Bridging the cyber and physical worlds using blockchains and smart contracts

Nikos Fotiou, Vasilios A. Siris, Spyros Voulgaris, George C. Polyzos
Motivation

• IoT devices have limitations and cannot interact with blockchains/smart contracts
  – Limited computational power and storage
  – Limited network connectivity
  – Security and trust issues

• The output of an actuation operation cannot be easily verified using cyber means
Contributions

• realistic approach for paid IoT interactions:
  ➢ limit loss in case of disruption of actuation
    o micro-payments for micro-transactions
    o make blockchain related micro-transactions efficient/inexpensive
• blockchain-based micro-payments to constrained IoT devices
  – incapable of
    • performing public-key encryption
    • (directly) participating in the blockchain
    • storing blockchain-related secrets.
• enable “payment delegation”
  – allowing users without blockchain credentials to pay
    • up to a pre-configured amount
    • for a specific service
• support many-to-one payments
  – enabling multiple users that share the same blockchain credentials to pay for a service
• a presently feasible solution
  – that relies on existing, already deployed technologies
H2020 **SOFIE: Secure Open Federation of Internet Everywhere**

- Applying Distributed Ledger Technology to
  - securely and openly federate IoT platforms
- *interconnected* distributed ledgers
  - decentralized business platforms
  - interconnection of diverse IoT systems
  - accessible metadata
  - open business rules on how to connect to platforms
  - securely record audit trails to be used to resolve disputes

http://www.sofie-iot.eu/ 

polyzos@aueb.gr
A solution

• We argue that the general cyber-real world interaction problem can not be easily solved

Damage control/limit potential loss
  – In case something goes wrong, the loss is a small pre-configured amount of money

• We leverage two existing solutions
  – Payment channels
  – Hash-based one time password (HOTP – RFC4226)
Setup

AS

Client

Device

Smart contract

polyzos@aueb.gr
High-Level System Perspective

- A client (or his owner) makes a “deposit” to a smart contract
- The client requests from an AS an “one-time password”
  - for invoking the actuation process for 1 time slot
- The password is exchanged for a “payment receipt”
- The receipt can be used by the AS to claim, from the smart contract, (part of) the deposit
- If a client needs more passwords, it produces more receipts...
High-Level System Properties

• A deposit is claimed using only a single payment receipt
  – even in the case of many-to-one payments
  – minimizes the interactions with the smart contract and makes the smart contract implementation simpler

• Payment receipts are provided off-chain
  – generation & validation of receipts involves only digital signatures computation
  – generation & evaluation of an one-time password involves the computation of a keyed hash message authentication code (HMAC)
  – this process is fast -> small time slots can be used
    • minimizing the losses in case of service disruption

• A device and an AS have to be pre-configured with a shared secret key
  – no further interaction is required between these two entities

• The channel client-device does not have to be secure
  – as opposed to the channel between a client and an AS

• Except from the validation of an one-time password, a device does not have to perform any other operation
BUILDING BLOCKS
Payment channel: setup

Contract

Deposit X tokens for AS

Client
Payment channel: Micropayments

AS

Client

Sign(k tokens)

Sign(2*k tokens)

Sign(3*k tokens)
Payment channel: closing the channel

AS

Contract

Client

Sign(3*k tokens)

pay 3*k

return X - 3*k

polyzos@aueb.gr
keyed Hash Message Authentication Code (HMAC)

One-Time Password (HOTP)
TRIVIAL CONSTRUCTION
Device access

Device access protocol:

1. AS signs a message with the client’s key and a counter:
   \[ \text{Sign}(\text{Client, amount}) \]
2. AS sends the signed message to the client:
   \[ \text{HMAC}(sk, \text{counter}) \]
3. Client signs the message with the client’s key and the incremented counter:
   \[ \text{Sign}(\text{Client, amount } \text{++}) \]
4. Client sends the signed message to the device:
   \[ \text{HMAC}(sk, \text{counter++}) \]
5. Device verifies the signature and sends the HMAC with the counter to the client:
   \[ \text{HMAC}(sk, \text{counter}) \]
6. Client verifies the HMAC and sends the HMAC with the incremented counter to the device:
   \[ \text{HMAC}(sk, \text{counter++}) \]
Channel close

- AS
- Contract
- Client Owner

Sign(Client, amount) → pay → return
CLIENTS WITHOUT ACCESS TO THE BLOCKCHAIN
Setup

Client owner

AS

Device

polyzos@aueb.gr
Payment channel setup

Deposit X amount for AS on behalf of PKclient
Device access

\[
\text{Device access}
\]

Device access protocol:

1. **AS** sends to **Client**:
   - \( \text{Sign}(\text{Client, amount}) \)
   - \( \text{HMAC}(sk, \text{counter}) \)

2. **Client** sends to **Device**:
   - \( \text{HMAC}(sk, \text{counter}) \)

3. **Client** receives from **Device**:
   - \( \text{HMAC}(sk, \text{counter}) \)

4. **Client** performs:
   - \( \text{Sign}(\text{Client, amount++}) \)
   - \( \text{HMAC}(sk, \text{counter++}) \)

5. **Client** sends to **Device**:
   - \( \text{HMAC}(sk, \text{counter++}) \)

6. **Device** receives:
   - \( \text{HMAC}(sk, \text{counter++}) \)
ONE CLIENT OWNER
MULTIPLE CLIENTS
Setup

Client owner

Device

Device

Device

Device

AS

polyzos@aueb.gr
Challenges

• 1. Store all legitimate public keys
• 2. Close the channel with a single transaction
Store all client keys in a Merkle tree

Client1

Client2

Client3

Client4
Payment channel setup

Contract

Deposit X amount for AS on behalf of Root

Client owner
Prove membership

H()

Client1

H()

Client2

H()

Client3

H()

Client4

polyzos@aueb.gr
Challenges

• 1\textsuperscript{st} Store all legitimate public keys
• 2\textsuperscript{nd} Close the channel with a single transaction
The straw man ledger

AS

Client X

Sign(Client X, amount)

HMAC(sk, counter)
The straw man ledger

AS

Client X

→ Sign(Clien X, amount)
The straw man ledger

Last Transaction

Sign(Client X, amount)
The straw man ledger

AS

Client Z

Last Transaction

Sign(Client X, amount)

Sign(Client Z, amount+1)

Sign(Client X, amount)
The straw man ledger

- AS
- Client Z

Last Transaction

- Sign(Client X, amount)
- Sign(Client Z, amount+1)

HMAC(sk’,counter’)

- Sign(Client X, amount)
- Sign(Client Z, amount+1)
The straw man ledger

AS

Last Transaction

Sign(Client Z, amount+1)

Sign(Client L, amount+2)

HMAC(sk'',counter'')

Client L

Sign(Client X, amount)

Sign(Client Z, amount+1)

Sign(Client L, amount+2)

polyzos@aueb.gr
Channel close

Sign(Client L, amount+2)

Check if Client L is authorized

pay

return
Implementation and Evaluation

• Implementation with Ethereum smart contracts
• Public-private key pairs with secp256k1
• HMAC with SHA256
• Merkle tree with keccak256
  – Hash function recommended for Ethereum Smart Contracts
• Cost of Open and Close:
  – 3rd construction: 4 cars
• Opening deposit: 14.5 sec
Conclusions

• realistic approach for paid IoT interactions:
  • blockchain-based micro-payments to constrained IoT device owners
    – payment delegation
  • efficiently support groups of clients (1 owner)
  • a presently feasible solution

Future Work

• Advanced Ledger and ILP
• Key revocation
Bridging the cyber and physical worlds using blockchains and smart contracts

Nikos Fotiou, Vasilios A. Siris, Spyros Voulgaris, George C. Polyzos

Thanks!

polyzos@aueb.gr