Ethereum Transaction and Smart Contracts among Secure Identities

*Francesco Buccafurri, Gianluca Lax, **Lorenzo Musarella** and Antonia Russo*
lorenzo.musarella@unirc.it, PhD Student
University «Mediterranea» of Reggio Calabria

2nd Distributed Ledger Technology Workshop (DLT 2019)

Pisa, 11/02/19

Agenda

01 **Background**

Blockchain, eIDAS, IBE

02 **Our Proposal**

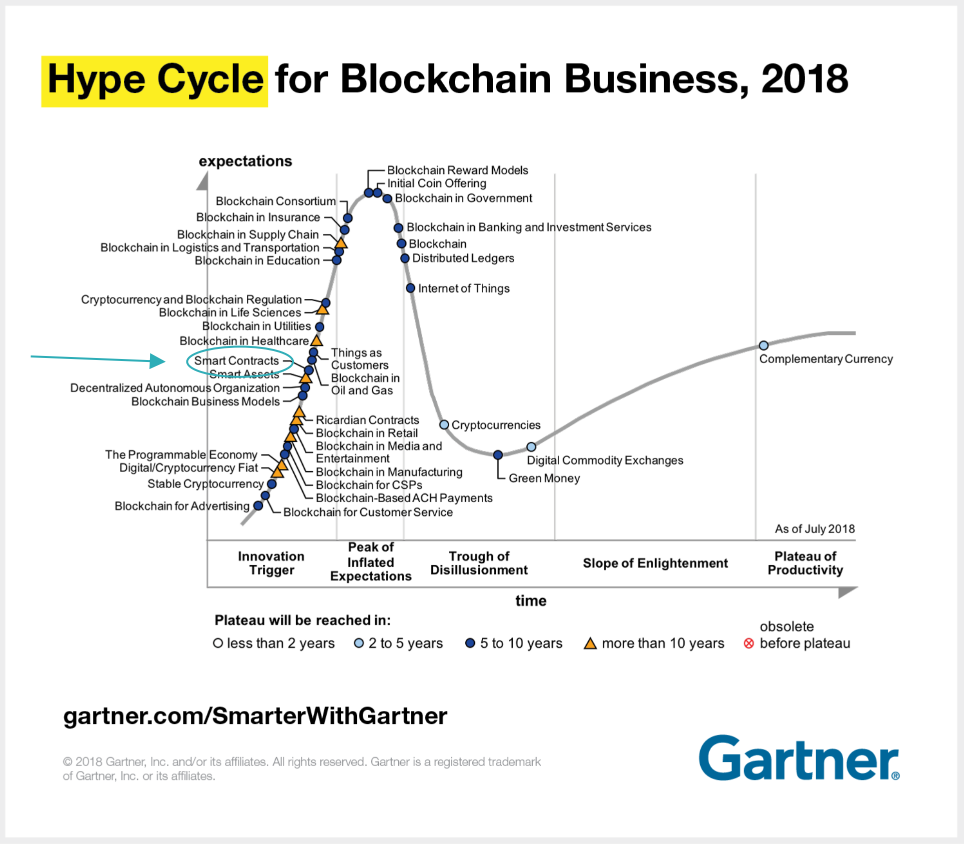
The scenario and our solution

03 **Conclusion**

01. Background

Blockchain, eIDAS, IBE

Blockchain


Hype Cycle for Blockchain Business, 2018
expectations
Blockchain Reward Models
Initial Coin Offering
Blockchain in Government
Blockchain Consortium
Blockchain in Insurance
Blockchain in Supply Chain
Blockchain in Logistics and Transportation
Blockchain in Education
Blockchain in Banking and Investment Services
Blockchain
Distributed Ledgers
Internet of Things
Cryptocurrency and Blockchain Regulation
Blockchain in Life Sciences
Blockchain in Utilities
Blockchain in Healthcare
Smart Contracts
Smart Assets
Decentralized Autonomous Organization
Blockchain Business Models
Things as Customers
Blockchain in Oil and Gas
Ricardian Contracts
Blockchain in Retail
Blockchain in Media and Entertainment
Blockchain in Manufacturing
Blockchain for CSPs
Blockchain-Based ACH Payments
Blockchain for Customer Service
The Programmable Economy
Digital/Cryptocurrency Fiat
Stable Cryptocurrency
Blockchain for Advertising
Cryptocurrencies
Digital Commodity Exchanges
Green Money
Complementary Currency
As of July 2018
Innovation Trigger
Peak of Inflated Expectations
Trough of Disillusionment
Slope of Enlightenment
Plateau of Productivity
time
Plateau will be reached in:
less than 2 years
2 to 5 years
5 to 10 years
more than 10 years
obsolete before plateau
gartner.com/SmarterWithGartner
© 2018 Gartner, Inc. and/or its affiliates. All rights reserved. Gartner is a registered trademark of Gartner, Inc. or its affiliates.
Gartner

Blockchain

Blockchain 2.0 and Smart Contracts

Two kinds of accounts:

1. EOAs
2. Smart Contracts (SC)

Messages and Transactions:

- Messages are sent from a SC to another SC
- Transactions are sent from EOAs

What if an EOA isn't registered yet on the service of the application platform implemented over Blockchain?

Public Digital Identity System

It must be compliant with the eIDAS regulation[1]

- Digital Identities are independent from the specific application platform. This allows the design of flexible, dynamic and interoperable services

- We refer to the Italian System of Public Digital Identity (SPID)

spid
Sistema Pubblico
di Identità Digitale

- It is necessary to find a secure way to link digital identities with Ethereum addresses

1. https://ec.europa.eu/futurium/en/content/eidas-regulation-regulation-eu-ndeg9102014

Identity Based Encryption

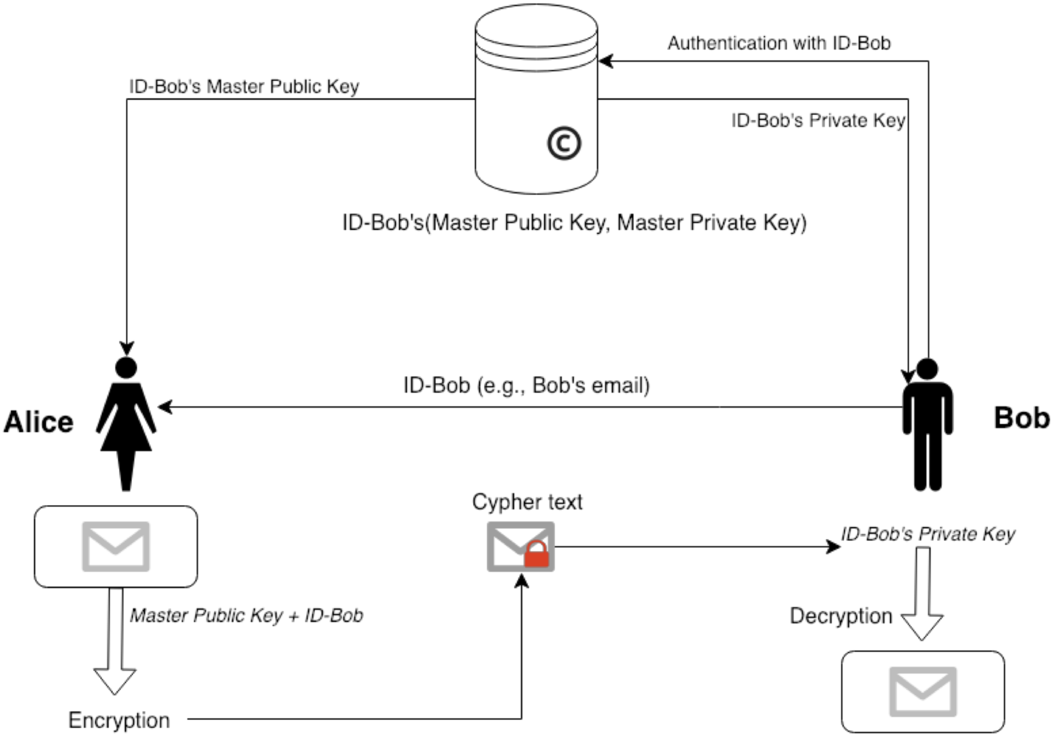
Basics

- Identity-based systems allow any party to generate a public key from a known identity value such as an ASCII string (e.g., email address);

- Identity-based systems requires a Private Key Generator (PKG) as Trusted Third -Party;

Identity Based Encryption

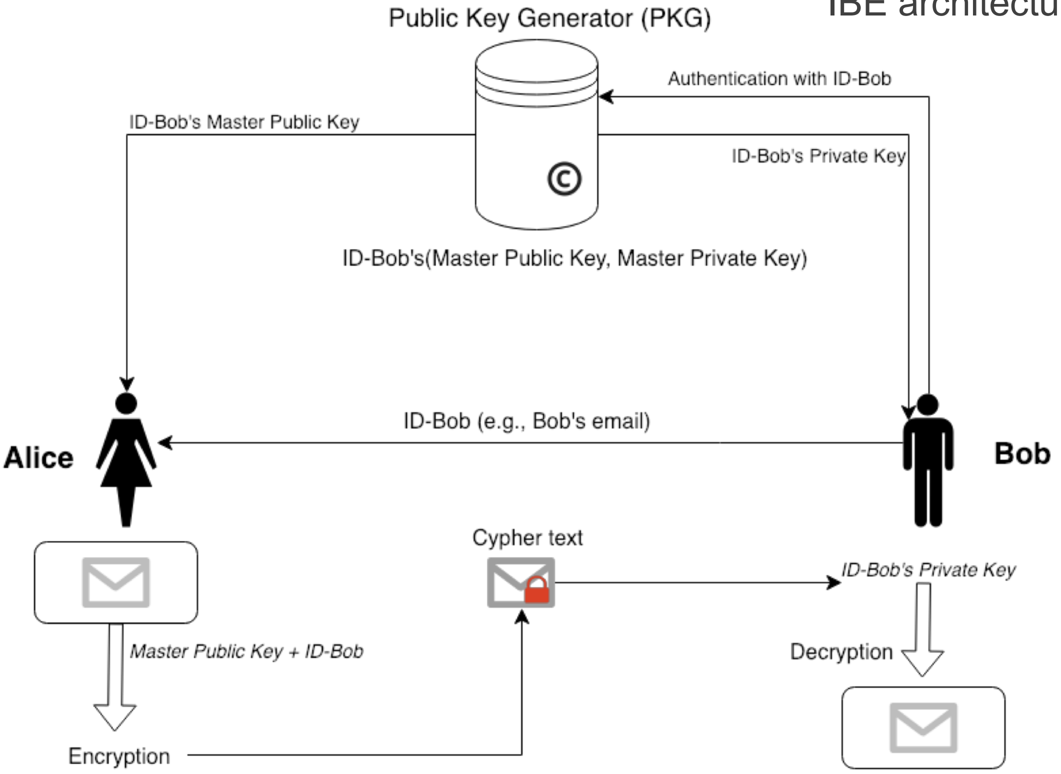
IBE architecture



- The PKG generates both Master Private Key and Master Public Key from the known identity (e.g Bob's email);
- The PKG stores the Master Private Key while it publishes the Master Public Key;
- In this way, there is no need to distribute public keys ahead of exchanging encrypted data;

Identity Based Encryption

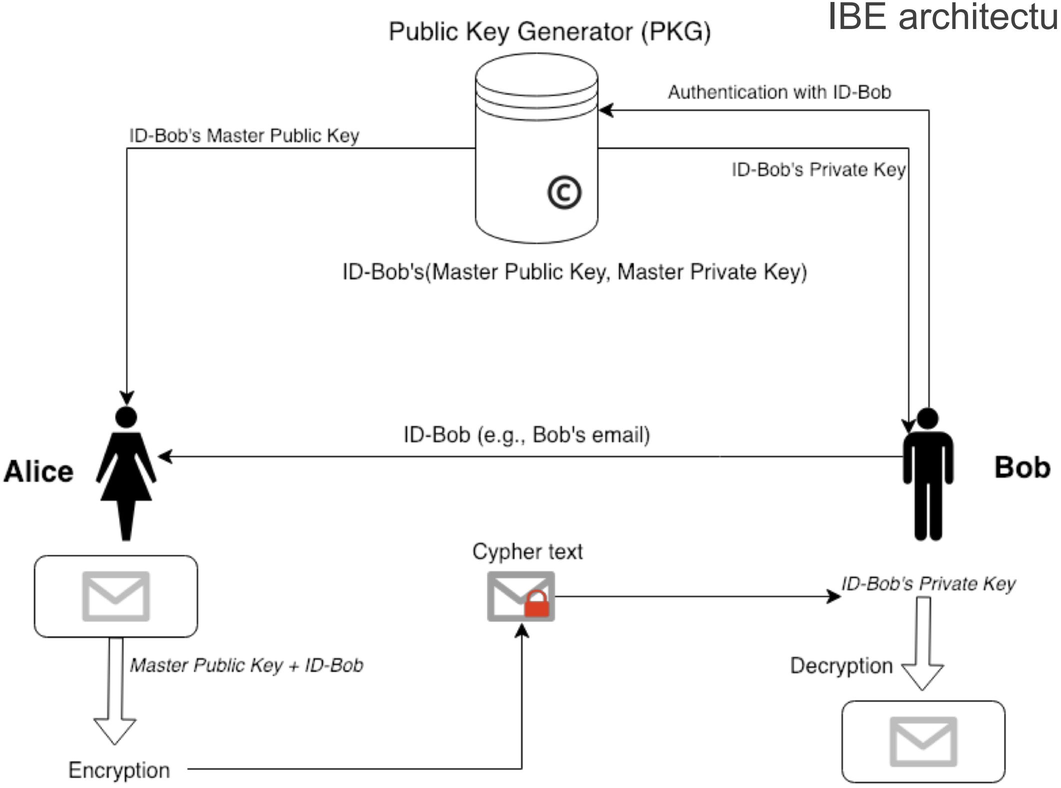
IBE architecture



- When **Alice** wants to send an encrypted messa ge to **Bob**, she has to know ID-Bob (e.g, Bob's email);
- She takes from PKG the Master Public Key of ID -Bob and she computes the Bob's public key cor responding to the identity by combining the Mast er Public Key and the ID-Bob;
- She sends the encrypted message;
- When **Bob** receives the message, he must auth enticate to the PKG to obtain his private key;

Identity Based Encryption

IBE architecture



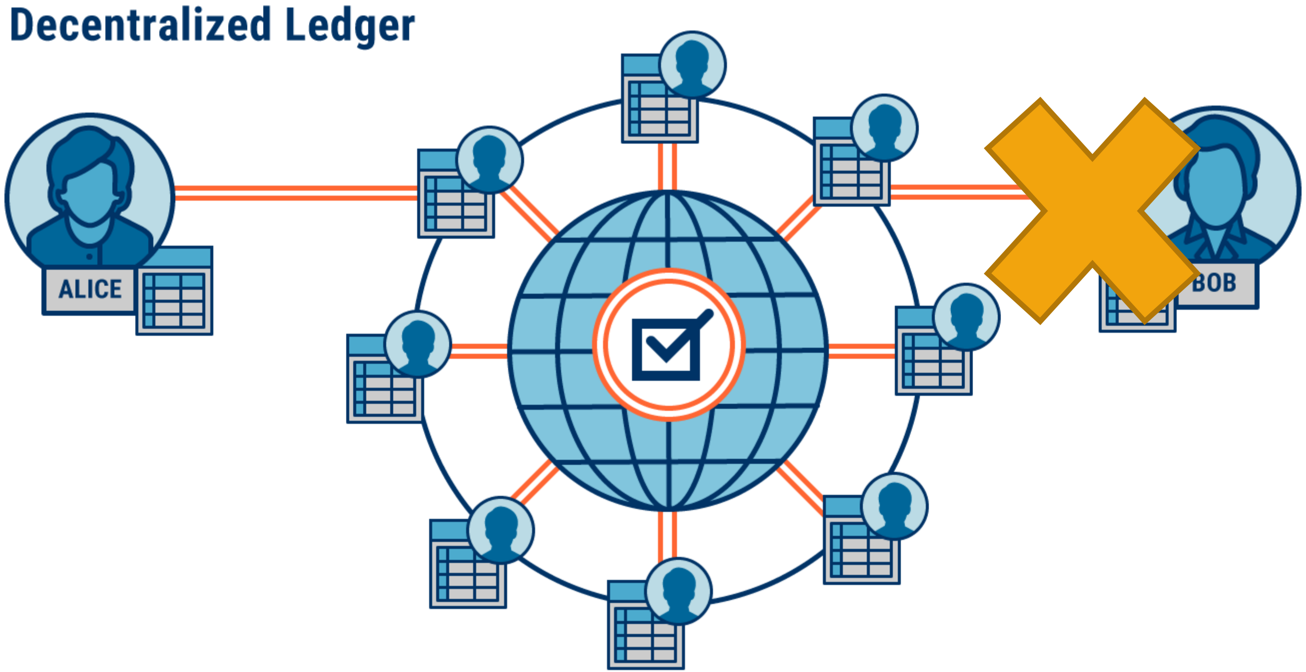
- After the successful authentication, PKG generates, from ID-Bob's Master Private Key, the private key and PKG gives it to **Bob**;
- Now **Bob** can decrypt the message received from **Alice**.

02. Our Proposal

The scenario and our solution

Our Proposal

The scenario



Our Proposal

The scenario: actors

- In our solution, we have the following types of entity:
  - An user using a digital identity for authentication;
  - A public identity digital system with Identity Provider IP;
  - An IBE system with PKG;
  - A Distributed Ledger allowing smart contracts (Ethereum).

Our Proposal

The scenario: types of operation carried out by users

1. *Digital Identity Registration:*

A public digital identity is identified by the pair $< username, IP >$, where *IP* is the identifier of the Identity Provider and the *username* is a string.

Furthermore, any Public Digital Identity System compliant with the eIDAS defines also an *Universal ID* UID.

Our Proposal

The scenario: types of operation carried out by users

2. *IBE private key gathering:*

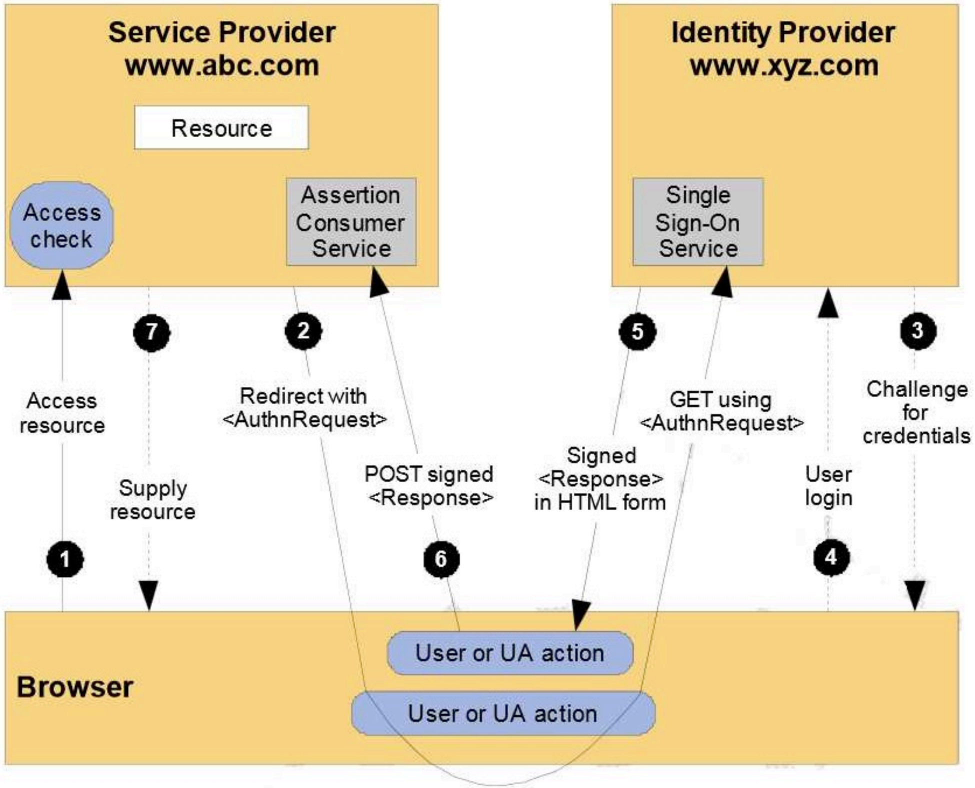
- As we said before, to obtain the IBE private key, a user must contact the PKG of the IBE service and he must authenticate successfully to the PKG;
- Then, the PKG authenticates the user by an eIDAS-compliant scheme.

Our Proposal

The scenario: types of operation carried out by users

- IBE acts as a Service Provider;

- The structure is compliant with SAML 2;



SAML 2

Our Proposal

The scenario: types of operation carried out by users

3. *Blockchain Binding*

- In this operation, an user associates his IBE public key IBE_P^K with his blockchain a ddress A;

- First, the user generates a pair of private and public blockchain keys;

- The blockchain address A is computed as the cryptographic hash of the public key;

Our Proposal

The scenario: types of operation carried out by users

3. *Blockchain Binding*

- Then, the user generates a transaction from A to A on the blockchain, having in the data field $< UID, E(A) >$;
- UID is the *Universal ID*, while $E(A)$ is the encryption of the blockchain address with the *IBE* private key, so that only him can compute $E(A)$;
- This transaction is called *binding transaction;*
- The user links his public digital identity to the blockchain address A.

Our Proposal

The scenario: types of operation carried out by users

4. Transaction

Let suppose that **Alice** wants to send to **Bob** v value (cryptocurrency, token, …) with a blockchain transaction/operation:

- First, she obtains the UID_{Bob} and she calculates the corresponding public key IBE_P^K;
- By calling a function of the smart contract, she looks for a *binding transaction B w ith* $< UID_{Bob}, E(A_{Bob}) >$ in the data field:
    - If she finds it, she uses IBE_P^K to deciphers $E(A_{Bob})$ to verify the authenticity of the signature;

Our Proposal

The scenario: types of operation carried out by users

4. *Transaction*

- If the check is ok, **Alice** has obtained **Bob**'s blockchain address A_{Bob} and she can proceed with the transaction.

- If **Alice** does not find the *binding transaction B with* $< UID_{Bob}, E(A_{Bob}) >$ in the data field, this means that **Bob** exists but he does not joined yet the blockchain.

- *So, what happens now?*

Our Proposal

The scenario: types of operation carried out by users

4. *Transaction*

- **Alice** generates a blockchain transaction from A_{Alice} to a Smart Contract A_{SC} specifying both UID_{Bob} and v;
- The smart contract will store this *sleeping transaction* from A_{Alice} to A_{Bob} with value v.

Our Proposal

The scenario: types of operation carried out by users

5. *Cashing*

- This operation is carried out by an user who wants to receive the *sleeping transaction* sent to him before his registration (in our example, **Bob**).

- **Bob** generates a blockchain transaction, named *cashing transaction* from him to the smart contract, specifying his UID_{Bob} in the data field (*cash* function in the smart contract code);

- If the smart contract finds a *binding transaction* corresponding, it computes the IBE_P^K calculated from the UID_{Bob}.

Our Proposal

The scenario: types of operation carried out by users

5. *Cashing*

- To do that, the smart contract uses an Oracle (we used Oraclize), which returns the A_{Bob} from the UID_{Bob} following these steps:
  - the Oracle looks for the IBE public key IBE_P^K associated to UID_{Bob} *and it tries to decipher* $E(A_{Bob})$ with IBE_P^K;
  - If it obtains A_{Bob}, the cashing process can continue because UID_{Bob} was successfully verified;
  - Else, the user who claimed the *sleeping values* was not really Bob.

Our Proposal

The scenario: types of operation carried out by users

5. Cashing

- At the end, the smart contract extracts from the stored *sleeping transactions* those sent to **Bob** (if they exists);

- It is generated a new transaction to A_{Bob} for each *sleeping transaction* found.

Our Proposal

Smart Contract

```
pragma solidity ^0.4.25;

import "github.com/oraclize/ethereum-api/oraclizeAPI_0.4.25.sol";
import "github.com/Arachnid/solidity-stringutils/strings.sol";

contract SleepingEther is usingOraclize {
    mapping(bytes32=>string) uidMapping; //mapping between queryID and bool
    mapping(string=>uint) payUid; //mapping between UID and eth value to send
    address public addr;
    using strings for *;
    string pi;

    function pay(string uid) public payable {
         payUid[uid] += msg.value; // add the ether addressed to uid
     }

     function cash (string uid) public payable{
         if(payUid[uid]>0)
             if (oraclize.getPrice("URL") <= address(this).balance) {
                pi = "URL".toSlice().concat(uid.toSlice());
                bytes32 queryId = oraclize_query("URL", pi);
                uidMapping[queryId]=uid;
            }
     }

    function __callback (bytes32 myid, string result , string uid) public {
         if (msg.sender != oraclize_cbAddress())
           revert ();
         bytes memory tempEmptyStringTest = bytes(result);
         if(tempEmptyStringTest.length != 0){
             addr = parseAddr(result);
             uint tot= payUid[uidMapping[myid]];
             addr.transfer(tot);
             payUid[uid]=0;
         }
     }
}
```

03. Conclusion

Conclusion

- We enable on Ethereum the possibility to send money to users without the need to know their blockchain address or when they are not registered yet on the service;
- The suitable use of the secure digital identity guarantees that only the correct user receives money.
- In this work, we only treat the case in which a given amount of cryptocurrency is transferred, but the transfer of tokens with identifier can be easily implemented by using, for example, the interface ERC721.

DIIES Dipartimento di INGEGNERIA dell'INFORMAZIONE, delle INFRASTRUTTURE e dell'ENERGIA SOSTENIBILE

Lorenzo Musarella

lorenzo.musarella@unirc.it

PhD Student