Multi-layered Security Technologies

for hyper-connected smart cities

D4.5 P2P level security and M-Sec blockchains

December 2019

Grant Agreement No. 814917

Multi-layered Security technologies to ensure hyper-connected smart cities with Blockchain, BigData, Cloud and IoT

Project acronym	M-Sec
Deliverable	D4.5 P2P level security and M-Sec blockchains
Work Package	WP4
Submission date	December 2019
Deliverable lead	Georgios Palaiokrassas (ICCS)
Authors	Georgios Palaiokrassas (ICCS), Orfeas Voutyras (ICCS), Xavier Cases Camats (WLI)
Internal reviewer	Arturo Madela (TST), Aamir Bokhari (YNU)
Dissemination Level	Public
Type of deliverable	DEM
Version history	- V01, 11/12/2019, Georgios Palaiokrassas, Content added - V02, 18/12/2019, Orfeas Voutyras, Section 4 completed - V03, 19/12/2019, Xavier Cases Camats, Contribution to Section 2.1 - V04, 19/12/2019, George Palaiokrassas, Ready for Internal Review - V05 19/12/2019, Aamir Bokhari, 1st Internal Review - V06 19/12/2019, Arturo Madela, 2nd Internal Review - V07 20/12/2019, Georgios Palaiokrassas, answers to comments - V1.0 20/12/2019, Georgios Palaiokrassas, ready for submission

The M-Sec project is jointly funded by the European Union's Horizon 2020 research and innovation programme (contract No 814917) and by the Commissioned Research of National Institute of Information and Communications Technology (NICT), JAPAN (contract No. 19501).

Table of Contents

List of Tables

List of Figures

1. Introduction

The current document, deliverable ‘D4.5 P2P level security and M-Sec blockchains’, provides the first version of M-Sec developments related to blockchain technology and P2P level security. In detail, it presents three different demonstrators as well as the corresponding services and gives installation details. It also provides plans about future developments until the next iteration of this deliverable, namely D4.6 on M30. The main focus of the presented demonstrators and tools is to implement the M-Sec blockchain framework to facilitate the convergence of IoT security with blockchains in order to support an innovative smart city platform.

1.1 Relationship to other work packages and tasks

Task T4.3 receives input from WP2 tasks in particular from Task2.1 – “Use cases description” (which identify and describe use cases) and Task 2.2 “M-Sec Pilots: Definition, setup and citizens involvement”, since the implementations support the Use Cases and Pilots. Additionally, there is a strong dependency with T2.3 “Overall Integration”.

At the same time, task T4.3 interacts with WP3 – “Requirements, architecture for hyper connected smart cities”, in particular with Task3.1 – “System level and User level requirements” where M-Sec requirements are defined and consolidated, and also, with Task3.2 – “M-Sec architecture”, where the M-Sec architecture is defined.

Furthermore, T4.3 has dependencies with the rest of WP4 “Multi-layered Security technologies” Tasks and more specifically with T4.1 for IoT security and related services based on blockchain technology, T4.2 “Cloud and data level security”, T4.4 “Application level security” and T4.5 “Overall end-to-end security” focusing on the integration with respective implementations for encrypted data storage.

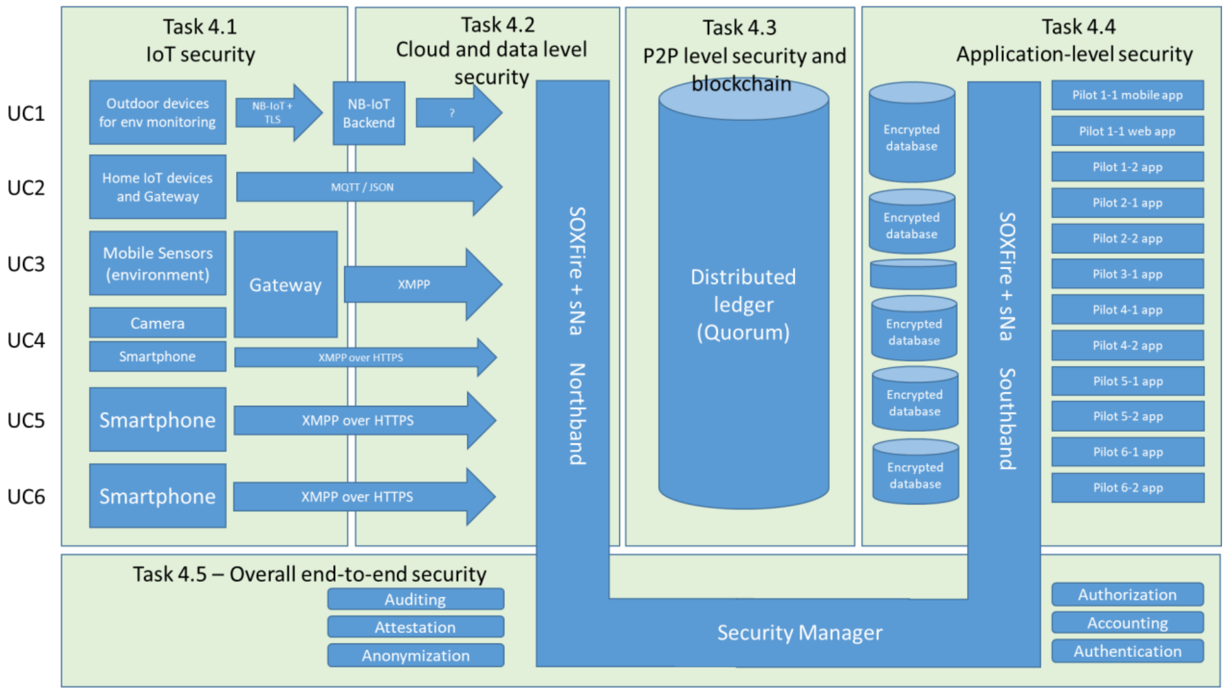


Figure 1. Overall M-Sec topology

1.2 Methodology followed

In order to enable the M-Sec paradigm, we researched different technologies and approaches of blockchain technology. We started the analysis with a detailed description of different blockchain frameworks, such as Ethereum, Hyperledger and Quorum, and tools we experimented with. Then we present three different demonstrators, which are the core of T4.3 and aim to support the different use cases and pilots.

2. Demonstration 1: Blockchain Framework and Middleware Services

2.1 General Description of the Prototype

The main focus of this Prototype is to implement the M-Sec blockchain framework, and to facilitate the convergence of IoT security with blockchains in order to support an innovative smart city platform. We used Ethereum-based blockchains as the basic foundation of M-Sec blockchain as it enables not only the exchange of value (M-Sec tokens) but also the enforcement of smart contracts, which provides an additional feature for the implementation and validation of the selected M-Sec use cases.

A milestone for the course of blockchain technology was the development of Ethereum project[1], offering new solutions by enabling smart contracts' implementation and execution. It is a suite of tools and protocols for the creation and operation of Decentralized Applications (DApps), "*applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third-party interference*".

It also supports a contract-oriented, high-level, Turing-complete programming language[2], allowing anyone to write smart contracts and create DApps. Smart contracts are mainly written in the programming language Solidity[3]-[4].

We have initially experimented with different Blockchain platforms, before concluding to Ethereum based blockchains such as Quorum and examined both public (unpermissioned) and private (permissioned) alternatives of the M-Sec blockchains. The most prominent among them was "Hyperledger", which is described in the next Section. Hyperledger implementation was considered as it can enable, through specific channels, the implementation of flexible blockchains with different permissions and authorization schemes.

The peer group management service is also part of the work covered in this task, as research will be pursued for defining how the blockchain networks are going to be self-organized and structured in the context of service provisioning so that they can form and operate the multi-layered architectures.

---

[1] J. Ray, "Ethereum Introduction," 11 12 2019. [Online]. Available: https://github.com/ethereum/wiki/wiki/Ethereum-introduction

[2]"White Paper," [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[3] Ethereum, "What is Ethereum?," [Online]. Available: http://www.ethdocs.org/en/latest/introduction/what-isethereum.html.

[4] Ethereum, "Solidity," Ethereum, [Online]. Available: http://solidity.readthedocs.io

Component Module 1: Hyperledger blockchain framework

We used three different open source Hyperledger Platforms and an overview of the overall architecture is presented in the diagram that follows (see Figure 2). The three projects are:

- Hyperledger Explorer: it provides details about the underlying blockchain such as the number of blocks, the transactions, the peers etc.
- Hyperledger Composer: it facilitates the development of smart contracts ("*chaincode*")
- Hyperledger Fabric: it provides the permissioned blockchain

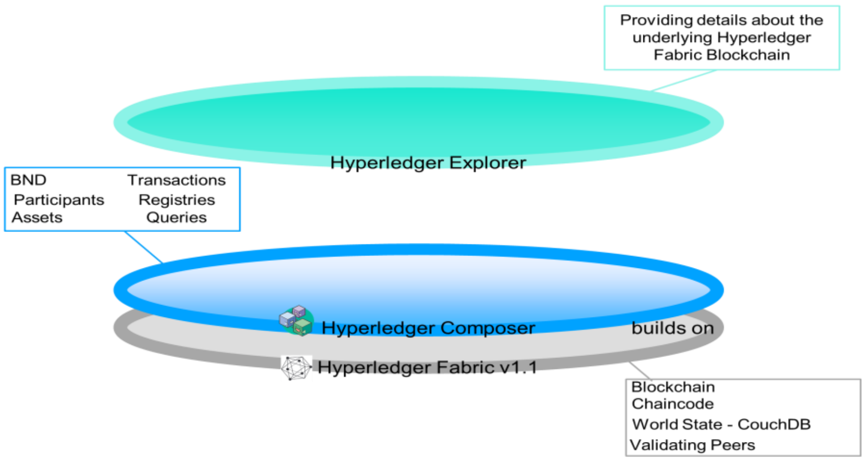
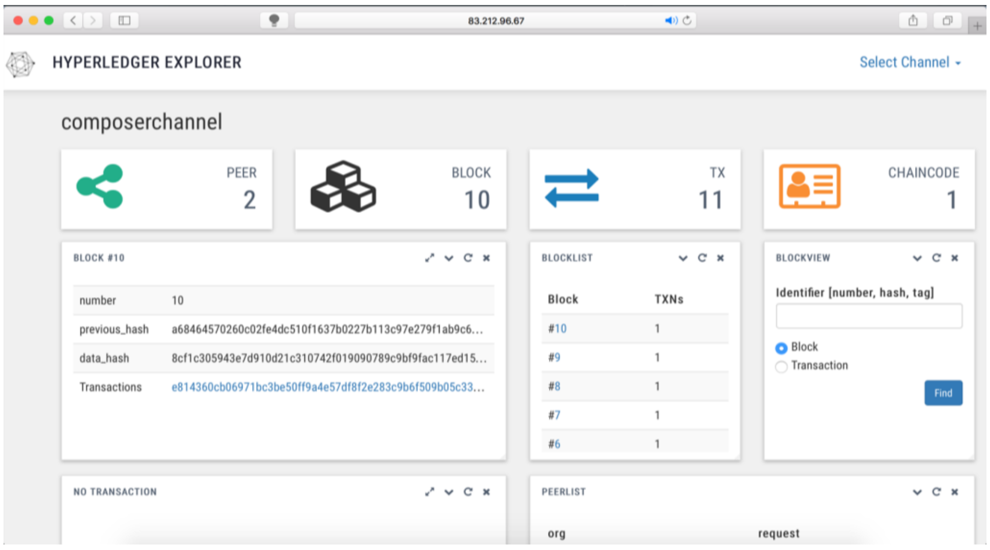


Figure 2. Representation of the architecture of our prototype based on Hyperledger

Our implementation was based on a permissioned blockchain, but we should note that by granting open access to all users, we would have a public-like blockchain.

Through Hyperledger Explorer (see Figure 3) we can inspect the Hyperledger Fabric Blockchain (Transactions, Blocks and others.)



composerchannel

PEER 2 | BLOCK 10 | TX 11 | CHAINCODE 1

BLOCK #10

number	10
previous_hash	a68464570260c02fe4dc510f1637b0227b113c97e279f1ab9c6...
data_hash	8cf1c305943e7d910d21c310742f019090789c9bf9fac117ed15...
Transactions	e814360cb06971bc3be50ff9a4e57df8f2e283c9b6f509b05c33...

BLOCKLIST

Block	TXNs
#10	1
#9	1
#8	1
#7	1
#6	1

BLOCKVIEW

Identifier [number, hash, tag]

Block [✓] Transaction

Find

NO TRANSACTION

PEERLIST

org	request

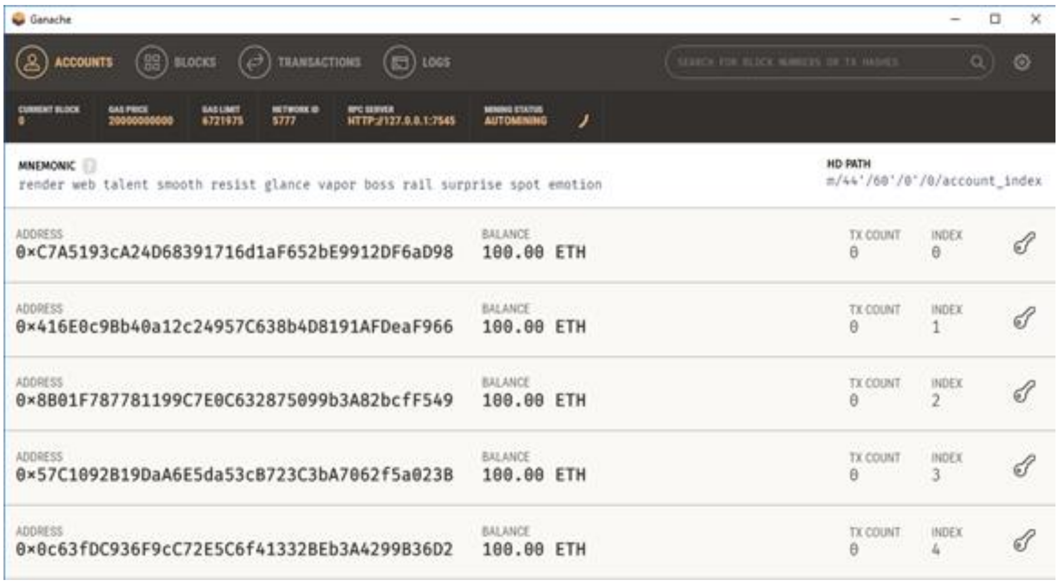
Figure 3. Hyperledger Explorer giving details about the underlying permissioned blockchain

Component Module 2: Ethereum & Quorum blockchain framework

In this Section we present the details regarding the blockchain platform, in which we develop the smart contracts that support the different use cases. As mentioned before, the different smart contracts are written in the programming language Solidity[5] .

1. *Private Ethereum Blockchain*

During the development process we have used a local private blockchain named Ganache[6], which allowed us extensive testing of the developed smart contracts. It provides a personal Ethereum blockchain which we can use to run tests, execute commands, and inspect the state while controlling how the chain operates. It provides a built-in explorer as shown in the following Figure 4 and allows us to quickly see the current status of all accounts, including their addresses, private keys, transactions and balances.



```
CURRENT BLOCK  GAS PRICE    GAS LIMIT  NETWORK ID  RPC SERVER              MINING STATUS
0              20000000000  6721975    5777        HTTP://127.0.0.1:7545   AUTOMINING

MNEMONIC                                                                    HD PATH
render web talent smooth resist glance vapor boss rail surprise spot emotion  m/44'/60'/0'/0/account_index

ADDRESS                                     BALANCE     TX COUNT  INDEX
0×C7A5193cA24D68391716d1aF652bE9912DF6aD98  100.00 ETH  0         0
0×416E0c9Bb40a12c24957C638b4D8191AFDeaF966  100.00 ETH  0         1
0×8B01F787781199C7E0C632875099b3A82bcfF549  100.00 ETH  0         2
0×57C1092B19DaA6E5da53cB723C3bA7062f5a023B  100.00 ETH  0         3
0×0c63fDC936F9cC72E5C6f41332BEb3A4299B36D2  100.00 ETH  0         4
```

Figure 4. Ganache Explorer allows us to examine all blocks, transactions, addresses and their balances

2. *Public Ethereum Blockchain*

Additionally, we deployed smart contracts on Public Ethereum Blockchain using browser IDE "Remix"[7]. Remix is an open source tool that supports smart contracts development on the browser and facilitates the deployment on local or public Ethereum-based blockchain platforms. We used Ropsten public Ethereum blockchain[8]. It is important to extensively test the smart contracts before we deploy them to the Quorum blockchain network (see next section), since the code can't be

[5] Ethereum, "Solidity," Ethereum, [Online]. Available: http://solidity.readthedocs.io.

[6] https:// www.trufflesuite.com/ganache
[7] https://remix.ethereum.org/
[8] https://ropsten.etherscan.io/

changed after deployment. To this direction, extensive testing was carried out on blockchains, by using these two testing solutions: Ganache Cli, as well as the Ropsten Test Net.

3. Quorum blockchain framework

Finally, the different smart contracts are written in the programming language Solidity[9] and are deployed on Quorum blockchain framework[10]. Quorum is a permissioned implementation of Ethereum which allows certified members to build and run decentralized applications that run on blockchain technology. It is open source platform and supports smart contract privacy. Both private and public smart contracts are validated by every node within the blockchain network. Additionally, Quorum provides privacy and transparency, both at transaction-level and network wide.

In each Quorum node consensus is achieved with the Raft or Istanbul BFT consensus algorithms instead of using Proof-of-Work. The P2P layer has been modified to only allow connections to/from permissioned nodes. In Ethereum the notion of Gas was introduced (the fee or pricing value required to successfully conduct a transaction or execute a contract on Ethereum blockchain platform), while in Quorum the pricing of Gas has been removed, although Gas itself remains.

One of the features of Quorum that are of great value for the component is the network and peer to peer permission management. This feature enables only the validated and authorized users to have access and be a part of the network. Also, Quorum provides enhanced transaction and smart contract privacy features. Permission-based nature of Quorum enables the constitution of private and public transaction getting the best of both worlds, open transactions are analogous to Ethereum but when it comes to the private transaction then it is confidential, and the data is not exposed to the public. Quorum adds privacy functions that allow for private transactions that are only visible to the transacting parties, while the other parties in the network would only see a hash. Finally, Quorum is considered to be very fast and being able to process even thousands of transactions per second, due to its efficient consensus mechanism which belongs to the family of Byzantine Fault Tolerance (BFT) mechanisms[11].

In order to develop and deploy the smart contracts to Quorum blockchain, we have used Truffle suite[12]. It is a development environment and testing framework using the Ethereum Virtual Machine (EVM). Additionally, we use Quorum Maker[13] (see Figure 5 below), which facilitates the deployment of smart contracts, offering visualization features to monitor the Quorum blockchain network and related blocks and transactions. Before we deploy the smart contracts to the blockchain network, we

[9] Ethereum, "Solidity," Ethereum, [Online]. Available: http://solidity.readthedocs.io.

[10] https://docs.goquorum.com/en/latest/].

[11] Vukolić M. (2016) The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication. In: Camenisch J., Kesdoğan D. (eds) Open Problems in Network Security. iNetSec 2015. Lecture Notes in Computer Science, vol 9591. Springer, Cham

[12] https://www.trufflesuite.com/

[13] https://github.com/synechron-finlabs/quorum-maker

extensively tested them on Ethereum blockchains using two testing solutions: Ganache Cli, as well as the Ropsten Test Net. Additionally, before being deployed on a larger Quorum Network, we used a Quorum test network consisting of seven nodes[14].

Smart contracts are written in Solidity so it is feasible to migrate from Quorum permissioned Blockchain Framework to public Blockchain Frameworks (e.g. Public Ethereum Network), since Solidity is the common programming language to Ethereum based blockchain frameworks.

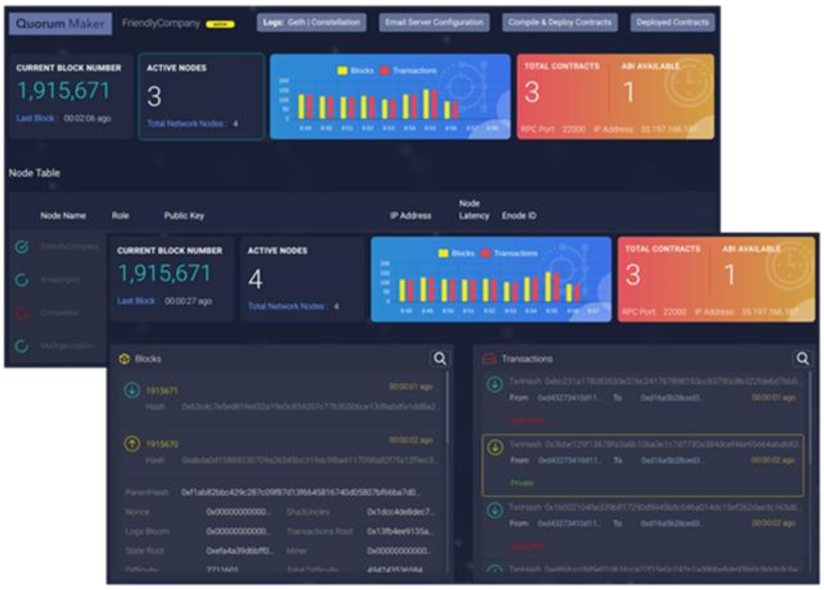


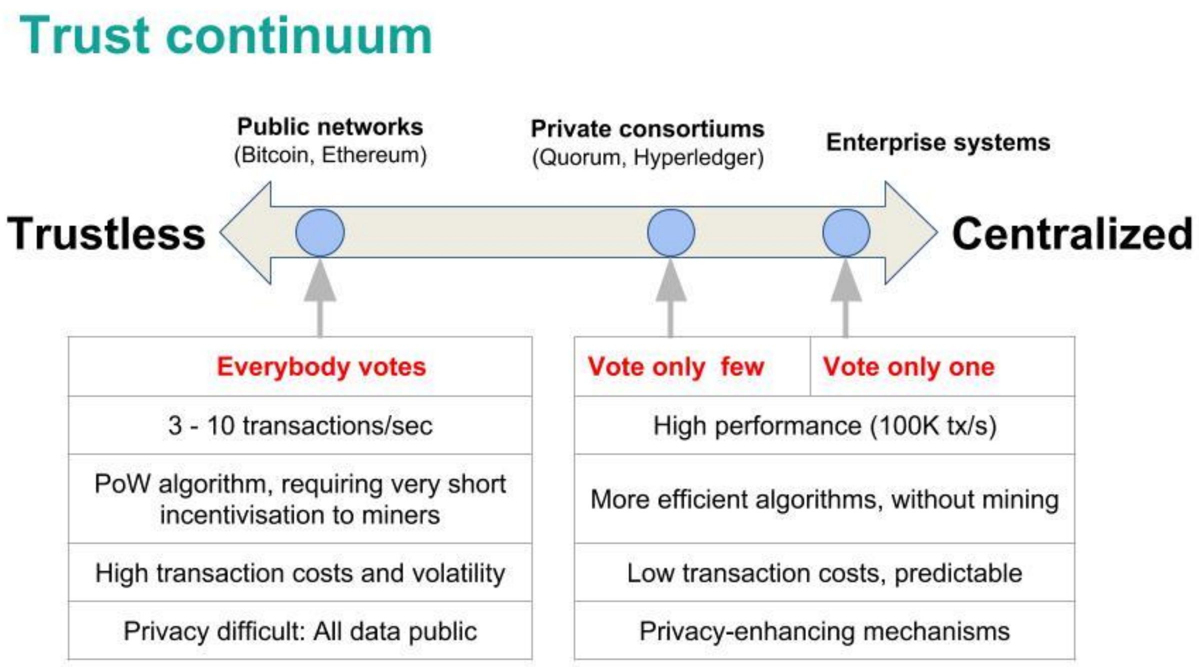
Figure 5. Details about blocks, transactions, addresses and smart contracts

4. ALASTRIA

Alastria is neither a public-permissionless network nor a private consortium, is a Public-Permissioned network it means that shares some of the properties of both types of networks, and it also has some requirements of its own.

Figure 6 describes the prevailing public-permissionless blockchain networks currently in production like Bitcoin or Ethereum have the very desirable property of being "*Trustless*". However, mainly due to the characteristics of the consensus algorithms used to achieve that property, they suffer from very well documented scalability problems. There are a lot of efforts being made to solve or alleviate the scalability problem, but as of today, the problem still exists and permissioned networks will always have several orders of magnitude better performance.

[14] https://github.com/jpmorganchase/quorum-examples/tree/master/examples/7nodes



Everybody votes	Vote only few	Vote only one
3 - 10 transactions/sec	High performance (100K tx/s)	
PoW algorithm, requiring very short incentivisation to miners	More efficient algorithms, without mining	
High transaction costs and volatility	Low transaction costs, predictable	
Privacy difficult: All data public	Privacy-enhancing mechanisms	

Figure 6. The Trust Continuum

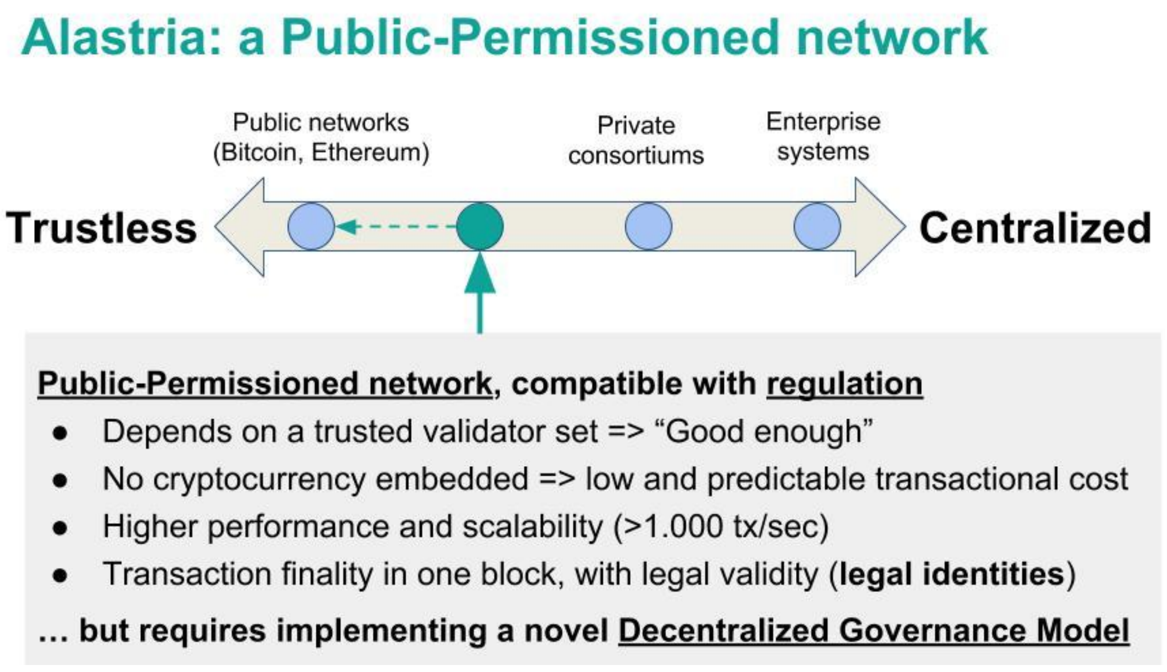
The Problems of the Public-Permissionless blockchain networks are:

- Scalabilty: The networks choose Decentralization and Security over Scalability.

  Taking into account the words of Vitalik Buterin describing the "*Blockchain trilemma*[15]", the trilemma claims that blockchain systems can only at most have two of the following three properties:
  - Decentralization, defined as the system being able to run in a scenario where each participant only has access to *O(c)* resources, where *c* refers to the size of computational resources available to each node (i.e. a regular laptop or small VPS "Virtual Private Server").
  - Scalability, defined as being able to process *O(n)* > *O(c)* transactions, where *n* refers to the size of the ecosystem in some abstract sense.
  - Security (or Safety), defined as being secure against attackers with up to *O(n)* resources

- Transaction Costs: High and Volatile
- Privacy: By default, in public blockchains like Bitcoin or Ethereum, transactions are executed by all nodes in the network, transactions are globally published and state data is not encrypted in most applications, so all participants have access to all data stored in the ledger without any restriction.

[15] https://medium.com/@aakash_13214/the-scalability-trilemma-in-blockchain-75fb57f646df



Alastria: a Public-Permissioned network

Public-Permissioned network, compatible with regulation

- Depends on a trusted validator set => "Good enough"
- No cryptocurrency embedded => low and predictable transactional cost
- Higher performance and scalability (>1.000 tx/sec)
- Transaction finality in one block, with legal validity (**legal identities**)

… but requires implementing a novel Decentralized Governance Model

Figure 7. Alastria in the Trust Continuum

In Public-Permissioned networks, the objective is to maximize decentralization and safety, even if this goes to the detriment of scalability. In this context, decentralization typically means the ability to transact anonymously but safely among individuals without the need for any intermediary acting as trusted party. It is often the case that the requirement to eliminate third parties is stronger than the requirement that the system be high-performance so it could be used as a general purpose transaction mechanism.

In Private Consortiums the objectives are generally different, and instead of trying to eliminate third parties at all costs, they try to use blockchain technology to improve efficiency and reduce costs of transaction among the partners composing the consortium. In many private consortiums, they want a shared database and transaction system so they can eliminate frictions and reduce costs of reconciliation.

As Figure 6 and Figure 7 describe, Alastria tries to be as public as possible, but without the disadvantages associated with public-permissionless networks.

As mentioned before, Alastria is not a Private Consortium but a Public-Permissioned network compatible with regulation instead. At a very high-level, the characteristics of Alastria are the following:

- It's permissioned, so every participant node has to be identified before it can participate in the network.
- No cryptocurrency embedded.
- A more efficient consensus algorithm, enabling higher performance and scalability.
- Transaction finality in one block, enabling legal validity of executed transactions.
- Implements legal identities of all participants.

For further information check the GitHub page of Alastria[16].

5. *Smart Contracts*

Smart Contracts are an instance of a computer program that runs on blockchain. In the case of permissioned blockchain such as Quorum, where only authorized users are able to interact with the ledger, an authorized user can create a contract by posting a transaction to the blockchain. It is important to notice that its code is fixed and cannot be changed after deployment. The code's execution is provoked by a received message either from a user of another contract and could provide utility to other contracts or require assistance from other Smart Contracts.

In this section we describe the different smart contracts developed to support the M-Sec use cases as well as some of the functionalities they provide.

1) M-Sec Token

A custom token was created specifically for research purposes. It is actually a cryptocurrency in the form of a smart contract running on Quorum Blockchain. It follows the ERC223[17] token standard. Prelaminar implementations followed the ERC20 token standard but ERC223 is a superset of the previous standard offering security improvements and more usability and backwards compatibility with any services and functionalities designed and developed for ERC20. As fully compliant with ERC223, it implements a set of functions and events, such as *name()*, *transfer()*, *totalSupply()* and *Transfer event* which is emitted to the blockchain when an amount of Tokens is transferred from a user to another. Some indicative developed methods are presented in the following **¡Error! No se encuentra el origen de la referencia.**.

This Token has different applications in the use cases. It is firstly used as a payment currency to exchange value among the users of the Marketplace. Another implementation and configuration of the M-Sec Token allows us to use it as a "*Social Token*". Users of the platform have an initial balance and particular users are rewarded with more token based on specific criteria such as for example:

i) the most active user,

ii) the most social user,

iii) the user who uploaded the most popular content.

This Token acts as a mean to tokenize a loyalty points program with rewards.

[16] https://github.com/alastria/alastria-platform/blob/master/en/Alastria-Core-Technical-Platform.md#alastria-core-technical-platform
[17] https://github.com/ethereum/EIPs/issues/223

Table 1: Overview of M-Sec Token's functions

```
contract ERC223 {
  uint totalSupply;
  function balanceOf();
  function name();
  function symbol();
  function decimals();
  function totalSupply();
  function transfer(to, value);
  function transfer(to, value, data);
  function transfer(to, value, data, custom_fallback);
  event Transfer(from, to, value, data);
}
```

In the following table more details are provided about the developed functions and events of the M-Sec Token:

Table 2: Detailed presentation of functions and events of M-Sec Token

Name	Input	Response	Description
totalSupply (function)	-	uint256 totalSupply	Get the total token supply
Name (function)	-	string _name	Get the name of token
Symbol (function)	-	bytes32 _symbol	Get the symbol of token
Decimals (function)	-	-	Get decimals of token
balanceOf (function)	address _owner	uint256 balance	Get the account balance of an account with address: address _owner
transfer (function)	address _to, uint _value	boolean	Transfer tokens, compatibility with ERC20
transfer (function)	address _to, uint _value, bytes _data	boolean	function that is always called when someone wants to transfer tokens. This function must transfer tokens and invoke the function *tokenFallback* if _to is a contract.
Transfer (event)	address indexed _from, address indexed _to, uint256 _value, bytes _data	-	Triggered when tokens are transferred and is emitted to the blockchain network
tokenFallback	address _from,	-	A function for handling token

(function)	uint _value, bytes _data	transfers, which is called from the token contract, when a token holder sends tokens

2) Item Manager Smart Contract

The Item Manager Smart Contract allows the interaction of item/content creators (e.g. photos, multimedia items, sensor data etc.) with the platform and the blockchain. A user is able to upload all the information and metadata related to an item. To this direction, we have created dedicated structs (Figure 8), which are a special feature of Solidity contract-oriented programming language, in order to store for each item, the details (e.g. tags, information, metadata) and the unique address of its owner.

```
struct item {
	address owner;
	string URI;
	uint256 price;
	string tag;
	string info;
}
```

Figure 8. Item Manager Smart Contract

3) Sensors Smart Contract

This smart contract records all the registered IoT sensors. It gives the possibility to register a sensor and to change its information afterwards as well. Dedicated Solidity structures were created to store this information and functions to allow its retrieval.

A structure that allows the storing of the information is the following:

```
	struct sensor {
		address sensor-Owner ;
		uint8 type-of-Sensor ;
		uint MSec-Token-Price ;
		uint32 timestamp-of-start ;
		uint16 frequency ;
		int32 latitude ;
```

```
            int32 longitude ;
            string url ;
        }
```

Figure 9. Sensors Smart Contract

4) In the following Table 3 more details are provided regarding our Sensors Smart Contract:

Table 3: Sensors Smart Contract details

Name	Input	Response	Description
registerSensor (function)	address sensor-Owner uint8 type-of-Sensor uint MSec-Token-Price uint32 timestamp-of-start uint16 frequency int32 latitude int32 longitude string url	Boolean success	Registration of a sensor to the dedication structure of the smart contract with the related information. Upon registration a verification of registration is returned
changeSensorInfo	uint8 type-of-Sensor uint MSec-Token-Price uint32 timestamp-of-start uint16 frequency int32 latitude int32 longitude string url	Boolean success	The owner of the sensor changes some of the fields for example the price in M-Sec Tokens or its position
BuySensorData	Uint32 sensorID uint32 fromTime uint32 toTime		This function is called when a user wishes to buy data for a specific time interval and communicates with M-Sec Token to perform this transaction

It is important to notice that functions like *changeSensorInfo* succeed only when the owner of the sensor (specific address) attempts to change the fields, otherwise the access is denied.

The function *BuySensorData* directly communicates with M-Sec Token smart contract, when a user wishes to buy data for a specific period. If the user has sufficient funds and the information is correct then the transaction will be successfully completed. Upon success the event *Transfer* is emitted to the network informing the users who watch the smart contracts that this transaction took place.

5) Know Your Customer Smart Contract

A huge number of financial banking transactions takes place every day. It is indicative that in July 2019 the Society for Worldwide Interbank Financial Telecommunication (SWIFT) recorded an average of approximately 32 million transactions per day. Blockchain can enable parties with no particular trust in each other to exchange digital data on a peer-to-peer basis with fewer or no third parties or intermediaries. In the recent report Scientific and Technical Research Report of European Commission on Blockchain[18], the need for *Know Your Customer* mechanisms is highlighted: "*the obligation of cryptocurrency exchanges and custodian wallet providers within the scope of EU regulation to implement mechanisms to counter money laundering and terrorist fundraising, such as 'know your customer' (KYC)* ".

It is evident that previously mentioned works involve value exchange through blockchain transactions and dedicated created smart contracts, making Know Your Customer process necessary. In this direction, we are presenting an approach which blends smart contracts for exchanging value in the IoT domain on a decentralized manner, integrating a KYC process handling *on chain* and *off chain* data.

Recent works have tried to tackle the problem of data management and KYC for blockchain applications. Shabair et al.[19] introduced a blockchain-based KYC proof of concept system and an orchestration tool for managing private blockchain environments over large scale test beds. In their work they highlight the need for additional research on security and privacy issues of blockchain applications. Norvill et al.[20] presented a demo of a system that allows automation and permissioned document sharing in order to simplify and reduce the work required by the KYC process, while Zhang and Yin[21] conducted a research on a digital copyright management system based on blockchain

[18]A. Anderberg et al., "Blockchain Now And Tomorrow," 2019.

[19] W. Shbair, M. Steichen, and J. François, "Blockchain orchestration and experimentation framework: A case study of KYC," in The First IEEE/IFIP International Workshop on Managing and Managed by Blockchain (Man2Block) colocated with IEEE/IFIP NOMS 2018, 2018.

[20] R. Norvill, M. Steichen, W. M. Shbair, and R. State, "Blockchain for the Simplification and Automation of KYC Result Sharing," in 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), 2019, pp. 9–10.

[21] X. Zhang and Y. Yin, "Research on Digital Copyright Management System Based on Blockchain Technology," presented at the 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chengdu, China, 2019.

technology. They focused mostly on PBFT (Practical Byzantine Fault Tolerance) consensus mechanism improved by Tendermint[22] replacing original Ethereum POW (Proof of Work), digital signatures and smart contracts to design user account management strategies, copyright review and applications for the needs of digital rights management. In our work we further explore the design and implementation of smart contracts for the KYC process on a decentralized approach.

Blockchain is beginning to transform industries and there is an increasing interest in exploring its potential for various production use cases, especially for supporting multi-party processes where members don't necessarily trust each other. However, there are many challenges that remain to be addressed such as trade-offs between respecting privacy and supporting transparency. Bhsaskaran et al[23] described the design of smart contracts for consent-driven and double-blind data sharing on the Hyperledger Fabric blockchain platform[24] into a KYC application, where the data are submitted, validated and kept within the ledger supporting different consent rules and privacy levels.

Vishwa et al.[25] presented a decentralized data management system for data privacy and control focusing on multimedia files. In their solution they use an external data lake, namely a centralized data storage solution on a cloud to store the transaction details of all the data added on the blockchain. In order to access the blockchain, a user signs up by broadcasting his identity and will be accepted by the consent of the majority of the nodes and will be provided his new identity and access permissions. In our approach we additionally use IPFS leading to a decentralized application and have successfully implemented smart contracts and software components, leveraging blockchain to automate tasks related to KYC process.

Our process of developing the smart contract to support KYC process is described through its use in the middleware services section.

6) Smart City Data Smart Contract

This smart contract focuses on managing data from the smart cities of Santander and Fujisawa and support the use cases. It directly communicates with other tools of M-Sec project such as encrypted data storage and offchain storage. It is an ongoing work and the final results will be described in details in the next iteration of the of the Deliverable D4.6 from T4.3. Additional flows are created as part of middleware services to support the interaction and integration with the rest of the platform.

[22] Jae Kwon, "Tendermint: Consensus without Mining." 2014.
[23] K. Bhaskaran et al., "Double-blind consent-driven data sharing on blockchain," in 2018 IEEE International Conference on Cloud Engineering (IC2E), 2018, pp. 385–391.
[24] "Hyperledger Fabric," Hyperledger
[25] A. Vishwa and F. K. Hussain, "A Blockchain based approach for multimedia privacy protection and provenance," in 2018 IEEE Symposium Series on Computational Intelligence (SSCI), 2018, pp. 1941–1945.

Component Module 3: Middleware Services

This component refers to all the implemented basic blockchain services that include services such as search and indexing of the P2P network resource, advertising & discovery services, and messaging services for exchanging messages between the peers.

1. *Know Your Customer*

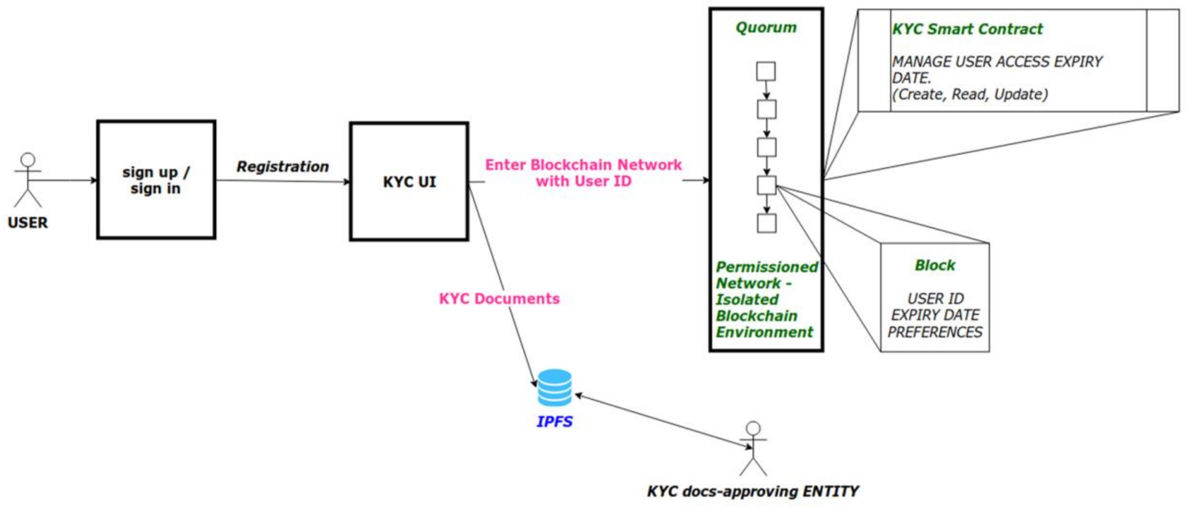

USER
sign up / sign in
Registration
KYC UI
Enter Blockchain Network with User ID
KYC Documents
IPFS
KYC docs-approving ENTITY
Quorum
Permissioned Network - Isolated Blockchain Environment
KYC Smart Contract
MANAGE USER ACCESS EXPIRY DATE.
(Create, Read, Update)
Block
USER ID
EXPIRY DATE
PREFERENCES

Figure 10. Overview of the KYC process for the M-Sec Platform

The KYC service as part of M-Sec Blockchain Middleware Services allows us to have a system where anonymity is maintained among user choices. The user identification has already been done outside the blockchain network, while no one inside the blockchain network is aware of the user's real identity. In this way M-Sec Platform will use KYC service to have services been delivered to users while hiding their true identity from their service provider or other users.

The M-Sec KYC Solution Concept includes the storage of personal data on an offchain database while the user is able to connect to the M-Sec Platform (and the blockchain network) using a special ID not relevant to his real identity. So the User Verification is conducted by an External Certificate Authority before accessing the system while the user uses the hash ID provided to him to interact with the System, as shown in the Figure 10.

Additionally, we have integrated the feature of the Expiration date. The System maintains an Expiry Date of users in the blockchain network. This information is stored within the smart contracts not in an external centralized database.

2. *IPFS: InterPlanetary File System*

Aiming to a more decentralized design we integrated blockchain with IPFS, a peer-to-peer version-controlled protocol and filesystem, run by multiple nodes, storing files submitted to it[26]. It combines distributed Hash Tables, Block Exchanges and Merkle Trees.

Using middleware, users are able to upload content to IPFS and place its unique hash code (address of the file) to the smart contracts running on Quorum blockchain. If we use a central database for storage, we benefit from the high throughput but this centralization does not coincide with the decentralized nature which blockchain advocates leading to a Single Point of Failure (SPOF) of the whole application. Facing the aforementioned drawback, IPFS being a peer-to-peer (p2p) file sharing system and Blockchain's complementary component, settled exceptionally the SPOF problem, furnishing low latency and data distribution.

3. *On-chain, off-chain data and access control*

One of our goals is to design a blockchain-based decentralized content marketplace, which enables trustless disintermediation between sensor owners (and more generally data owners) and consumers. Using a dedicated created cryptocurrency (M-Sec Token) for payments, a consumer can buy data on the marketplace without involving a marketplace intermediary. This refers to the research and development of data privacy-enhancing mechanisms along with data access control and privacy policies that are necessary for the M-Sec framework. Moreover, it deals with the separation of data, meaning to identify what needs to be pushed on blockchain and what to remain off-chain, a decision that is always critical when designing blockchain platforms

4. *Transaction Handler*

One of the main and most important features of the Quorum is the private transaction mechanism. Transaction privacy is achieved by using the Ethereum Transaction Model and enhancing it with new parameters that specify the nodes in which the transactions should be published. The Constellation layer of Quorum that contains the transaction Manager and Enclave module is responsible for the private transaction handling. All the public transactions follow the already established p2p Ethereum network flow.

Additional mechanisms are implemented that:

i. allow only authorized users to commit a transaction and have access to the blockchain,
ii. verify the identity of user using cryptography algorithms,
iii. in case he is about to receive some data/service in exchange of M-Sec Tokens it is verified that he has already made the purchase.

[26] Chen, Y., et al.: An improved P2P file system scheme based on IPFS and Blockchain. Big Data (Big Data), IEEE International Conference on (2017).

The Transaction Handler could be regarded as a flow providing a layer before blockchain that performs a first process of the potential transactions to formulate them and optimize and verify the content to be inserted in the blockchain.

5. Upload Handler

This part of the Middleware Services provides functionalities for efficiently performing actions related to Assets. As an asset we could consider a file, a multimedia item, a dataset that could be described with a predefined set of fields such as:

i. Title
ii. Timestamp of start
iii. Timestamp of end (whether applicable)
iv. Owner/Creator name (or Address)
v. Price in M-Sec Tokens
vi. Description
vii. Location (latitude and longitude)
viii. URL related to the storage of the asset

All these functionalities are related to Smart Contracts in which we have defined Solidity structs keeping record of uploaded assets/items and we have additionally define related fields for metadata. These functionalities include:

i. Uploading of an item by providing its details, as specified previously so it can be registered to the Item Manager Smart Contract.
ii. Browsing through all the available items registered in the smart contract.
iii. After specifying some criteria, the user is able to view an asset and its metadata.

6. Write/Update Metadata of Asset

This service is strongly connected to the Transaction Handler. As an indicative case, only the authorized users are allowed to update the metadata of an item. The user who has the right to update is the owner of the item or a user with a specific permission. The service handles the communication with the smart contracts and checks the rights of a user.

2.2 Package Information & Installation Instructions

Required Tools and dependencies

- Truffle Suite
- Solidity Programming Language
- Quorum Blockchain

Install Truffle Suite

Truffle suite:

```
npm install truffle -g
```

More installation instructions could be found in the following link: https://www.trufflesuite.com/truffle

Install Solidity

Ethereum, "Solidity," Ethereum, [Online]. Available: http://solidity.readthedocs.io.

Install Quorum Blockchain Network

https://github.com/synechron-finlabs/quorum-maker

https://github.com/jpmorganchase/quorum-examples/tree/master/examples/7nodes

Licensing

Quorum, the go-ethereum library (i.e. all code outside of the cmd directory) is licensed under the GNU Lesser General Public License v3.0

Solidity is licensed under GNU General Public License v3.0

3. Demonstration 2: IoT Marketplace

The goal is to create decentralized IoT ecosystems and validate their viability and sustainability. To this direction we define and implement a novel marketplace where smart objects can exchange information, energy and services through the use of virtual currencies allowing real-time matching of supply and demand enabling the creation of liquid markets with profitable business models of the IoT stakeholders. In this section we cover the basic technical implementation details of the M-Sec marketplace: market participants, from IoT devices to humans using mobile applications are able to exchange data and value through the M-Sec blockchain implementation.

3.1 General Description of the Prototype

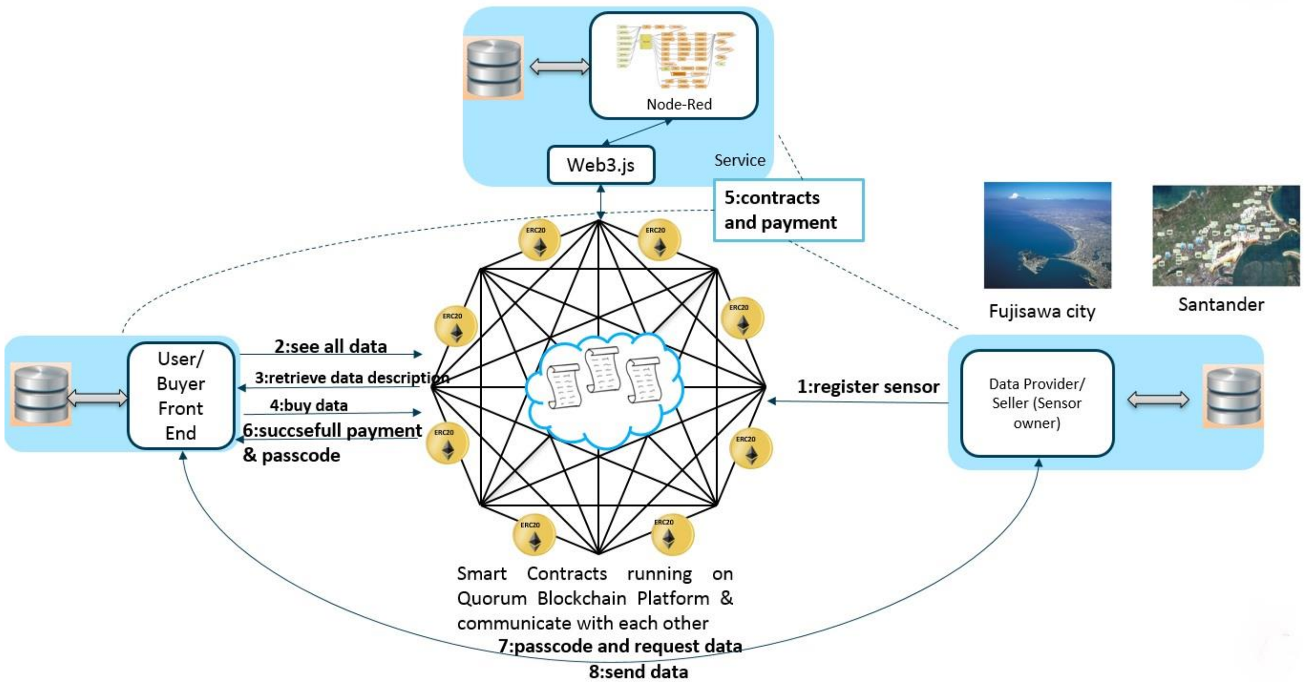

Node-Red
Web3.js
Service
5:contracts and payment
Fujisawa city
Santander
User/ Buyer Front End
2:see all data
3:retrieve data description
4:buy data
6:succsefull payment & passcode
1:register sensor
Data Provider/ Seller (Sensor owner)
ERC20
Smart Contracts running on Quorum Blockchain Platform & communicate with each other
7:passcode and request data
8:send data

Figure 11. Overview of the M-Sec IoT Marketplace and its components

In the previous Figure 11, we can see an overview of the developed marketplace and its components, explained in detail through for a specific example use of it.

1. The owner of a sensor/data source who wishes to make his data available for purchase or exchange register himself to the dedicated created smart contract providing information about the type of the data, their frequency, the price, the location etc.

2. A User of the M-Sec Platform who acts here as a potential buyer using our developed front end can see all the available sensors and their data
3. Upon finding some interesting data he/she can retrieve additional detailed descriptions about them and then
4. Buy the data of interest using M-Sec Tokens, which is a cryptocurrency in the form of a smart contract running in on blockchain presented in previous section
5. The deployed smart contracts communicate with each other to verify the sufficient funds of the buyer and complete the purchase by transferring funds from the balance of the buyer to the one of the data owner. The developed Node-Red flows also assist in this process connecting the different components of the system
6. In the case of successful payment, when the buyer has sufficient funds and after the tokens are transferred, a passcode is returned to the buyer necessary for accessing the purchased data
7. The buyer communicates with the platform and the API of the data owner and using the transactions details requests the data
8. The desired data is returned to the buyer in a predefined format such as JSON

Components

Component Module 1: Node-Red Flows

In order to orchestrate the different components and services we have used Node-Red and have developed several flows. Node-Red is a powerful visual tool for wiring together hardware devices, APIs and web-services, create flows that connect distributed components into a common IoT application[27].

We developed different flows for the different parts of the IoT Marketplace.

During the development of the system we simulated the IoT weather sensors provided by public APIs and for this simulation we used an API provided by Dark Sky[28]. Using Node-RED features we created flows that request current weather data for several locations from the Dark Sky API and then save these data (air temperature, relative humidity, pressure, visibility, wind speed and direction, sky cloud coverage, dew point, UV radiation and the columnar density of total atmospheric ozone layer) into a local database. We also exposed a RESTful API in order to serve the data to the users when requested. When a request is received, the API key is checked. If it is correct, the data responding to the specified time intervals is retrieved from the database and then sent to the requester.

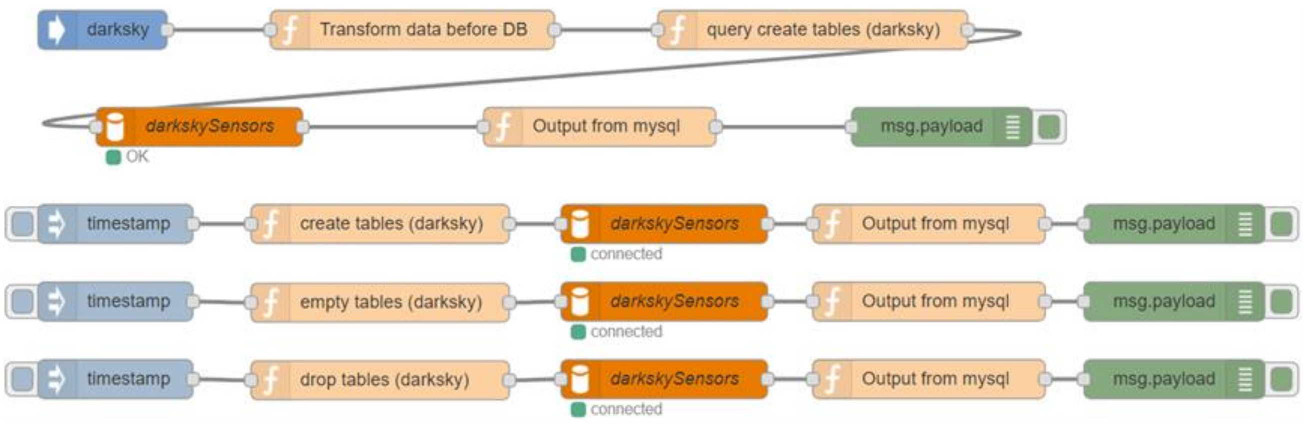


Figure 12. Node-Red Flow for the simulation of IoT sensor data

Component Module 2: Blockchain Smart Contracts

Different smart contracts are implemented in the programming language Solidity to support the different use cases.

Some of the developed smart contracts were described in detail previously:

1. M-Sec Token: Digital cryptocurrency in the form of a smart contract in Solidity language running on blockchain.

[27] https://nodered.org/docs/
[28] The Dark Sky Company, LLC, "Dark Sky," The Dark Sky Company, LLC, [Online]. Available: https://darksky.net/dev

2. Sensors Smart Contract: responsible for registering sensors and recording transactions for IoT sensor data
3. Smart City Data Smart Contract: responsible for handling data from Smart Cities

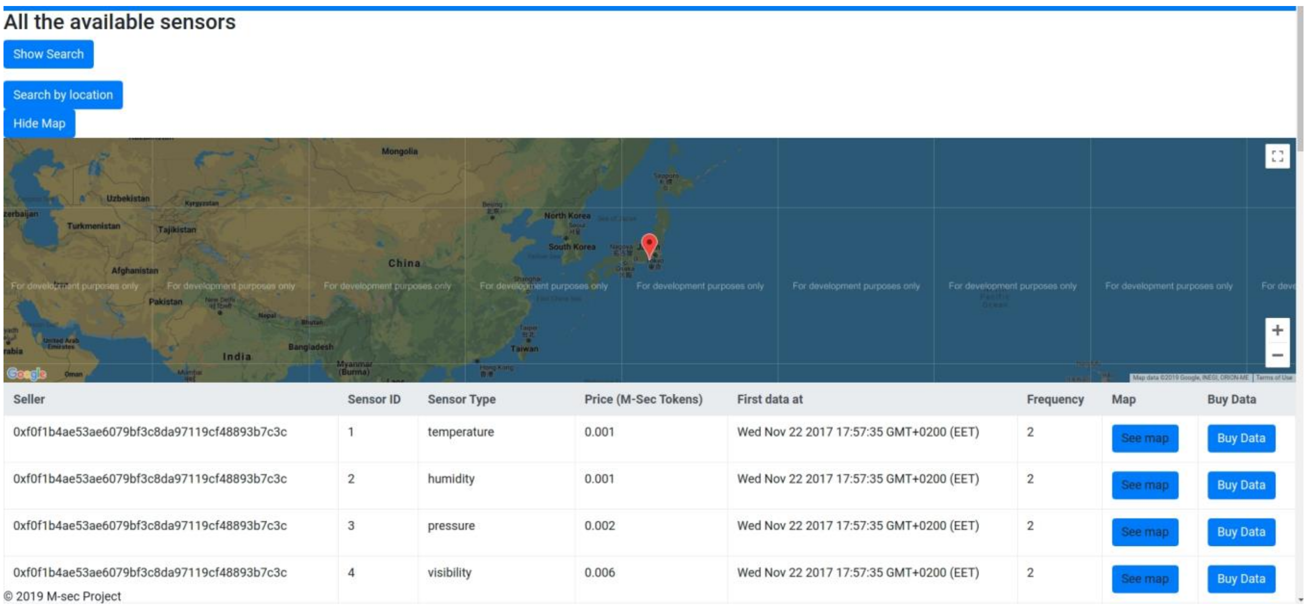
Component Module 3: Web Application

This web application provides interfaces between the users and the blockchain. It provides functionalities helping users interact with the smart contracts deployed on Quorum Blockchain and access data they have bought. It also allows sending transactions to and reading data of transactions and smart contracts. It also "protects" users from misreading or mistyping info when sending a transaction.

We have used different languages and technologies to create these interfaces such as JavaScript, Bootstrap, HTML, jQuery, Nodejs. Some of the developed interfaces are described below with screenshots and details. We have used Web3.js to interact with the deployed smart contracts.

The user searches in all the available sensors registered in the Smart Contracts the sensors of interest specifying details in the corresponding fields such as the location, the type the data (temperature, starting date and time, frequency etc.), as shown in Figure 13.

'

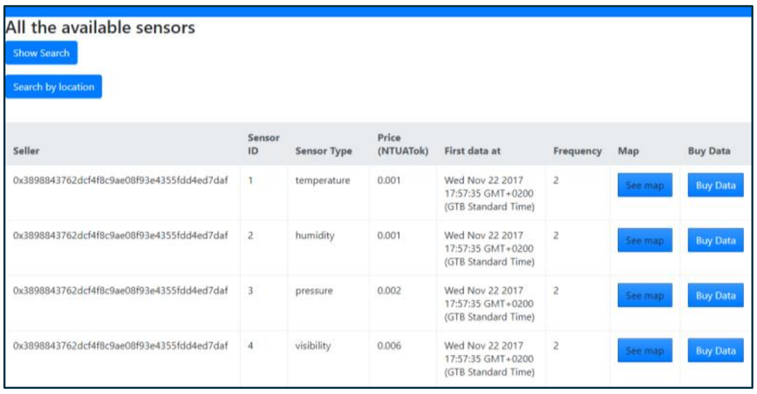


Seller	Sensor ID	Sensor Type	Price (M-Sec Tokens)	First data at	Frequency	Map	Buy Data
0xf0f1b4ae53ae6079bf3c8da97119cf48893b7c3c	1	temperature	0.001	Wed Nov 22 2017 17:57:35 GMT+0200 (EET)	2	See map	Buy Data
0xf0f1b4ae53ae6079bf3c8da97119cf48893b7c3c	2	humidity	0.001	Wed Nov 22 2017 17:57:35 GMT+0200 (EET)	2	See map	Buy Data
0xf0f1b4ae53ae6079bf3c8da97119cf48893b7c3c	3	pressure	0.002	Wed Nov 22 2017 17:57:35 GMT+0200 (EET)	2	See map	Buy Data
0xf0f1b4ae53ae6079bf3c8da97119cf48893b7c3c	4	visibility	0.006	Wed Nov 22 2017 17:57:35 GMT+0200 (EET)	2	See map	Buy Data

© 2019 M-sec Project

Figure 13. Graphical User Interface enabling searching of sensors in the smart contracts running on blockchain

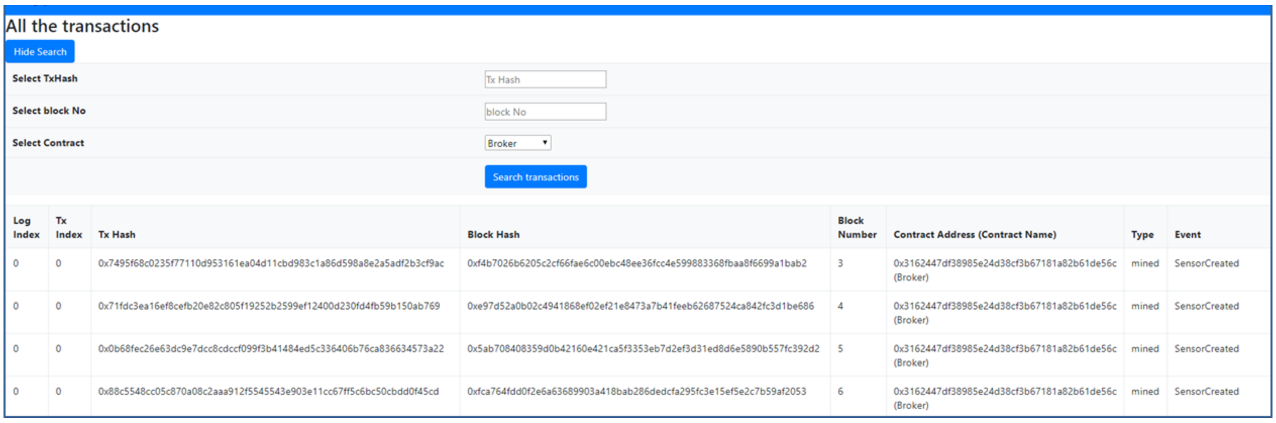
After specifying all the required information, a query is submitted to the smart contracts running on the Quorum blockchain and a list of all the available sensors is returned with information of the address of the data owner, the sensor type (temperature, pressure, visibility etc.), the frequency, a link opening a map and the option for the user to buy these data using M-Sec Tokens, as shown in Figure 14.


All the available sensors

Seller	Sensor ID	Sensor Type	Price (NTUATok)	First data at	Frequency	Map	Buy Data
0x3898843762dcf4f8c9ae08f93e4355fdd4ed7daf	1	temperature	0.001	Wed Nov 22 2017 17:57:35 GMT+0200 (GTB Standard Time)	2	See map	Buy Data
0x3898843762dcf4f8c9ae08f93e4355fdd4ed7daf	2	humidity	0.001	Wed Nov 22 2017 17:57:35 GMT+0200 (GTB Standard Time)	2	See map	Buy Data
0x3898843762dcf4f8c9ae08f93e4355fdd4ed7daf	3	pressure	0.002	Wed Nov 22 2017 17:57:35 GMT+0200 (GTB Standard Time)	2	See map	Buy Data
0x3898843762dcf4f8c9ae08f93e4355fdd4ed7daf	4	visibility	0.006	Wed Nov 22 2017 17:57:35 GMT+0200 (GTB Standard Time)	2	See map	Buy Data

Figure 14. Graphical User Interface of the returned results after the query to the smart contract

An overview of the blockchain and the transactions included in each block is provided in our developed Explorer interface,as shown in Figure 15. The user is able to search for specific blocks, transactions, users, contracts and see the related activity.


All the transactions

Select TxHash	Tx Hash
Select block No	block No
Select Contract	Broker

Log Index	Tx Index	Tx Hash	Block Hash	Block Number	Contract Address (Contract Name)	Type	Event
0	0	0x7495f68c0235f77110d953161ea04d11cbd983c1a86d598a8e2a5adf2b3cf9ac	0xf4b7026b6205c2cf66fae6c00ebc48ee36fcc4e599883368fbaa8f6699a1bab2	3	0x3162447df38985e24d38cf3b67181a82b61de56c (Broker)	mined	SensorCreated
0	0	0x71fdc3ea16ef8cefb20e82c805f19252b2599ef12400d230fd4fb59b150ab769	0xe97d52a0b02c4941868ef02ef21e8473a7b41feeb62687524ca842fc3d1be686	4	0x3162447df38985e24d38cf3b67181a82b61de56c (Broker)	mined	SensorCreated
0	0	0x0b68fec26e63dc9e7dcc8cdccf099f3b41484ed5c336406b76ca836634573a22	0x5ab708408359d0b42160e421ca5f3353eb7d2ef3d31ed8d6e5890b557fc392d2	5	0x3162447df38985e24d38cf3b67181a82b61de56c (Broker)	mined	SensorCreated
0	0	0x88c5548cc05c870a08c2aaa912f5545543e903e11cc67ff5c6bc50cbdd0f45cd	0xfca764fdd0f2e6a63689903a418bab286dedcfa295fc3e15ef5e2c7b59af2053	6	0x3162447df38985e24d38cf3b67181a82b61de56c (Broker)	mined	SensorCreated

Figure 15. Graphical User Interface of the Explorer

3.2 Package Information & Installation Instructions

Required Tools and dependencies

The following tools and dependencies are required to install and use the IoT Marketplace:

- NodeRed
- Nodejs
- MySQL
- Ethereum/Quorum Blockchain
- Nodejs and Javascript

Install NodeRed

- **Node-Red**: Node-RED is a powerful visual tool for wiring together hardware devices, APIs and web-services, create flows and connect distributed components into a common IoT application [https://nodered.org/].
    - Installing Node-Red: The easiest way to install Node-RED is to use the node package manager, npm, which comes with Node.js [https://nodered.org/docs/getting-started/installation]. Installing as a global module adds the command "node-red" to your system path:
        - For Ubuntu:
            - sudo npm install -g --unsafe-perm node-red
        - For Windows, execute CMD with administrator rights and then execute:
            - npm install -g --unsafe-perm node-red
    - Version: We have installed Node-Red v0.17.5
    - Running: after installing Node-Red as a global npm package, open a terminal and run the "*node-red*" command. You can then access the Node-RED editor by pointing your browser at: http://localhost:1880
    - After accessing the editor, you have to left click on the menu button (three lines on the top right corner), then click on manage palette, switch to the install tab and search for the node-red-contrib-neo4j package and install it. This will add the node required by our flows ensuring the dependency.

Install Nodejs

- **Node.js**: Before installing Node-Red, a Node.js installation is required. We have installed Node.js version v8.9.3.
    - On Ubuntu machines we have to run the following commands:
        - sudo apt-get update
        - sudo apt-get upgrade
        - sudo apt-get install node.js -y
        - sudo apt-get install npm -y
    - On Windows machines we can download the appropriate installer from https://nodejs.org/en/download and execute it.

Install MySQL

- We used the MariaDB SQL, but any other SQL relational database can be used. Full instructions of how to install MariaDB database can be found here: https://downloads.mariadb.org/.

Install Front End

- Front End: We have developed a web front end, useful for end users of our application. It provides a Graphical User Interface
    - Based on HTML, Javascript, Vue Javascript framework and other libraries
    - Running: it is deployed on our server (and cloud servers as well) and accessible in http://snf-755174.vm.okeanos.grnet.gr

Install Java

- Most of the systems used are built on top of java engines so a Java distribution needs to be installed in the system before anything else.
    - On Ubuntu machines a simple list of commands is enough to install the latest distribution of Java:
        - sudo add-apt-repository ppa:webupd8team/java
        - sudo apt-get update
        - sudo apt-get install -y oracle-java8-installer
        - sudo apt-get update
    - On Windows machines we have to download the appropriate installer from https://java.com/en/download and execute it

Install Ethereum/Quorum Blockchain

Instructions are provided in the previous Section related to the Blockchain demonstrator.

Operating System

- We have tested the platform on Windows 10 and Ubuntu 18 but all of the software listed here is available in a large number of other distributions.

Okeanos

- Okeanos: We have deployed our Node-Red and Neo4j services to Okeanos cloud service for Greek Research and Academic Community
    - https://okeanos.grnet.gr/home/

Licensing (if applicable)

Since ICCS/NTUA is a non-profit Academic Research Body, we will be releasing all related M-Sec results as open source contributions under Open Source licenses. Concretely, permissive licenses, as are not restrictive licenses and it can be used to create a proprietary good, allowing a commercial exploitation and ensuring high impact. Examples of those are: Apache, BSD, etc.

Demonstration 3: Trust & Reputation Management

3.3 Introduction

Trust and Reputation (T&R) models have been proposed by many researches as an innovative solution for guaranteeing a minimum level of security between two entities of a distributed system that want to have a transaction or interaction. Thus, many studies, works and models have been designed, carried out and developed in this direction, leading to a current solid research field on which both academia and industry are focusing their attention. Many methods, technologies and mechanisms have been proposed in order to manage and model trust and reputation in systems such as P2P networks[29], ad-hoc ones[30], wireless sensor networks[31] or even multi-agent systems[32]. Such methods have been used in many environments like P2P networks, Wireless Sensor Networks (WSN), Vehicular Ad-hoc Networks (VANETs), Identity Management Systems, Collaborative Intrusion Detection Networks (CIDN), Cloud Computing Systems, Application Stores and of course the IoT.

T&R management is a very useful and powerful tool in environments where a lack of previous knowledge about the system can lead participants to undesired situations, specifically in virtual communities where users do not know each other at all or, at least, do not know everyone. It is in those cases where the application of trust and reputation mechanisms is more effective, helping a peer to find out which is the most trustworthy or reputable participant to have an interaction with, preventing thus the selection of a fraudulent or malicious one. Most of the current T&R models in the literature follow four general steps which are described by Marti and Garcia-Molina[33] (Figure 15):

1. **Collecting information** about a certain participant in the community by asking other users their opinions or recommendations about that peer.
2. **Aggregating all the received information** properly and somehow computing a score for every peer in the network.
3. **Selecting the most trustworthy or reputable entity** in the community providing a certain service and effectively having an interaction with it, assessing posteriori the satisfaction of the user with the received service.
4. **Punishing or rewarding** according to the satisfaction obtained, adjusting consequently the global trust (or reputation) deposited in the selected service provider.

5.

[29] F. Almenarez, A. Marin, C. Campo, C. Garcia, "PTM: a pervasive trust management model for dynamic open environments", First workshop on pervasive security and trust, Boston, USA; 2004.

[30] M. Moloney, S. Weber, "A context-aware trust-based security system for ad hoc networks", Workshop of the 1st International Conference on Security and Privacy for emerging areas in communication networks, Greece; 2005, pp. 153–60.

[31] Boukerche, L. Xu and K. El-Khatib, "Trust-based security for wireless ad hoc and sensor networks", Computer Communications 2007.

[32] J. Sabater and C. Sierra C, "REGRET: reputation in gregarious societies", Proceedings of the 5th International Conference on Autonomous Agents, Canada, 2001.

[33] S. Marti and H. Garcia-Molina, "Taxonomy of trust: categorizing P2P reputation systems", Computer Networks 2006.

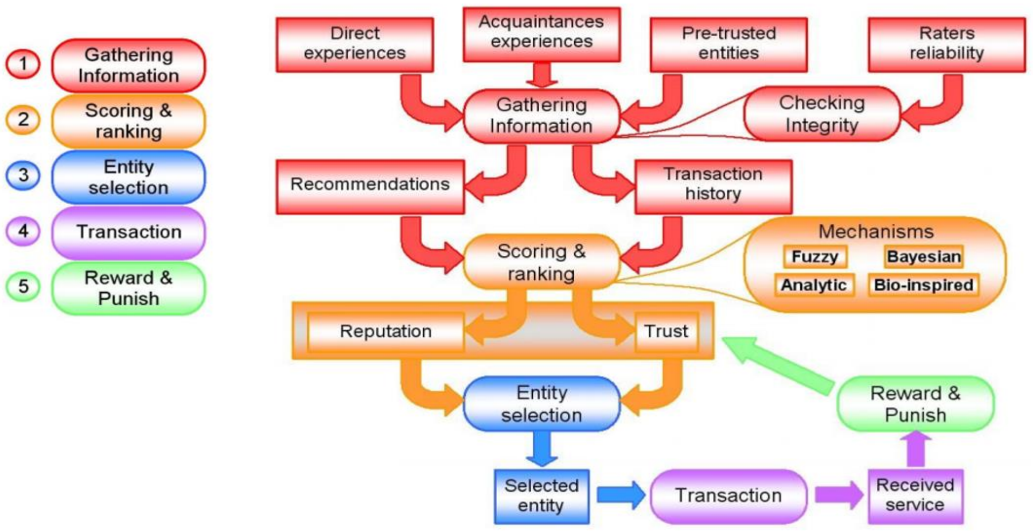


Figure 16: General steps followed in T&R models.

Currently, the idea of using a T&R engine on top of the Blockchain Middleware Services and the IoT Marketplace (already described in the previous sections) is being investigated. Such an engine would enhance the security mechanisms of M-Sec and make it possible to evaluate the actual content being shared through the Blockchain and the Marketplace, thus ensuring the trustworthiness of the several actors participating in the exchange or sharing of information, data and services.

The M-Sec T&R model (M-Sec T&RM)

Different models manage concepts such as Trust or Reputation in many different ways. Although there are some generic data structures for the domain of T&R provided for example by the Open Reputation Management Systems (ORMS) of OASIS[34], **there are no standards** for concepts like Trust and Reputation. In this subsection we try to provide some clear definitions of the main concepts that build up the M-Sec T&R model, and the main features that characterize it. In M-Sec T&RM we define Trust and Reputation as follows:

- **Trust:** The expectation that an interaction will be satisfactory based on *our personal experience*.
- **Reputation:** The belief that an interaction will be satisfactory based on the experience of *our social circle*.

Node A will have a high Trust index for Node B if the services provided from Node B to Node A have been evaluated from Node A positively. Node A will have a high Reputation index for Node B if the services provided from Node B have been evaluated from the social circle of Node A positively.

Definitions

The distinction between a trust and a reputation model is not always clear. However, in our opinion, those models making an explicit use of other participants' recommendations could be categorized as reputation models while the rest could be considered just as trust models.

Let's assume that actor-1 wants to find out some social characteristics of actor-2 for a specific service offered. The following terms can then be defined:

[34] OASIS: https://www.oasis-open.org/committees/orms

- **<u>Popularity (P):</u>** A counter which monitors how many times actor-2 has received or may receive a request (how many "hits" it has). The Popularity Index is an accumulative and comparative indicator, and is used to determine the stability of Reputation and Trust.
- **<u>Trust (T):</u>** The belief of actor-1 that actor-2 is going to deliver the correct service based on previous interactions of actor-1 with actor-2. The Trust Index of actor-2 provided by actor-1 is a property which states how many times actor-2 has successfully shared its services with actor-1. Trust is "subjective", because it is estimated from perspective of the individual trustor (actor-1 in this case).
- **<u>Reputation (R):</u>** The belief of actor-1 that actor-2 is going to deliver the correct service based on previous interactions of other actors. The Reputation Index can be calculated from the Trust that other actors (apart from actor-1) have on actor-2. In other words, this metric determines the belief of others on an actor and is useful especially when actor-1 does not have enough data to extract a Trust Index for actor-2 (because e.g. there are no interactions between the two actors yet).
- **<u>Reliability (R')</u>**: An absolute indicator of the performance of the actor that quantifies its efficiency to offer successfully its services relatively to its ideal or normal operation. The Reliability Index should be based on criteria like: response time upon request, ability to communicate, quality of service provided, etc.
- **<u>Dependability (D):</u>** A social measure combining all the above social measures. It can be simply derived by the expression $\boldsymbol{D = a \cdot T + b \cdot R + c \cdot R' + d \cdot P}$ where *a, b, c* and *d* (non-negative integers) are the weights of the measures and $\boldsymbol{a + b + c + d = 1}$. For this calculation, Popularity has to get normalized. By selecting the appropriate weights, we can provide the expression of the Dependability Index that we want. For example, when there are only a few interactions between actor-1 and actor-2, then the Trust Index should have a low weight and the Reputation Index should have a high weight. This means that the weights should change dynamically and be set according to the users or developers preferences.

General Features

Reputation connects closely to the concept of Trust, but there is a clear difference, which can be illustrated by the following two scenarios:

- Actor-1 trusts actor-2 because Actor-2 has a good Reputation. This reflects that Reputation can be used to build Trust.
- Actor-1 trusts actor-2 despite the bad Reputation of Actor-2. This reflects that even if actor-1 knows the Reputation of actor-2, actor-1 has its own private knowledge (e.g. direct experience with actor-2) which is considered to be more important.

Generally, an actor can be evaluated only by information gathered from other actors. Its Dependability can be calculated by each and every other actor of the community (a subjective estimation) or by the whole system (a more, but not totally, objective estimation). Depending on its (subjective or objective). Both big and small time-windows are used to quickly detect malicious or unsatisfactory behaviour and avoid the fast redemption of blacklisted actors. Moreover, feedback from recent interactions has a higher weight than this of older actions.

Benevolent actors should have more opportunities than newcomers. As a result, newcomers with 0 interactions with other actors will have Reputation equal to 0. However, an extra rule has to be applied to the model we have designed to give the opportunity to newcomers that have a low Reputation (because of the small number of interactions with other actors) to be chosen as service providers at some point and start building their Reputation. For example, 10% of the recommendations from the platform should introduce newcomers to the rest of the community. The same applies for actors which have low Reputation due to malicious or unsatisfactory behaviour in the past. In other words, this rule enables the **social integration** and **reintegration** of the actors to the system. Moreover, this rule is necessary for the first moments of the social community that may be born from M-Sec, as the network, at its **initial state**, will not have any actors with high Reputation.

It should be noted that, in contrast with many T&R models, we choose to use **different** Trust and Reputation **scores** for different services provided by the members of the network. This feature helps as face quite many security threats. For example, abuse of a high achieved Reputation is easily avoided.

Calculation of Trust & Reputation

In M-Sec T&RM, only the idea of subjective Trust is modelled, as we claim that subjectiveness is embedded in Trust's meaning. Strong Trust on an actor cannot and should not be affected by claims of a third party. In order to model Trust, the experiences based on which the Trust is calculated need to be modelled. Thus, we need memory. For that purpose, the M-Sec Blockchain can be used to store the "social" interactions between actors and the evaluations of the corresponding services. Some crucial attributes that have to be stored in these Log Files are:

- **Satisfaction (s):** This value is essentially a subjective QoS indicator. The Satisfaction is automatically derived by the absolute values of the service based on their correctness. For example, a sensor that suddenly reports a really high temperature will be assigned a satisfaction rating based on the correctness of this report. If there is a fire, the Satisfaction is high, but if the is not, the Satisfaction is zero. Since an actor that regulates the alarms can consult more than one sensors, a malicious or faulty sensor will quickly lose any trust. If the sensor is fixed, the Social Reintegration part of the system will allow it to build trust again.
- **Weight (w):** This is a value indicating how crucial the service is for the well-being of the actor. It is used in order to prevent a malicious actor from providing a minor service well and then exploiting the built Trust and providing a crucial service poorly. Due to this value, it is difficult for the Trust index to increase just because of minor services, whereas it can drop quickly in case of a crucial service with low quality.
- **Fading factor (f):** When new interactions take place, the importance of older ones should decrease. The fading factor addresses this issue and forces peers to stay consistent with their previous behaviour. Old interactions have lower fading factor values, so an actor cannot misbehave relying on its good history. The fading factor makes the Social Reintegration of ex-malicious nodes possible, meaning that if they become benevolent, it is possible for them to get a second chance and form new ties with the network. Of course multiple incidents of misbehaviour can get an actor permanently black-listed. This fading factor can be set by the system administrator so that the actors take under consideration the last N interactions with any other actor.

When an actor wants to calculate the **Trust Index** of another one, it looks into the appropriate Log Files in the Blockchain and calculates the trust value as the weighted average of the log entries using:

$$\mu_t^k = \frac{\sum_{i=1}^{N}(s_i \cdot w_i \cdot f_i)}{W} \qquad (1)$$

where W is the normalization co-efficient which ensures that the trust value will be between [0,1] and is calculated by:

$$W = \sum_{i=1}^{N}(w_i \cdot f_i) \qquad (2)$$

The **mean value (μ)** is a measure of the overall observed behaviour of the actor and indicates the expected satisfaction value of the next interaction. However, it is needed to know how confident we can be about the value of μ i.e. how much the satisfaction from the service may actually deviate from μ. Thus, the **standard deviation (σ)** of the behaviour is also calculated. To reduce the computational overhead, the calculation of the later occurs simultaneously with the calculation of the mean value following the formula:

$$\sigma_t^k = \frac{\sqrt{\sum_{i=1}^{N}(s_i{}^2 \cdot w_i \cdot f_i) \cdot W - \left(\sum_{i=1}^{N}(s_i \cdot w_i \cdot f_i)\right)^2}}{W} \qquad (3)$$

Finally, we define Trust as:

$$\boldsymbol{T^k = \mu_t^k - \sigma_t^k} \qquad \mathbf{(4)}$$

To sum up, μ shows the satisfaction that actor-1 should expect from actor-2, while σ shows how predictable the behaviour of actor-2 is. This means that if $T = 0.5$ then there is an 84% probability that the satisfaction for the service will be 0.5 or greater. That way the service providers that are not consistent and have an ever changing and oscillating behaviour will have lower Trust indexes even if their μ value is higher.

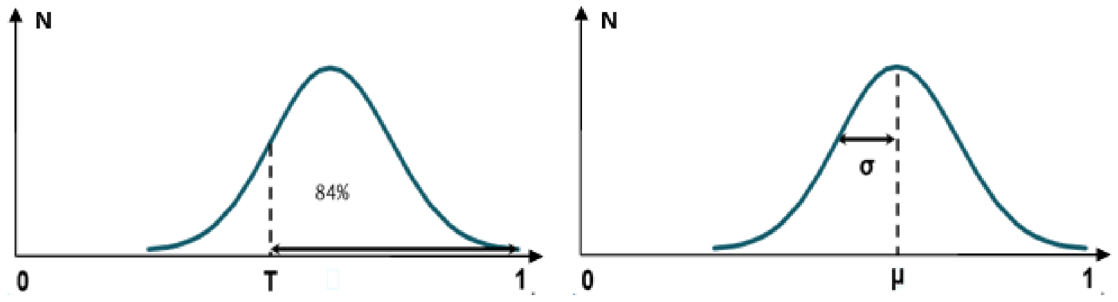


Figure 17: Calculation of the Trust Index of an actor.

Similar approach is being followed to calculate the **Reputation Index**, although this metric needs to extract more interactions logs from the Blockchain.

3.4 Testing

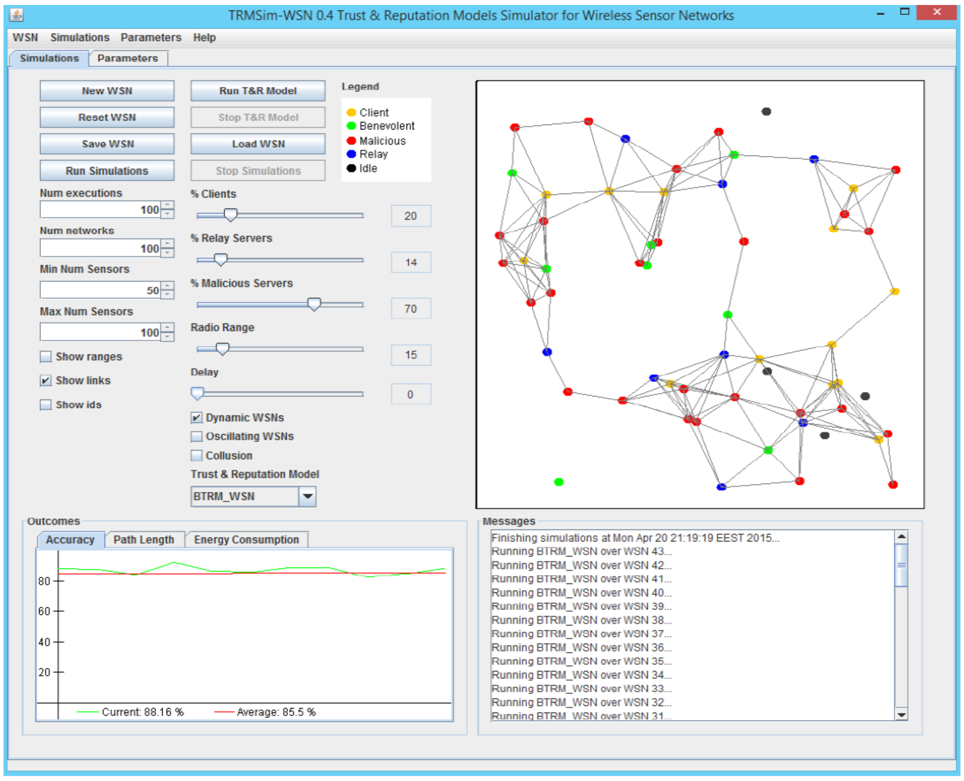
TRMSim-WSN

In order to test our T&R model, we used TRMSim-WSN[35], a simulator for T&R models. The TRMSim-WSN is a Java-based T&R models simulator aiming to provide an easy way to test a trust and/or reputation model over WSNs and to compare it against other models.

The TRMSim-WSN is, as far as we know, the state-of-the-art simulation platform for confidence-renowned systems. It is aimed at simulating algorithms for reputation and trust management in WSN systems, but the same principles can apply to IoT systems in general. The simulation can be run over a single randomly generated WSN or over a set of networks. The user is able to define parameters of the network, such as the percentage of clients and that of malicious nodes. Network topologies may also be loaded from and saved to XML files. Sample trust and reputation models have been included and an API is offered which provides a template for the users to help them easily load new T&R models to the simulator[36]. For the tests, parameters that can be configured are: number of executions, number of random networks to be tested, % of Malicious Actors, Collusion between Malicious Actors, Oscillating behaviour of actors, etc.

[35] F. G. Marmol and G. M. Perez, “TRMSim-WSN, Trust and Reputation Models Simulator for Wireless Sensor Networks”.
[36] F. G. Marmol, “Implementing and Integrating a new Trust and/or Reputation Model in TRMSim-WSN”.

To evaluate our T&R model we compared it with three predominant (as of today) T&R models (Eigentrust, PeerTrust and PowerTrust) as well as with a relatively new system known as BTRM (Bio -Inspired Trust and Reputation Model) that applies a biological algorithm known as Ant-Colony System.



Num executions 100
Num networks 100
Min Num Sensors 50
Max Num Sensors 100
☐ Show ranges
☑ Show links
☐ Show ids
% Clients 20
% Relay Servers 14
% Malicious Servers 70
Radio Range 15
Delay 0
☑ Dynamic WSNs
☐ Oscillating WSNs
☐ Collusion
Trust & Reputation Model BTRM_WSN
Current: 88.16 %
Average: 85.5 %

```
Finishing simulations at Mon Apr 20 21:19:19 EEST 2015...
Running BTRM_WSN over WSN 43...
Running BTRM_WSN over WSN 42...
Running BTRM_WSN over WSN 41...
Running BTRM_WSN over WSN 40...
Running BTRM_WSN over WSN 39...
Running BTRM_WSN over WSN 38...
Running BTRM_WSN over WSN 37...
Running BTRM_WSN over WSN 36...
Running BTRM_WSN over WSN 35...
Running BTRM_WSN over WSN 34...
Running BTRM_WSN over WSN 33...
Running BTRM_WSN over WSN 32...
Running BTRM_WSN over WSN 31...
```

Figure 18: TRMSim-WSN.

We run simulations both in simple networks and in networks with dynamic entry or oscillating behaviour of actors. Measurements of the average satisfaction were made at various percentages of malicious actors (10%, 50% and 90%). The results are given in the next figure.

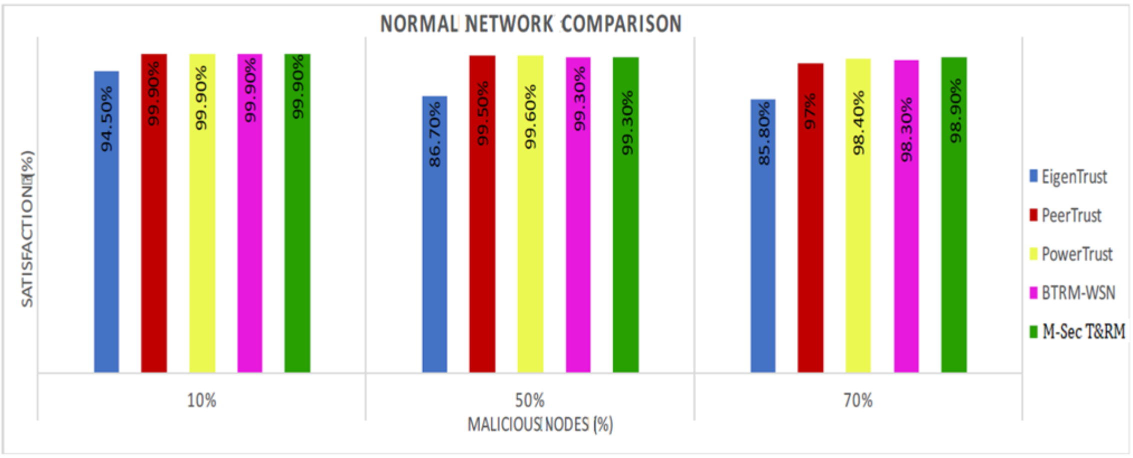


Figure 19: Normal Network Comparison

From the above, it can be seen that our models performance is comparable to that of the over models (and in some cases better).

During Y2, we are planning to continue the simulations for our T&R model, focused not only on the average satisfaction metric, but also on other important characteristics, such as scalability of the system. Once this level is completed, depending on the results, the model will be implemented as a mechanism on top of the M-Sec Blockchain and Marketplace and be tested accordingly.

4. Conclusion

In the previous section we presented three different demonstrators as part of T4.3.

In our next steps we are going to further explore the potential of smart contracts to support the different M-Sec Use cases and integrate middleware services with the rest of the components.

Trust levels over a trustless IoT infrastructure will be further researched in order to allow the convergence of the specific implementations in a broader smart city context. Blockchains will be studied also as a technology foundation upon which values can be exchanged in an IoT infrastructure, enabling thus devices to buy services from other devices.

Finally, a model that integrates Trust and Reputation model within IoT Marketplace and smart contracts will be further examined.