AQQUA: Augmenting Quisquis with Auditability

George Papadoulis [1] Danai Balla [1 2] Panagiotis Grontas [1]
Aris Pagourtzis [1 2]

[1]National Technical University of Athens

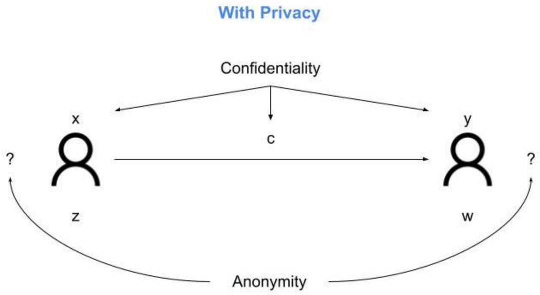
[2]Archimedes/Athena RC

AtheCrypt, May 23rd, 2024

Most Blockchains are Pseudonymous

- We can trace transactions and reveal user identity [Mei+16]
- No privacy
  - Personal data leakage
  - Front-running attacks
  - "tainted" currency

Privacy on the Blockchain



- Malicious incentives/illegal activities: (e.g. money laundering, tax evasion)
- TornadoCash is blacklisted
- Monero has been delisted from popular exchanges

A middle ground

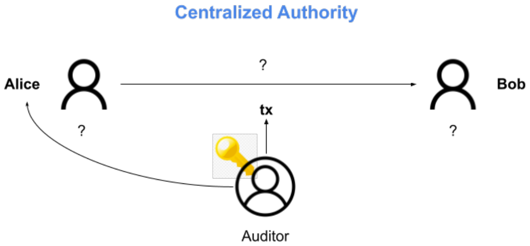
Combine Privacy with Auditability

Guarantees both system and participants comply with financial regulations and laws

Two approaches [CBC21]

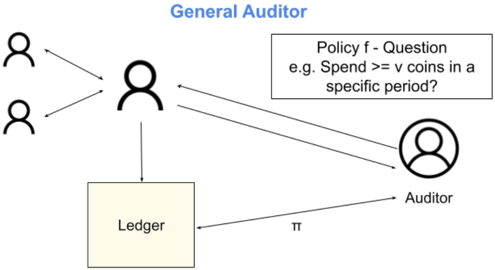
- Centralized Authority
- General Auditor

Auditability with a Centralized Authority



- Zcach extension [GGM17], PRCash[Wüs+19]
- The centralized authority gains too much power

Auditability with a General Auditor



- PGC [Che+20] - confidential, trades anonymity for auditability
- MiniLedger [CB21] - anonymous, confidential, concurrent transactions always have conflict

AQQUA: Augmenting Quisquis with Auditability

Goal

Construct efficient, anonymous, confidential, auditable system

AQQUA

- Augment Quisquis DPS [Fau+19] by adding a **general auditor**.

Quisquis:

- Fully private (anonymity & confindetiallity).
- Constant storage cost w.r.t. number of transactions
- Supports concurrent transactions

Auditability vs Privacy

Auditor questions

- upper bound on amount the user sent/received in a period of time
- non-participation
- exact value sent/received in a transaction

Challenges when combining privacy with auditability

- anonymity, user can hide accounts (and thus amounts)
- need to know which users are in the system

Authorities & State

- Add a Registration Authority that maps user's real identity with an initial public key.
- Split state to two sets:

UTXOSet

Accounts
$acct_1$
$acct_2$
$\dots$
$acct_m$

UsersSet

Initial public key	Number of accs
$\mathtt{pk}_{0_1}$	$\mathtt{com}(\#\mathtt{accs})_1$
$\mathtt{pk}_{0_2}$	$\mathtt{com}(\#\mathtt{accs})_2$
$\dots$	$\dots$
$\mathtt{pk}_{0_n}$	$\mathtt{com}(\#\mathtt{accs})_n$

Building Blocks for Accounts [Fau+19]

UPK

Group $(\mathbb{G}, p, g)$ where DLOG is hard

- $\mathsf{pk} = (g^r, g^{r\cdot \mathsf{sk}}) = (g_1, g_2)$, r random
- Update pk: Pick random s, $\mathsf{pk}' = (g_1^s, g_2^s)$

Indistinguishability: pk and pk′ are computationally indist. (DDH)

ElGamal commitments with UPKs

- $\mathtt{com}(\mathsf{pk}, v; r) = (c, d) = (g_1^r, g^v g_2^r)$
- homomorphism: Pick d, s,
  $\mathtt{com}' = \mathtt{com} \cdot (g_1^s, g^d g_2^s) = (g_1^{r+s}, g^{v+d}, g_2^{r+s})$
- Rerandomize: d $= 0$

Accounts

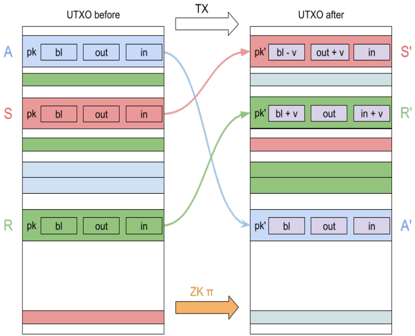
Account acct $= (\mathrm{pk}, \boxed{\texttt{bl}}, \boxed{\texttt{out}}, \boxed{\texttt{in}})$

- bl, account's balance
- out, in, the total amount that the account has sent/received

Accounts can be updated

- Update $\mathrm{pk} = (g, h) \to \mathrm{pk}' = (g^{r}, h^{r})$
- Re-randomize the commitments.
- Modify the commited value through homomorphic property.

Transactions



$$\mathrm{TX}(S, R, A, \vec{v} = (-v, +v, 0, \ldots, 0))$$

- Sender's balance reduced by v
- Receiver's balance increased by v
- Anonymity set no value change
- Update `in`, `out` accordingly
- Updates all accounts
- Shuffle outputs

ZK-proofs

- ZK-proof:
  - amount is substracted only from accounts that sender owns
  - preservation of value
  - correct update
- ZK proofs obtained as combination of Sigma protocols for:
  - Knowledge of sk (DL)
  - Correct Update (DDH)
  - Correct shuffling of accounts (Bayer-Groth shuffle)
- Can be made non-interactive using Fiat-Shamir transform.

AQQUA Properties

- **Non-growing UTXO**:
  - all inputs accounts can be removed from UTXOSet (spent)
- **Anonymity**: hide the link between inputs/outputs accounts
  - indistinguishability of UPK scheme
  - hiding property of commitment scheme
  - zero-knowledge property of the NIZK proofs
- **Theft prevention**: users can only move funds from accounts they own
  - unforgeability property of UPK scheme
  - binding property of commitment scheme
  - soundness property of the NIZK proofs

Supported Policies

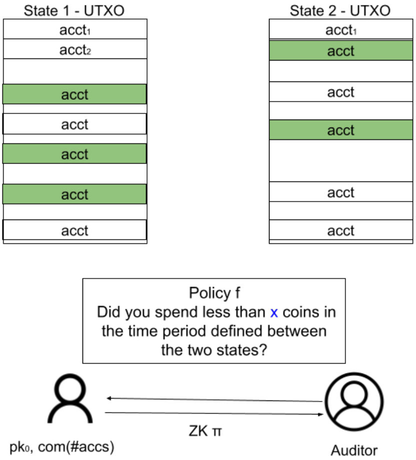
Policy Predicates

We capture policies as predicates f over

- $\mathtt{pk}_0$: initial public key
- $(\mathsf{state}_1, \mathsf{state}_2)$: time period
- $\mathtt{aux}$: auxiliary information

- **Sending limit policy** f_{slimit}
- **Receiving Limit policy** f_{rlimit}
- **Transaction Value Limit** f_{txlimit}
- **Non-participation** f_{np}

Audit Example



For each snapshot:

- Open $\boxed{\texttt{\#accs}}$
- Reveal `#accs` accounts to `AA`
- Prove opening of commitment and ownership of revealed accounts
- $\boxed{v_j} = \prod_{i=1}^{\#\texttt{accs}_j} \text{acct}_{j_i} . \boxed{\texttt{out}}$
- $\boxed{v} = \boxed{v_2} \,/\, \boxed{v_1}$
- Prove that opening of $\boxed{v} < x$ $(\sum \texttt{out}_2 - \sum \texttt{out}_1 < x)$

Audit Properties

- **Audit soundness**: there cannot be a successfully verified audit generated by a user who is non-compliant
  - binding property of commitment scheme
  - soundness property of the NIZK proofs

Note

The user reveals their accounts only for the two snapshots. The authority cannot learn any information about the rest of the states (indistinguishability property of accounts).

Conclusion & Future Work

- Based on Quisquis, we constructed an auditable private decentralized system
- Introduce authorities which do not intervene in the normal flow of transactions
- Stable state size
- Supports concurrent transactions

Future Work:

- Sound audit proofs while revealing only number of accounts to auditor instead of the accounts themselves.
- Convert audit proofs to be designated-verifier.

Thank You!

Questions?