OCash: Fully Anonymous Payments between Blockchain Light Clients

Adam Blatchley Hansen[1], Jesper Buus Nielsen[1], Mark Simkin[2]

[1]Aarhus University, [2]Flashbots

Anonymous Transactions

Many constructions exist for anonymous transactions on blockchains.

However:

Most users are not running their own personal full node!

Light clients (eg, users cell phones,) rely on querying full nodes to send/receive transactions.

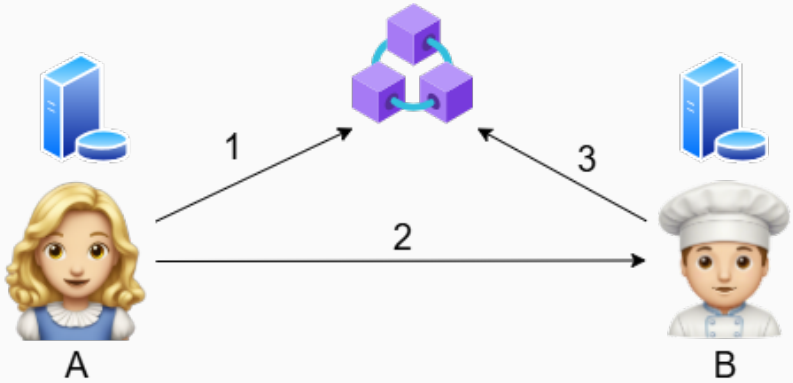
Question

Can we support light clients and at the same time provide strong anonymity guarantees?

1. Provably Secure anonymous Transactions
2. With strong anonymity (sender cannot see when receiver redeems coin)
3. Where light clients have privacy against full nodes

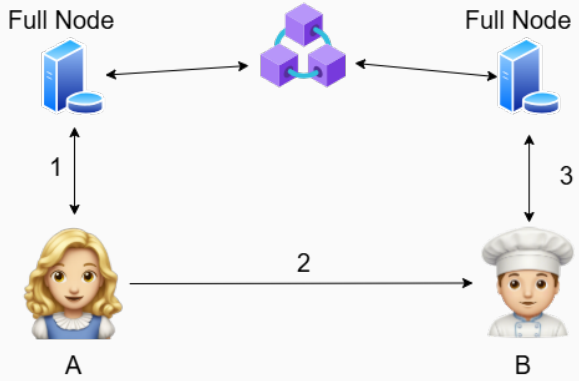
Anonymous payments

1. Coin
2. Opening information
3. Claim Coin



Light Client Anonymous Payments

1. Coin
2. Opening Information
3. Claim Coin



Light Client Anonymous Payments

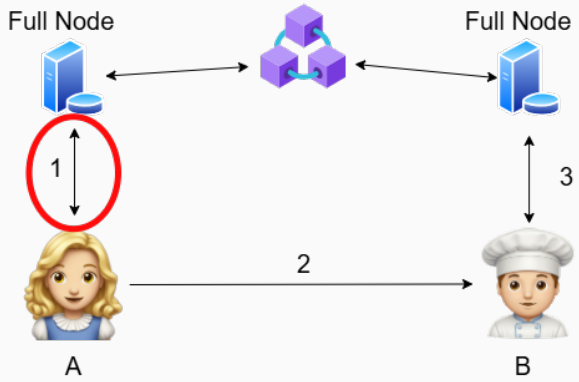
Whats the problem?

Accumulator Solutions (high level)

$C \leftarrow comm(A, B, r, tid)$

$\text{Coin} = (r, tid, C)$

1. Alice posts commitment C to chain, added to accumulator (merkle tree)

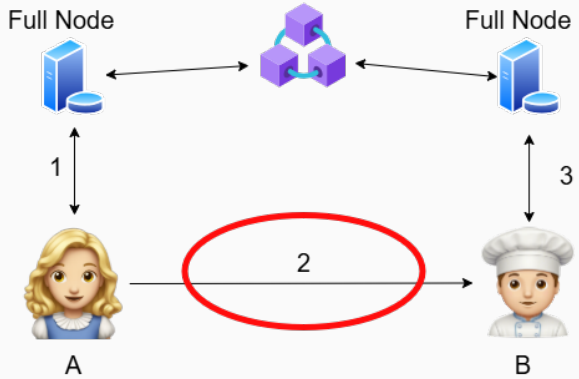


Accumulator Solutions (high level)

$C \leftarrow comm(A, B, r, tid)$

Coin $= (r, tid, C)$

2. Alice sends Coin to Bob

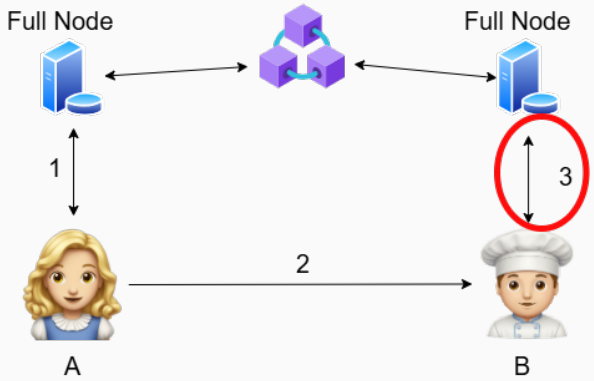


Accumulator Solutions (high level)

$C \leftarrow comm(A, B, r, tid)$

Coin $= (r, tid, C)$

3. Bob uses (r, tid) to prove knowledge of a C in merkle tree with identifier tid



Accumulator Solutions (high level)

Claiming:

$C \leftarrow comm(A, B, r, tid)$

Coin $= (r, tid, C)$

Bob publishes NIZK showing (high level)

1. There exists some commitment C in the merkle tree s.t.
2. Commitment C includes tid (Public)
3. Commitment C includes B (Private)
4. Commitment C includes r (Private)
5. And revealed tid hasn't been used before

Accumulator Solutions (high level)

Claiming:

$C \leftarrow comm(A, B, r, tid)$

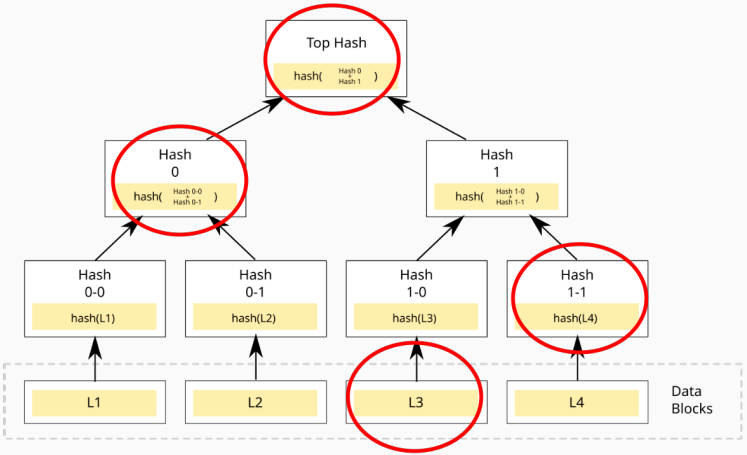
Coin $= (r, tid, C)$

Bob publishes NIZK showing (high level)

1. **There exists some commitment C in the merkle tree s.t.**
2. Commitment C includes tid (Public)
3. Commitment C includes B (Private)
4. Commitment C includes r (Private)
5. And revealed tid hasn't been used before

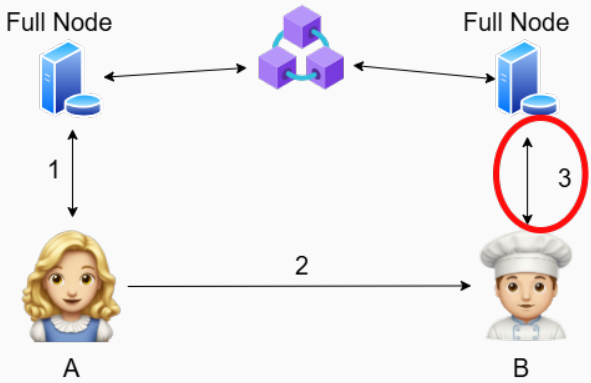
Accumulator Solutions (high level)

.



Accumulator Solutions (high level)

Bob needs state from Node



Contributions

Our Contributions

Model this problem in the UC framework

Propose scheme (based on oram) and prove security in this framework

Define and construct several building blocks

- ANCO (Anonymous Coin Friendly Encryption)
- SOROM (Strongly Oblivious Read-Once Maps)
- CRaB (Compressible Randomness Beacons)

This Talk

This talk

Model this problem in the UC framework

Propose scheme (based on oram) and prove security in this framework

Define and construct several building blocks

- **ANCO (Anonymous Coin Friendly Encryption)**
- **SOROM (Strongly Oblivious Read-Once Maps)**
- CRaB (Compressible Randomness Beacons)

Ocash: Idea

Bob needs to download enough state to prove presence of commitment

Full node shouldn't learn anything about which commitment

Ocash: Idea

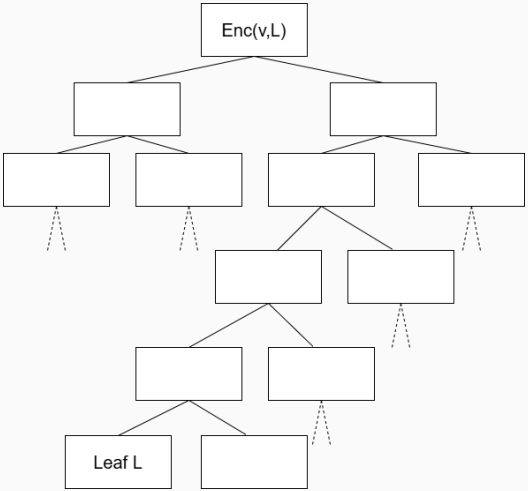
We want to obliviously read:

sounds like ORam!

Path ORam: Allows clients to read/write arbitrary data obliviously, with only poly-logarithmic overhead.

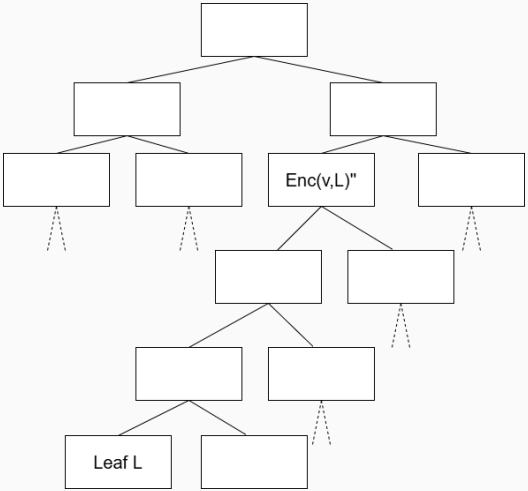
Ocash: Path Oram (high level)

Encrypted value inserted at root bucket (with random target leaf)



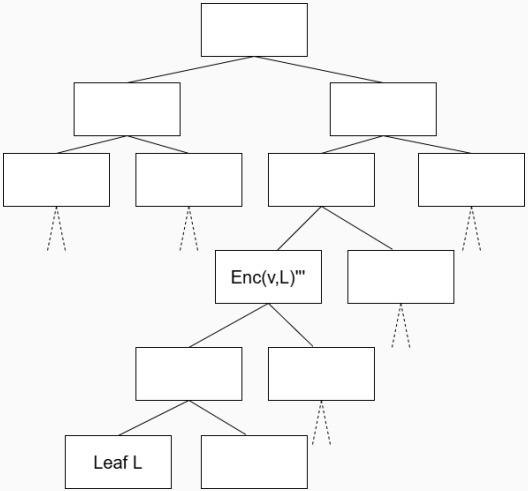
Ocash: Path Oram (high level)

Maintainence steps (obliviously) move value towards leaf



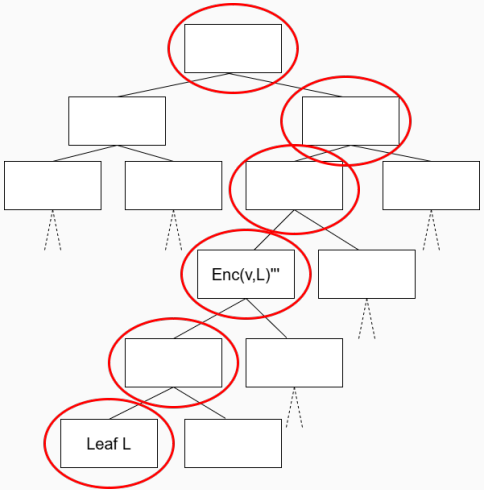
Ocash: Path Oram (high level)

Maintainence steps (obliviously) move value towards leaf



Ocash: Path Oram (high level)

To read, client reads entire path to leaf L. ($\log(n)$ overhead)



Ocash: Idea

Bob needs to download enough state to prove presence of commitment

Full node shouldn't learn anything about which commitment

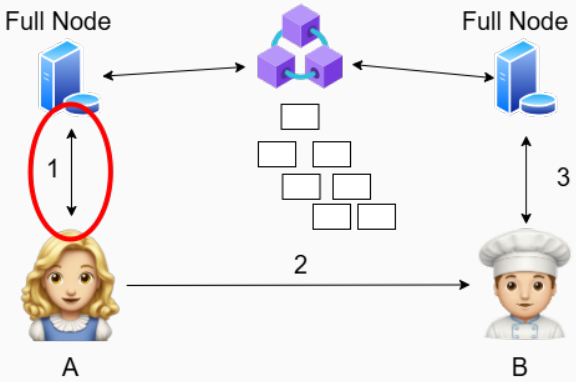
We have a trusted party (later committee) run a "read only oram" of Commitments!

OCash: (high level)

$C \leftarrow$ *commitment*

$\text{Coin} = (L_a, tid, C)$

1. Alice posts commitment C to chain, encrypts L_a to committee



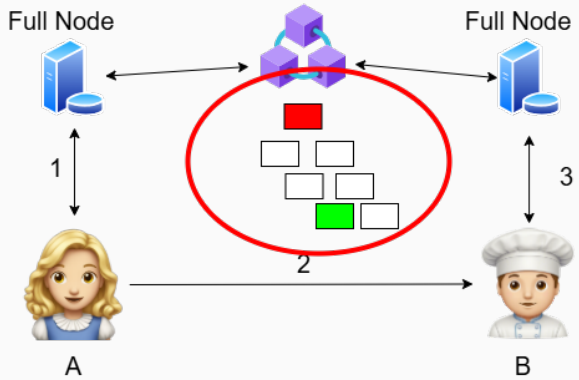
OCash: (high level)

Committee:

Get L_c from random beacon,

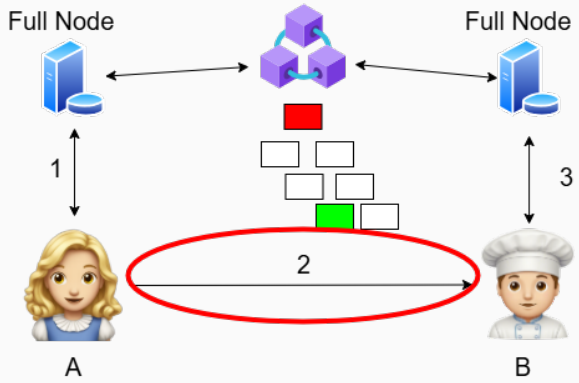
$L = H(L_c, L_a)$

Insert C into top bucket, with target leaf L



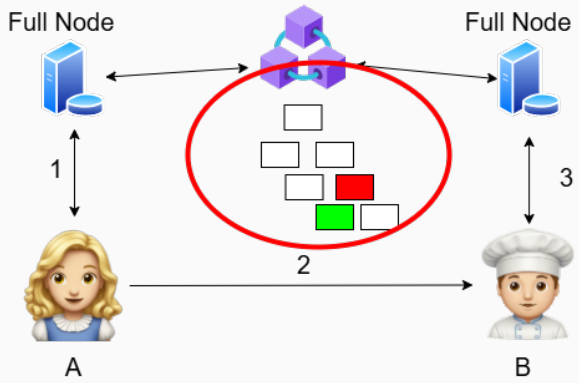
OCash: (high level)

Send opening information and L_a to Bob



OCash: (high level)

More coins inserted, Committee maintain data structure



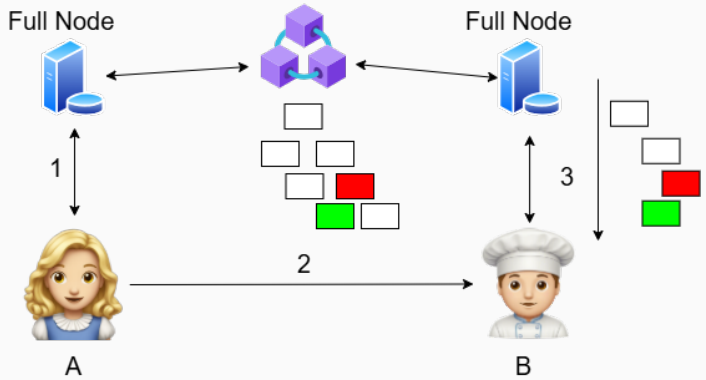
OCash: (high level)

3.

Bob requests random beacon history (CRaB key)

Bob computes $L = H(L_c, L_a)$

Bob requests path to L, nizk OR proof over all buckets



ANCO

$$tid = g_0^{\xi} \cdot g_1^{A} \cdot g_2^{B} \cdot g_3^{n_A} \cdot g_4^{a}$$

OCash "Coins" Coins are encryptions of the *tid* pedersen commitment, using a special PKE scheme

1. Re-randomisable public key encryption scheme (without knowing key)
2. Key-indistinguishability under re-randomisation
3. Strong message binding (can only decrypts to one message, regardless of key)

ANCO

KeyGen: $pp = (g_0, q)$
$x \xleftarrow{\$} \mathbb{Z}_q$
$h = g_0^x$
$(ek = (g_0, h), dk = x)$

Encryption: $pp = (g_0, q), ek = (g, h), m$
$\rho \xleftarrow{\$} \mathbb{Z}_q^*$
$\rho' \xleftarrow{\$} \mathbb{Z}_q$
$\mathsf{CT} = (g_0^\rho, h^\rho, g_0^{\rho'}, h^{\rho'} m)$

ANCO

Decrypt: $pp = (g_0, q), dk = x, CT = (A, B, C, D)$
If $B = A^x$ let $m = DC^{-x}$,
otherwise let $m = \perp$.

Rerandomise: $pp = (g_0, q), ct = (A, B, C, D)$
$\rho \xleftarrow{\$} \mathbb{Z}_q^*$
$\rho' \xleftarrow{\$} \mathbb{Z}_q$
Output $ct' = (A^\rho, B^\rho, A^{\rho'} C, B^{\rho'} D)$

$$\text{Coin} = \text{Enc}(tid) = (g^{\rho}, h^{\rho}, g^{\sigma}, h^{\sigma} \cdot \underbrace{g_0^{\xi} \cdot g_1^{A} \cdot g_2^{B} \cdot g_3^{n_A} \cdot g_4^{a}}_{tid})$$

Claiming Coin

1. Bob receives SOROM branch of $log(N)$ ANCO Ciphertexts.
2. Bob received dk from Alice, so can recognise and decrypt tid.
3. Bob publishes tid, with a NIZK showing an OR proof of decryption to tid over all commitments in the path

Note on proofs

DLOG relations + Sigma Protocols

By $(A, B, C, D' = D \cdot tid^{-1})$, we reduce the OR proof to showing that there is one tuple for which there exists a w such that $A^w = B$ and $C^w = D'$.

This can then be efficiently computed using the "one-out-of-many" DLOG Sigma Protocol (GK15, $log(l)$ blowup for l elements.)

We also give Sigma Protocols for sending/collecting encrypted amounts.

Path Oram Security model vs Sorom

CRaB

CRaB key

Compressed random beacon, allowing fast evaluation of any previous index

Strong Anonymity

Revealing tid breaks strong anonymity

$ohid = PRF(K, hid) = g_5^{1/(K+hid)}$

Output is run through a PRF before being revealed.

The Committee

Requires a trusted party or committee to maintain the ORAM CRaB

Make SOROM verifiable $\rightarrow$ can only break privacy

Even if deadlocks/dies, can still claim all coins.

The Committee

Requires a trusted party or committee to maintain the ORAM CRaB

For proof of stake committees, verifiable MPC

Is a heavy (neccesary?) assumption.

Future work

Other primitives? (PIR?)
Efficient SOROM/ORAM verifiable MPC
Recursive SNARK style constructions?

Thank You!

Thanks for listening!

Full construction, UC security modelling, proofs and more in the full paper.

https://eprint.iacr.org/2024/246

Any Questions?