DARKID

Blockchain based anonymous distributed ID system

Proof of Concept

19th January 2018

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

arnaucode.com

@arnaucode

Main Concept

- Guarantee a decentralized login system
- Make sure that register users are real ones
    - Guarantee that there are no bots generating large amounts of fake accounts
- Ensure that only verified users can generate anonymous IDs (darkID)

Motivation

- Traditional Login Systems are owned by few big corporations
    - “Information is power. But like all power, there are those who want to keep it for Themselves.”
      - Aaron Swartz
- With centralized systems the power to censor, ban, delete users is in few hands
- We need to build an internet with decentralized systems, to decentralize power

Basic Functionalities

- Verify the identity of an user
  - Based on the
    - Username, Email, Phone, ID card, ...
- Generate anonymous ID (darkID) and get that ID signed by a serverSigner with high reputation
- Inject the darkID into the ethereum blockchain using a Smart Contract
- Authenticate in the platforms using the darkID, that is public in the ethereum blockchain

Elevator pitch

- This is Marie
- Marie wants to surf the net, without being identified
- But also, Marie don’t want a net full of bots and fake accounts
- Marie wants a login system based on cryptography, to ensure anonymity
- Also wants to ensure that only verified users can use the login system
- Marie loves decentralized systems
- Marie loves ethereum smart contracts
- Marie loves darkID

Technologies used

- Desktop App
  - Angularjs + Electron + Go lang
- Backend
  - Go lang
  - Solidity (Ethereum)

First Prototype - 'blockchainIDsystem'

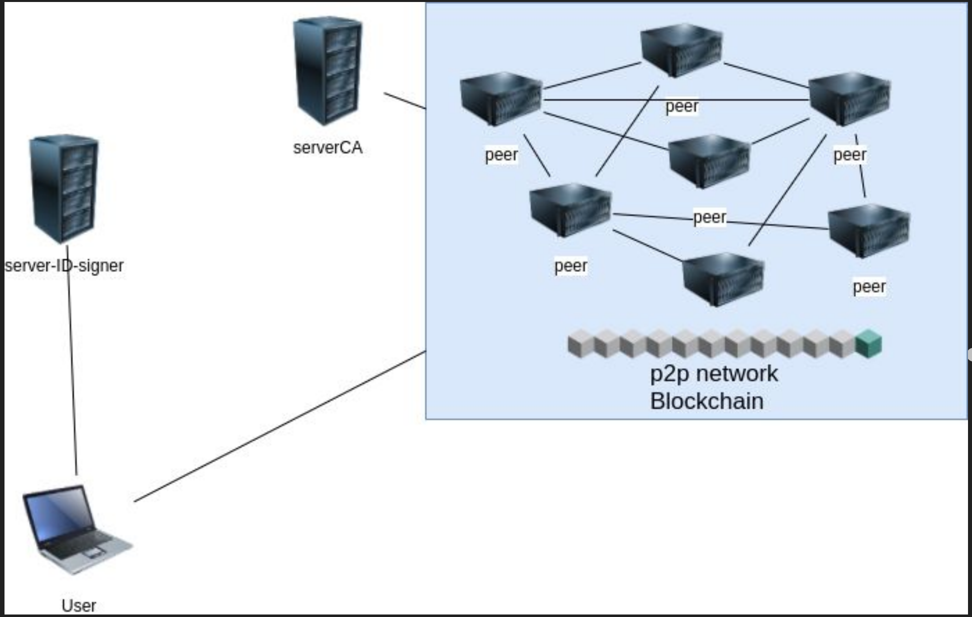
Everything from scratch:

- Written own RSA library
- Written own peer-to-peer network
- Written own blockchain algorithms over the p2p network
- Written Server-ID-Signer
- Written Desktop App

https://github.com/arnaucode/blockchainIDsystem

Project not finished and currently abandoned

First Prototype



- peers
  - the peers of the p2p network that runs the blockchain
- serverCA
  - Is a REST server that has been certified (is the Certified Authority) to validate the peers that will be able to participate of the blockchain.
  - Have the webapp (frontend) to validate peers through a GUI interface
  - The GUI frontend webapp allows also to view the current peers of the network and the blocks of the blockchain
- server-ID-signer
  - The server where the user creates a non anonymous account
  - Also is the server that blind signs the Anonymous ID of the users
  - Have the webapp (frontend) to interact through a GUI interface
  -

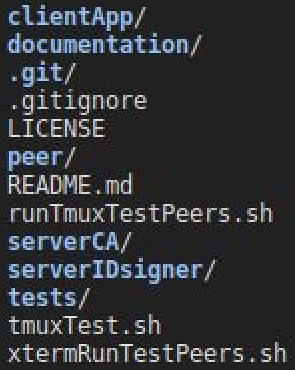
First Prototype

Problems:

- Implement a production ready peer-to-peer network is not so easy
  - Gnutella, distributed hash tables, freenet, …
- Use own cryptographic algorithms for real world solutions is not a good idea
- Implement a blockchain system needs to add some of Proof-of-Work, Proof-of-Stake, Proof-of-Cooperation, … system → too much code for a small university subject project

First Prototype - 'blockchainIDsystem'

First Prototype short demo



```
clientApp/
documentation/
.git/
.gitignore
LICENSE
peer/
README.md
runTmuxTestPeers.sh
serverCA/
serverIDsigner/
tests/
tmuxTest.sh
xtermRunTestPeers.sh
```

- At some point, I realized that this is not the way
- Better use ethereum smart contracts instead of developing own p2p network and own blockchain (assuming small amount of time for developing this university subject project)

2nd prototype:

darkID

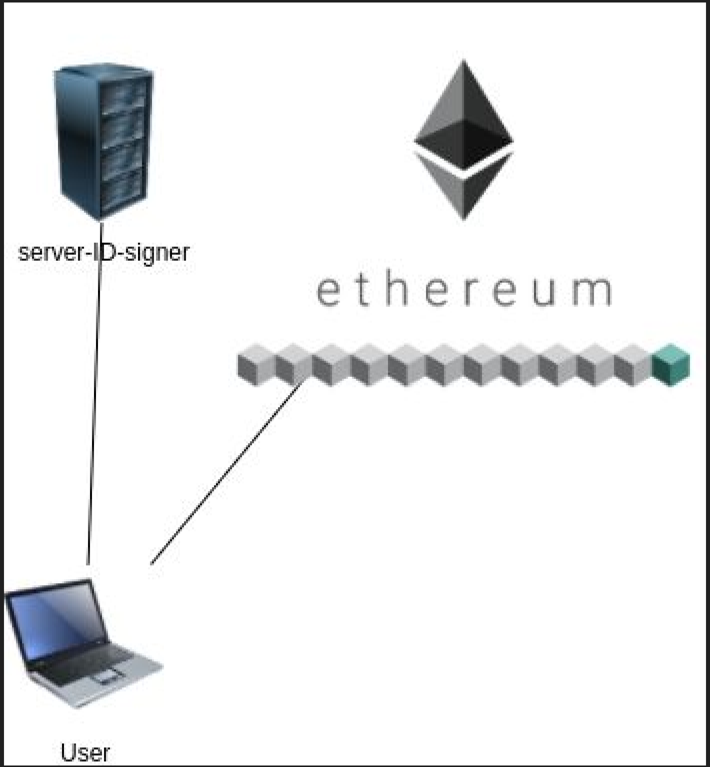
Blockchain based anonymous distributed ID system

https://github.com/arnaucode/darkID

2nd Prototype: darkID

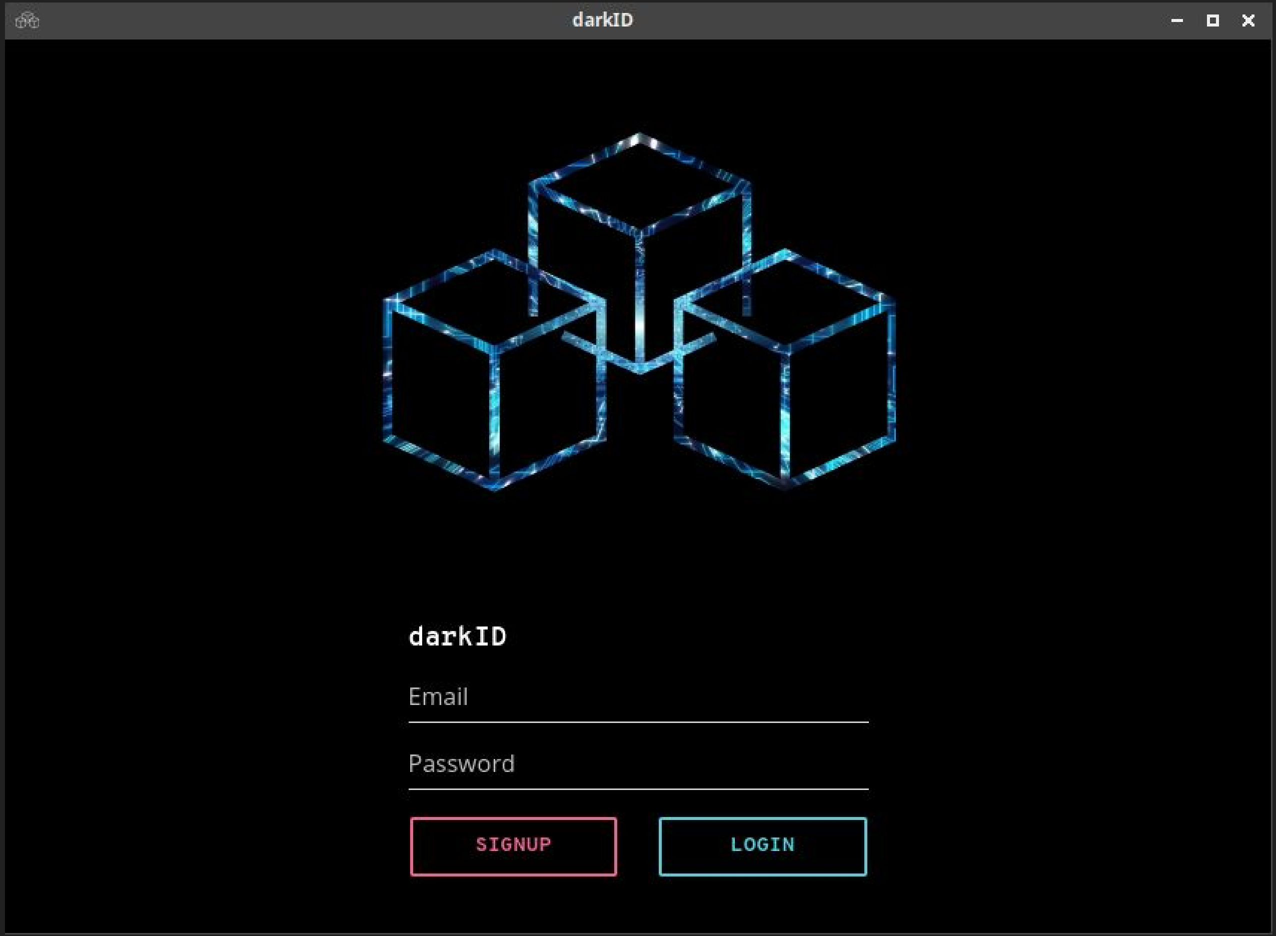
- Instead of developing own p2p network and own blockchain → use ethereum blockchain and ethereum Smart Contracts
- Use of Go existing cryptographic algorithms
- Written Server-ID-Signer
- Written Desktop App

DARKID

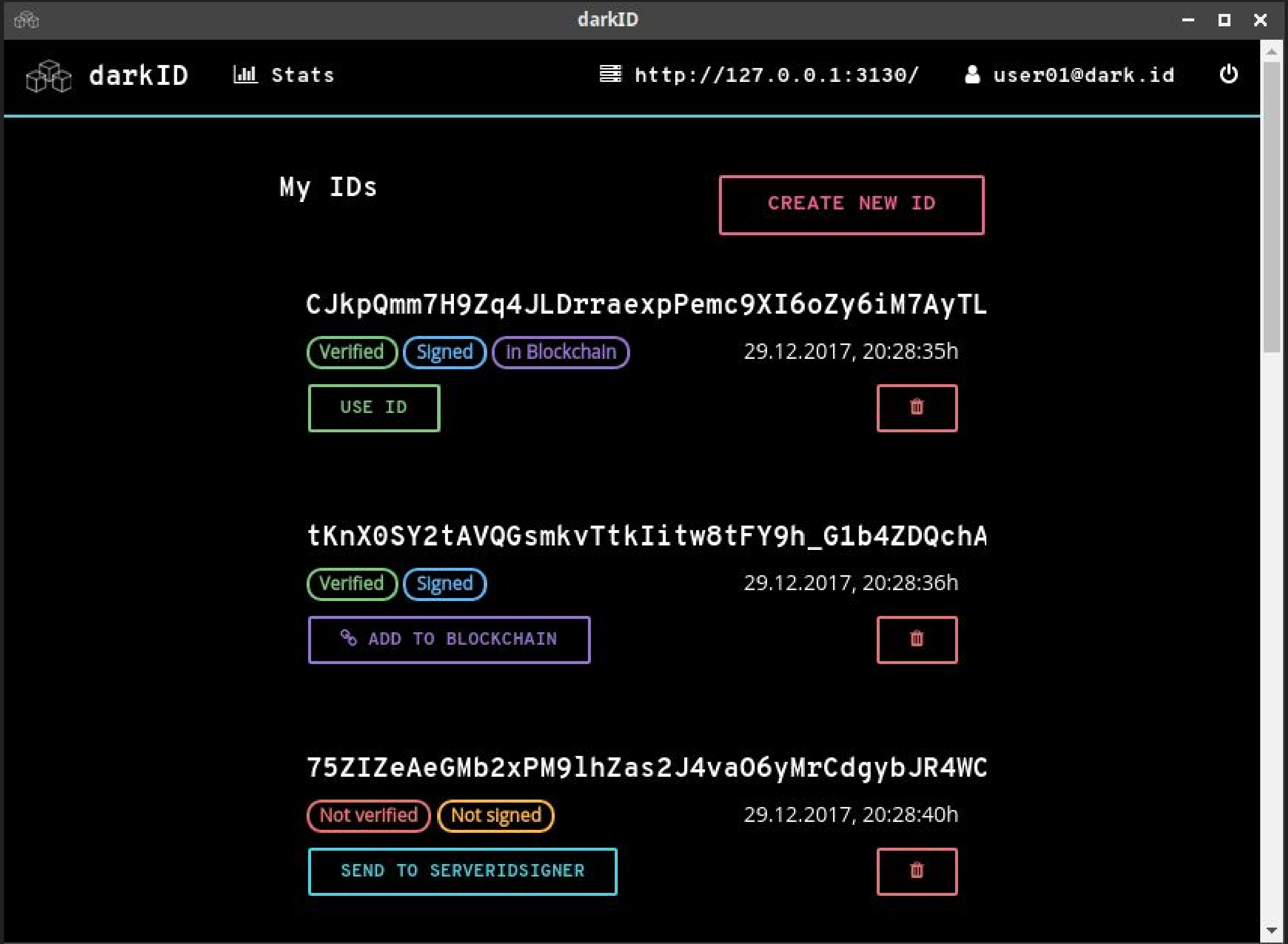


- server-ID-signer
  - The server where the user creates a non anonymous account
  - Also is the server that blind signs the Anonymous ID of the users
  - Have the webapp (frontend) to interact through a GUI interface
- Desktop App
  - Asdf
- ethereum Smart Contracts
  - Where the darkIDs are stored

Desktop App



Desktop App



darkID

darkID   Stats   http://127.0.0.1:3130/   user01@dark.id

My IDs

CREATE NEW ID

CJkpQmm7H9Zq4JLDrraexpPemc9XI6oZy6iM7AyTL

Verified Signed In Blockchain   29.12.2017, 20:28:35h

USE ID

tKnX0SY2tAVQGsmkvTtkIitw8tFY9h_G1b4ZDQchA

Verified Signed   29.12.2017, 20:28:36h

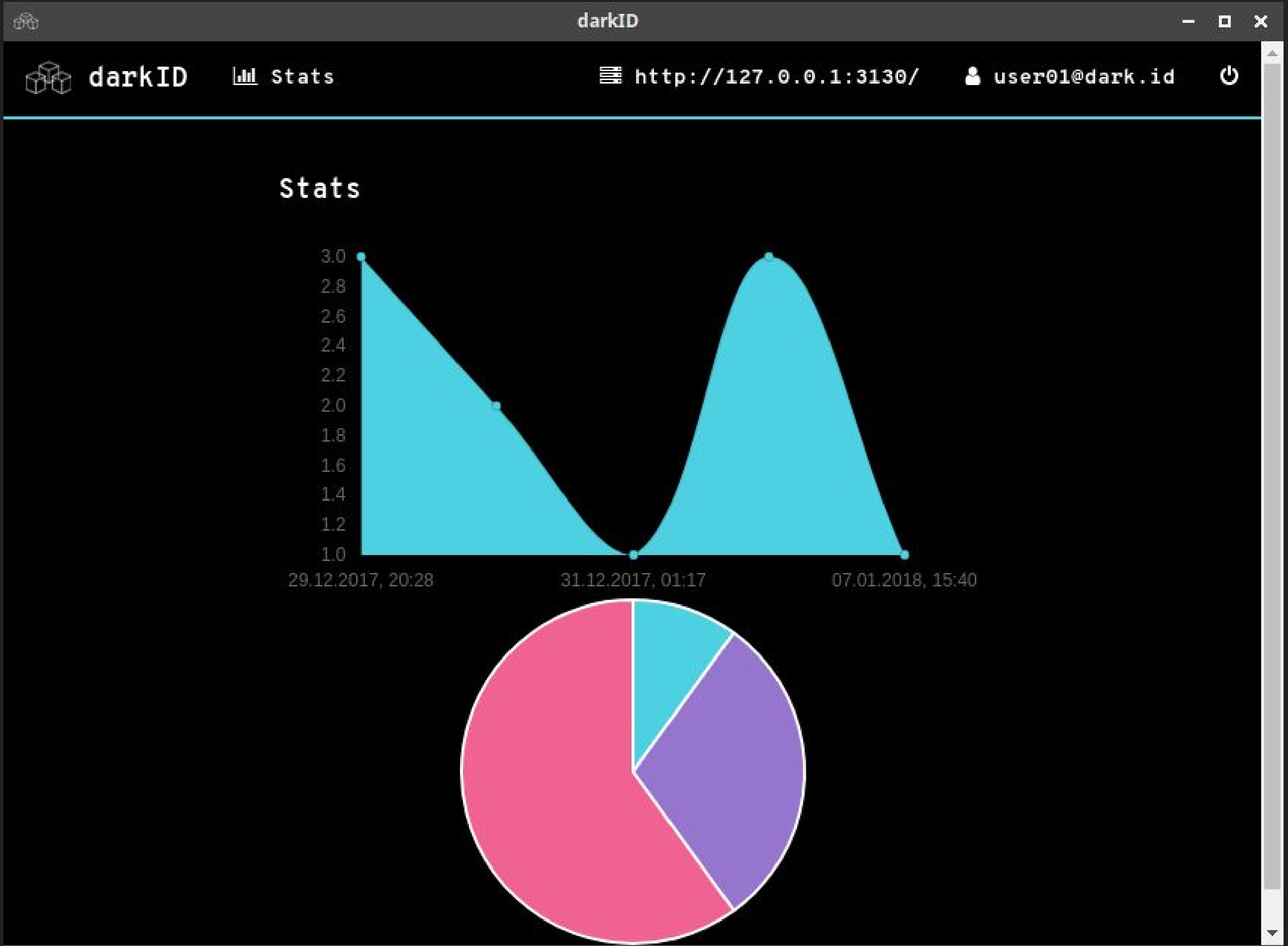
ADD TO BLOCKCHAIN

75ZIZeAeGMb2xPM9lhZas2J4va06yMrCdgybJR4WC

Not verified Not signed   29.12.2017, 20:28:40h

SEND TO SERVERIDSIGNER

Desktop App



darkID
darkID
Stats
http://127.0.0.1:3130/
user01@dark.id
Stats
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
29.12.2017, 20:28
31.12.2017, 01:17
07.01.2018, 15:40

How darkID works? Step by step process

1. The user registers a non anonymous user (using email, phone, password, etc), and performs the login with that user
2. The user, locally, generates a RSA key pair (private key & public key)
3. The user blinds his Public-Key with the server-ID-signer Public-Key
4. The user's Public-Key blinded, is sent to the server-ID-signer
5. The server-ID-signer Blind Signs the Public-Key blinded from the user, and returns it to the user
6. The user unblinds the Public-Key signed by the server-ID-signer, and now has the Public-Key Blind Signed by the server-ID-signer

How darkID works? Step by step process

7. The user sends the Public-Key blind signed to the p2p network
8. The peers verify that the Public-Key Blind Signed is correctly signed by the server-ID-signer, if it is, they add the Public-Key to the Ethereum Blockchain, inside a new block
9. Then, when the user wants to login into a platform, just needs to put his Public-Key
10. The platform goes to the Ethereum Blockchain, to check if this Public-Key is registered in the blockchain
11. The platform sends a message encrypted with the user Public-Key, and the user returns the message decrypted with the Private-Key, to verify that is the owner of that Public-Key

RSA and Blind Signature

RSA encryption system

https://en.wikipedia.org/wiki/RSA_cryptosystem

- Public parameters: (e, n)
- Private parameters: (d, p, q, phi, sigma)
- Public-Key = (e, n)
- Private-Key = (d, n)
- Encryption: $c \equiv m^e \pmod{n}$
- Decryption: $c^d \equiv (m^e)^d \equiv m \pmod{n}$

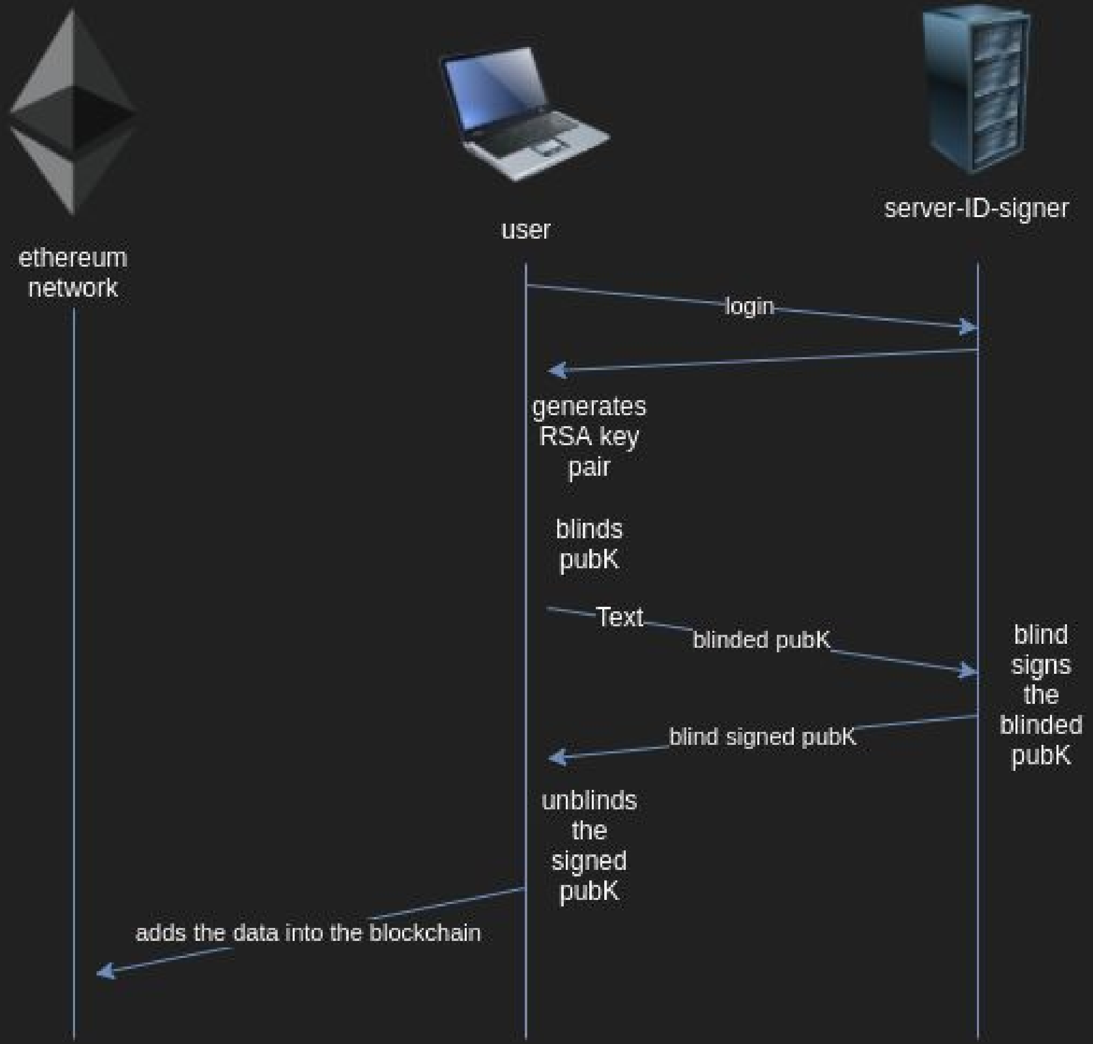
Blind signature process

https://en.wikipedia.org/wiki/Blind_signature

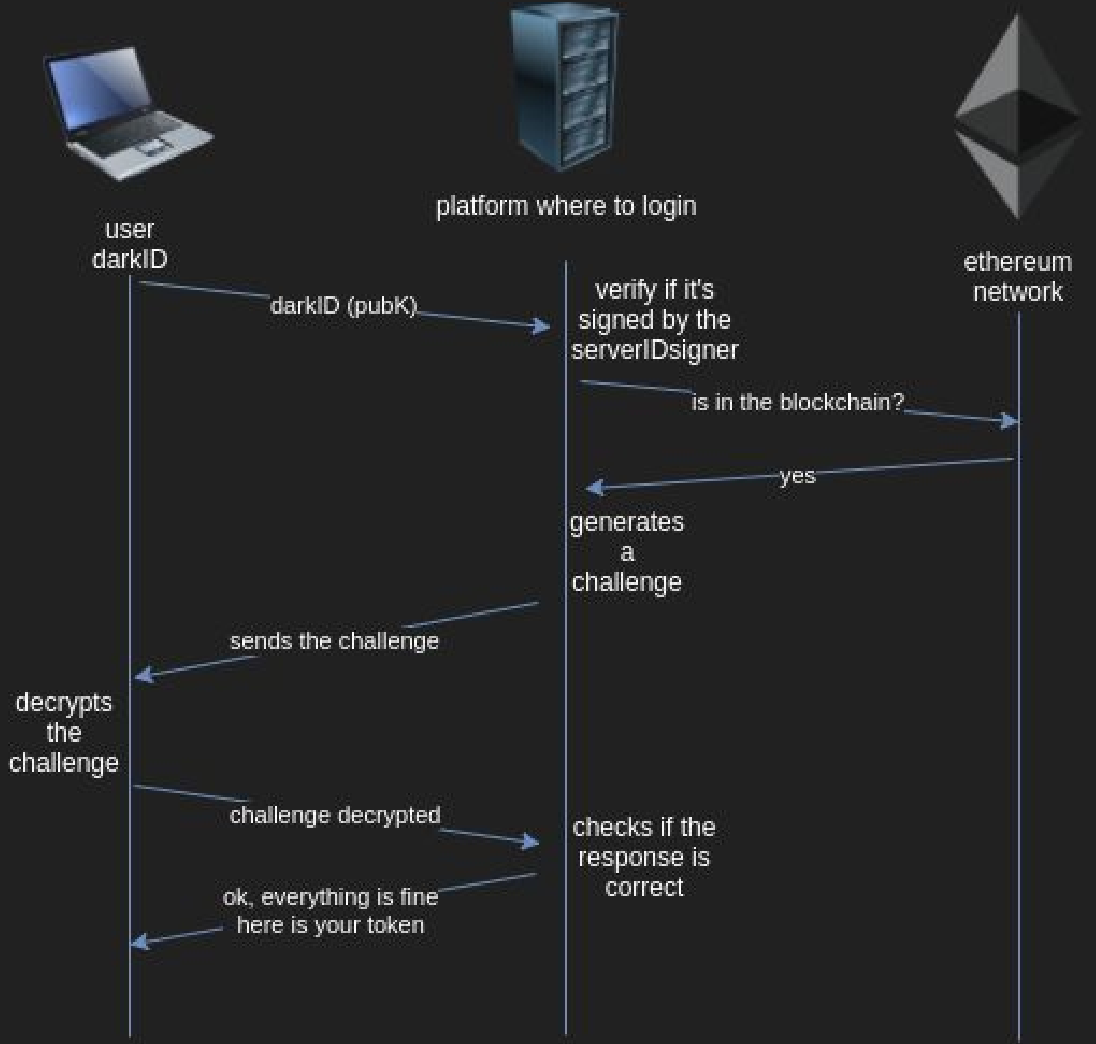
- m is the message (in our case, is the Public-Key of the user to be blinded) $m' \equiv mr^e \pmod{N}$
- server-ID-signer blind signs m' $s' \equiv (m')^d \pmod{N}$.
- user can unblind m, to get m signed $s \equiv s' \cdot r^{-1} \pmod{N}$
- This works because RSA keys satisfy this equation $r^{ed} \equiv r \pmod{N}$
- and this

$$s \equiv s' \cdot r^{-1} \equiv (m')^d r^{-1} \equiv m^d r^{ed} r^{-1} \equiv m^d r r^{-1} \equiv m^d \pmod{N}$$

darkID Generation

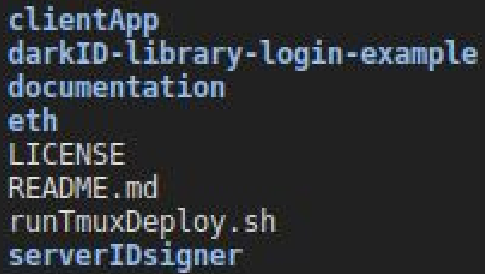


DARKID LOGIN



darkID

Demo time



```
clientApp
darkID-library-login-example
documentation
eth
LICENSE
README.md
runTmuxDeploy.sh
serverIDsigner
```

Conclusions

- It was funny starting to implement a p2p network and a blockchain from scratch
- But makes no sense having not so much time in a short university subject
- ethereum smart contracts have a lot of potential uses
- Current version is far from a stable release for real use
- Can be an option for a Final Degree Project:
  - Develop an entire blockchain from scratch, with some alternative to PoW and PoS
- darkID have lots of applications
  - For social networks, anonymous voting systems, leaks system, …
  - Better use over Tor network