Coinjoin done right

(and anti-Sybil with RIDDLE)

waxwing (Adam Gibson)

10th December 2022

(no affiliation)

Outline

1. The need for coinjoin
2. The problem with coinjoin
3. SDMC
4. The Sybil problem in JM, LN and others
5. Cost imposition strategies
6. Fidelity bonds
7. PoDLE
8. RIDDLE

The need for coinjoin

Hard money

Hard money

Onchain is the industrial layer.

Hard money

Onchain is the industrial layer.

Bitcoin is like financial uranium.

Hard money

Onchain is the industrial layer.

Bitcoin is like financial uranium.

But buying and selling bitcoin on a CEX is like financial anthrax.

Hard money

Onchain is the industrial layer.

Bitcoin is like financial uranium.

But buying and selling bitcoin on a CEX is like financial anthrax.

Hard money

Onchain is the industrial layer.

Bitcoin is like financial uranium.

But buying and selling bitcoin on a CEX is like financial anthrax.

SWIFT not Starbucks points.

Why is privacy hard?

Digicash was perfectly ("information theoretically") private. It failed, badly.

Why is privacy hard?

Digicash was perfectly ("information theoretically") private. It failed, badly.

Onchain activity is intrinsically about power and politics. "The payments they don't want you to make".

Why is privacy hard?

Digicash was perfectly ("information theoretically") private. It failed, badly.

Onchain activity is intrinsically about power and politics. "The payments they don't want you to make".

Consumer activity will be entirely offchain.

Why is privacy hard?

Digicash was perfectly ("information theoretically") private. It failed, badly.

Onchain activity is intrinsically about power and politics. "The payments they don't want you to make".

Consumer activity will be entirely offchain.

The DNA of blockchains is to be public. Making blockchain activity private is never more than partial, and has unpleasant tradeoffs.

Why is privacy hard? 2

Onchain privacy is a political act - coinjoins, sidechains, alt-currencies (Monero, Zcash).

Why is privacy hard? 2

Onchain privacy is a political act - coinjoins, sidechains, alt-currencies (Monero, Zcash). These things all have value, but not solving the problem.

Why is privacy hard? 2

Onchain privacy is a political act - coinjoins, sidechains, alt-currencies (Monero, Zcash).

These things all have value, but not solving the problem.

Monero, Zcash have not succeeded, even though they correctly see the problem of non-cryptographic onchain privacy, because they neglect the “DNA”.

Failure modes are catastrophic, and have already occurred.

Why is privacy hard? 2

Onchain privacy is a political act - coinjoins, sidechains, alt-currencies (Monero, Zcash).

These things all have value, but not solving the problem.

Monero, Zcash have not succeeded, even though they correctly see the problem of non-cryptographic onchain privacy, because they neglect the “DNA”.

Failure modes are catastrophic, and have already occurred.

Why is privacy hard? 2

Onchain privacy is a political act - coinjoins, sidechains, alt-currencies (Monero, Zcash).

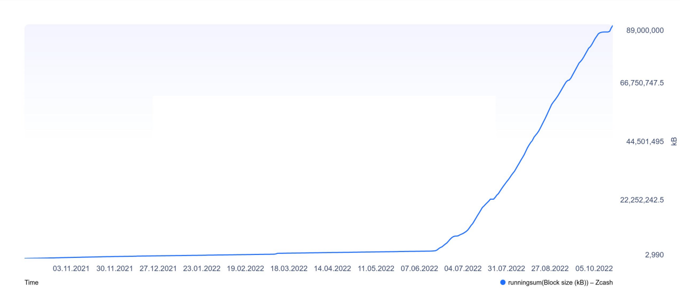
These things all have value, but not solving the problem.

Monero, Zcash have not succeeded, even though they correctly see the problem of non-cryptographic onchain privacy, because they neglect the “DNA”.

Failure modes are catastrophic, and have already occurred.

A lightning payment is an apolitical act.

Why is privacy hard? 3



h/t Jameson Lopp; this is Zcash being "attacked" with spam transactions.

Perfect privacy + "coins" means never forgetting old state.

It accumulates infinitely (nullifiers)[1].

[1] https://z.cash/technology/zksnarks/

The problem with coinjoin

The problem with coinjoin

"Make every spend a coinjoin" ..?

The problem with coinjoin

“Make every spend a coinjoin” ..?

Centrally coordinated, smooth experience. Can’t choose amount.

The problem with coinjoin

“Make every spend a coinjoin” ..?

Centrally coordinated, smooth experience. Can’t choose amount.

Market based, not smooth experience, on chain fee watermark.

The problem with coinjoin

“Make every spend a coinjoin” ..?

Centrally coordinated, smooth experience. Can’t choose amount.

Market based, not smooth experience, on chain fee watermark.

Both: public (hence ’political’).

The problem with coinjoin

“Make every spend a coinjoin” ..?

Centrally coordinated, smooth experience. Can’t choose amount.

Market based, not smooth experience, on chain fee watermark.

Both: public (hence ’political’).

Economically incentivised via CISA? No.[2]

[2] https://github.com/ElementsProject/cross-input-aggregation/blob/master/savings.org

SDMC

Steganographic

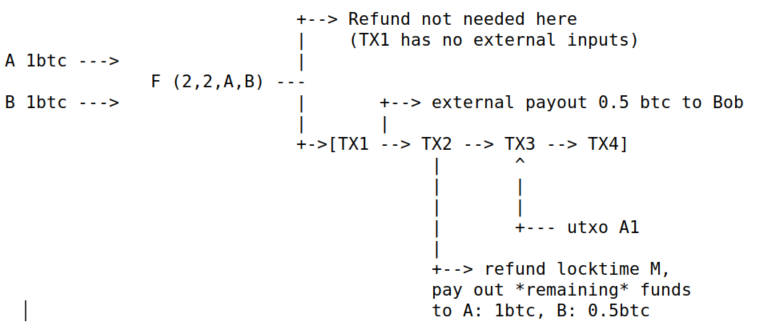
Decentralized

Market-based

Coinjoins

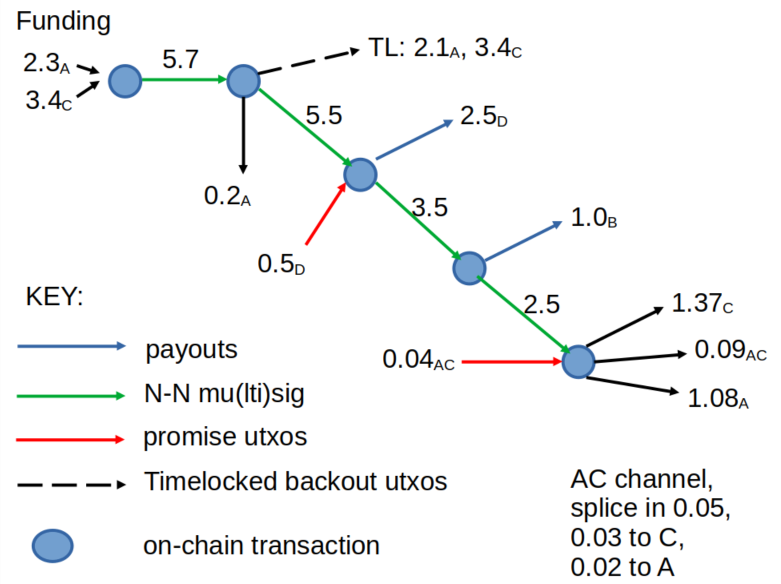
Steganographic?

CoinjoinXT basic idea:



```
                            +--> Refund not needed here
                            |    (TX1 has no external inputs)
A 1btc --->                 |
              F (2,2,A,B) ---
B 1btc --->                 |       +--> external payout 0.5 btc to Bob
                            |       |
                            +->[TX1 --> TX2 --> TX3 --> TX4]
                                         |       ^
                                         |       |
                                         |       |
                                         |       +--- utxo A1
                                         |
                                         +--> refund locktime M,
                                         pay out *remaining* funds
  |                                      to A: 1btc, B: 0.5btc
```

The kitchen sink!



Advantages - 1

- Every transaction, except arguably the last, looks like a normal spend.

Advantages - 1

- Every transaction, except arguably the last, looks like a normal spend.
- All the transaction negotiation can happen right at the start (A, C, D)

Advantages - 1

- Every transaction, except arguably the last, looks like a normal spend.
- All the transaction negotiation can happen right at the start (A, C, D)
- MuSig2 or ECDSA-2P can hide the use of multisig

Advantages - 1

- Every transaction, except arguably the last, looks like a normal spend.
- All the transaction negotiation can happen right at the start (A, C, D)
- MuSig2 or ECDSA-2P can hide the use of multisig
- This sub-graph is not distinguishable; analyst doesn't see this as an event

Advantages - 2

Ins:

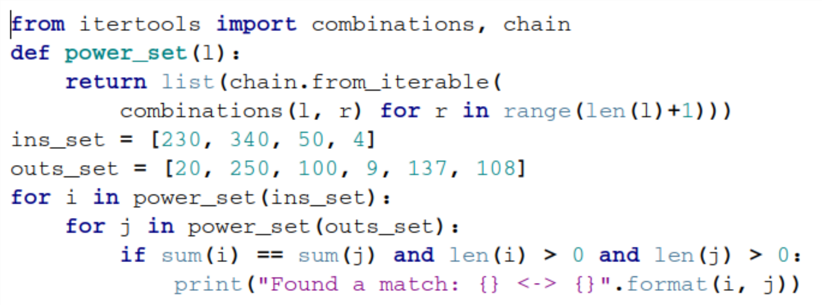
$$\begin{bmatrix} 2.3 \\ 3.4 \\ 0.5 \\ 0.04 \end{bmatrix}$$

Outs:

$$\begin{bmatrix} 0.2 \\ 2.5 \\ 1.0 \\ 0.09 \\ 1.37 \\ 1.08 \end{bmatrix}$$

Considering power sets, there are $\sim 2^{10}$ subset-pairs to consider. And none of them work!

Advantages - 3



```
from itertools import combinations, chain
def power_set(l):
    return list(chain.from_iterable(
        combinations(l, r) for r in range(len(l)+1)))
ins_set = [230, 340, 50, 4]
outs_set = [20, 250, 100, 9, 137, 108]
for i in power_set(ins_set):
    for j in power_set(outs_set):
        if sum(i) == sum(j) and len(i) > 0 and len(j) > 0:
           print("Found a match: {} <-> {}".format(i, j))
```

outputs:

```
Found a match:  (230, 340, 50, 4) <->
(20, 250, 100, 9, 137, 108)
```

Advantages - 4

Subset-sum fails in 3 ways:

- There are multiple subset sum solutions

Advantages - 4

Subset-sum fails in 3 ways:

- There are multiple subset sum solutions
- The analyst cannot find the subgraph!

Advantages - 4

Subset-sum fails in 3 ways:

- There are multiple subset sum solutions
- The analyst cannot find the subgraph!
- There is an inter-participant (not external) payment

Advantages - 4

Subset-sum fails in 3 ways:

- There are multiple subset sum solutions
- The analyst cannot find the subgraph!
- There is an inter-participant (not external) payment

Ideally all 3 can occur, but arguably the second is the most important.

Disadvantages

All at once? Timelocks, but:

Disadvantages

All at once? Timelocks, but:

Means coordinating delayed payments

Disadvantages

All at once? Timelocks, but:

Means coordinating delayed payments

Might mean reverting fully to taker/maker model (see next)

Disadvantages

All at once? Timelocks, but:

Means coordinating delayed payments

Might mean reverting fully to taker/maker model (see next)

Offchain-onchain privacy bleed is cool but . . .

Disadvantages

All at once? Timelocks, but:

Means coordinating delayed payments

Might mean reverting fully to taker/maker model (see next)

Offchain-onchain privacy bleed is cool but . . .

Size difference cannot be overwhelming or subset-sum break is lost

Disadvantages

All at once? Timelocks, but:

Means coordinating delayed payments

Might mean reverting fully to taker/maker model (see next)

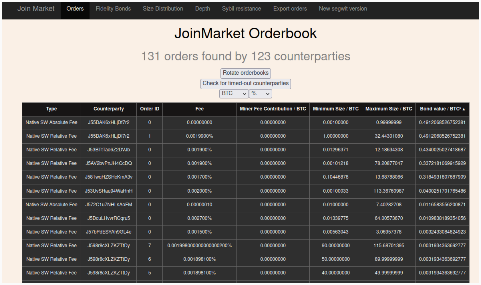
Offchain-onchain privacy bleed is cool but . . .

Size difference cannot be overwhelming or subset-sum break is lost

Payment “hashes”: PTLC required for atomicity with adaptor.

Markets and Fees

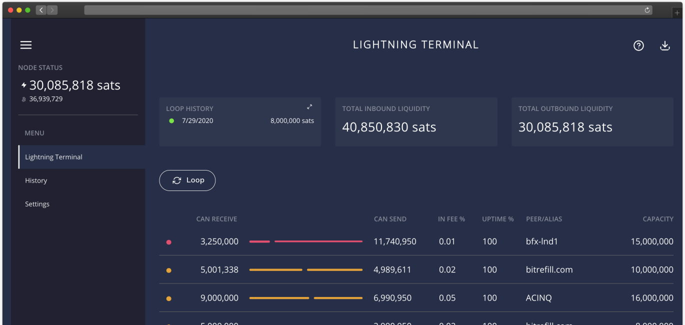
Liquidity



JoinMarket Orderbook

131 orders found by 123 counterparties

Type	Counterparty	Order ID	Fee	Miner Fee Contribution / BTC	Minimum Size / BTC	Maximum Size / BTC	Bond value / BTC² ▲
Native SW Absolute Fee	J55DAK6xHLjDf7r2	0	0.00000000	0.00000000	0.00100000	0.99999999	0.4912068526752381
Native SW Relative Fee	J55DAK6xHLjDf7r2	1	0.0019900%	0.00000000	1.00000000	32.44301080	0.4912068526752381
Native SW Relative Fee	J53BTtTao6Z2DVJb	0	0.001900%	0.00000000	0.01296371	12.18634308	0.4340025027418687
Native SW Relative Fee	J5AV2bvPnJH4CcDQ	0	0.001900%	0.00000000	0.00101218	78.20877047	0.3372181069915929
Native SW Relative Fee	J581wqHZSHcKmA3v	0	0.001700%	0.00000000	0.10446878	13.68788066	0.3184931807687909
Native SW Relative Fee	J53UvSHau94WaHnH	0	0.002000%	0.00000000	0.00100033	113.36760987	0.0400251701765486
Native SW Absolute Fee	J572C1u7NHLsAoFM	0	0.00000010	0.00000000	0.01000000	7.40282708	0.0116583556200871
Native SW Relative Fee	J5DcuLHvvrRCqru5	0	0.002700%	0.00000000	0.01339775	64.00573670	0.0109838189354056
Native SW Relative Fee	J57bPdESYAh9GL4e	0	0.001500%	0.00000000	0.00563043	3.06957378	0.0032433084824923
Native SW Relative Fee	J598r8cXLZKZTtDy	7	0.0019980000000000000200%	0.00000000	90.00000000	115.68701395	0.0031934363692777
Native SW Relative Fee	J598r8cXLZKZTtDy	6	0.001898100%	0.00000000	50.00000000	89.99999999	0.0031934363692777
Native SW Relative Fee	J598r8cXLZKZTtDy	5	0.001898100%	0.00000000	40.00000000	49.99999999	0.0031934363692777



NODE STATUS

30,085,818 sats

36,939,729

LOOP HISTORY

● 7/29/2020 8,000,000 sats

TOTAL INBOUND LIQUIDITY

40,850,830 sats

TOTAL OUTBOUND LIQUIDITY

30,085,818 sats

	CAN RECEIVE	CAN SEND	IN FEE %	UPTIME %	PEER/ALIAS	CAPACITY
●	3,250,000	11,740,950	0.01	100	bfx-lnd1	15,000,000
●	5,001,338	4,989,611	0.02	100	bitrefill.com	10,000,000
●	9,000,000	6,990,950	0.05	100	ACINQ	16,000,000
●	5,000,000	2,990,950	0.02	100	bitrefill.com	8,000,000

The problem - onchain fees

Coordinating to get liquidity - pay to participate (JM, liquidity ads etc.)

The problem - onchain fees

Coordinating to get liquidity - pay to participate
(JM, liquidity ads etc.)
For coinjoin, *on chain fees leave a trace*.

The problem - onchain fees

Coordinating to get liquidity - pay to participate (JM, liquidity ads etc.)

For coinjoin, *on chain fees leave a trace*.

Inputs of payer are easy to identify. This means no privacy from receiver, one of the best things a coinjoin **could** give.

The problem - onchain fees

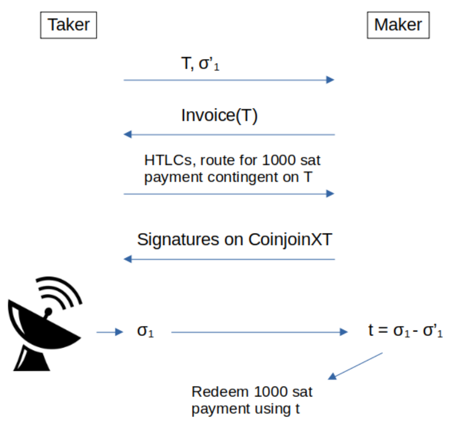
Coordinating to get liquidity - pay to participate (JM, liquidity ads etc.)

For coinjoin, *on chain fees leave a trace*.

Inputs of payer are easy to identify. This means no privacy from receiver, one of the best things a coinjoin **could** give.

Outputs **may** be hidden from snooper, but may not: asymmetric behaviour on-chain.

The solution? - offchain fees


Taker
Maker
T, σ'₁
Invoice(T)
HTLCs, route for 1000 sat payment contingent on T
Signatures on CoinjoinXT
σ₁
t = σ₁ - σ'₁
Redeem 1000 sat payment using t

HODL invoice - signature adaptors (Schnorr)

The solution? - offchain fees - 2

Taker holds adaptor secret q.

The solution? - offchain fees - 2

Taker holds adaptor secret q.

Makers set up invoices that unlock to preimage $q + \alpha_n$.

The solution? - offchain fees - 2

Taker holds adaptor secret q.

Makers set up invoices that unlock to preimage $q + \alpha_n$.

If only 1 Maker, then the adaptor atomicity works:

The solution? - offchain fees - 2

Taker holds adaptor secret q.

Makers set up invoices that unlock to preimage $q + \alpha_n$.

If only 1 Maker, then the adaptor atomicity works:

$\sigma'_t = k_t + \mathbb{H}(\tilde{P}, \tilde{R} + Q, tx)x_t$

The solution? - offchain fees - 2

Taker holds adaptor secret q.

Makers set up invoices that unlock to preimage $q + \alpha_n$.

If only 1 Maker, then the adaptor atomicity works:

$\sigma'_t = k_t + \mathbb{H}(\tilde{P}, \tilde{R} + Q, tx)x_t$

Taker sends Q and σ'_t, Maker verifies.

The solution? - offchain fees - 2

Taker holds adaptor secret q.

Makers set up invoices that unlock to preimage $q + \alpha_n$.

If only 1 Maker, then the adaptor atomicity works:

$\sigma_t' = k_t + \mathbb{H}(\tilde{P}, \tilde{R} + Q, tx)x_t$

Taker sends Q and σ_t', Maker verifies.

Then Maker sends his own partial sig σ_m.

The solution? - offchain fees - 2

Taker holds adaptor secret q.

Makers set up invoices that unlock to preimage $q + \alpha_n$.

If only 1 Maker, then the adaptor atomicity works:

$\sigma'_t = k_t + \mathbb{H}(\tilde{P}, \tilde{R} + Q, tx)x_t$

Taker sends Q and σ'_t, Maker verifies.

Then Maker sends his own partial sig σ_m.

When Taker broadcasts tx and full σ, Maker can find q:

The solution? - offchain fees - 2

Taker holds adaptor secret q.

Makers set up invoices that unlock to preimage $q + \alpha_n$.

If only 1 Maker, then the adaptor atomicity works:

$\sigma'_t = k_t + \mathbb{H}(\tilde{P}, \tilde{R} + Q, tx)x_t$

Taker sends Q and σ'_t, Maker verifies.

Then Maker sends his own partial sig σ_m.

When Taker broadcasts tx and full σ, Maker can find q:

$q = \sigma - \sigma_m - \sigma'_t$.

The solution? - offchain fees - 2

Taker holds adaptor secret q.

Makers set up invoices that unlock to preimage $q + \alpha_n$.

If only 1 Maker, then the adaptor atomicity works:

$\sigma'_t = k_t + \mathbb{H}(\tilde{P}, \tilde{R} + Q, tx)x_t$

Taker sends Q and σ'_t, Maker verifies.

Then Maker sends his own partial sig σ_m.

When Taker broadcasts tx and full σ, Maker can find q:

$q = \sigma - \sigma_m - \sigma'_t$.

Then uses q to settle the Lightning invoice.

The solution? - offchain fees - 3

Last step doesn't work with $N > 1$ makers;

$\sigma - \sigma_m \neq \sigma_t$.

We need **every** maker to be assured that when tx is broadcast, they get q.

🤔

The solution? - offchain fees - 4

The solution is a lot like “multiparty S6”[3] :

1 Taker, 2 Makers, coordinate aggregated key P_{agg}.

All 3 share hashes of Q_i and hashes of R_i as commitments (3 round Musig).

The Q_i and R_i values are then revealed (all to all).

All parties calculate aggregated nonce $\sum R_i$.

Taker constructs payment hashes for invoices $Q_1 + Q_3$, $Q_1 + Q_2$.

[3] https://reyify.com/blog/multiparty-s6

The solution? - offchain fees - 5

Then each party can make their signature adaptors, e.g.:

$\sigma'_1 = k_1 + \mathbb{H}(P_{agg}||R + Q_1 + Q_2 + Q_3||m)x_{agg,1}$

Each other party can verify those:

$\sigma'_1 G =? = R_1 + \mathbb{H}(\ldots)P_{agg,1}$

When all satisfied, any can start sending "full" partials: Maker 2 (index 3) can send:

$\sigma_3 = k_3 + q_3 + \mathbb{H}(P_{agg}||R + Q_1 + Q_2 + Q_3||m)x_{agg,3}$

The solution? - offchain fees - 6

and if full signature appears on chain, can deduce $q_1 + q_2$ by subtraction:
$\sigma - \sigma_3 - \sigma'_1 - \sigma'_2 = q_1 + q_2$
... and $q_1 + q_2$ is the preimage for his Lightning payment, thus he atomically claims the payment if the funding utxo is spent.

SDMC - the sales pitch

- A market solution to coordination

SDMC - the sales pitch

- A market solution to coordination
- No on chain footprint of that market

SDMC - the sales pitch

- A market solution to coordination
- No on chain footprint of that market
- A steganographic style of coinjoin - each tx in the tree can look like a payment

SDMC - the sales pitch

- A market solution to coordination
- No on chain footprint of that market
- A steganographic style of coinjoin - each tx in the tree can look like a payment
- Preserve the no-cross-block interactivity property of coinjoin

SDMC - the sales pitch

- A market solution to coordination
- No on chain footprint of that market
- A steganographic style of coinjoin - each tx in the tree can look like a payment
- Preserve the no-cross-block interactivity property of coinjoin
- Reduced chain bloat from reduced tx sizes from increased anon set from stega- property.

Remaining issues

- Does it matter how private off chain payments are?

Remaining issues

- Does it matter how private off chain payments are?
- Staggered times - is it too inconvenient?

Remaining issues

- Does it matter how private off chain payments are?
- Staggered times - is it too inconvenient?
- Remembering signed txs for a while?

Remaining issues

- Does it matter how private off chain payments are?
- Staggered times - is it too inconvenient?
- Remembering signed txs for a while?
- Very big (e.g. 10BTC), off chain bleed through fails?

Remaining issues

- Does it matter how private off chain payments are?
- Staggered times - is it too inconvenient?
- Remembering signed txs for a while?
- Very big (e.g. 10BTC), off chain bleed through fails?
- PTLC required for offchain fee part.

Imposing a cost

Ways of imposing cost

Lock coins.

Ways of imposing cost

Lock coins.

Burn coins.

Ways of imposing cost

Lock coins.

Burn coins.

Directly pay.

Ways of imposing cost

Lock coins.

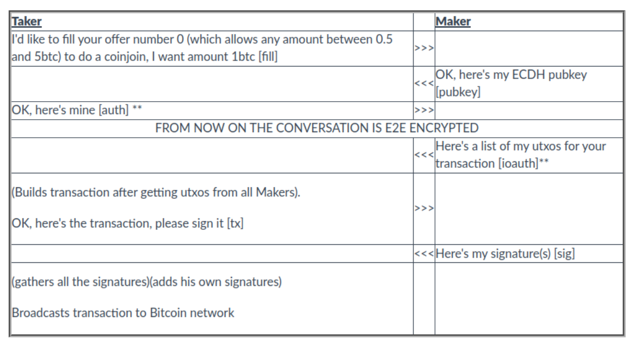
Burn coins.

Directly pay.

Proof of work.

Probing to deanonymize

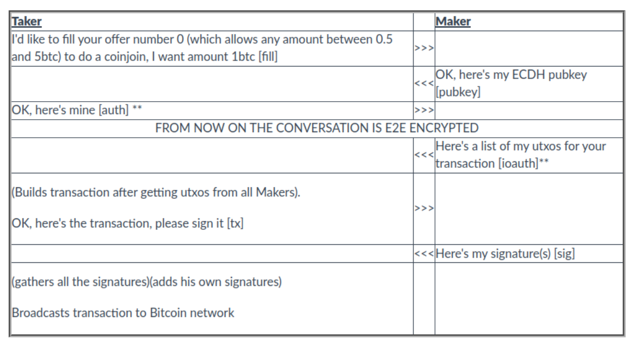
In Joinmarket, 2016:



Taker		Maker
I'd like to fill your offer number 0 (which allows any amount between 0.5 and 5btc) to do a coinjoin, I want amount 1btc [fill]	>>>	
	<<<	OK, here's my ECDH pubkey [pubkey]
OK, here's mine [auth] **	>>>	
FROM NOW ON THE CONVERSATION IS E2E ENCRYPTED		
	<<<	Here's a list of my utxos for your transaction [ioauth]**
(Builds transaction after getting utxos from all Makers). OK, here's the transaction, please sign it [tx]	>>>	
	<<<	Here's my signature(s) [sig]
(gathers all the signatures)(adds his own signatures) Broadcasts transaction to Bitcoin network		

Probing to deanonymize

In Joinmarket, 2016:



Taker		Maker
I'd like to fill your offer number 0 (which allows any amount between 0.5 and 5btc) to do a coinjoin, I want amount 1btc [fill]	>>>	
	<<<	OK, here's my ECDH pubkey [pubkey]
OK, here's mine [auth] **	>>>	
FROM NOW ON THE CONVERSATION IS E2E ENCRYPTED		
	<<<	Here's a list of my utxos for your transaction [ioauth]**
(Builds transaction after getting utxos from all Makers). OK, here's the transaction, please sign it [tx]	>>>	
	<<<	Here's my signature(s) [sig]
(gathers all the signatures)(adds his own signatures) Broadcasts transaction to Bitcoin network		

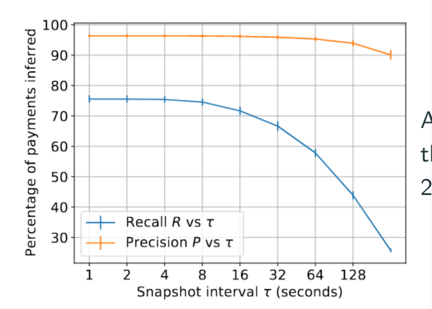
The attacker must *race* to keep up; if they only sample occasionally, they don't get the data to relate old states to new states.

Probing to deanonymize - 2

A similar dynamic in Lightning.

Probing to deanonymize - 2

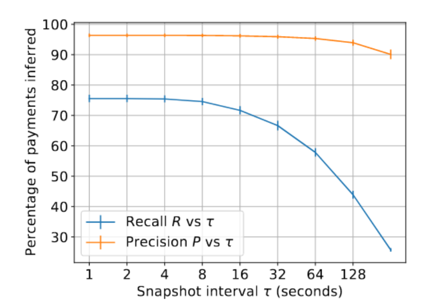
A similar dynamic in Lightning.



An Empirical Analysis of Privacy in the Lightning Network - Yousaf et al 2021

Probing to deanonymize - 2

A similar dynamic in Lightning.

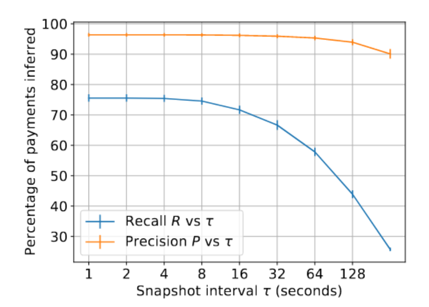


An Empirical Analysis of Privacy in the Lightning Network - Yousaf et al 2021

Notice that short snapshot intervals are required (depends on traffic).

Probing to deanonymize - 2

A similar dynamic in Lightning.

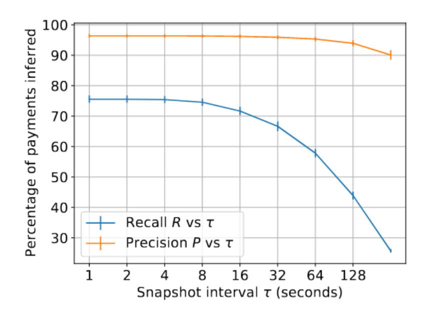


An Empirical Analysis of Privacy in the Lightning Network - Yousaf et al 2021

Notice that short snapshot intervals are required (depends on traffic).
Why? Payments *overlap*. (Old state → new state, again).

Probing to deanonymize - 2

A similar dynamic in Lightning.



An Empirical Analysis of Privacy in the Lightning Network - Yousaf et al 2021

Notice that short snapshot intervals are required (depends on traffic).
Why? Payments *overlap*. (Old state → new state, again).
But, they are also sufficient!

BTC as resource to exhaust

Jamming.

BTC as resource to exhaust

Jamming.

Subtle versions: protocol aborts pre-conf; low fees.

BTC as resource to exhaust

Jamming.

Subtle versions: protocol aborts pre-conf; low fees.

Obvious case: Lightning channel jamming:

BTC as resource to exhaust

Jamming.

Subtle versions: protocol aborts pre-conf; low fees.

Obvious case: Lightning channel jamming:

- Capacity
- Slot jamming

BTC as resource to exhaust

Jamming.

Subtle versions: protocol aborts pre-conf; low fees.

Obvious case: Lightning channel jamming:

- Capacity
- Slot jamming

Proposals to solve are numerous![4][5][6]

[4] https://github.com/t-bast/lightning-docs/blob/master/spam-prevention.md

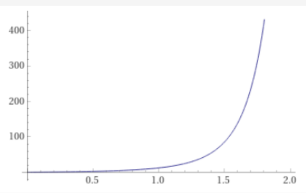
[5] https://jamming-dev.github.io/book

[6] https://github.com/s-tikhomirov/ln-jamming-simulator/blob/master/unjamming-lightning.pdf

Solutions

An idealist's framework

- No identities - no blame apportioning
- Ideal - superlinear cost to attack
- Ideal - *imperceptible* cost for normal protocol usage



Example - privacypass

Privacy Pass

System for getting tokens for solving CAPTCHAs that can be used anonymously.

Example - privacypass

Privacy Pass

System for getting tokens for solving CAPTCHAs that can be used anonymously.

Specifically rate-limits/throttles through “human” work.

Example - privacypass

Privacy Pass

System for getting tokens for solving CAPTCHAs that can be used anonymously.

Specifically rate-limits/throttles through “human” work.

Uses simple CDH-based blinding (cf blind signatures).

Example - privacypass

Privacy Pass

System for getting tokens for solving CAPTCHAs that can be used anonymously.

Specifically rate-limits/throttles through "human" work.

Uses simple CDH-based blinding (cf blind signatures).

Easy for a low-volume user.

Example - privacypass

Privacy Pass

System for getting tokens for solving CAPTCHAs that can be used anonymously.

Specifically rate-limits/throttles through “human” work.

Uses simple CDH-based blinding (cf blind signatures).

Easy for a low-volume user.

Requires a central server for the service.

Example 1 - PoDLEs

Need to 'use' utxos to access the service (tokens based on utxo keys)

Example 1 - PoDLEs

Need to 'use' utxos to access the service (tokens based on utxo keys)

PODLE (DLEQ) is not private from the counterparty:

Example 1 - PoDLEs

Need to 'use' utxos to access the service (tokens based on utxo keys)
PODLE (DLEQ) is not private from the counterparty:
Commit: $\mathbb{H}(Q = x_u * J)$ where J is NUMS

Example 1 - PoDLEs

Need to 'use' utxos to access the service (tokens based on utxo keys)

PODLE (DLEQ) is not private from the counterparty:

Commit: $\mathbb{H}(Q = x_u * J)$ where J is NUMS

Reveal: $\mathbf{DLEQ}(P_u, Q), \quad U$

Example 1 - PoDLEs

Need to 'use' utxos to access the service (tokens based on utxo keys)

PODLE (DLEQ) is not private from the counterparty:

Commit: $\mathbb{H}(Q = x_u * J)$ where J is NUMS

Reveal: $\textbf{DLEQ}(P_u, Q), \quad U$

Commitments are broadcast aggressively, so can't be reused.

Example 1 - PoDLEs

Need to 'use' utxos to access the service (tokens based on utxo keys)

PODLE (DLEQ) is not private from the counterparty:

Commit: $\mathbb{H}(Q = x_u * J)$ where J is NUMS

Reveal: $\mathbf{DLEQ}(P_u, Q), \quad U$

Commitments are broadcast aggressively, so can't be reused.

Multiple tokens? Just generate J with a counter.

Example 2 - RIDDLEs

RIDDLE:

- ring sig/ZKP means utxo never revealed.

Example 2 - RIDDLEs

RIDDLE:

- ring sig/ZKP means utxo never revealed.
- **linkable** ring sig with broadcast: one usage

Example 2 - RIDDLEs

RIDDLE:

- ring sig/ZKP means utxo never revealed.
- **linkable** ring sig with broadcast: one usage
- tags for multiple usages

Example 2 - RIDDLEs

RIDDLE:

- ring sig/ZKP means utxo never revealed.
- **linkable** ring sig with broadcast: one usage
- tags for multiple usages
- compact proof/sig size: logarithmic in anon set

Example 2 - RIDDLEs

RIDDLE:

- ring sig/ZKP means utxo never revealed.
- **linkable** ring sig with broadcast: one usage
- tags for multiple usages
- compact proof/sig size: logarithmic in anon set
- sublinear verify time?

RIDDLEs - scaling

```
A1o7rjCKHp+SskFhdPxrb/y0ThBb31QZ5h+HfME0RI06A3gJJEgEGQz1r9e7ExKvLB9Md
9cvWzdnfMeUdbEWrjhuAnVEkOUd1Kh23VMft8qIyJozWUGzi31r1Nf+bostBM94AwCtPB
NpKS/A8P9uDRswYj68oodjt4h7z7FD8yKKnePwAm1KreAaMu4LpGkTNPc8YWp6HalEloo
9EK26PrbFLbMxA3orj96BSgfXnBB2m4II9ENa+taEtNkno0I9GukS1a6dA2403BAC/w4I
A2NbLJufCilpSAluIow9b+kxHlq5kqe/A5j35Y+HBANs1B4YaKxeDczae9bthibo1dvtm
KsRdZ9DAuhz5YUY2priqCHYFQL0/heDN6E4bOWlN5/fgGO6HMpQAtdhnzIFiZEhBJ/bAC
mU+rxmdw7GO4VX7JvPAxEb5dR6Aq9KZxexxmmi0BVguwyx6bD0uHxD7U7GEpkFQ0eRGoi
VA3d78myUJDA74ZWCX5lvgEaF8eNY7L2nqByAj0GiEW2tAmbQnnGo8P1Abo9WmGJPzgEe
XILcqKWMCXcZuRMOjkzRAuBamBH73Hf0F6w8E/YhilcnTccKYb2mWp+p06U16JkQAyNju
vTxC6vloS5TV0WPMy/bIjUw9FwoBFdIAua5pk6/A7E9UpoKfg97kQXt707qfP5jOwj9+G
puKQoT5e6Kq7JzA9b5h00UO3RRxrU+x/dp58ZTESqf0lgZOUv6++pIr7qgAujxddeheGN
UVWVZi5U2Brsn5tyjcc0eMHkgeYbTS5YwAryTlI0RJVGIkJQm5Ngv8vQinTWsON0fA3J2
I5rTOyycAjVvLQkuoHO/qQqVYS37CFOfaEOng+AWqXECNZaS70lpN6v9VClqNAysjSTOY
mWXjNWUh0aqeDB8XPnK2rySOFLnrTeGDBpvZPGUYQTV06XUMI15JP/AfNifdN8sP504h7
n8rh+PRsKbfAgb9WDTTfW5NOc6cOoMGol5PKO76j4XrNL/uWwcnPNVBTrE8jkDmNeDHll
kyHZFos2YsYycHAZCjbTgdlJVCvImCcz2eQdaYY2nWujrF4wuK5QEyVK2RXwI8eqC8ZgX
q4Fgt65Q/42+3tUY6dwZZ9hrFOR5LwHnaI0NM+GStqwSD80Jl+17xel7crQo15H8WRnAj
NZiATBGM90dwq+lMYxI6v8qtu9ZNKlWQUaIU5cu5plYuKTHz+h8QVpJnG3TEsMJcj5bbQ
H4dQocmSr0a3q7plgYLLFzbdPhGrMrZigBPh6f0ZJlkCFSy6t38tGbw1AiqCrn64ByUv2
E4t7y+V0ewvZLACshi0acgCXor3GMpg3f7tRq8pQwciq0iqltqi0Z06mIXLKhsCwRW+q3
TNulNwe485cNYBhiWl8FwyT2CxDYVTWVmZ7exMdYFggBuV7OPly3K8Od8Z0AF2OiZLFP1
uz5VY3DgAsZ9VSdfMbnAOnZx2Btzfq9/LEnY0fWxsaOUgWWV4BnUYaLtWdMmJAiO2bB92
liKe5G2hmAgyibci7jebR7WksehkKV3O9j2HJ6ddET8w+5eEYjqOFQeWLKvfL78OqdKCJ
hiPuu5VCrxw1oqGTAdHjRQrEKIG0zZWZqfyyEvm1ZHIu2yKSiVlSu2bG7fKoAvJyg1pP+
XaeYiQzE+KJavWiboEMjsPyoKdhVd4nl8ck=
```

RIDDLE - scaling - 2

That was about 6K taproot keys on signet, with a linking tag.[7]

[7] https://reyify.com/blog/little-riddle

RIDDLE - scaling - 2

That was about 6K taproot keys on signet, with a linking tag.[7]
(Linkable) ring sig with log scale communication for ECDL.

[7] https://reyify.com/blog/little-riddle

RIDDLE - scaling - 2

That was about 6K taproot keys on signet, with a linking tag.[7]

(Linkable) ring sig with log scale communication for ECDL.

Work of Groth, Bootle, MRL (triptych).

[7] https://reyify.com/blog/little-riddle

RIDDLE - scaling - 2

That was about 6K taproot keys on signet, with a linking tag.[7]

(Linkable) ring sig with log scale communication for ECDL.

Work of Groth, Bootle, MRL (triptych).

But all DL based schemes were linear verify, which isn't OK for 10K keys!

[7] https://reyify.com/blog/little-riddle

RIDDLE - scaling - 2

That was about 6K taproot keys on signet, with a linking tag.[7]

(Linkable) ring sig with log scale communication for ECDL.

Work of Groth, Bootle, MRL (triptych).

But all DL based schemes were linear verify, which isn't OK for 10K keys!

Pairings based schemes can achieve the goal but: secp256k1.

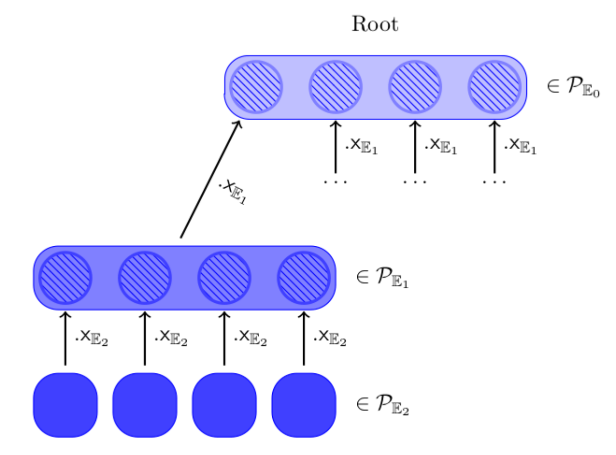
[7] https://reyify.com/blog/little-riddle

RIDDLE - scaling - 3

Merkle proofs are log-time-verify.

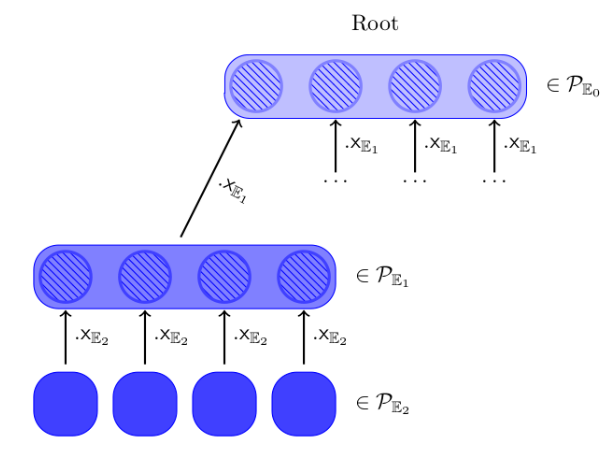
RIDDLE - scaling - 3

Merkle proofs are log-time-verify.


Root
$\in \mathcal{P}_{\mathbb{E}_0}$
$.\mathrm{x}_{\mathbb{E}_1}$
$.\mathrm{x}_{\mathbb{E}_1}$
$.\mathrm{x}_{\mathbb{E}_1}$
$.\mathrm{x}_{\mathbb{E}_1}$
$\cdots$
$\cdots$
$\cdots$
$\in \mathcal{P}_{\mathbb{E}_1}$
$.\mathrm{x}_{\mathbb{E}_2}$
$.\mathrm{x}_{\mathbb{E}_2}$
$.\mathrm{x}_{\mathbb{E}_2}$
$.\mathrm{x}_{\mathbb{E}_2}$
$\in \mathcal{P}_{\mathbb{E}_2}$

RIDDLE - scaling - 3

Merkle proofs are log-time-verify.



Curve Trees - Practical and Transparent Zero Knowledge Accumulators, Campanelli et al, 2022

Cheap to verify RIDDLEs?

secp/secq 2-cycle (Poelstra)[8], see also “pasta”(Hopwood)[9] etc.

[8] https://moderncrypto.org/mail-archive/curves/2018/000992.html

[9] https://electriccoin.co/blog/the-pasta-curves-for-halo-2-and-beyond/

Cheap to verify RIDDLEs?

secp/secq 2-cycle (Poelstra)[8], see also "pasta"(Hopwood)[9] etc.

Shallow Merkle tree with Pedersen hashing (algebraic)

[8] https://moderncrypto.org/mail-archive/curves/2018/000992.html

[9] https://electriccoin.co/blog/the-pasta-curves-for-halo-2-and-beyond/

Cheap to verify RIDDLEs?

secp/secq 2-cycle (Poelstra)[8], see also "pasta"(Hopwood)[9] etc.

Shallow Merkle tree with Pedersen hashing (algebraic)

Sublinear verify even though ZKP of openings is linear (Bulletproofs)

[8] https://moderncrypto.org/mail-archive/curves/2018/000992.html

[9] https://electriccoin.co/blog/the-pasta-curves-for-halo-2-and-beyond/

Cheap to verify RIDDLEs?

secp/secq 2-cycle (Poelstra)[8], see also "pasta"(Hopwood)[9] etc.

Shallow Merkle tree with Pedersen hashing (algebraic)

Sublinear verify even though ZKP of openings is linear (Bulletproofs)

Basic results of paper are at 20ms verify even for 1M commitments

[8] https://moderncrypto.org/mail-archive/curves/2018/000992.html

[9] https://electriccoin.co/blog/the-pasta-curves-for-halo-2-and-beyond/

Cheap to verify RIDDLEs?

secp/secq 2-cycle (Poelstra)[8], see also "pasta"(Hopwood)[9] etc.

Shallow Merkle tree with Pedersen hashing (algebraic)

Sublinear verify even though ZKP of openings is linear (Bulletproofs)

Basic results of paper are at 20ms verify even for 1M commitments

Using KZG10 style polynomial commitments even more optimal, but pairings needed

[8] https://moderncrypto.org/mail-archive/curves/2018/000992.html

[9] https://electriccoin.co/blog/the-pasta-curves-for-halo-2-and-beyond/

Cheap to verify RIDDLEs?

secp/secq 2-cycle (Poelstra)[8], see also "pasta"(Hopwood)[9] etc.
Shallow Merkle tree with Pedersen hashing (algebraic)
Sublinear verify even though ZKP of openings is linear (Bulletproofs)
Basic results of paper are at 20ms verify even for 1M commitments
Using KZG10 style polynomial commitments even more optimal, but pairings needed

Gobbledeygook? No worries: bottom line we can get the ∼3kB proof for 10K keys and still have our server/routing node not hang on 100% CPU.

[8] https://moderncrypto.org/mail-archive/curves/2018/000992.html
[9] https://electriccoin.co/blog/the-pasta-curves-for-halo-2-and-beyond/

Maybe a broad strategy here?

- Use RIDDLEs for high anonymity and low cost
- Use Chaumian tokens (e.g. privacypass) for modulating spending, lightweight and cheap to use
- Use service-side throttling to vary cost based on market conditions

Is RIDDLE just a Stake Certificate?

Yes, it is an attempt to implement that idea, except:

Is RIDDLE just a Stake Certificate?

Yes, it is an attempt to implement that idea, except: There is no attempt to identify a *type* of taproot utxo.

Is RIDDLE just a Stake Certificate?

Yes, it is an attempt to implement that idea, except: There is no attempt to identify a *type* of taproot utxo.

What is the difference between a 'real' channel utxo and a 'fake' one?

Is RIDDLE just a Stake Certificate?

Yes, it is an attempt to implement that idea, except:
There is no attempt to identify a *type* of taproot utxo.
What is the difference between a 'real' channel utxo and a 'fake' one?
So RIDDLE only does ZKPOK of key for utxo, plus one-timeness (tokenized).

Is RIDDLE just a Stake Certificate?

Yes, it is an attempt to implement that idea, except: There is no attempt to identify a *type* of taproot utxo.

What is the difference between a 'real' channel utxo and a 'fake' one?

So RIDDLE only does ZKPOK of key for utxo, plus one-timeness (tokenized).

https://jamming-dev.github.io/book/6-reputation.html

Is RIDDLE just a Stake Certificate? - 2

Is RIDDLE just a Stake Certificate? - 2

Probing can definitely be “honest”.

Is RIDDLE just a Stake Certificate? - 2

Probing can definitely be “honest”.

Payments can be streamed, channels can be faked.

Is RIDDLE just a Stake Certificate? - 2

Probing can definitely be “honest”.

Payments can be streamed, channels can be faked.

Align cost with resource usage and forget labelling usage types.

Is RIDDLE just a Stake Certificate? - 2

Probing can definitely be “honest”.

Payments can be streamed, channels can be faked.

Align cost with resource usage and forget labelling usage types.

“Identity is the problem, not the solution.”

Sketch of a future with probing defence in LN

- RIDDLEs let nodes buy Chaumian tokens (CTs) anonymously from peer nodes to allow routing to work.

Sketch of a future with probing defence in LN

- RIDDLEs let nodes buy Chaumian tokens (CTs) anonymously from peer nodes to allow routing to work.
- In baseline operation, the cost in CTs per routing attempt is very low. Ordinary user won't see a cost worth mentioning.

Sketch of a future with probing defence in LN

- RIDDLEs let nodes buy Chaumian tokens (CTs) anonymously from peer nodes to allow routing to work.
- In baseline operation, the cost in CTs per routing attempt is very low. Ordinary user won't see a cost worth mentioning.
- Rep tokens could also be spent as an 'optimistic mode' thing.

Sketch of a future with probing defence in LN

- RIDDLEs let nodes buy Chaumian tokens (CTs) anonymously from peer nodes to allow routing to work.
- In baseline operation, the cost in CTs per routing attempt is very low. Ordinary user won't see a cost worth mentioning.
- Rep tokens could also be spent as an 'optimistic mode' thing.
- If traffic starts to get higher, start to ramp up the CTs per routing attempt cost. For a user who just wants to do **one** payment, they should still be fine but an attacker sending a stream is going to use up a lot of resources.

Sketch of a future with probing defence in LN

- RIDDLEs let nodes buy Chaumian tokens (CTs) anonymously from peer nodes to allow routing to work.
- In baseline operation, the cost in CTs per routing attempt is very low. Ordinary user won't see a cost worth mentioning.
- Rep tokens could also be spent as an 'optimistic mode' thing.
- If traffic starts to get higher, start to ramp up the CTs per routing attempt cost. For a user who just wants to do **one** payment, they should still be fine but an attacker sending a stream is going to use up a lot of resources.
- In the limit, a full speed probing attempt against a lot of nodes is going to require consuming a huge amount of utxo creation resources (absolute limit: block size), or otherwise a huge payment cost to buy from others (trust? reblinding?).

Final thoughts

- Anonymous environments are adversarial, and tough.

Final thoughts

- Anonymous environments are adversarial, and tough.
- Bitcoin lets us use cost to address that issue.

Final thoughts

- Anonymous environments are adversarial, and tough.
- Bitcoin lets us use cost to address that issue.
- But costs must be especially low for low resource usage.

Final thoughts

- Anonymous environments are adversarial, and tough.
- Bitcoin lets us use cost to address that issue.
- But costs must be especially low for low resource usage.
- RIDDLE is $\sim$ practical today, if taproot is used, and can be improved (crypto).

Final thoughts

- Anonymous environments are adversarial, and tough.
- Bitcoin lets us use cost to address that issue.
- But costs must be especially low for low resource usage.
- RIDDLE is $\sim$ practical today, if taproot is used, and can be improved (crypto).
- Joinmarket can notably improve its privacy model using this.

Final thoughts

- Anonymous environments are adversarial, and tough.
- Bitcoin lets us use cost to address that issue.
- But costs must be especially low for low resource usage.
- RIDDLE is $\sim$ practical today, if taproot is used, and can be improved (crypto).
- Joinmarket can notably improve its privacy model using this.
- At least some of the nastier attacks on LN could be addressed with this overall strategy (we hope!).

Thank you

Contact info:

@waxwing@x0f.org (mastodon)

https://github.com/AdamISZ

blog: https://reyify.com/blog (email there)

gpg: 4668 9728 A9F6 4B39 1FA8 71B7 B3AE 09F1 E9A3 197A