Privacy-preserving KYC on Ethereum

Alex Biryukov, Dmitry Khovratovich, **Sergei Tikhomirov**

SnT, University of Luxembourg

9 May 2018
CWI, Amsterdam, The Netherlands

Outline

Identity is data that represents a user

Identity is used for:

- Authentication: proves the user is who they claim to be;
- Authorization: ensure the user is eligible for an action.

In cryptographic terms, user is represented by a private-public key pair.

Centrally managed identity

- Prevalent model today
- User delegate identity management to companies, get access using password
- Risks: identity theft, central point of failure

Decentralized identity

- Putting users in charge of managing their data
- Can be implemented using blockchains
- Does it respect privacy?
- Does it comply with regulations?

Bitcoin

- A decentralized digital currency [Nakamoto 2008]
- Combines cryptography and economics to prevent double spending without a trusted third party

Ethereum: generalized blockchain

- A blockchain-based application platform [Buterin 2014]
- Key feature: Turing complete programming

Ethereum tokens

- A popular use case for smart contracts
- A fungible unit of value maintained by a smart contract
- ERC20 is the de-facto standard token API
- Decentralized exchanges – a promising direction

ERC20 functions

- `transfer` – send tokens to an address
- `approve` – allow other user to transfer my tokens
- `transferFrom` – send other user's tokens (only if `approved`)

A decentralized KYC-compliant identity

Our identity management design for financial services is:

- Decentralized (on-chain)
- Privacy-preserving
- Can be made compliant
- Extendable to many application types (consider a token exchange as an example)

Cryptographic accumulator

- A cryptographic primitive: absorbs algebraic objects
- Provides interface to verify whether a value was accumulated
- Preserves privacy: individual values are not disclosed

Accumulator-based identity workflow (1/2)

- A KYC **Provider** publishes a contract with an empty accumulator
- A **User** interacts with the **Provider** (possibly offline) and gets their value accumulated
- The **Provider** issues a *witness* s.t. the **User** can later prove their eligibility

Accumulator-based identity workflow (2/2)

To prove eligibility, a user submits an (atomic) zero-knowledge proof of the statement:

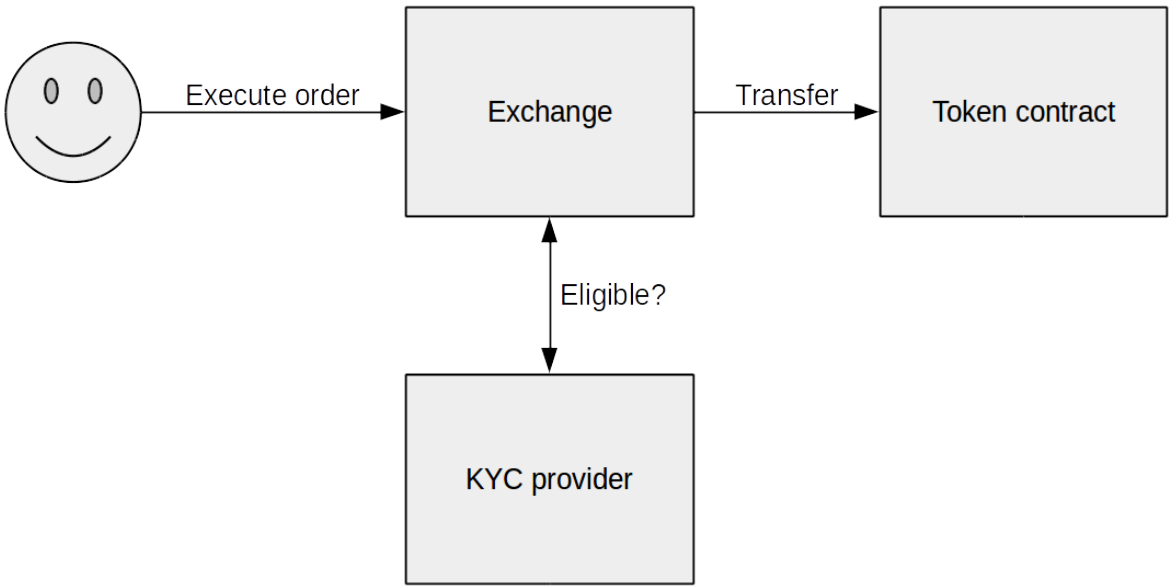
- I know the private key corresponding to `msg.sender`;
- I know a signature and a witness for *some* value which was previously accumulated.

KYC Provider interface

- `add(user, token)` – makes user eligible
- `remove(user, token)` – makes user not eligible
- `isEligible(user, token)` – check if the user is eligible

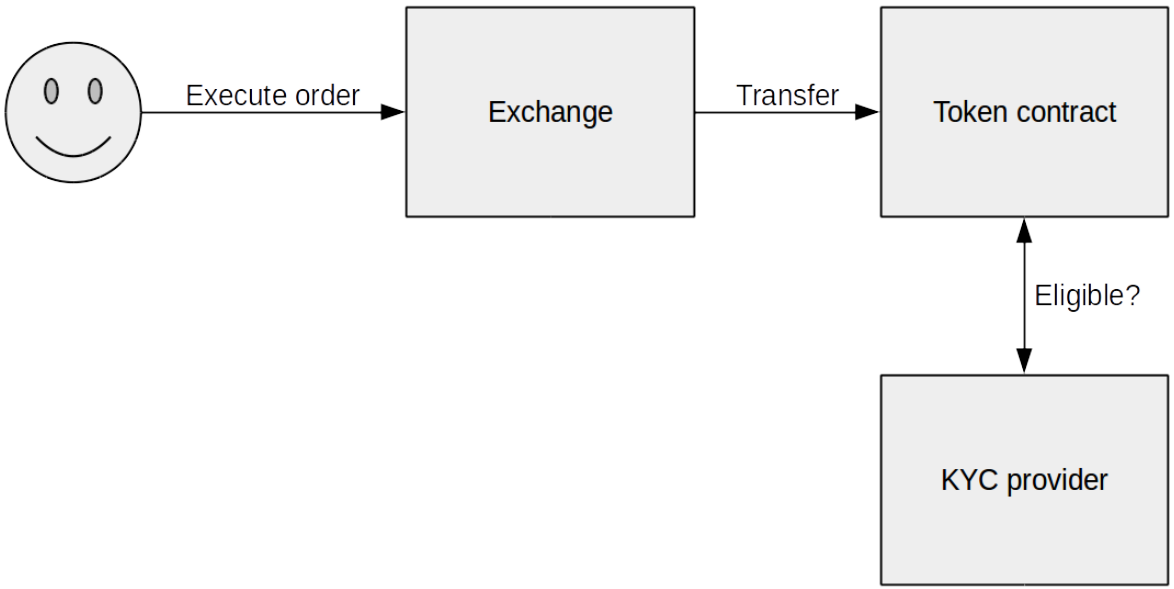
Use case 1: compliant exchange

- An exchange verifies users before making transactions
- Traded tokens do not need to be aware of KYC



Use case 2: compliant token

- A token verifies users before making transactions
- Services (exchanges) do not need to be aware of KYC



Implementation details

- A PoC implementation (not privacy-preserving): joint 1st prize at the Luxblock hackathon in May 2017
- (The team also included: Daniel Feher, Dmitry Khovratovich, Aleksei Udovenko, Maciej Zurad)
- Accumulator implementation depends on new opcodes: currently Ethereum does not natively support all required cryptographic operations
- Updating the accumulator is expensive if done on-chain

Conclusion and future work

- Ethereum provides ways to encode and enforce digital agreements
- Cryptography allows for additional eligibility checks which minimally impact the users' privacy
- Many technical challenges to overcome before realizing this idea

Research question

Can we leverage sophisticated cryptography in public blockchains to provide stronger security and privacy guarantees?

Questions?

- cryptolux.org
- s-tikhomirov.github.io