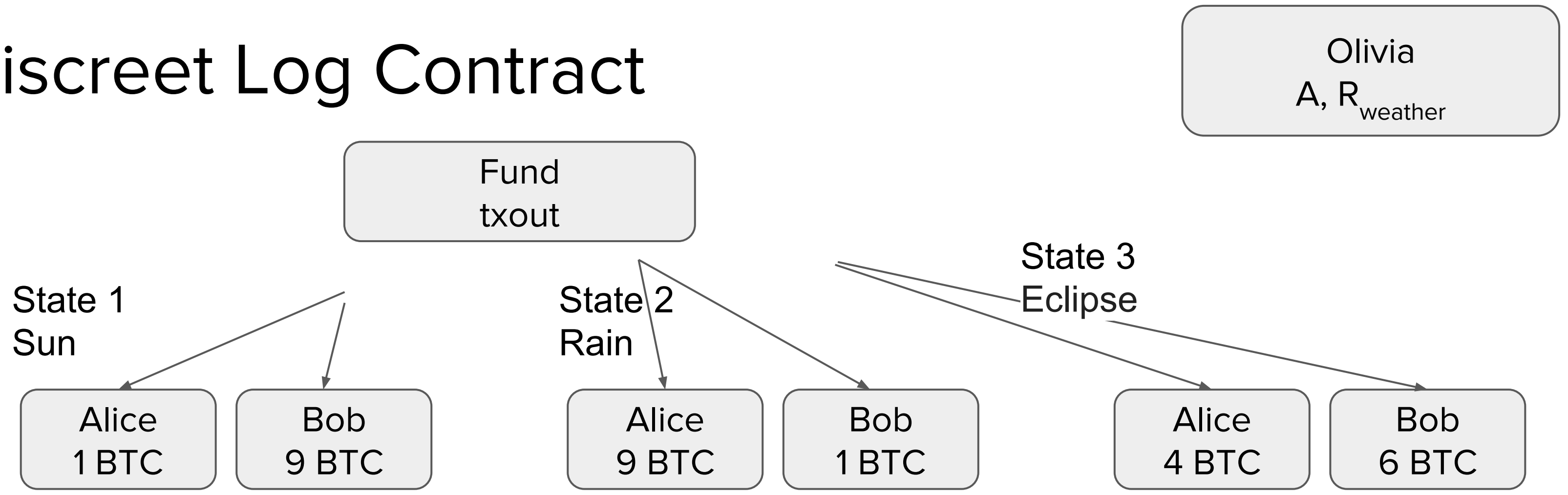
Output script

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

```
In Lightning, PubY is "correct", and
pubX is only used in case of fraud
```

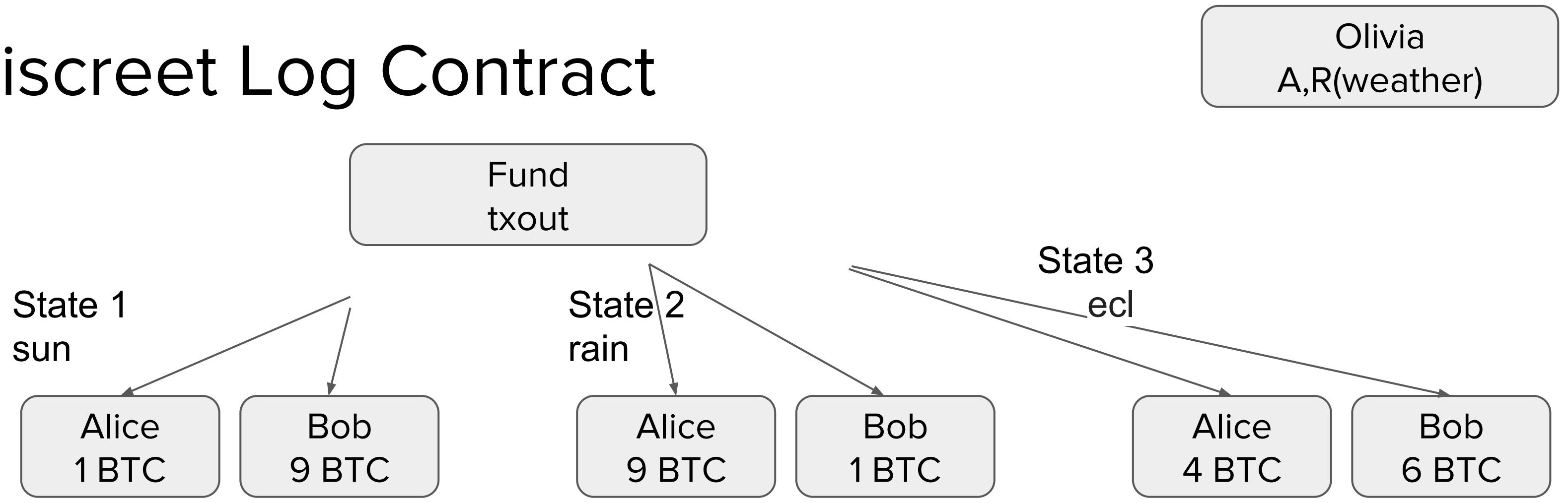
```
In DLC, PubY is "wrong", PubX is the
signature based key and "correct"
```

Discreet Log Contract



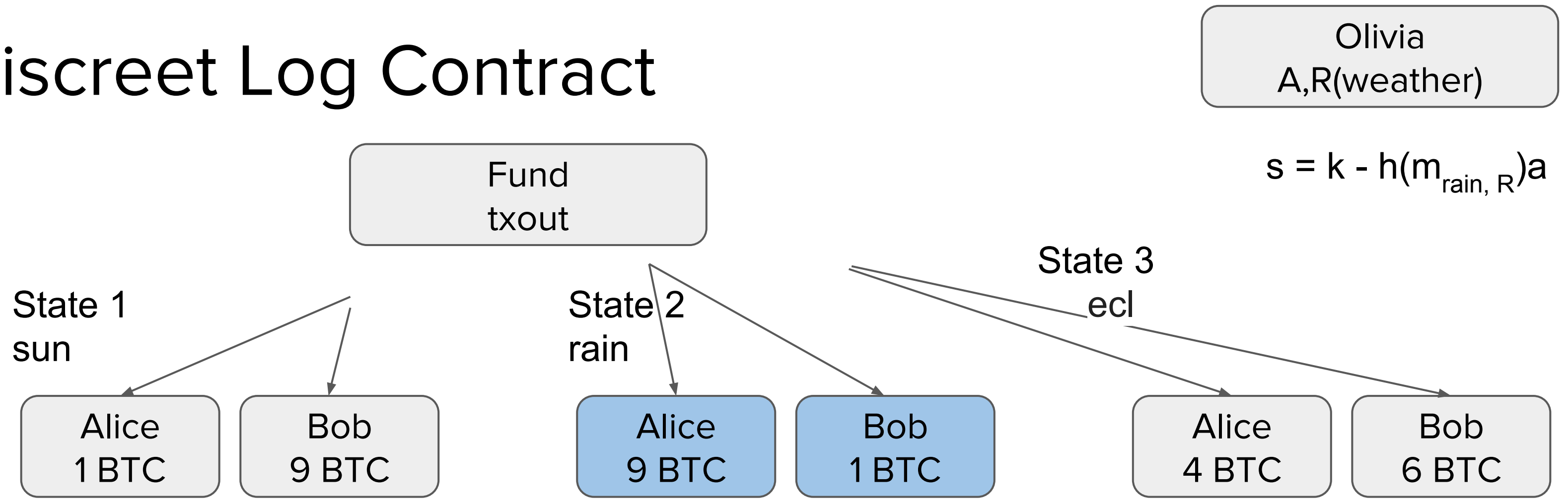
Alice & Bob build a contract
Looks like LN, but instead of making outputs sequentially, they make them all at once.
Instead of 'most recent' determining validity, Olivia's signature determines validity.
Olivia can't see the contract, (it's unbroadcast) and wouldn't recognize her part of the keys even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

Discreet Log Contract



It rained. Olivia signs the message "rain"

Discreet Log Contract

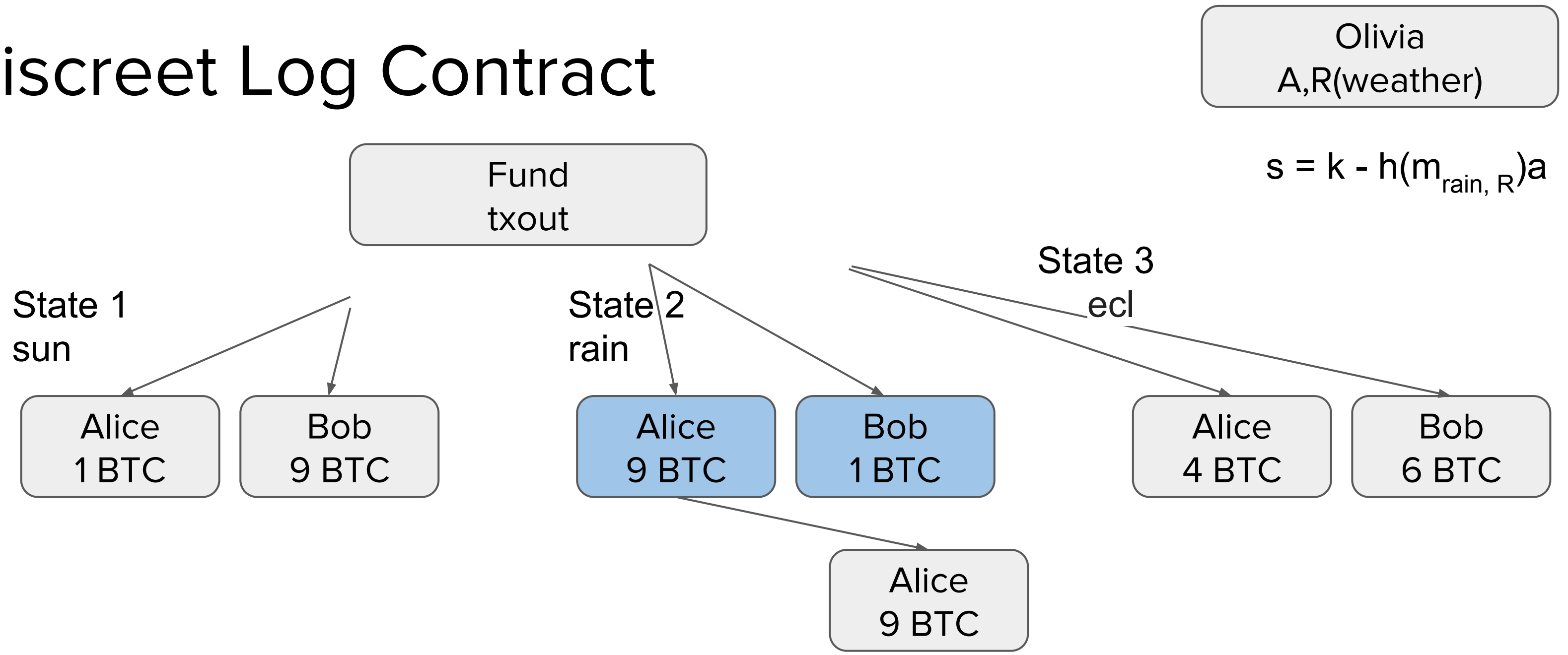


Olivia's signature is s_{rain} which is a partial private key

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.

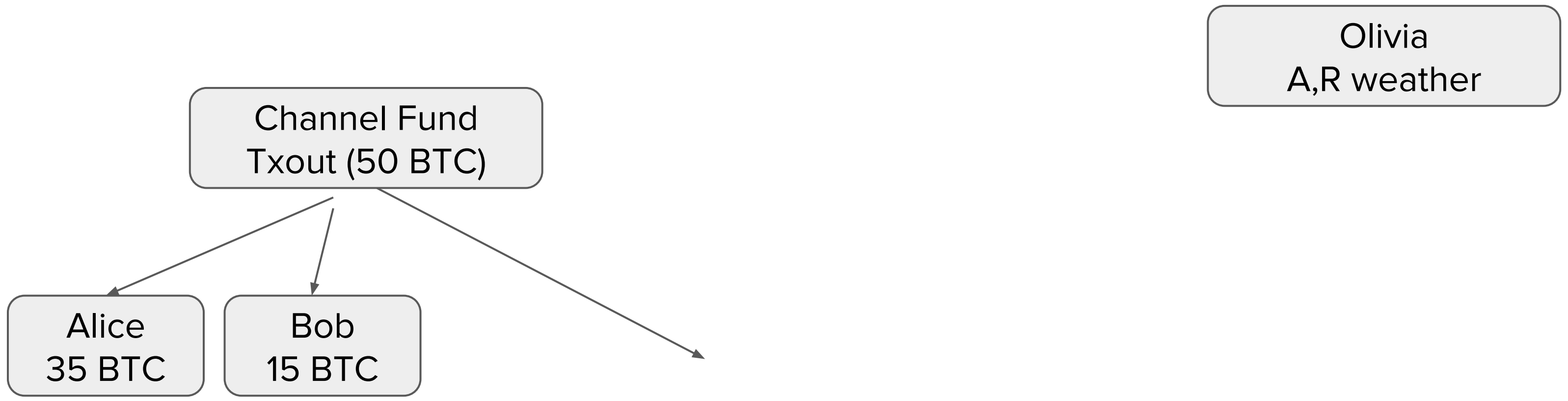
An incorrect signature is public.

Olivia doesn't know about the contract

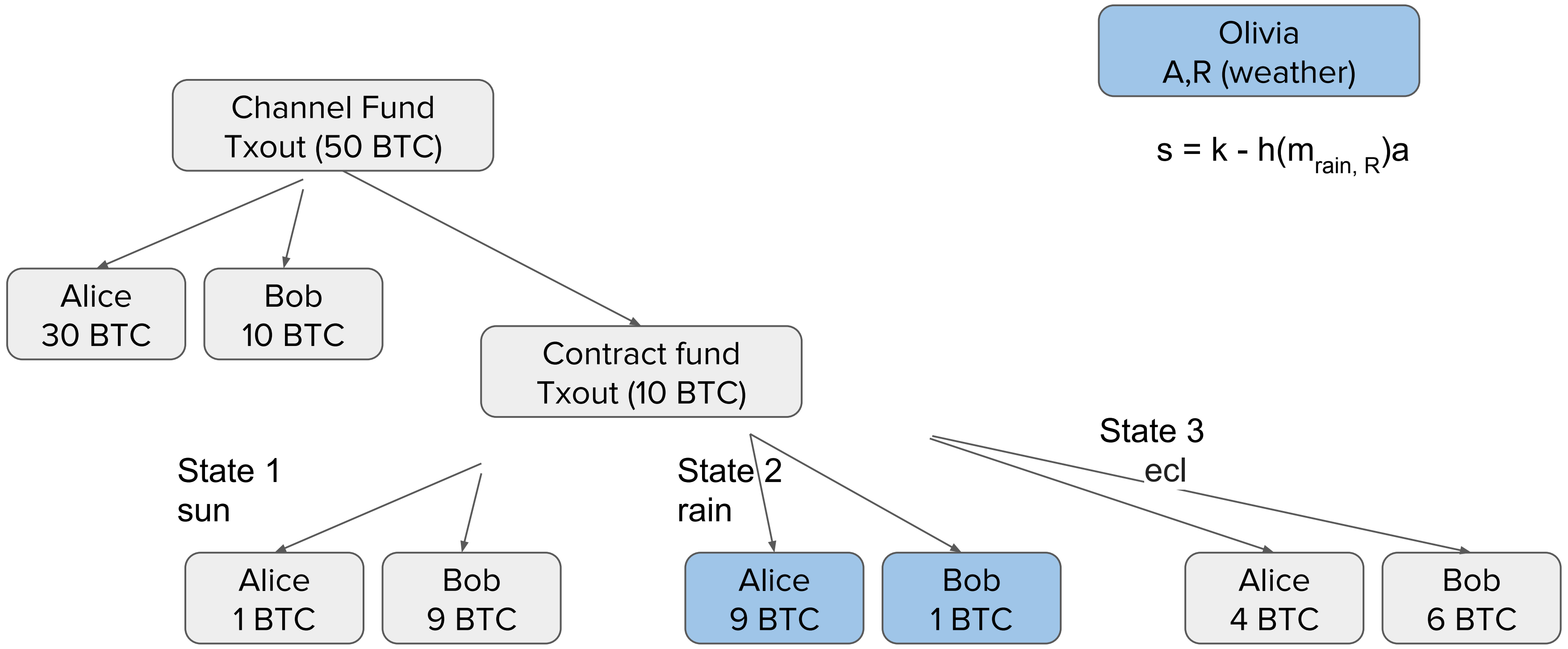
DLCs within channels

Make a DLC output from an LN channel

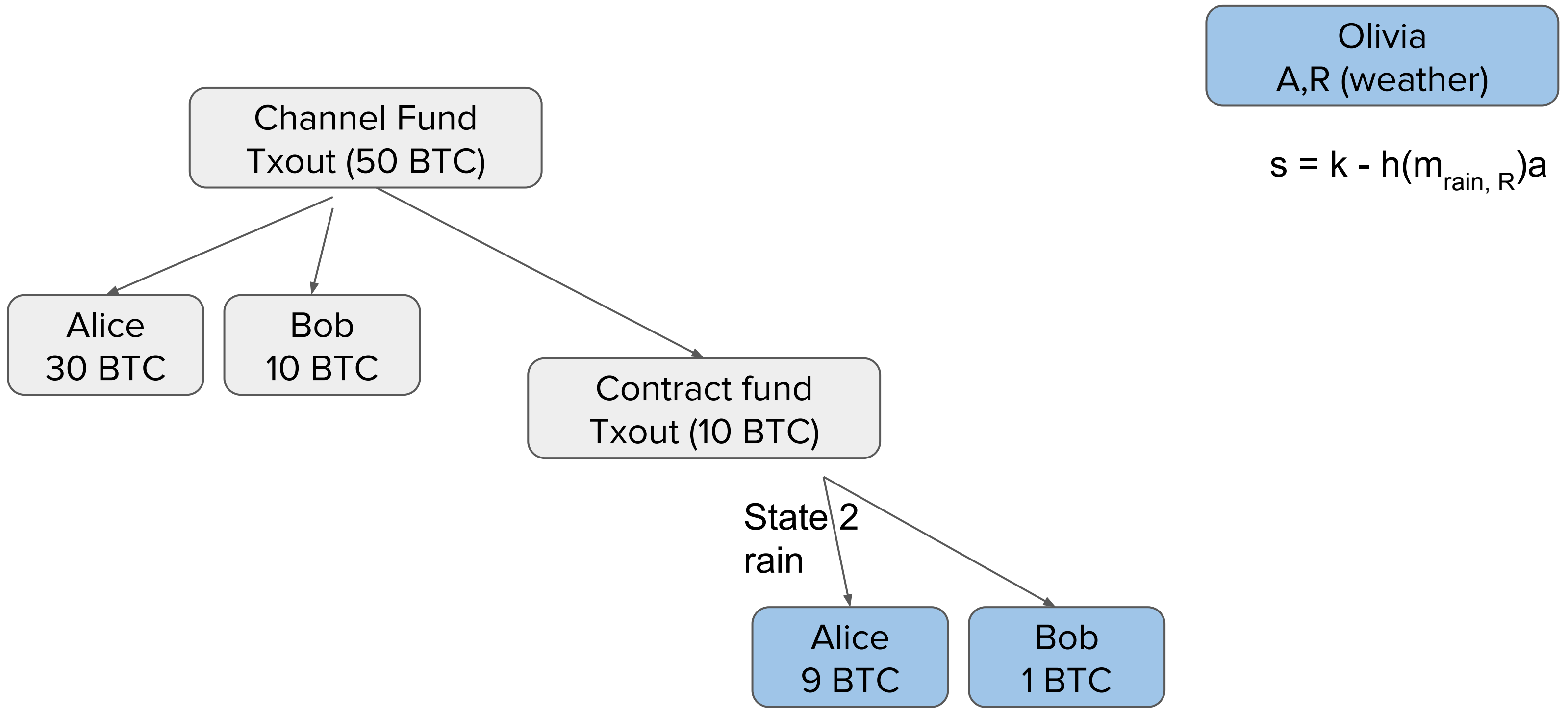
If parties cooperate, 0 txs get broadcast to the blockchain



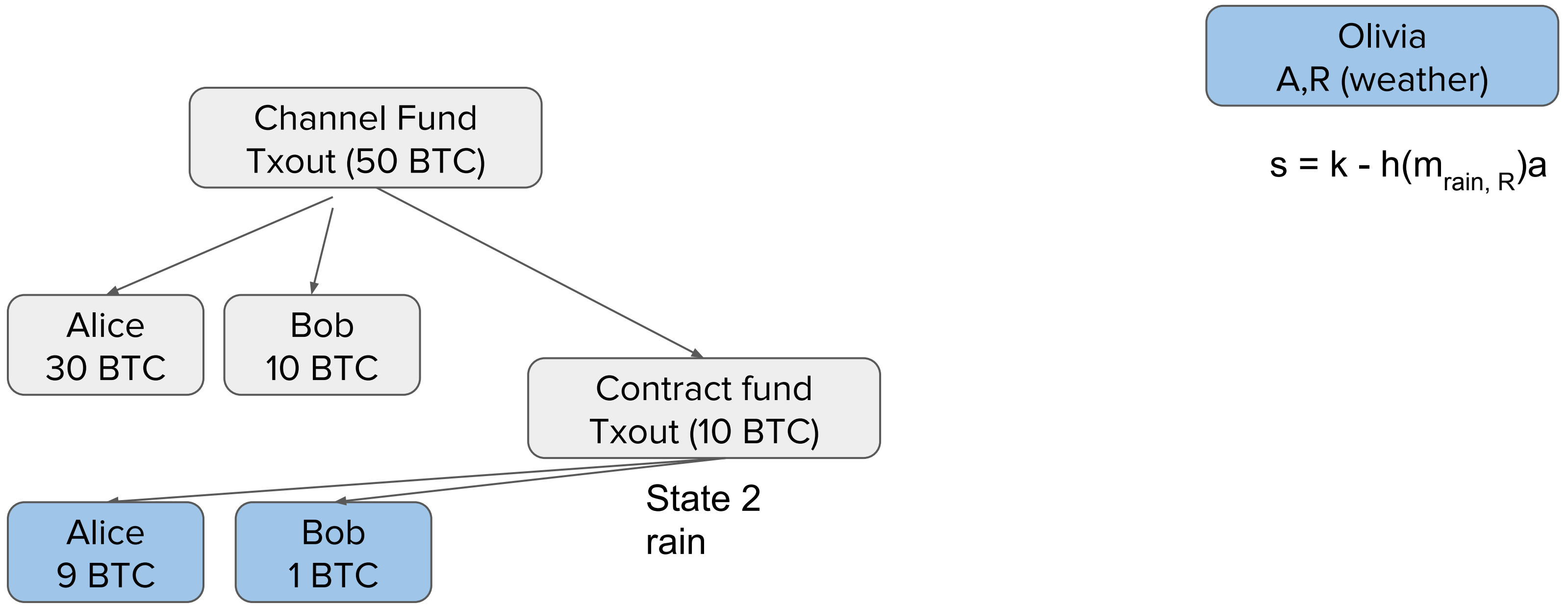
Alice & Bob have a normal LN channel



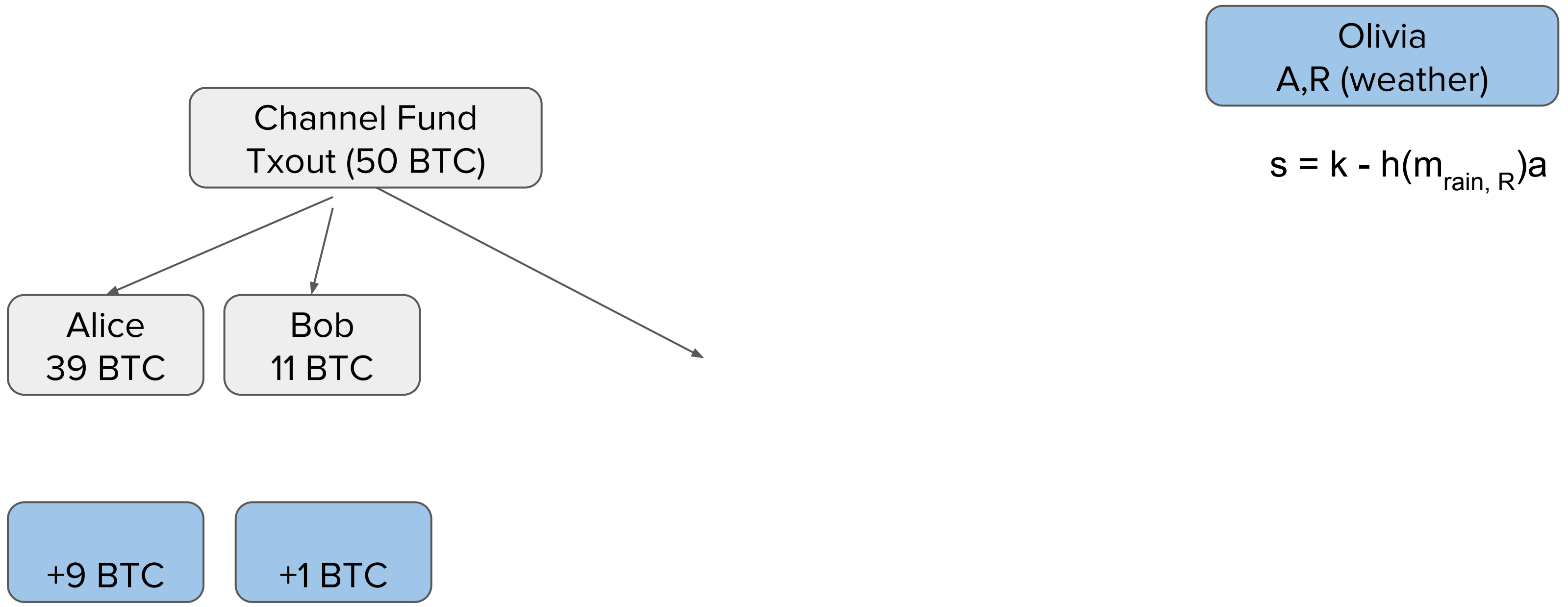
With Olivia's s_{rain} Alice can close both the channel, and the contract.
(some delays are required)



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)



$s = k - h(m_{rain,\ R})a$

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 these contracts

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

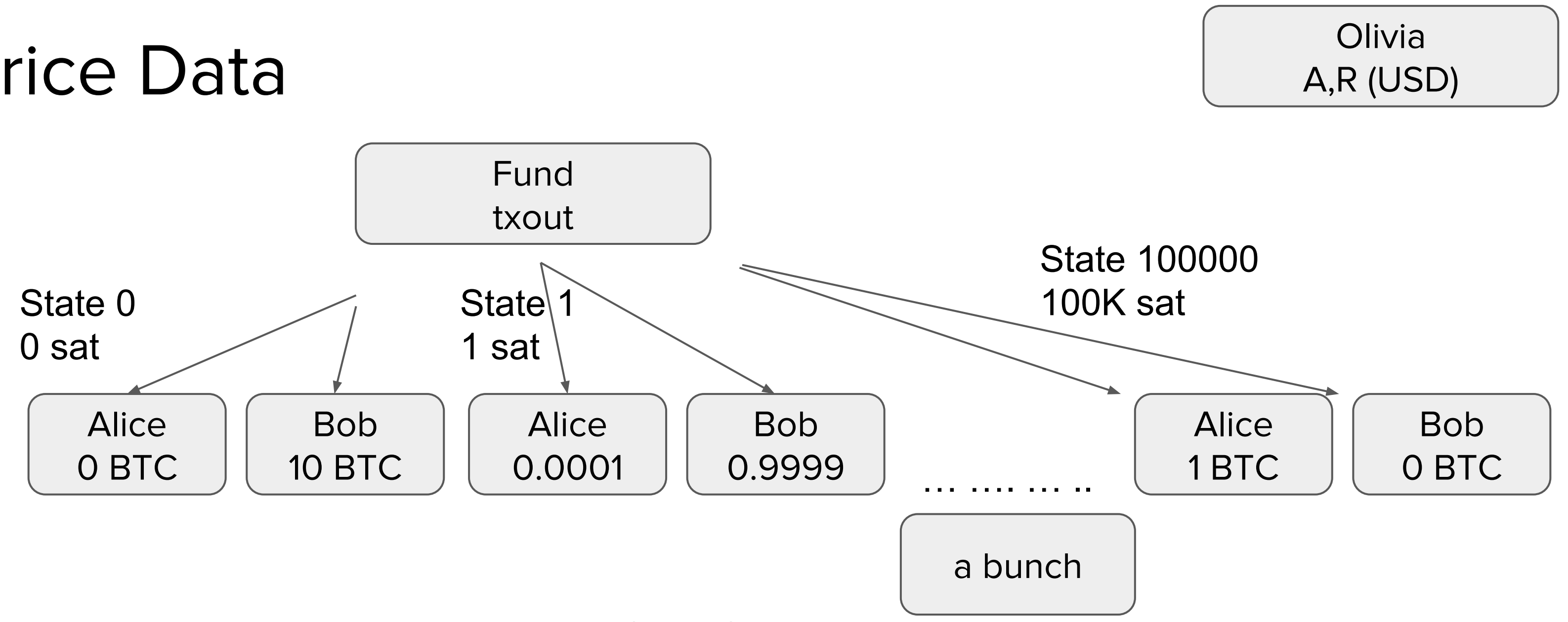
If broadcast to the network, it's still not clear that it was a contract. The oracle's sG pubkey is not detectable or decidable.

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)

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.

Future / Questions

```
Will add functionality to
github.com/mit-dci/lit
comments / questions welcome
adiabat on freenode IRC
```