

BIGIoT – Bridging the Interoperability Gap of the Internet of Things

Deliverable 3.3c: Security and Privacy Design for Smart Objects

Version 1.3

Delivery Date: 21.08.2018

This project has received funding from the European Union’s Horizon 2020 research and innovation programunder grant agreement No 688038.

|  |  |
| --- | --- |
| Document Information | |
| Responsible Person and Affiliation | Stefan Schmid (BOSCH) |
| Due Date / Delivery Date | August 31, 2018 |
| State | Draft (One of: Draft, Reviewed, Final) |
| Reviewers | Karina Rehfeldt (TUC), Sebastian Kaebisch (SIEMENS) |
| Version | 1.3 |
| Confidentiality | Public |

|  |  |  |
| --- | --- | --- |
| Version | Description of Changes | Date of Resolution |
| 1.0 | Non-harmonized draft version | 18.07.2018 |
| 1.1 | First harmonized draft version | 27.07.2018 |
| 1.2 | Draft version for the internal reviewers | 10.08.2018 |
| 1.3 | Final version for the internal reviewers | 21.08.2018 |
| 1.4 | Final version | 31.08.2018 |

#### List of Authors

|  |  |  |
| --- | --- | --- |
| Organization | Authors | Main organizations’ contributions |
| BOSCH | Stefan Schmid | Section 2. Review of the entire document. |
| UPC | Juan Hernández-Serrano,  Carlos Dolader,  Joan Bel,  Jose L. Muñoz,  Oscar Esparza,  Olga León | Sections 2, 3, 4 and 5. Review of the entire document |
| AAU | Lars Mikkelsen,  Henrik Rasmussen,  Hans-Peter Schwefel,  Tatiana Madsen | Introduction, sections 1 and 4. Major revisions of the entire document |
| SIEMENS | Arne Bröring | Conclusions and outlook. Review of the entire document. |

#### Abbreviations

|  |  |
| --- | --- |
| Abbreviation | Meaning |
| **BCN** | Barcelona (Pilot) |
| **BIG** | The BIGIoT crypto token. It should/could be mapped to Euros (EUR). |
| **BIGIoT** | Project title: Bridging the Interoperability Gap of the Internet of Things |
| **DLT** | Distributed Ledger Technology |
| **DRW** | Defect Remediation Window. It is a privacy KPI. |
| **ETH** | Ether. The cryptocurrency of Ethereum |
| **IoT** | Internet of Things |
| **NG** | Northern Germany (Pilot) |
| **PIE** | Piedmont (Pilot) |
| **KPI** | Key Performance Indicator |
| **SC** | Smart Contract |
| **WPRT** | Weighted Privacy Risk Trend. It is a privacy KPI. |

Table of Contents

[Summary 7](#_Toc522613278)

[1 Design, development and evaluation of BIGIoT on the Blockchain 8](#_Toc522613279)

[1.1 Overview on Marketplace Functionalities 9](#_Toc522613280)

[1.2 Blockchain-based Marketplace Design and Implementation 10](#_Toc522613281)

[1.3 Qualitative Analysis of Security and Availability 16](#_Toc522613282)

[1.4 Testbed Assessment 17](#_Toc522613283)

[1.5 Analysis of Blockchain Forking 24](#_Toc522613284)

[1.6 Conclusion and Outlook 27](#_Toc522613285)

[2 Towards a European federation of BIGIoT marketplaces 29](#_Toc522613286)

[2.1 Introduction 29](#_Toc522613287)

[2.2 Architecture 31](#_Toc522613288)

[2.3 BIG: the BIGIoT token 33](#_Toc522613289)

[2.4 Privacy aspects 34](#_Toc522613290)

[2.5 Use of state-of-the-art technologies 34](#_Toc522613291)

[2.6 State-of-the-art 39](#_Toc522613292)

[2.7 Conclusions 40](#_Toc522613293)

[3 Data Exchange protocol 42](#_Toc522613294)

[3.1 Introduction 42](#_Toc522613295)

[3.2 State of the art 42](#_Toc522613296)

[3.3 Data Exchange with Random Verification 43](#_Toc522613297)

[3.4 Potential attacks 53](#_Toc522613298)

[3.5 Properties achieved 54](#_Toc522613299)

[3.6 Conclusions and future work 56](#_Toc522613300)

[4 The BIGIoT privacy risk rating methodology 58](#_Toc522613301)

[4.1 From the OWASP RRM to a BIGIoT RRM 59](#_Toc522613302)

[4.2 Evaluating the risk 59](#_Toc522613303)

[4.3 Factors for estimating likelihood 62](#_Toc522613304)

[4.4 Factors for estimating impact 65](#_Toc522613305)

[5 Privacy analysis of the BIGIoT pilots 68](#_Toc522613306)

[5.1 Key Performance Indicators (KPI) 70](#_Toc522613307)

[6 Conclusions & Outlook 75](#_Toc522613308)

[References 77](#_Toc522613309)

[ANNEX A. Risk Assessment 82](#_Toc522613310)

#### List of Figures

[Figure 1: Modified architecture for BIGIoT functionality implementation on Ethereum 11](#_Toc522613311)

[Figure 2: Flowchart of offering creation process in the Offering SC. 12](#_Toc522613312)

[Figure 3 Offering Query flow initiated from Consumer Client 14](#_Toc522613313)

[Figure 4 shows the performance obtained during tests of offering creations. 19](#_Toc522613314)

[Figure 5: Graph showing the performance of FMR search when there are no matching offerings. 21](#_Toc522613315)

[Figure 6: Graph showing the performance obtained with AMR-search. 21](#_Toc522613316)

[Figure 7: Graph showing results from offering modification testing on Blockchain offerings. 23](#_Toc522613317)

[Figure 8: Graph of performance of performing deletion operations on existing offerings in the blockchain. 23](#_Toc522613318)

[Figure 9 Example of situation leading to a chain fork 24](#_Toc522613319)

[Figure 10. Architecture for the marketplace generation 31](#_Toc522613320)

[Figure 11. Types of ledgers 36](#_Toc522613321)

[Figure 12. Data exchange protocol. Interactions with the SC 47](#_Toc522613322)

[Figure 13. merkle tree of encrypted data. In this example, MRC=H0123 49](#_Toc522613323)

[Figure 14. merkle tree of keys. In this example figure MRK = Hk0123 50](#_Toc522613324)

[Figure 15. Weighted Privacy Risk Trend (WPRT) 72](#_Toc522613325)

[Figure 16. Technical and business vulnerabilities 73](#_Toc522613326)

[Figure 17. Total and average Defect Remediation Window (DRW) in hours 74](#_Toc522613327)

#### List of Tables

[Table 1: results of FMR offering query testing with no and 10 attributes specified respectively when searching for the same matching offering (The matching offering is located at offering 32.) 20](#_Toc522613328)

[Table 2 First usable block given the average delay of the worst connection in the P2P network 26](#_Toc522613329)

[Table 3. Notation for the data exchange protocol 48](#_Toc522613330)

[Table 4. Likelihood and impact levels 60](#_Toc522613331)

[Table 5. Overall risk severity indicator 61](#_Toc522613332)

[Table 6 - Computing overall likelihood 61](#_Toc522613333)

[Table 7 - Computing overall impact. 62](#_Toc522613334)

[Table 8 - Use case clusters. 68](#_Toc522613335)

[Table 9 - Array of weights for the BIGIoT risk assessment. 69](#_Toc522613336)

# Summary

This deliverable is the second and final update of D3.3 – Security and Privacy Design for Smart Objects. The document presents the work performed in T3.3 of the BIGIoT project up to month 31. The document strongly relies on the BIGIoT architecture (see Deliverable D2.4) and on the marketplace design of BIGIoT (see Deliverable D4.1), which are recommended as prior reading.

This deliverable contains five main results. The first three results discuss the use of blockchain concepts in an extended BIGIoT system and are a completely new work that is included in only this version of the deliverable. The forth section is an update of what was presented in D3.3b. The fifth section extends and analyses the privacy indicators that were defined as a joint work with Task 5.4 and Deliverable 5.4b.

More in detail, a description of an Ethereum based design and implementation of the BIGIoT marketplace together with an analysis of the impact of the used blockchain features in a testbed assessment is documented in Section 1.

Section 2 introduces the use-case of ‘market federation’ and discusses how conceptually such use-case can be supported by the use of blockchains.

Section 3 introduces a protocol that uses the blockchain to allow a consumer to buy or just check a representative subset of providers’ data for testing while protecting both the consumer against malicious providers offering intentionally non-representative subsets of data, and the provider against releasing data to consumers without getting paid.

Section 4 provides updates on the BIG IoT Risk Rating Methodology presented in version D3.3b. The analysis performed following this methodology, and other previously introduced security analysis, has been reported to the third and final implementation iteration of BIGIoT services in the pilots. The detailed results for each service are shown in the Annex A.

Finally, section 5 defines and computes three privacy related KPIs. Although these KPIs have a limited number of evolution steps over time (as BIGIoT only had 3 development iterations), the KPI evolution nevertheless shows the increasing awareness and attention of the BIGIoT team to security and privacy related issues.

# Design, development and evaluation of BIGIoT on the Blockchain

So far, the BIGIoT Marketplace operates as a central entity where offerings are published and discovered, and from where access tokens are distributed and identities managed. The advantage of the centralized approach is that everything is collected at one place, but this can also be a disadvantage. If the marketplace becomes unavailable none of the consumers and providers could discover each other and initiate new connections. For this reason, this chapter explores how the marketplace functionalities can be made available in a distributed manner, while at the same time maintaining integrity in the data stored in the marketplace.

A distributed marketplace implementation can be realized using methodologies such as replication of the marketplace storage along with backup servers hosting the marketplace. However, new technologies are here investigated that may give a marketplace other characteristics such as increased availability, transparency, data security, and data integrity.

Blockchain is the common name for a continuous chain of blocks, each containing a set of transactions. The blocks are linked together with the cryptographic hashes, where each block contains the hash of previous blocks; as a consequence, blocks in the chain are in principle immutable. Blockchain is still an emerging technology, but with promises of distributed immutable data storage and distributed trust many use cases are envisioned. Furthermore, a key functionality offered by some blockchains is the ability to define and store Smart Contracts (SCs). SCs are a form of automated reliable agreements backed by consensus algorithms and strong cryptography. We could say that they are “programs” that are automatically executed by the nodes on the blockchain based on a set of conditions or triggers. Since SCs are backed up by the blockchain, they are reliable and perform exactly as they were programmed to. Using SCs is especially interesting for the case of a marketplace, where data contracts can be offered and subscribed to [1] [2].

This section investigates how a selection of marketplace core functionalities can be realized utilizing a blockchain. This is done based on a testbed implementation using the Ethereum blockchain, in which the performance is investigated. Furthermore, the consistency problem of blockchain stored data is investigated using a stochastic model where also network delays are taken into account. This chapter is based on the work done in the master student project at AAU [3] and a shorter version of this analysis is in the process of being published [4].

## Overview on Marketplace Functionalities

The BIGIoT marketplace allows providers and consumers of IoT related offerings to easily find each other in order to exchange data and services [5]. The marketplace also enables providers to monetize offerings, and in that way adding further incentive to publish offerings in the marketplace. Core functionalities of the BIGIoT marketplace are:

* **Account management**. Providers and consumers need to register an account at the marketplace. This account will be used later for authorization of services in order to allow access to data, or to associate earnings from use of data.
* **Create an offering**. The marketplace stores a description of the data that the provider wants to offer to other services or applications. Associated parameters often include data category, endpoint, input and output data formats, license, pricing models, geographic information and more.
* **Modify an offering**. The provider can also modify an offering, e.g. adding more input or output data fields or changing the license.
* **Offering query**. The offering is discoverable by other services by making an offering search query, which is matched with a subset of the available offering description parameter fields. The parameters in the query can contain values such that the offerings are matched based on these values, e.g. a specific category or a maximum price. The search query will return a set of matching offerings and the consumer can select one or more of them for subsequent use.
* **Subscribe to offering and related access management**. When a consumer subscribes to an offering, the access is managed by the marketplace. One method is to assign access tokens, which then are verified by the service that is offering the data before access is allowed.
* **Billing of offering use**. The marketplace will bill the consumer based on the amount of data consumed and/or the duration of the consumption. In the token based approach, the service providing the data offering would have to report to the marketplace how much data has been consumed and for how long.

If a consumer requests access to an offering from a provider with a valid access token, the provider should trust that it is issued correctly from the marketplace. Likewise, if a consumer discovers an offering, it should be a trustworthy offering from the provider in question, e.g. validated by the marketplace.

Of these core functionalities of the marketplace, a selection of three has been implemented using the Ethereum blockchain: 1) **Offering Creation**, 2) **Offering Query** and 3) **Offering Modification and Deletion**.

## Blockchain-based Marketplace Design and Implementation

For implementation of the selected marketplace functionalities a development version of the Ethereum blockchain is used. Ethereum is designed for creating applications utilizing the blockchain technology via the Ethereum Virtual Machine (EVM). The EVM contains the functionality to create a private P2P network and a private chain on that network. The developers can create their own decentralized applications (ÐApps) on that chain. One of the useful tools to create ÐApps is the creation of SCs, since they allow users to create code, which can then be executed on the chain. With SCs it is possible to deploy other functionality and thereby create decentralized applications.

The programming language used in this work to create SCs is Solidity, a contract-oriented programming language that is at present the primary language on Ethereum. Since SCs are located on the chain, they provide the same transparency and security that any other block on the chain would have. As contracts are contained in blocks, they cannot be deleted from a chain once they are committed. The only way to remove the contracts’ functionality from the chain is to include a method to disable them.

At the time of writing, the main chain in Ethereum takes up 629 gigabytes of storage and this is far more than can be expected for the end user devices or IoT devices. While off-chain transactions are possible, it is not possible to read and validate the state of the chain without having a full local copy of the chain. Therefore, the architecture of the BIGIoT marketplace build upon an Ethereum chain excludes the end users in the chain. The proposed architecture is presented in Figure 1.

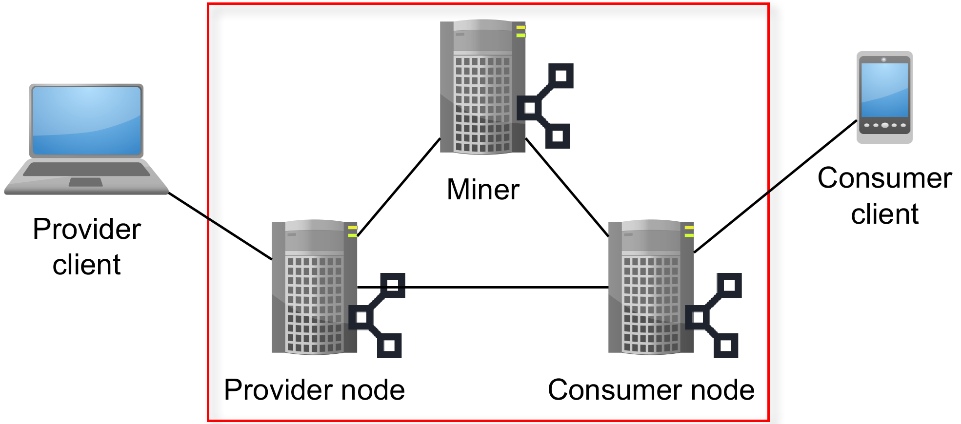


Figure 1: Modified architecture for BIGIoT functionality implementation on Ethereum

Elements inside the red square have a full copy of the chain and helps distribute changes, while nodes outside do not have access to the chain (referred to as foreign nodes or clients).

* **Consumer Nodes:** Consumers are Ethereum nodes that can interact with and modify the chain. They perform Search queries and subscription actions.
* **Consumer Client:** Consumer Clients are the end users of Consumer nodes and each Consumer should make sure it has resources to support their clients. This replaces the BIGIoT Consumers in the current BIGIoT system.
* **Provider Nodes:** Providers are Ethereum nodes that can interact with and modify the chain. Their primary task is to create offerings and manage their endpoints. These nodes replace Providers in the current BIGIoT system.
* **Provider Client:** Provider Clients is where a Consumer client will retrieve data from.

On a blockchain only two types of interactions are allowed; read and write. Any action that changes a chain costs Ethers, as such action is a computational task for the mining nodes. Reading from a blockchain is a local task performed by a node and therefore does not require mining.

The marketplace functionalities selected to be realized are subsequently described individually in terms of how they are designed and implemented in the following.

### Offering Creation

To ensure that offerings can be created by Providers, the functionality of the contract is open to all users in the chain. Any account on the chain with sufficient funds of Ethers to create a transaction can therefore create an offering. As the fields of such an offering, to a large extent, are customizable by the users, there is a need to validate the inputs given to the contract. The general functionality of creating an offering is described by the flowchart in Figure 2.

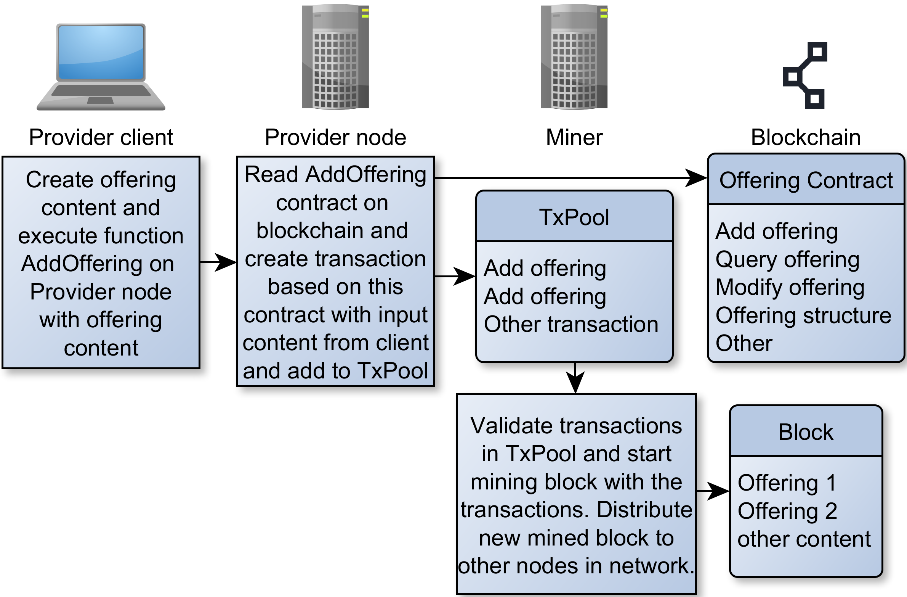


Figure 2: Flowchart of offering creation process in the Offering SC.

The provider node receives the content of the offering from a provider client. The provider node creates an offering transaction by using the Add offering functionality which is part of the Offering contract stored on the chain, and adding the transaction in the queue to be added in blocks to the chain.

A miner will now validate the content of the offering based on the Add offering rules defined in the Offering contract. Given the content is valid the miner includes this transaction (possibly along with other transactions) in a new block and starts mining it. If the block is successfully mined, the miner distributes it to other nodes in the network. If the other nodes validate the block, it will be accepted as the next block on the chain.

For realizing this, the language Solidity [6] is used, which is designed specifically for Ethereum. Since Solidity is still in development there are some general programming language features missing. One limitation of Solidity is that it is not possible to create 2-dimensional arrays. As a string in Solidity is seen as an array of chars, it is thereby not possible to return an array of strings. As it is possible to convert a string to the byte32 format, it then becomes possible to store them in an array. This will however limit the input and output parameters to a 32-byte size, but this is deemed sufficient in this case. Therefore, whenever a user adds inputs or outputs they are converted from strings to bytes32 and placed in a byte array. The reverse of this conversion is done at the Offering Query described in next section.

The advantage of using Solidity with the Ethereum blockchain is that the needed libraries are deployed on the chain and from there linked to the necessary SC. The link between them ensures that the functionality of the library can be used in the contract from any node and that functionality will be the same.

As this function modifies the chain by adding an offering to the storage, it requires a transaction to do so. This means paying a fee in Ethers in order for this transaction to be processed and stored in a block. Once this is done, the offering will be on the chain.

The SC concept is used in the following way when adding a new offering. When a provider client wants to create a new offering, it creates a transaction towards the Offering SC. This transaction works as a function call towards the Add offering function in the contract and must therefore contain the necessary parameters of the offering to be created. This transaction, once validated, works as a pointer towards the SC as well. This means when a node reads this transaction/block it understands that it belongs to, and specifies, fields in the offering SC. This transaction will therefore specify areas such as price, endpoint, id, license and so on. This information is then gathered in an offering struct which is accessible via the offering contract, allowing a potential consumer to simply utilize the offering contract to find the necessary information regarding the offering, without itself having to visit the block it was created in. In this implementation the fields for describing the offering are kept simple such that the query function likewise can be kept simple. However, to add more complex functionality and data structures this must be defined in the Offering Contract. The offering contract now contains all information and functionality needed for discovering it via offering query, and for subscribing to it.

### Offering Query

Before a consumer can consume data described in an offering, it should first find the offering that fulfils the data requirements. However, since a consumer client is a foreign node it cannot search through offerings directly on the chain but it must do this via a consumer node, which then returns the matching offerings to the consumer client.

The search functions are defined as part of the Offering contract on the blockchain, which means that all consumer nodes will employ the same search. This also means that it is not possible to inject a malicious search function in the system, e.g. causing one or more offerings never to be returned. The Offering Query flow is illustrated in Figure 3.

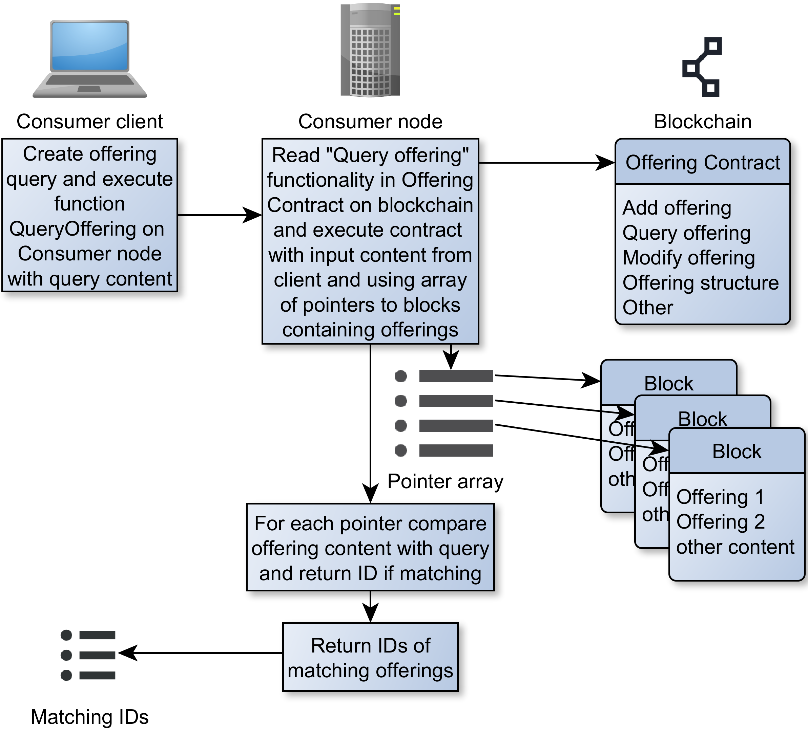


Figure 3 Offering Query flow initiated from Consumer Client

Two Offering Query functions have been implemented: One returning the first offering that match the criteria, referred to as First Matching Result Search (FMR-Search). And one returning a list of all offerings matching the criteria, referred to as All Matching Results Search (AMR-Search).

Solidity is used for interaction with the chain. Solidity does not support returning of arrays of structures [6], so it is not possible to return a list of the actual offerings, but instead the search functions return an ID or a list of IDs of the offerings for FMR and AMR respectively.

The miners add offerings to blocks and store them on the chain, where multiple offerings can be stored in the same block. External to the blockchain an array is created and maintained locally at each consumer that keeps track of which blocks contain which offerings, i.e. a pointer for each offering to which block contains it. By using an array external to the chain, it is possible to delete offerings in the sense that pointers can be removed from the array. The search function takes a query description as input, which contains the fields and field values to match with the offerings. The search function then iterates through the local pointer array and for each pointer it reads the offering it points to and compares the content to the search query. If the offering matches the search query, the ID of the offering is returned.

For AMR-Search all offerings on the chain are compared, and the IDs of the matching offerings are returned in an array. Unfortunately, Solidity does not support dynamically changing the size of arrays, so every time a new offering is matched a new array is created which is increased in size. The reason for this limitation in memory handling is that memory operations in solidity cost ethers because normally they are done and distributed on the chain. For this reason, the amount of memory operations performed are kept at a minimum. However, in this realization the memory operations are done locally and never written to the chain and distributed. Optimally the AMR-Search result data handling should be handled in another language to avoid this issue, but for this proof of concept implementation Solidity is used for simplicity.

When the IDs are returned from Consumer Node to Consumer Client based on the search query the client use them as input to another function on the Consumer Node that returns the offering content from the chain based on its ID.

### Offering Modification and Deletion

Deleting transactions from a blockchain can never be done, as such operation contradicts the logic of the blockchain concept. However, it is necessary to be able to make offerings invalid. For instance, the offering could become outdated or the Provider could discover an error in the created offering. In any case making the offering invalid is necessary. As deleting block cannot be done, Solidity has implemented an overwrite functionality, which overwrites the memory which was first reserved by the block containing the offering creation. This will create a new transaction overwriting the original offering. Therefore, the data inside the contract is removed, however, that data still exists in the chain in the original block. Over time a provider might want to change the content of an offering. Such change can be realized by deleting the existing offering and creating a new one. However, such approach would require the Consumer to subscribe to a new offering, and it is therefore desirable to modify existing offerings instead.

The offering parameters which can be changed are the ones, whose changes are keeping the offering still intact. This means parameters such as inputs, outputs and endpoints can be changed, as these are specifying how and where the offering can be used. Other parameters, such as offering price and license cannot be changed, as they fundamentally modify the offering, effectively creating a new offering. If modifications to these areas are required, the offering must instead be deleted and a new one created to replace it. To ensure that no malicious or accidental deletion or modifications of offerings are performed, the account which created the offering must also be the one to delete/modify it. If the source of the transaction does not match the offerings creator, then it cannot be changed. This implies that any offering created by a lost account can never be removed. This is checked based on the offering deletion function defined in the Offering SC which is stored on the chain. The actual check is performed when a miner is about to mine the transaction, in which case it checks if the transaction is legal.

As ‘Offering Modification and Deletion’ requires to store data on the blockchain, this use-cases also requires a transaction (and mining and ethers).

## Qualitative Analysis of Security and Availability

If a consumer node becomes unavailable it is not possible for the connected consumer clients to make subscriptions, queries or get authenticated, which is required to obtain an access token. In this case it is however only a fraction of the marketplace consumers that become disconnected. A similar observation can be made for provider nodes and provider clients.

In a centralized marketplace scenario all consumers and providers would become disconnected in case of a marketplace failure. Furthermore, depending on the setup of storage of offerings and subscriptions, these would have to be recreated in the worst case in the centralized scenario. In the blockchain case all offerings and subscriptions would remain intact and available as long as one of the chain nodes remain online.

Another key aspect to evaluate when realizing the blockchain based approach is related to security. As blockchain is founded on distributed trust and thus it is vital to evaluate how this potentially can be exploited maliciously as there is no central entity that can be entrusted with maintaining the system integrity, which is the case in the traditional marketplace approach.

One issue related to the distributed trust is called the 51% problem, referring to that if 51% of the mining power is controlled in a blockchain network it is possible to decide which blocks are accepted and which are not. This means that it is possible to deny service for specific clients. It is furthermore possible for the miner with the majority of mining power to double-spend [7] Ethers as the attacker secretly can mine on a private chain which will grow faster than the parallel main chain. The attacker can spend ethers on the main chain and then introduce the private chain to the network, which will be accepted as the main chain. In doing this, ethers spent on the previous main chain will be cancelled and returned, to be available to be spent again.

An important part of the blockchain concept is the transparency in transactions. Any transaction made can be viewed by all parties. This is necessary in order for miners and other nodes in the network to validate balances of accounts as well as allow transactions between users. In this project, the transparency ensures that any node is able to validate a transaction as well as access the SCs. However, it also means that any user has full access to the content and structure of the SCs. Any potential vulnerability in the contracts are then visible to potential malicious users. Another problem is that users can see other transactions. This allows a company to determine what subscriptions competitors have and thereby create similar competing services.

## Testbed Assessment

To evaluate the performance of the realized marketplace functionalities the testbed is used to assess the time needed for various operations. The architecture and the functionalities from the previous section have been implemented using a P2P network of nodes, where all peers share the same private Ethereum blockchain using the Geth framework [8]. Due to the blockchain mechanisms to adjust the mining complexity, the number of nodes in the network does not change the mining speed of blocks and therefore the testbed architecture can be simple. The architecture consists of a miner, a provider node and a consumer node connected in a ring structure. The connection is done via a local Ethernet. The nodes use the following common setup: Ubuntu 16.04, Geth version 1.8.10, Node.js version 8.11.1, Truffle version 4.1.0, npm version 5.7.1, Docker version 1.13.1.

As transactions are required to perform tasks in the offering contract it is necessary to test if the performance of the mining node is as expected. It is therefore important to note:

* The chain size does not directly impact transaction performance
* Internal parameters such as gas limit and difficulty does

To ensure consistency in testing steady state values for the latter parameters are used. This ensures that each test experiences as close to the same conditions as possible.

### Offering creation

The method for testing the creation offerings is the following:

* A new, empty chain is created with a mining node and a user
* The user migrates the offering SC to the chain
* The user creates 4096 offering transactions towards the SC function addOffer (4096 is the default max value of allowed transactions which can be stored in an Ethereum node)
* Once the Transaction Pool (TxPool) from the user is synchronized with the miner node, the miner node starts mining
* The starting block ID and the last block ID is logged for later analysis

By filling up the miners TxPool it represents the scenario of 4096 offerings arriving at the same time. The TxPool maximum size of 4096 transactions can be increased as needed, to allow more simultaneous transactions in the network, i.e. adding new, modifying and subscribing to offerings. This limitation should be explored in terms of the expected traffic in the BIGIoT scenario, but it is deemed sufficient for this test setup in order to test the timing of the realized functionalities. Creating an offering takes an estimated 425000-525000 gas to execute, this means that given the gas limit of 8 million (current gas limit in the public Ethereum chain), a block can contain roughly 16-18 offerings. We use the average of 17 to represent the amount of offerings processed in one block for these calculations. The expectation of the tests is that the results should closely follow this calculation as the expected block mining time is adjusted constantly to be as close to 14 second average as possible. This is also seen on Figure 4.

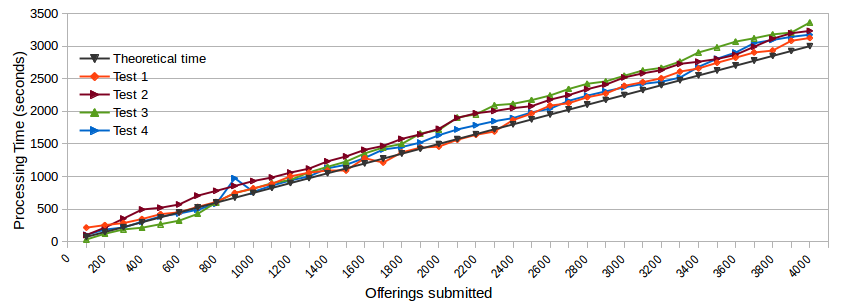


Figure 4 shows the performance obtained during tests of offering creations.

As is seen on the figure the overall trend of the four tests are identical. They each increase linearly based on the amount of offerings to mine. It is clear that the average performance of offering creation follows the theoretical calculations closely. There are almost no deviations in the comparisons between the two. The system therefore is not affected significantly when increasing the amount of offerings as the scaling is linear with the amount of offerings in the queue in front of it. However, this also speaks about the systems scalability to the extent of how it will handle an increase in offering creation rates. The results show it takes about 3224 seconds to process 4096 offerings. It is assumed that the rate of blocks mined is constant and that each block contains the same amount of offerings.

This is the maximum rate at which offerings can arrive on average without overflowing the miner’s TxPool. If an overflow happens, the miner will not register the transaction and the transaction would have to be resent at a later time. This can cause problems for the node who created the transaction. To ensure transaction order in a node, each transaction contains a node nonce. This nonce indicates the order in which transactions have been created by the node. Every time a node creates a transaction it will increase the nonce by 1. This is necessary to ensure that dependant transactions are not processed in the wrong order. If a transaction is lost, the following transactions from that node can become starved until the lost transaction is retransmitted. This problem it not limited to offerings but is present in any transaction-based solution.

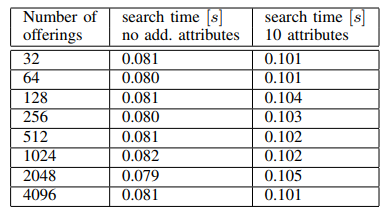
### Offering Query

The search time depends on two factors, namely the amount of offerings and the amount of parameters used to describe offerings. While queries depend on the amount of offerings in the chain, it does not depend on other scalability factors such as nodes in the network or how the network is connected. The following parameters are adjusted to gain a more complete picture of the offering query's performance.

* Number of offerings in the chain: This number will start at 25 offerings and increase to 212 offerings.
* Size of the offering query: This test will compare the performance when no additional search attributes in addition to offering category are specified with the case when there are 10 additional search attributes.

In a first set of tests, the FMR algorithm is executed for a search requests that finds a specific match within the first 32 offerings in the blockchain. Therefore, the size of the blockchain subsequently is not affecting the search time, as the columns of Table 1 show. When specifying more attributes in the query (here an additional 10 attributes to match), the search time is increasing, in this specific case by approximately 20ms or 25%.

Table 1: results of FMR offering query testing with no and 10 attributes specified respectively when searching for the same matching offering (The matching offering is located at offering 32.)



To evaluate the worst case search time, the search time is now evaluated for FMR-Search, when there are no matches for different amounts of offerings in the chain. The result for such experiment is presented in Figure 5. In that scenario, the FMR algorithm has to loop through the full list of offerings. Therefore the search time also depends on the number of existing offerings: An additional 4000 offerings adds about 375ms in search time in Figure 5.

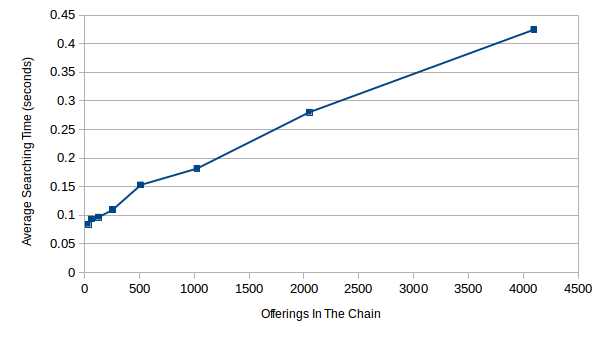


Figure 5: Graph showing the performance of FMR search when there are no matching offerings.

When running AMR search, the algorithm always has to process the whole list of offerings. Therefore, a similar dependence on number of offerings as for FMR in Figure 6 is expected.

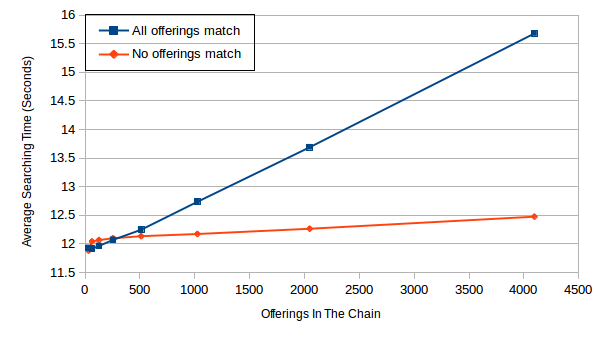


Figure 6: Graph showing the performance obtained with AMR-search.

Figure 6 shows the execution time of AMR search when there are no matching offerings (lower, red curve) in comparison to the case when ALL offerings match the request. An increase of about 0.59 seconds can be observed for the scenario of no matching, when the number of offerings is increased from 32 to 4096; that value is consistent to the increase observed in Figure 5. In the scenario where ALL offerings match (upper blue curve in Figure 6), the increase of search time from 32 to 4096 offerings is much stronger, here 3.75 seconds. This stronger increase is caused by the construction of the return data structure, to which each matching result is added by the FMR algorithm. In addition, this operation is not very efficient in the current Solidity version. The inefficiency of constructing even mildly complex data structures in Solidity can be observed even more strongly when comparing the left end of Figure 6, around 11.9s, with Figure 5, around 0.1s: Despite both algorithms need to process the same small list of offerings, AMR needs around 11.8s more to execute. This additional time is caused by the extraction of the array of matching IDs from Solidity to an application written in another language (in this case JavaScript), and for this reason is not directly associated with the search time. The only difference between the FMR and AMR algorithms in this case is that AMR creates a candidate list for storing the matching results. These results show that either the memory handling in the current Solidity execution environment is poor or returning arrays from SCs to JavaScript is in the current implementation very inefficient.

### Offering Modification

The method for testing the modifications of offerings is the same as creating them. A final step is added to the process which is the modification of the offerings. As modifying offerings is a smaller process than creating them, the gas it takes is lower and therefore more modifications can be processed per block. Modifications take roughly 30000 gas to complete and this allows for 270 modifications to be processed per block. However, as they still depend on the mining of blocks it is expected that shares the linear behaviour with the offering creation, but with lower values due to the increase in transactions per block. This is also evident with the test results seen in Figure 7. Again, they all share the same tendency as the theoretical calculations. However, as the blocks start to contain hundreds of transactions the lines have several straight lines as all modifications in the same block has the same completion time.

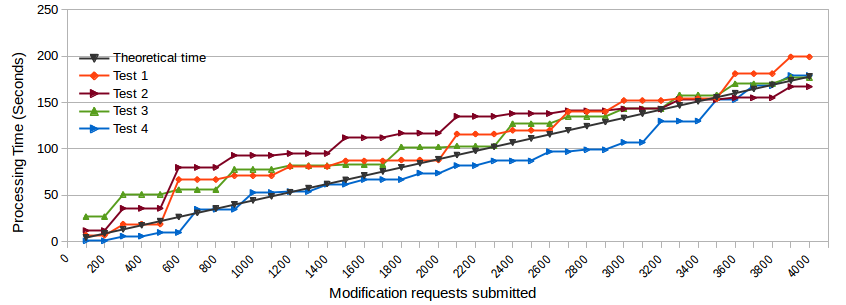


Figure 7: Graph showing results from offering modification testing on Blockchain offerings.

The same calculation for modifications can be performed to determine the processing rate of transactions:

### Offering Deletion

As deleting offerings is almost identical to modifying (modification: 30000 gas, deletion: 27000 gas) it is expected that they will share close to identical behaviour. This is also seen on Figure 8.

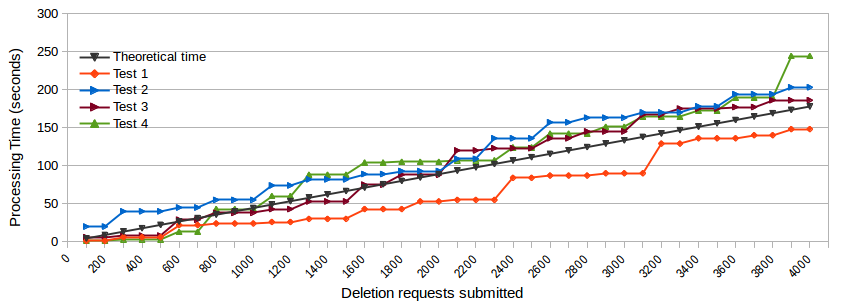


Figure 8: Graph of performance of performing deletion operations on existing offerings in the blockchain.

The same calculation for deletions can be performed to determine the processing rate of transactions:

## Analysis of Blockchain Forking

Working with a blockchain, one should be aware of inconsistency problems that can occur due to non-negligible delays needed for the distribution of mined blocks across the network. A situation can occur when two or more nodes have mined blocks almost simultaneous and they started to distribute the blocks to their peers. The first arriving block will be chosen by a peer to be added to the chain. When different blocks arrive first to different nodes, different instances of the same chain are created. Two or more separate blockchains will be active. This situation is called *chain forking*. It is illustrated in Figure 9.

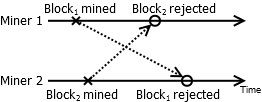


Figure 9 Example of situation leading to a chain fork

The fork is resolved when a new block is mined and distributed to all other peers in the network and the chain to which this block belongs has the most accumulated amount of work (i.e. this chain is the longest). The other nodes with another instance of the chain switch to this chain as it is considered to be a better alternative (the path with the most combined difficulty is always preferred). Other chains are eliminated by overwriting blocks that differ from the "winning" chain.

The fact that unintentional forks can occur in the chain leads to an important problem in the form of when a transaction can be deemed committed. In the considered scenario of a marketplace operation, forking can potentially lead to situations when consumers subscribe to an offering that in the future will be removed or the subscription transaction towards the offering could be lost, thereby not granting the user access to the data. In general, forking can result in double spending.

To ensure that a block stays in the chain typically means waiting until multiple blocks has been mined on-top of it, as this lowers the chances of the blocks being replaced. This introduces an additional delay, besides the delays considered in the testbed evaluation. Since it can be quite significant, it is important to include it into analysis of timing behaviour of the system. The purpose of this section is to provide a quantitative estimate of a number of extra blocks to be mined such that the block in question would remain in the main chain with high probability. This is especially important when dealing with payments since log times would badly affect the payment latency and the usability of the system.

In this subsection we set up a simple analytical model that provide a probability that a fork lasts more than n blocks. The number of blocks involved in the fork is called *fork age*. Two main parameters influence the fork age: the mining difficulty and block distribution delays. The harder it is to mine a block, the lower is the probability that a fork occur but the longer time it takes to resolve it. Larger delays of block distribution in the network means there is a bigger chance that individual nodes receive different mined blocks to be added to the chain.

The modelling is done under the following assumptions:

* All miners have equal mining power, i.e. the block mining times are independent and identically distributed (i.i.d.)
* The difficulty has converged, i.e. the rate of mined blocks in the entire network is on average 14 seconds.
* All miners are working on the same block number.

The first two assumption can be reasonable when operating on a private chain, while for public blockchains such as Bitcoin or Ethereum miners hash rates varies dynamically, even though difficulty targets are adjusted continuously [9]. The last assumption allows derivation of a simple analytical model, while we believe that relaxing this assumption would not have a big impact on the results: indeed, if the forks exists with different chain lengths, it is easier to resolve them, as any new block generated by the longest subchain will be accepted by all nodes belonging to other shorter subchains.

We are interested to find probability p that a mined block is distributed successfully to all other nodes in the network. This corresponds to a situation when no forking has occurred, the chain length is increased by one and all miners proceed to mine next block. Let T be a random variable denoting time needed to distribute a block to another node. Let τ be a mining time.

The time τ to mine a block is known to be well-modelled by an exponential random variable (see e.g. [10]). Let α be the mining rate of the process (in our implementation one block is mined every 14 secs on average). The distributed block will be successfully appended to another miner's chain, if no alternative block is mined during the time elapsed from the block generation until the block is received. Using introduced notation, we can write

for every node i in the network.

The distribution time of a block is mainly dependent on the time needed to send it over a communication network connecting nodes, as communication delay constitutes the major part of the overall distribution delay. Communication delay is determined by network topology, types of links used, communication protocols applied, as well as different dynamic factors, e.g. network conditions and traffic load. Here we follow a widely accepted approach in the literature to model communication delay as a geometrically distributed random variable (see e.g. [11]) We use the continuous equivalent, the exponential distribution. We assume that T is i.i.d. exponential random variable with rate λ. Under the assumption of exponential mining time and exponential distribution time, the probability of a successful distribution of a mined block is:

Now probability q that a fork will last for n or more blocks can be found as q=(1-p)n. For certain values of α and λ one can estimate the probability that the block can be changed when it is placed n blocks deep in the chain. Since we would like to have high probability that the blocks are confirmed after a number of transactions, for numerical evaluations presented further we have selected q=10-5. Table 2 shows a few examples of what the first usable block is depending on the mean communication network delay. In case communication delays between different nodes have different mean values, we can consider the value for the worst link in a fully connected P2P network and use this to make an estimation of the first usable block.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Delay mean | 50ms | 100ms | 250ms | 1s | 5s | 10s | 20s |
| First usable block | 2 | 3 | 3 | 5 | 9 | 14 | 22 |

Table 2 First usable block given the average delay of the worst connection in the P2P network

We compare the results in Table 2 with the current implementations of the popular blockchains. In Bitcoin a transaction is confirmed once it is 6 blocks deep in the chain [12]. EtherumWallet client states that a block needs 11 confirmations, i.e. it should be 12 blocks deep to be confirmed [12]. These numbers are not dependent on network conditions or other factors and there is no clear guidance why these numbers are selected. Looking at our results, it can be concluded that even a smaller number of blocks will be sufficient having good network connections. However, our analysis has a limitation of considering only the impact of non-intentional forks. As far as we know, this is the first attempt to connect consensus on a blockchain with experienced communication delays.

## Conclusion and Outlook

This section has summarized the architecture and implementation of the BIGIoT IoT marketplace based on the Ethereum blockchain. Three central use-cases of the marketplace are implemented and investigated: 1) **creation of offerings by providers**; 2) **modification/deletion of offerings**; and 3) **search for offerings by consumers**. As forks can occur in the network it is important that consumers do not use a new offering instantly. It is necessary to specify a waiting period before blocks are used. This waiting period has been defined as block age and shall be integrated into the design of the offering query.

As Solidity is a relatively new programming language, it lacks some features. One feature is the ability to return tuples and it has therefore been necessary for the SCs to perform the search internally. It was observed that some operations in Solidity or at its interface to JavaScript are extremely time consuming; these were operations which store data inside the SC such as arrays.

While Ethereum can add positive characteristics to the system it also comes with a cost. This Section has shown that it is possible to achieve the necessary functionality of the BIGIoT Exchange part on the Ethereum platform, however there are several trade-offs between performance, security, availability as well as development and operational requirements for users and BIGIoT themselves. One critical limitation is the offering query time, which was shown to have some limitations in terms of speed when returning the matching offerings. This could be explored further by making the search of offerings a completely off chain action, meaning using an external database with information and state about the offerings that are added to the chain.

Future work with respect to this blockchain implementation of the BIGIoT marketplace involves:

* Validation of the stochastic model for blockchain forking in the testbed using a network emulator.
* Further studies related to realizing accounting and payment functionalities using the blockchain technology and especially using SCs.
* Investigate the consequences when replacing the Ethereum blockchain with other blockchains.
* In this test setup the blockchain setup used is a unpermissioned public shared ledger where all nodes have the right to add transactions to the chain. This means that the management of the chain and thereby offerings are distributed to all nodes. Alternatively, a permissioned public shared ledger could be used where only a selected group of nodes are allowed to add transactions. This would mean that the distributed setup would still be in effect, while the control of the marketplace would be kept intact.

Despite the advantages on availability and security and despite the advantages provided by the transparency of SCs, advanced use-cases of marketplace federation, may benefit even more from the blockchain features and are therefore discussed in the next section.

# Towards a European federation of BIGIoT marketplaces

## Introduction

As stated in [13], nowadays the main barrier for igniting an IoT data marketplace are:

1. The lack of homogenization/interoperability among the different IoT platforms, services, markets. Missing standards, common APIs and data models make it difficult for data consumers to discover relevant data assets, access the data and also integrate the data in a uniform manner, especially when it is combined from multiple providers.
2. The lack of an efficient service discovery that eases the market exchange by, e.g., providing automated consumer/provider searches, data exchange and payments.

BIGIoT overcomes these difficulties by defining:

* **Central Marketplaces**. The different stakeholders (data providers and consumers) can meet on central marketplaces.
* **Semantic Models**. Semantic descriptions of data and services allows matching of providers’ offerings and consumer’s demand. No direct information exchange and negotiations are required, allowing completely independent data providers and consumers to meet and exchange and use data in a meaningful way.

However, there are still some challenges to be addressed:

* **Platform centralisation**. Central data marketplaces are often closed ecosystems that are in the hands of few established players or a consortium that decide on the rules, policies, etc. As business interest of ruling members often conflicts with those of new applicants, their request to join an ecosystem might be denied. This is a major barrier for growing a competitive data economy, where all stakeholders can freely participate under fair conditions.
* **Regulatory specificity** (sector, local, regional, national, international). A successful marketplace would need to comply with all the regulations. Moreover, the marketplace could offer legal-related services to participants. Conflicting regulations in terms of sectors/type of services or locations may render infeasible to have a single marketplace instance.
* **Personal data security.** Sharing of sensitive personal or industrial data assets demands high security standards for data spaces/marketplace platforms, interfaces, and special hooks for data owners to control with whom their data is exchanged.
* **Lack of transparency.** Data marketplaces normally don’t provide a complete historic of agreements between the different actors.
* **Untrusted management**. Lack of public/verifiable managing practices constitutes a trust barrier for new actors. In addition, the rights for using the data are often defined in contractual agreements among the involved parties.
* **Reputation of new actors**. Lack of previous transactions between consumers and new providers is another clear entry barrier to the marketplace. In nowadays marketplaces trust is built upon reputation and therefore it’s difficult to a new provider to compete with already-established providers.

In this section, we explain how to properly addressed these challenges by defining a federation of marketplaces connected through a shared backplane that is assured and audited by a distributed ledger. In short, the proposed solution relies on three keystones:

1. **Federation of marketplaces**. Federation of marketplaces competing for more consumers/providers removes wall-gardens among marketplaces and enables exchange of data across providers and consumers of different marketplaces. Such federation will encourage decentralization, trusted management practices and lower fees, among other things. Moreover, the individual marketplaces will be able to address the local/sector regulations and requirements.
2. **Reliable ledger for payments/contracts**. A distributed ledger (e.g. a blockchain) allows all the actors (consumers, providers, marketplaces) to easily inspect and verify payments and contracts. Moreover, using SCs, which are immutable and auditable, ensures that all actors are going play their expected role. Trust will rely on certainty and not on (past) reputation. This is expected to be a key feature to draw new customers (providers and/or consumers) to the ecosystem. This approach (distributed ledger and SCs) can also be adopted by individual marketplaces for handling local contracts and payments.
3. **Common protocols for data exchange**. DLT-based management of immutable and auditable SCs opens new possibilities for trusted data exchange, such as non-refusal protocols and statistical data verification. This will allow, for example, trusted exchanges without the need of a trusted third party.

## Architecture

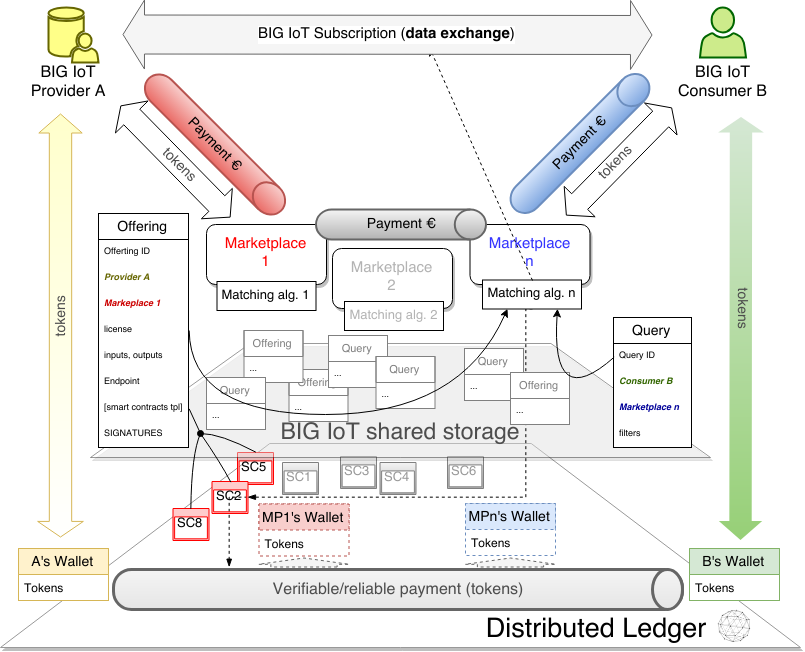


Figure 10. Architecture for the marketplace generation

The approach is based on three layers:

1. **direct P2P connections between consumers and providers** for the actual data exchanges (BIGIoT subscriptions);
2. **a BIGIoT shared storage** (mainly) for consumers’ queries and providers’ offerings; and
3. **a Distributed Ledger** that audits and validates the BIGIoT shared storage and implements the necessary SCs.

The BIGIoT shared storage will store offerings and queries and make them accessible for the matching algorithm in a fast-enough and reliable way. Although any cloud storage would work, as better depicted in Section 2.5.1, Ethereum SWARM [14], BigChainDB [15] or IPFS [16] are state-of-the-art solutions that could fit the goal; especially the latter one, which is to date the most mature one.

Offerings and, optionally, queries are linked to SCs that can be called to agree on a BIGIoT subscription. The capability of running SCs on top of the DLT provides a big improvement since they can run without any censorship or third party interference, and allows to avoid a counterparty risk. Therefore, it provides a perfect solution to implement a two-way protocol requiring trusted identification, neutral execution of previously agreed-upon checks and payment processing. To deploy and interact with the SCs, smart wallets are required. The wallets store stakeholders keying material that allows to redeem crypto tokens and allow users to execute SCs.

Although JSON-LD is still a valuable format, current BIGIoT offerings and queries description formats need to be changed as explained next.

An offering is created by a specific provider on a specific marketplace. Therefore, we need also to store identities for providers and marketplaces. A potential format is the one defined in [17], which also makes use of JSON-LD. The ID of any stakeholder must show proof of ownership of the smart wallets they own.

An offering supports one or a set of SCs (denoted as SC1, SC2, etc. in Figure 10). A basic set of SCs templates should be provided from the very beginning; although new SCs can be added later on. The offering must store the inputs needed to create and execute a SC for a specific subscription and consumer, with at least the wallet id (public address) that the provider uses for this offering.

Since offerings are on a shared space, marketplaces have access to all of them and can try to orchestrate connections between their consumers and providers assigned to any other marketplace. This fact enables a fair competition between marketplaces for the best matching algorithm and customer support.

## BIG: the BIGIoT token

Crypto currencies are not money (yet) mainly by two reasons: 1) **volatility**; and 2**) lack of regulation**. As a result, most of the companies need to rely on a (strong) fiat money: in Europe, Euro.

BIGIoT could use dedicated tokens to create electronic transactions between the different entities: consumers, providers and marketplace. The token could be called BIG. A token is an asset, not a coin. However, being part of the marketplace federation means agreeing on a value for the token. For simplicity, the federation could simply agree that 1 BIG is a digital crypto representation of 1 EUR.

The marketplaces are responsible for the exchange between the token and the local fiat currency where the marketplace is regulated, but only based on their operation; that is to say, marketplaces are not exchangers. Notice that it makes no sense to make business exchanging BIG and EUR since the value is not changing.

Consumers and providers exchange their BIG to EUR in their respective marketplaces. Marketplaces must also have traditional payment channels with the other marketplaces.

The issuers of BIGs could be the marketplaces, but should better be traditional banks. Several banks are already working in issuing a digital crypto asset equivalent to Euro, indeed. For example, Banco Sabadell in Spain is developing BSToken with such a goal (https://github.com/BancoSabadell/bs-token). Involving the bank sector should be a priority for properly igniting the ecosystem.

The main advantages of using BIG instead of fiat money (i.e. Euros) are:

* **Reduced backbone for payments**. The DLT is enough to manage secure payments of BIG among the federation stakeholders.
* **Reduced bureaucracy**. Money redistribution is automated by the SCs. Marketplaces should only take care of the exchange to/from fiat money (EUR) of the final balance of their customers.
* **Reduced fees**. Although this is very related to the underlying distributed ledger technology and the protocols used for the payment channels, state-of-the-art solutions spend less fees than traditional payment options.
* **Transparency, auditability and accountability**. All the SCs are public. Therefore, there operation is transparent. Moreover, as they are run on a DLT, their actions can be easily audited and accounted.
* **Automated, verifiable and reliable payments**. Payments are automated by the BIGIoT libraries and since payment channels will be backed up by a DLT, they will be reliable and perfectly verifiable.

The costs reduction and the enhanced trust on the system will allow a competitive platform that will incentivize all parties (providers and consumers) to participate in this ecosystem.

## Privacy aspects

Marketplaces and data owners demand high security standards in order to share industrial or personal data assets. The new European privacy regulation (GDPR) requires an unprecedented level of transparency and control for end users. SCs enable a solution to accomplish the transparency required laying down great flexibility in case changes are needed. Using novel SCs that involve end user as data owners (in case of personal data assets) the GDPR can be satisfied as data owners are able to give or revoke their explicit consent to share their data.

Wallets manage the necessary keying material for enabling the various stakeholders (incl. end-users) to interact with the SCs. Despite inherent privacy issues exist, solutions are emerging, such as **transactional confidentiality**. All the content being public, including state history and changes, allows external parties running a node to trace potentially confidential information. A working countermeasure is using zero-knowledge Succinct Non-interactive ARgument of Knowledge (zk-SNARKs) [18]. It would keep their spending records and the balances of their wallets anonymous, not letting the nodes that are executing payments or running SCs know the identities of the counterparties.

## Use of state-of-the-art technologies

### BIGIoT shared storage

The shared storage offers the federated marketplaces a common storage for the shared offering descriptions as well as the semantic models that are needed to describe the offerings in an interoperable manner. This shared storage can be built using several existing distributed storage solutions or simple cloud storage. A common API is integrated into each marketplace to interact with the shared storage. The interface with the BIGIoT shared storage should account for the following:

* Key management, authentication and authorization.
* Registration of resources from providers: namely the semantic descriptions of offerings.
* Accounting of provided and consumed resources by auditing the SCs actions managed in the DLT layer.

Swarm [14], BigChainDB [15] and IPFS [16] offer comprehensive solutions for an efficient decentralised storage layer. The main properties offered by these architectures are low-latency retrieval, resilience to node’s disconnections, liveliness and censorship-resistance. Moreover, these solutions can provide a versioned registry of all the content that has been stored.

The shared storage is backed up (in terms of verifiability) and complemented by a DLT (potential DLT options are described in Section 2.5.2).

### Distributed Ledger

A DLT is a shared and synchronized database maintained by a set of participants known as nodes. DLTs require the use of consensus protocols ensuring that all nodes agree on what information is stored and the historical order of the inputs (often referred to as transactions). Depending on the number of nodes storing the information and the consensus protocol implemented, properties are varying in several aspects, e.g. the level of decentralization, accessibility, auditability and integrity.

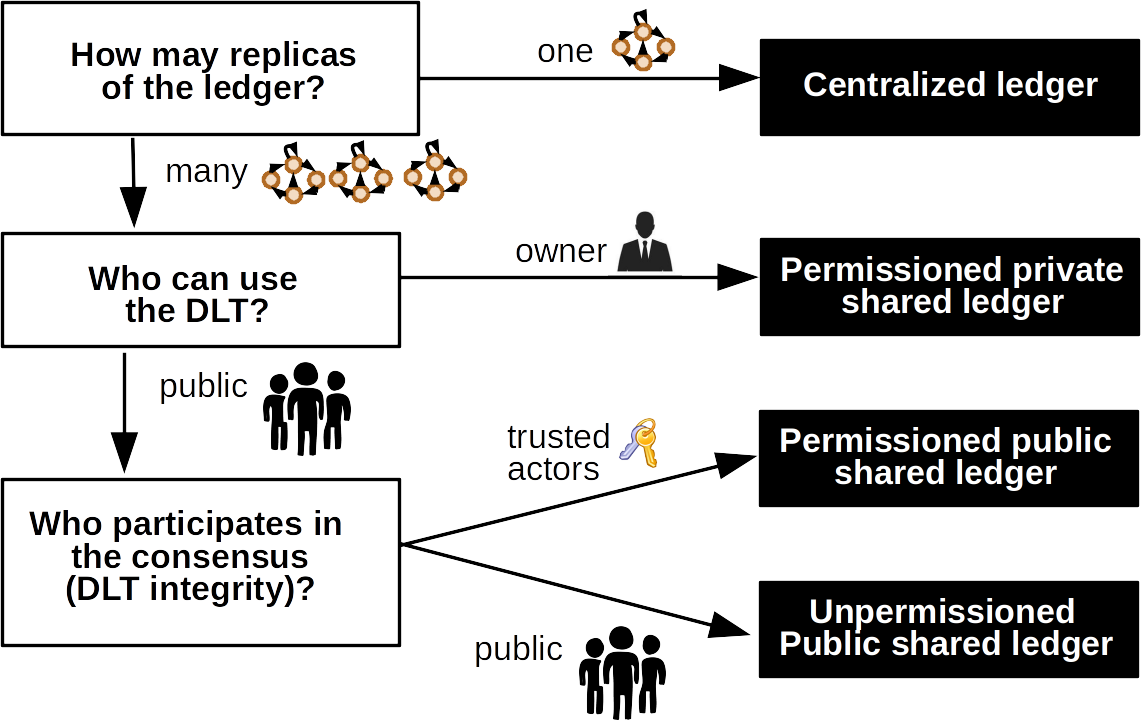


Figure 11. Types of ledgers

As shown in Figure 11, there exists three main categories of distributed ledgers:

1. **Permissioned private shared ledgers**. Write permission is centralized to one organization while read permissions can be public or restricted. Since the consensus is often run by a “small” set of nodes, it could provide a more efficient and scalable ledger at the cost of less centralization and resilience against node failures. If read permissions are public and the consensus algorithm is hard enough to ensure that the blockchain cannot be rewritten, these ledgers could also be audited but, since all the nodes are operated by the same organization, it is difficult to trust their operation. In our opinion, permissioned private shared ledgers are senseless and do not provide any benefit over standard cloud approaches.
2. **Permissioned public or consortium shared ledger**. Nodes running the consensus protocol are pre-selected among all the participants (public) or, more commonly, because of being part of a consortium. In the latter case, decentralization is only partially reached, but higher efficiency and scalability could be achieved as a result (the numbers of nodes is low enough to maintain high throughput). If read permissions are public, these ledgers could also be audited by an external third party.
3. **Unpermissioned public shared ledger**. The network is public in terms of writing and reading. So anyone can read and audit the data, and directly participate in the consensus process. Decentralization and immutability are preserved. On the other hand, unpermissioned public ledgers have low efficiency because all nodes participating in the consensus algorithm must replicate the same deterministic operations to add information in the ledger.

### Consensus algorithms

Writing to the shared ledger in a sorted, agreed manner while preserving security and providing fault tolerance is the main goal of consensus mechanisms. Algorithms have to be resilient to transactional problems, including double spending attacks, delays or errors messaging, and node failures: disconnections, erratic behaviour and malicious behaviour. To sum up, consensus must ensure a consistent global/general state. Several solutions exist to achieve this goal: proof-of-work (PoW), proof-of-stake (PoS), practical byzantine fault tolerance (PBFT), delegated proof-of-stake (DPoS), and more are yet to come.

Byzantine Generals’ problems [19] was the first work to formally address the agreement/consensus problem that appears when malicious generals are participating in a decision that must be agreed upon by all generals in a coordinated attack. The Bitcoin network [20] presents a solution to Byzantine fault tolerance with a Proof-of-work (PoW) protocol. Miners, participants that write the information in the ledger, must spend resources to find a nonce that, hashed with the information that is going to be included, must be smaller than a given value.

Assuming that more than half of all miners are honest, the properties accomplished in PoW protocol are:

* **Censorship resistance**. All transaction will be included in the network no matter who is creating them.
* **Reversion resistance or immutability**. Transactions already included in the blockchain cannot be changed or deleted.
* **Robustness**. The network can tolerate that some nodes are faulty or offline.

These properties created a new paradigm where decentralization is truly achieved and no central entity was in control of the network. Furthermore, in case of Bitcoin, besides the privilege of choosing what and in what order the information is added to the Blockchain, the same mining process incorporated a reward for miners in order to incentivise participation, and created a new economic system. PoW is a protocol that was predominantly designed for unpermissioned public shared ledgers; although with several efficiency (energy consumption is very high), scalability and latency challenges that are not yet solved.

Another alternative to PoW is proof-of-stake (PoS) where nodes that write information in the ledger are selected according to the amount of stake (e.g. currency) they own. Several cryptocurrencies adopted such an approach, such as PeerCoin [21] and the (potentially) upcoming Ethereum Casper, which combines both PoW and PoS. These projects hoped that PoS would reduce the vast amount of energy required for mining with PoW at no cost in terms of security and decentralization. The PoS community was enthusiastic about their new consensus method, but sceptics were quick to cite some theoretical security issues facing PoS. For example, with PoS nothing stops an attacker (there is no cost in mining) to mine a fork and create a hard fork. This flaw is known as the nothing at stake problem [22] and is often addressed with a parallel penalisation system.

A variant of PoS is the Delegated PoS (DPoS) where owners of the currency delegate the generation (mining) and validating processes to nodes they choose based on some given criteria. DPoS protocols perfectly fit in permissioned ledgers where nodes can be pre-selected by assigning them some stake.

Another option for the consensus algorithm is the “classical” practical byzantine fault tolerance (PBFT) [23], which was a first approach for solving the Byzantine Generals problem. This algorithm can tolerate *k* failing nodes if there are 3*k*+1 nodes available. PBFT relies on the classical agreement paradigm of 1 identity 1 vote. Therefore, it is unpractical for unpermissioned distributed ledgers where any participant could create several identities (sybil attack) and disrupt the consensus voting protocol. For this reason, PFBT can be only used in permissioned ledgers, such as e.g. Hyperledger Fabric [24]. While PBFT requirement of a permissioned ledger could be seen as a constraint since it is not a truly distributed approach; on the other hand, PBFT could save a lot of resources and thus could be more efficient and scalable.

### Initial choice of technologies enabling the BIGIoT federation of marketplaces

A permissioned or consortium-shared ledger among the marketplace owners and data providers would fit quite well assuming that all players agree which nodes participate in the consensus algorithm (for writing on the ledger). An initial approach will be to allow nodes run by the federating marketplaces and/or financial or legal entities (e.g. public administration bodies or banks) watching over the appropriate operation of the market.

Obviously, if such a consensus cannot be achieved, a unpermissioned public ledger (e.g. Ethereum mainnet) would be needed. However, it would increase the fees in transactions costs and confirmations time compared to a permissioned ledger.

Another option would be to define some central authorities that will have control over the network, introducing nodes that validate operations to secure tax payments and compliment regulations. This solution would need some degree of expertise from authorities which may be a barrier to execute.

In all cases, the chosen ledger technology should be immutable, auditable and support the SCs that a marketplace federation may require. Whether permissioned or not, the most mature DLTs that could suit the needs of the BIGIoT federation are Ethereum [8] and Hyperledger Fabric [24].

## State-of-the-art

A solution to join the BIGIoT shared storage and the distributed ledger have been presented in Section 1. In the solution, an Ethereum blockchain has been used to create an architecture where a set of SCs run some of the core BIGIoT functionalities, such as offering creation, modification and deletion. The experiment places consumers and providers, who create and update offerings, directly as miner nodes in the new network, in order to test the performance of their solution. Besides making it clear that special care should be taken to define the SCs in order to avoid unnecessary costs and undesired delays, the approach stills needs additional functionalities that should be implemented not in a DLT layer but in shared-data layer. The BIGIoT shared-data layer, which is core to the BIGIoT federation, is therefore a natural evolution of what is presented in Section 1.

To the best of our knowledge, the closest solution to what we are presenting here is the streamr project [25] which present a solution using an API and what they call broker nodes. The API functionalities include everything related with the shared-storage layer explained in this work: creation and subscription of offerings. Broker nodes handle the traffic between providers and consumers and are coordinated in order to distribute the traffic among all the nodes. This way the all data is partitioned and each node manages their own information allowing to make the system scalable.

Broker nodes report checksums for their assigned partitions to the network coordinator SC and deliver data to any SC subscribers. Although executing these operations has an associated cost (in Ethereum gas) that broker nodes should pay, the cost can be compensated because the nodes are rewarded with tokens (which are assumed to have more market value than the costs). The same token is also used as the currency for the buy and sell data processes.

The approach that we are studying in this section is conceptually different to streamr since, in our approach, no intermediaries (broker nodes) are in place when the data is exchanged between a provider and a consumer. From our point of view, any intermediary can and should be avoided to enhanced the system efficiency and decentralization. Moreover, our solution will not rely on the appreciation of the crypto currency to be economically viable.

## Conclusions

DLT have great potential to enable the creation of an open, trusted and scalable data economy based on federating marketplaces based on BIGIoT and other participating partners.

DLT will help the participating marketplace operators, who collectively run the **consensus-based, self-governing and decentralised backplane**, as well as the data owners, providers and consumers to establish trust in the overall solution.

In particular, we evaluated the following capabilities:

* **Decentralised storage** and access to semantic descriptions of the offered data assets in order to enable data discovery across today’s silos. This enables federation among the individual data spaces and marketplaces, without the need of central control or coordination that has to be trusted by all parties.
* **Immutable and auditable SCs** for the trading of data assets across data space and marketplace boundaries. All stakeholders, namely data providers (for confirmation of the offer and its conditions, e.g. license, price and SLAs) and data consumers (for agreement of the contract conditions). This solution can also be adopted by individual marketplaces for handling local contracts.
* **Crypto token** to provide a transparent, cost-efficient and fast payment solution for trading data assets among the participating data spaces and marketplaces. The crypto token will also incentivise data providers to offer their data assets. The tokens can also be used by the federating marketplaces as an internal payment medium.

Moreover, for the actual exchange and transport of data among data providers and consumers, we recommend to leverage the BIGIoT data access API. This has the advantage that data can be efficiently exchanged/transported and remain on the provider infrastructure – without the need to store or pass this through any central or intermediary nodes.

# Data Exchange protocol

## Introduction

The use of data within a business has become an increasingly important aspect to the business success. Research has shown that proper use of big data techniques helps identify new insights, optimize operating processes and take better and faster decisions. In this context, ecosystems have grown to fulfil the data needs of diverse actors in the market: data suppliers, data custodians, data aggregators… Therefore, players in the market not only collect the data they are generating and analyse it, but increasingly rely on third party data to enhance its business value. The remaining challenges of the sector are, notably, regulatory complexity, privacy, the difficulty of making proper data agreements and the **difficulty of valuing data and convincing customers of its value without giving it away** [26].

The creation of marketplaces addresses many of this problems. It helps to join interests between providers and consumers in a platform where both parties can meet each other and trade information. It solves the integration problem of connecting users and providers; however, some other drawbacks still remain. This work will focus on the problem of convincing consumers of data value, which can be seen as a form of **lack of trust in data providers**: a problem that still cannot be solved without previous confidence establishment. It manifests mainly as an entry barrier to providers in the market [27], hurting competence and thus reducing utility for consumers. A protocol will be presented to address this problem in a data trade, while preserving the necessary security, privacy and fairness characteristics that a marketplace should have.

## State of the art

A standard solution for trust establishment/maintenance is to use identifying information, since requiring e-mails, credits cards and other data constitutes an entry barrier to fraudsters. In pure data markets, being essentially business to business, it is even more feasible to demand identification warranties. In some cases, it can be also used as a negative incentive by, for example, recurring to law enforcement if tangible fraud is committed. These are however tedious and expensive processes and may not be feasible when the problem is that the quality of the data is not fulfilling expectations.

From a consumer/client perspective, credit card payments often work as a hidden escrow service in e-commerce, as there are many options to get reimbursed if fraud occurs. Reputation systems are also a common way to penalize perceived malicious actors and reward legitimate ones, for example with more visibility.

Normally a combination of all these techniques is used in successful e-commerce websites [26]. However, these methods are incompatible with anonymity. On top of it, they pose entry barriers for new actors.

## Data Exchange with Random Verification

Achieving the exchange of virtual products with many parties while minimizing risks is the main goal of virtual commerce. In order to exchange value safely it is essential that i) consumer is getting the product that it is paying for and ii) provider is paid. This is achieved, many times, without any strict protocols, just by existing trust. Counterparties know each other and are confident, by previous experience or by future interest, that no intent to scam will be made by other parties. When stronger assurance than that provided by direct mutual trust is needed, a trusted third party (TTP), whom all the parties trust, guarantees the process is carried out to all parties. It solves the crucial aspect of minimizing risk but comes with an extra cost for all parties and the need for a viable TTP.

In general, Distributed Ledger Technologies (DLTs) can be seen as a paradigm shift when it comes to the need of trusted third parties for something as critical as payment processing. The use of public-key cryptography provides strong assurance and easy verifiability when attributing actions (payments) to specific participants (cryptocurrency owners). Using DLTs, all participants in the network can maintain a synchronized version of a database (who owns what) without the need for a central authority (TTP) that guarantees integrity, fairness or availability.

The protocol hereby presented for data exchange shares the goals previously stated for virtual commerce. It also provides a solution to the lack of trust in data providers, and the data they provide, conceptually different from those explained before: offering a sample of the data to the consumer. Therefore, its goals are:

1. **That the consumer gets a fair sample of the data being traded before committing itself**. The protocol must ensure that the whole data is statistically consistent with this sample and so that the sample is not influenced by the provider. Thus the consumer can assess its value realistically.
2. **That the provider is paid, if and only if, the consumer has access to the data**. Thus the transaction is correct: the consumer cannot get the data without paying and the provider cannot get paid without giving away the data.
3. **That the solution is cost-efficient**. Thus the cost of running the protocol is much smaller than the value of the data it can trade. Due to high fees on public ledgers, the cost of the designed solution must remain mostly fixed and amount of data saved on the network mostly independent of the quantity of real data traded.

### Design of the protocol

The design presented here is for selling a set of non-real-time data, although the protocol can also work with a flow of real-time data by paying for batches of real-time data, obviously with some delay that must suit the service policies. The design rationale of our solution is depicted next and more details are provided in subsequent sections.

The protocol begins by generating a sequence of encryption keys to create a set of cryptograms that result from sequentially encrypting the fragments of clear data with the generated keys.

The provider delivers the complete set of cryptograms to the consumer. Obviously, it would be totally impractical to exchange that amount of data using the ledger as a storage. Instead, the exchange of cryptograms between the provider and consumer is done off-blockchain. This approach is completely aligned to the current BIGIoT philosophy of direct connections between providers and consumers which is clearly stated in Deliverable D2.4 and also followed in the approach presented in Section 1.

Next, the provider deploys a SC in a ledger that stores the root of two merkle hash trees. One root is the root of the tree formed hashing the cryptograms and the other root is the one resulting from hashing the keys. A merkle tree [28] is a structure that allows for efficient and secure verification of content in a large body of data. This structure helps verify the consistency and content of the data.

Commitments using root hashes are very efficient, in this case, since two hashes are stored in the ledger. These roots will allow to prove whether an element is or is not a cryptogram or a key and its position in the merkle tree. Using this, we will be able to assure to the consumer that the provider cannot alter the committed data set.

Then, some encryption keys are revealed by the provider in order to disclose a fair data sample that could convince the consumer about the value of the data set. At this moment, it is very important for the producer to make sure that, with the set of disclosed keys, it is not possible for the consumer to derive any previous or posterior keys of the sequence of keys used to encrypt the data; although the consumer must be able to easily derive all the keys when it agrees to buy the data set. A simple way to get this is to create the keys as hashes of a (initially private) seed and the index of the key in the sequence. This construction has the properties that we want: without the seed, the consumer cannot derive any other keys but once the seed is known, it is easy to regenerate the sequence of keys. Notice also that, since the seed will be public at the end of the process, all the off-blockchain traffic including cryptograms must have been exchanged using a secure channel.

Another important issue that our protocol has to solve is how to select and exchange the keys to disclose the fair data sample. The important aspect here is that the provider cannot control this process because, if this happens, the provider could decide to use a biased (not fair) data sample. Then, we have two options, either let the SC decide the indexes to be disclosed and or let the consumer decide them. We use the second option because then, the selection and exchange can be done totally off-blockchain. Doing this on-blockchain requires the availability of a random selection algorithm within the SC, which is not a trivial issue for distributed ledgers.

Since getting the sample has no cost for the consumer (it is off-blockchain), a potential attack could be a consumer trying to get a large amount of free data by repeating the process of getting small samples. In such a case, the provider can decide how many times it allows getting a sample set without making a final deal and it can use the marketplace to blacklist abusive consumers.

Finally, if the fair sample convinces the consumer, it will send a transaction to the corresponding SC with the payment to buy the data. Then, the provider discloses the seed that allows the consumer to derive all the keys, decrypt the previously received cryptograms and access the data. If everything is OK, the protocol ends after some time by sending the payment to the producer or by letting the producer withdraw the payment.

The final aspect that our protocol has to solve is when the provider either uses incorrect keys (keys that do not correspond to the sequence generated by the seed) or sends incorrect cryptograms. In both cases the consumer will be unable to decrypt a set of cryptograms to obtain the data. For such a reason, our protocol adds an optional final step that could be used to resolve conflicts by allowing the consumer to challenge the provider to proof that it sent a valid key for a given valid cryptogram. Conflict resolution an exceptional situation that is, in general, discouraged because it has costs for either the consumer or the provider.

When the subscription is created, the provider locks an amount of crypto currency that could be used to execute the conflict resolution. If the consumer is not able to decrypt a cryptogram (either because of an incorrect key or an incorrect cryptogram), the consumer executes the conflict resolution on the SC. If the provider cheated, the SC will refund the consumer and the cost of the execution will be charged to the provider by unlocking the money locked for conflict resolution. If the provider didn’t cheat, the consumer will have to pay the execution of the conflict resolution in the SC, so that there are not incentives for a consumer to cheat.

Although explained and detailed below, for reference purposes Figure 12 pictures the protocol operation and the interactions between the different stakeholders and the SC. Moreover, Table 3 summarizes the notation in use.

We assume a marketplace model similar to Section 1, in which offerings are stored via SCs on the DLT. Figure 12 does not repeat the full marketplace concept from Section 1 but instead focuses on the interaction with the SC during the search and pre-subscription phase. Red text in the figure denotes calls to SC functions, the ovate shapes represent the different states of the SC.

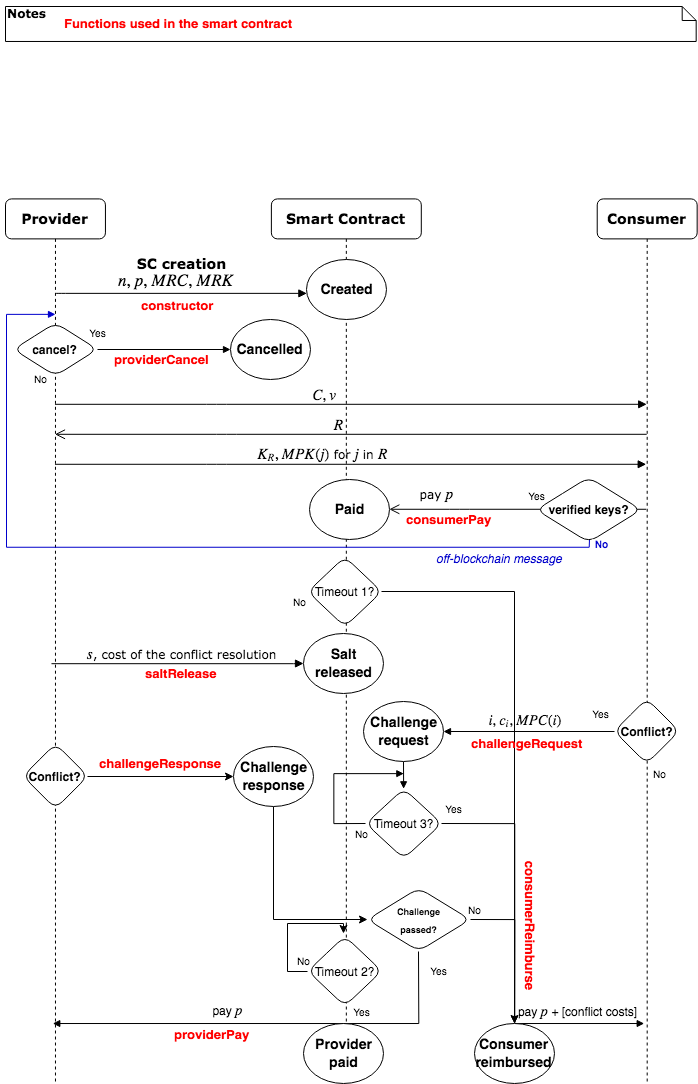


Figure 12. Data exchange protocol. Interactions with the SC

|  |  |
| --- | --- |
| Hash(*X*) | Digest obtained from applying a cryptographically secure hash function Hash to *X*. In a Ethereum implementation it would naturally be keccak256. |
| Enc(*ki*, *x*) | Symmetric encryption of message *x* with key *ki* |
| Dec(*ki*, *x*) | Symmetric decryption of ciphertext *x* with key *ki* |
| *n* | Number of data samples. |
| *D* = {*d1*, *d2*, … , *dn*} | Data samples. |
| *C* = {*c1*, *c2* , … , *cn*} with *ci* = Enc(*ki*, *di*) | Encrypted data samples. |
| *s* | Cryptographically secure random number for key creation (salt). |
| *K* = {*k1*, … , *kn*} with *ki* = Hash(*i* + *s*) | Symmetric keys used to encrypt data samples di and decrypt encrypted data samples *ci.* |
| MRK | Root of the merkle hash tree containing all the elements of *K*; that is to say that MRK is the merkle root of the tree with leaves *k1*, …, *kn*. |
| MRC | Root of the merkle hash tree containing all the elements of *C*; that is to say that MRC is the merkle root of the tree with leaves *c1*, …, *cn*. |
| MPC(*i*) | Merkle proof for leaf *i* of the merkle hash tree with leaves *c1*, …, *cn*. |
| MPK(*i*) | Merkle proof for leaf *i* of the merkle hash tree with leaves *k1*, …, *kn*. |
| *V* | Number of data samples to be revealed. |
| *R* = {*r1*, *r2*, … , *rv*} | Indexes of the samples to be revealed |
| *p* | Price |

Table 3. Notation for the data exchange protocol

#### Step 1. Data preparation and SC creation (provider)

The provider must prepare the data, splitting it in different samples. There will be cases where data splits “naturally” (e.g. records of a database, where each record is a sample) and others where samples must be generated from a compact source of data (e.g. video). Data must be sorted, using the most valuable variable, in case there are multiple ones. This will avoid a potential attack explained in Section 3.4.

A random seed number, *s*, must be generated in order to create the keys that will encrypt each sample of the whole data separately. Once data is split into different samples and a seed is created, the provider must create two merkle hash trees:

1. Merkle tree of encrypted data: each leaf will be an encrypted data sample *ci*. An example is shown in Figure 13.
2. Merkle tree of keys: each leaf will be the key that deciphers the same element in the Encrypted data merkle tree. An example is presented in Figure 14.

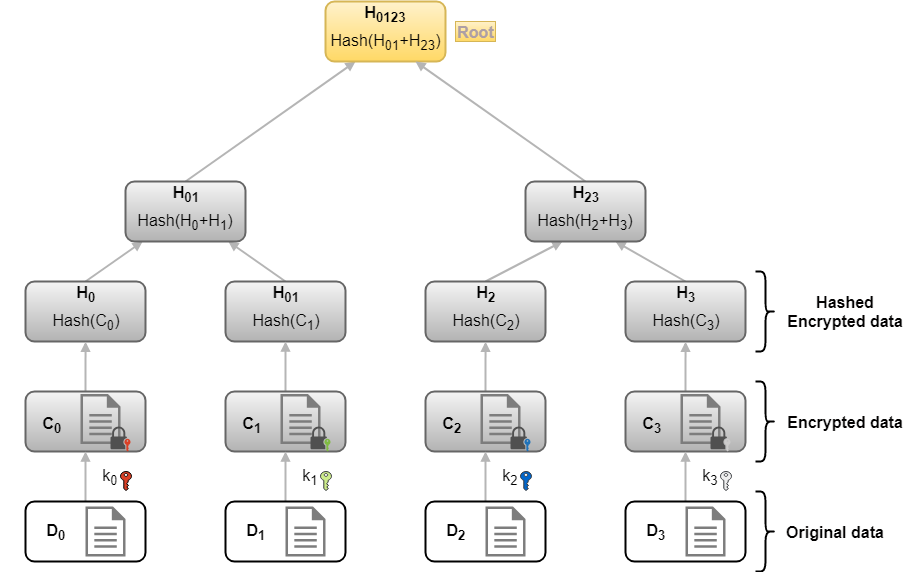


Figure 13. merkle tree of encrypted data. In this example, MRC=H0123

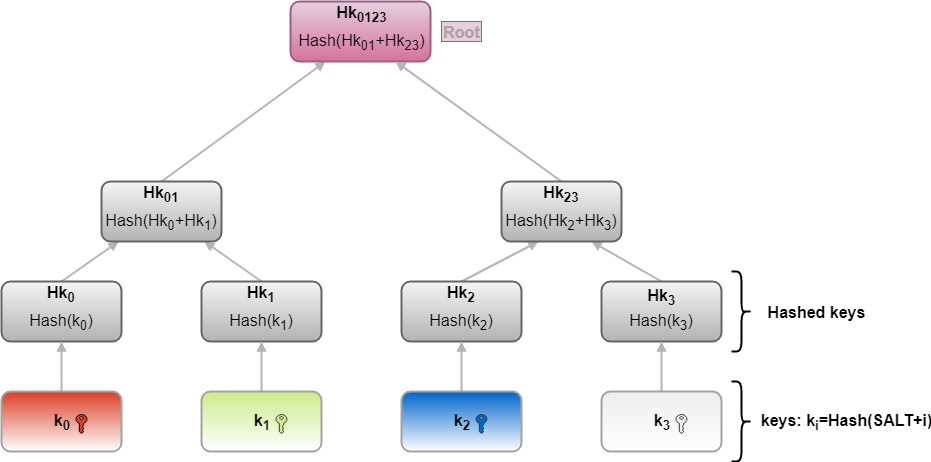


Figure 14. merkle tree of keys. In this example figure MRK = Hk0123

Now the provider must deploy a public SC showing the specifications of the data offered, including a description of the data offered, the price of the whole data and the total number of samples it consists of.

* (Provider→SC) n: the total number of samples.
* (Provider→SC) p: price of the data willing to sell.
* (Provider→SC) MRC: the root of the encrypted data merkle tree of the encrypted data. In addition to its use in the protocol, it will act as a check that bulk data was received correctly.
* (Provider→SC) MRK: the root of the keys merkle tree of the keys that encrypts each sample.

#### Step 2. Data shipping (provider)

Once a consumer has made contact, the provider sends off-blockchain:

* (Provider→Consumer) C={ci}: all data samples encrypted.
* (Provider→Consumer) v: number of data samples to be revealed.

These items do not provide any relevant information about the data to the consumer, other than already known aspects such as size. The off-blockchain channel used has to be encrypted to avoid a potential attack explained in Section 3.4.

#### Step 3. Selection of the samples to be revealed (consumer)

Once the consumer has checked that he has received the information correctly, he sends to the provider:

(Consumer→Provider) R: indexes of the samples the consumer wants to check. Notice that they will be random samples because the consumer does not have any information about the data.

#### Step 4. Sample keys revelation (provider)

The provider sends to the consumer:

* (Provider→Consumer) : the keys that decipher the samples that B has chosen.
* (Provider→Consumer) : the merkle proofs[[1]](#footnote-1) that ensure these keys are part of the original key tree, and so have not been altered.

#### Step 5. Payment (consumer)

Now, the consumer has all the information needed to make a decision on whether to buy the data or not. Protocol-wise, all he needs is to make sure the keys revealed correspond to the original key tree by using the merkle proofs. Not doing so would make him vulnerable as the decrypted samples may not correspond to the data set. As to whether the information in the whole data is worth the price, the question goes beyond the scope of this paper.

* (Consumer→SC) payment.
  + SC triggers Timeout 1: if the provider does nothing, the consumer is reimbursed and the protocol ends; otherwise the providers gets paid once the timeout triggers.

#### Step 6. Salt release (provider)

Once the consumer has made the payment, and thus has shown interest to buy the data:

* (Provider→SC) cost of executing the conflict resolution in the DLT native crypto currency. In order to guarantee a fair potential conflict resolution in the future, the provider locks some money in the SC that could be used to execute the conflict resolution if the provider cheats.
* (Provider→SC) *s*: the random seed from which the keys for each index have been created.

Now the consumer has access to the key generator and can decrypt the data, it will just check that the sample keys, provided in step 4 for the indexes chosen in step 3, have been truly generated using this random seed, as explained in step 1.

If the result is positive, the provider is guaranteed not to have cheated. In this case, the consumer should not do any further action and the payment to the provider will happen once the timeout triggers.

If it is not possible to obtain all the sample keys with the salt, there exists no guarantee that the samples come from the full data. In this case, the provider should challenge the consumer to prove its honesty and two optional steps are required for conflict resolution. To put time limits to those actions, the:

* SC triggers Timeout 2: if the consumer does nothing, the provider gets the payment and the protocol ends.

#### [optional step] Conflict resolution: specific key challenge (consumer)

Once the consumer finds a key that could not be used to decrypt an encrypted data sample (either because of an incorrect key or incorrect cryptogram), it provides its index to the SC and the affected encrypted data sample. This step obviously has a cost that must be paid initially by the consumer to the SC.

* (Consumer→SC) i: failing key index.
* (Consumer→SC) ci: the affected encrypted data sample.
* (Consumer→SC) MPC(i): the merkle proof of the affected cryptogram (can be checked against the already published MRC).
* The SC triggers Timeout 3: if the provider does nothing, the consumer is reimbursed, the cost of executing the SC is paid with the money previously locked by the provider, and the protocol ends.

#### [optional step] Conflict resolution: Challenge resolution (provider)

First, the provider must prove that the key was generated correctly and that it was part of the original key set by providing the correspondent merkle proofs:

* (Provider→SC) MPK(*i*)

The SC will generate the key using the salt *s* and the index *i* and will check it against the received merkle proof MPK(*i*) and the previously stored merkle root MRK (received in step 2). If the result is positive, the provider is guaranteed not to have cheated with the sample keys.

However, there is still a risk that the consumer had cheated with the encrypted data sample (bad or fake encryption). Therefore, the SC will check that the affected *ci* is part of the encrypted data samples *C* originally released by the provider by checking *ci* against the received MPC(i) and the previously stored MRC (received in step 2). If *ci* is part of the original set and it cannot be decrypted with the generated key, the provider cheated with it.

If the provider cheated, all the payments done by the consumer are reimbursed, the conflict resolution cost (for executing the SC) is paid with the money that the provider locked in step 6, and the SC is finalized and killed. If the consumer cheated, all the costs of executing the conflict resolutions will be paid by the consumer.

## Potential attacks

### Eavesdropping - in Step 2 - by an external party

A potential attacker eavesdropping the channel between consumer and provider could get a copy of the cryptogram. Thus, if the protocol finishes correctly, the s will be revealed and he could get access to the data for free. To prevent it, the channel must be encrypted as stated earlier. For the same reason, if during a protocol run the consumer receives the cryptogram and then cancels, this cryptogram and the associated SC cannot be reused another time.

### Data Replication - in Step 1 - by the provider

When the value of the data in a dataset is not linear, as is the case in an email database, where duplicated samples are worth nothing, an attack is possible: the provider could make many replicas of the data and merge them. If the amount of different samples before replication is big enough, the samples revealed to the consumer will all be provably different. Thus, the consumer will not get a fair idea of the database content. To avoid this problem, the consumer should get confident that the data are sorted using the most meaningful variables; e.g. by asking for some sets of consecutive samples, which may help to estimate the data replication ratio.

### Offer Replication - in Steps 1-3 - by the provider

As the consumer performs the data evaluation with a reduced amount of random samples, there exists some variability on its value perception. Consequently, a malicious provider could try repeating the selling process multiple times until the samples chosen are good enough for the consumer to accept the trade. The variability of the value perception in the sample sets depends exclusively on the amount of samples. Therefore, the consumer must ensure there are enough samples provided so that it is not profitable for the provider to repeat the process multiple times, given the fixed costs of the SC creation. Another possible solution would be for the provider to block/burn some crypto value (e.g. Ethers) as a guarantee that the attack is not profitable for him, regardless of the sample amount.

### Request Replication - in Steps 1-5 - by the consumer

In a similar way as in the previous attack, a consumer could engage in the protocol multiple times without any intention of buying the whole data, and accumulate free samples during the process. For this to be non-viable, the provider should ensure the value of a revealed sample set is small enough compared to the fixed costs of the consumer. Like in the previous case, burning/blocking some Ether could be an alternative solution.

## Properties achieved

### Simultaneous probabilistic verification

It is important to note that the probabilistic verification is not only accomplished for the quality of the data but also for the authenticity of the keys.

If the keys sent to the consumer in step 4 are not generated with the provided salt, the consumer will realize comparing the committed keys with the keys created using the salt.

On the other hand, notice that if there exist a committed key that (i) is not included as part of the verification set KR and (ii) that is not generated by the salt, then, when the consumer decrypts the corresponding cryptogram, it will get a useless sample. In this case, the consumer will not be able to distinguish between a situation in which the provider has encrypted the data with an invalid key (not generated by the salt) or has included a random (i.e. non valuable) as sample. This situation can be fixed enforcing the samples to follow some type of format.

### Efficiency

We consider our protocol fairly efficient because of the following reasons: The size of the sample (v) provided to show the whole data value need not be proportional to the size of this whole data (n). This permits to work with big datasets or real-time big-data services with reduced costs. Furthermore, the interaction with the distributed ledger, which is the most expensive part, is fixed and relatively small.

* Amount of data transferred between consumer and provider: O(N)
* Amount of operations in the consumer and the provider: O(N)
* Amount of data stored in the distributed ledger: O(1)
* Amount of operations in the distributed ledger: O(1)

### Fairness

We consider the samples revealed have no inherent value, which should be close to true. The amount of data revealed this way is not proportional to the size of the whole data, it should just depend on its nature. Thus, if the whole data is big enough, the value of the revealed samples becomes negligible.

**From the provider perspective**, the only way he could be cheated is if the consumer has access to the data and he still does not receive the payment. This cannot happen as the provider will only reveal the salt s once the payment is in the SC, and the SC will award him the money if the consumer does not complain or, if doing so, he can still prove having generated the keys correctly. As previously stated, the SC is guaranteed to work as expected with very high probability. As the keys are hashes originating from a pseudo-random number, there is no viable way for the consumer to obtain them, other than paying.

**From the consumer perspective**, as he has checked the sample keys are contained in the keys merkle tree, once the salt s is revealed and he checks the sample keys can be generated from it, all the keys are provably guaranteed to have been generated the same way. The consumer will then be able to decipher the whole data in the same way it has deciphered the samples, and those will necessarily be part of the whole. If some key cannot be generated from the salt, although being part of the merkle tree, and consequently he will probably not be able to decipher part of the database, he can complain to the SC. If he does so, due to the properties of merkle trees and hash functions, the provider will be unable to invent a merkle proof that tricks the SC to think that the true key obtained with the salt was included in the original merkle tree. Again, the SC is provably guaranteed to work as expected. The consumer has chosen the revealed samples indexes while already having the cryptogram so, whatever information it decrypts before payment, has to probably be like the rest of the information.

### Liveness

After the SC is created, the provider can cancel it if no consumer comes. After the payment, the timeouts ensure the counterparty can finalize the execution favourably to her if the party expected to act does not do so in time. This way, being the economic incentives right, the protocol is guaranteed to enter some of the final states.

### Non repudiation / transferable proofs for external parties

The protocol is essentially a non-repudiation of receipt scheme: once the consumer shows interest (makes the payment), if the provider receives the money (no challenge posed or challenge completed successfully), the consumer cannot deny having access to the data. This takes into account the impossibility to deny access to the Ethereum network. However, the records kept in the blockchain can also be used for non-repudiation of origin. Both parties can prove to an external party (law enforcement, for example) they have bought from/sold to the data in question: they just need C, s, the public logs of the protocol run and a proof they own the address used. As it is not viable to find a collision of the hash function, MRC and MRK logged in the SC creation will prevent other data to be faked as bought/sold this way.

## Conclusions and future work

We have proposed a protocol for secure data exchange that makes use of SCs in a DLT. In our protocol the consumer can check a subset of the data before the actual payment and both the consumer and the provider are protected. The provider will be always paid if the data is released and the consumer will only pay if and only if the data has been properly received and the initial set of sample data addressed the expectations. By the time of writing this deliverable (July 2018), we are working on an Ethereum implementation of this protocol that is expected to be released on October 2018 and published as a journal article a few months later.

A major enhancement to the protocol would be to redesign it in a way that the SC and the cryptogram could be reused. The first two steps of the protocol would only be run once and the sample selection would be the same for all potential consumers, bringing in many advantages. First, the costs would be reduced as only a SC would be used independently of the number of failed sale attempts. Second, the use of the same random sample would make essentially disappear the first three presented potential attacks. Third, offer replication and request replication would not be possible by design. Fourth, samples could be bigger as its value is only given away once to everybody. And sixth, eavesdropping would not be possible either as many people having copies of the cryptogram should not be dangerous. Proxy re-encryption technology is something that we have to definitely explore in order to be successful implementing this major enhancement.

If this redesign were not possible, some incentive system should be designed to avoid the attacks, as commented in their section. Fixed costs for buyer and seller actions could prevent offer/request replication. Also permissioning and other forms of censorship could be used, although they would defeat the main purpose of the protocol.

A relatively easy improvement would be to use an extended version of the merkle trees that guarantees not only that some information is included in the tree, but also where it is placed in the tree. By doing this, the challenge part could be reduced from two steps to one. In place of the buyer challenging the seller to prove a correct key is in the tree, the seller would directly prove to the SC that a wrong key is in some specific position in the tree. It would be simpler, with less costs, and the buyer could do any complains wanted as he would be paying them entirely.

Another future work is to design meaningful sizes of data samples for different types of data.

# The BIGIoT privacy risk rating methodology

Among the different security and privacy requirements that have been identified to ignite a secure and reliable IoT ecosystem [29], **privacy by design** is key. Indeed, it has been literally requested (“Data protection by design” - DPD) in the Article 25 of the General Data Protection Regulation [30], which has become enforceable on May 2018.

DPD refers to building privacy features from the very beginning of the design process instead of modifying or adding new features at a later stage. This fact involves the consideration of privacy in the full software development life cycle (SDLC).

Notice that introducing privacy in the SDLC does not necessarily imply added costs. The cost of a data breach differs for every organization. As stated in [31], on average, all organizations in European countries such as France and Germany will incur the cost of at least 3 million euros for a data breach. According to this, investing in DPD should be understood as a factor of economical savings and not the other way round. It can help in saving direct operational costs due to data breaches and indirect cost due to the consequent loss of reputation.

However, the appropriate privacy measures will strongly depend on the identified risks. In the end, security always comes down to making a risk assessment. Identifying the risks is of paramount importance in order to fully understand the security and privacy threats and to support a mitigation strategy. A risk rating methodology (RRM) is therefore key to establish metrics that allow to evaluate the severity of the vulnerabilities. Therefore, every identified issue should have a corresponding score that should allow to properly prioritize how and when it is addressed on the SDLC. An appropriate RRM can also be a valuable tool to achieve compliance with the GDPR in a near future.

There are several proposals for risk analysis in specific IoT ecosystem [32] [33] [34] as well as many works identifying privacy issues, challenges and implications [35] [36] [37]. However, to the best of our knowledge, at 1st quarter of 2018 there were not yet a clear methodology for the analysis and integration of privacy-related risk assessment results into the SDLC, in a sense of properly assigning a priority to development issues/actions derived from the analysis. This is reason why we developed the BIGIoT RRM, that has been already published in the InterOSS workshop [38].

## From the OWASP RRM to a BIGIoT RRM

There are many different approaches to risk analysis [39] [40] [41]. The BIGIoT RRM is mainly based on the OWASP RRM [41] with some modifications to address the specifics of the BIGIoT ecosystem. The OWASP approach, and therefore the BIGIoT one, are based on these standard methodologies.

The OWASP RRM is a general-purpose RRM that covers evaluation of the potential attacker, the exploit evaluation and detection, and finally, the evaluation of the technical and business impact in different domains. While the idea would be to get a global risk rating methodology that could be applied in any case, vulnerabilities that could be critical to one organization may not be for another.

The BIGIoT RRM extends the OWASP RRM to better address the IoT domain and to add more flexibility in terms of setup. Specific to the IoT domain is that IoT platforms collect and process big quantities of data from myriads of sensor units. The collected data is then used by services and applications to provide services to users, sometimes also based on input from users.

By following the BIGIoT RRM, it is possible to estimate the severity of all of these risks to the business and make an informed decision about what to do about those risks. The obtained rating will help to ensure the business doesn't get distracted by minor risks while ignoring more serious risks that are less well understood.

Mainly two factors are taking into account in a standard risk model: likelihood and impact; being the risk usually measured as

## Evaluating the risk

Once a security and/or privacy risk that needs to be rated is identified, the tester needs to gather information about the threat agent involved, the attack that will be used, the vulnerability involved, and the impact of a successful exploit on the business. There may be multiple possible groups of attackers, or even multiple possible business impacts. In general, it is advisable to be cautious by using the worst-case option, which will result in the highest overall risk.

We focus on the risk analysis of privacy and data breaches and their consequences in the IoT scenario. With such a purpose, we consider as baseline the risk factors of the OWASP RRM. Then, we select and tune the factors that are relevant for our scenario and we also propose new key factors that, from our point of view, are lacking in OWASP RRM.

Once we have the list of factors, as detailed below, we score the factors from 1 to 9, where 1 is low impact or likelihood and 9 is high. Unlike with the OWASP approach, we also assign weights to the factors considering the idiosyncrasy of the analysed scenario.

The overall likelihood and impact (both technical and business) are computed as the weighted arithmetic mean of their factors. Therefore, the overall values are calculated by taking the sum of the scores multiplied by their weights and dividing by the sum of the weights. Estimating the associated risk to the business is just as important, since the resources available to fix the vulnerabilities will often depend on it.

As shown in Table 4, the results are then classified into 3 levels; 0 to 3, 3 to 6, and 6 to 9, which are denoted as **low**, **medium** and **high** respectively.

|  |  |
| --- | --- |
| Likelihood and Impact Levels | |
| 0 to <3 | low |
| 3 to <6 | medium |
| 6 to 9 | high |

Table 4. Likelihood and impact levels

The tester should think through the factors that determine both the likelihood and the impact levels, and identify the key "driving" factors that are controlling the result. The tester may discover that their initial impression was wrong by considering aspects of the risk that were not obvious.

Once we have a rate for the likelihood and the impact level, the overall risk indicator is evaluated by combining them as shown in Table 5. In the overall indicator, besides de **low**, **medium** and **high** levels, we have two interesting additional edge cases:

* **critical**: these are risks with high probability and high impact attacks. These risks are the ones that should be addressed by the company in first place.
* **negligible**: these are risks with low probability and low impact attacks. A company may not address them or just ignore them depending on the available resources.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Likelihood** | | |
| **Impact** | low | medium | high |
| high | medium | high | critical |
| medium | low | medium | high |
| low | note | low | medium |

Table 5. Overall risk severity indicator

If it is necessary to defend the ratings or make them repeatable, then it is necessary to go through a more formal process of rating the factors and calculating the result. Remember that there is quite a lot of uncertainty in these estimates and that these factors are intended to help the tester arrive at a sensible result. This process can be supported by automated tools to make the calculation easier: An excel spreadsheet is provided (see ANNEX A. Risk Assessment) to help testers with the analysis of risks with the BIGIoT RRM.

The first step is to select one of the options associated with each factor and enter the associated number in the table. Then simply take the weighted average of the scores to calculate the overall likelihood. An example can be found Table 6 (factors are described in Sections 4.3 and 4.4.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Threat agent factors | | |  | Vulnerability factors | | | |
| Skill level | Motive | Opportunity | Size | Ease of discovery | Ease of exploit | Awareness | Intrusion detection |
| Weight | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Score | 5 | 2 | 7 | 4 | 3 | 6 | 9 | 2 |
|  | Overall likelihood (weighted average)=4.75 (**medium**) | | | | | | | |

Table 6 - Computing overall likelihood

Next, the tester needs to figure out the overall impact. The process is similar here. In many cases the answer will be obvious, but the tester can make an estimate based on the factors, or they can average the scores for each of the factors. Again, less than 3 is low, 3 to less than 6 is medium, and 6 to 9 is high. An example can be found in Table 7.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Technical Impact | |  | Business Impact | | |
| Loss of privacy | Loss of accountability | Financial damage | | Reputation damage | Privacy violation scale |
| Weight | 1 | .5 | .7 | | 1 | 1 |
| Score | 9 | 8 | 1 | | 2 | 4 |
|  | Overall technical impact=8.67 (**high**) | | Overall business impact=2.48 (**low**) | | | |

Table 7 - Computing overall impact.

Once the tester estimates likelihood and impact, as shown in Table 5, it can now combine them to get a final severity rating for this risk. Note that if they have good business impact information, they should use that instead of the technical impact information. However, if they have no information about the business, then technical impact is the next best thing.

In the example above, the likelihood is medium and the technical impact is high, so from a purely technical perspective it appears that the overall severity is high. However, note that the business impact is actually low, so the overall severity could be best described as low as well. This is why understanding the business context of the vulnerabilities you are evaluating is so critical to making good risk decisions.

In the following sections, we broke down the factors that make up the likelihood and the impact.

## Factors for estimating likelihood

Once the tester has identified a potential risk and wants to figure out how serious it is, the first step is to estimate the likelihood. At the highest level, this is a rough measure of how likely this particular vulnerability is to be uncovered and exploited by an attacker. It is not necessary to be over-precise in this estimate. Generally, identifying whether the likelihood is low, medium, or high is sufficient.

There are a number of factors that can help determine the likelihood. The first set of factors are related to the threat agent involved. The goal is to estimate the likelihood of a successful attack from a group of possible attackers. Note that there may be multiple threat agents that can exploit a particular vulnerability; therefore, it is usually best to use the worst-case scenario. For example, an insider may be a much more likely attacker than an anonymous outsider, but it depends on a number of factors.

Note that each factor has a set of options, and each option has a likelihood rating from 0 to 9 associated with it. A weighted sum of these numbers will be used later to estimate the overall likelihood. The weights depend on how relevant the various factors are in the IoT scenario, and which factors are more important than others. As a starting point all factors are assigned the weight of 1 and then reduced according to importance and relevance, and factors with 0 weight are dropped.

### Threat agent factors

The first set of factors are related to the threat agent involved. The goal here is to estimate the likelihood of a successful attack by this group of threat agents. Use the worst-case threat agent.

#### Skill level (weight: 1)

This factor describes the required skill level of the attacker to perform the attack, which is relevant in determining how realistic the exploit is. Note the description is changed from the OWASP description.

For reference: security penetration skills (1), network and programming skills (4), advanced computer user (5), some technical skills (7), no technical skills (9)

#### Motive (weight: 0.5)

This factor describes how motivated the attacker is to perform the exploit. The importance of this factor in BIGIoT is currently reduced, since the focus of the risk analysis is to evaluate the severity of data breach, and to a lesser extent the payoff of the attacker.

Low or no reward (1), possible reward (5), high reward (9).

Estimating the potential reward is so difficult that the weight of this factor is usually reduced. As a reference, in BIGIoT a default value of 0.5 is assumed.

#### Opportunity (weight: 1)

This factor describes the opportunity or resources needed to perform the exploit. Much like the skill level, this factor also helps in determining how realistic the exploit is.

Full access or expensive resources required (1); special access or resources required (5); can be done by readily available off-the-shelf equipment, without any constraints on physical presence or special software (9).

#### Size (weight: 1)

This factor describes the scope of the attack, i.e. in terms of how large the group of potential victims is. Note the description is changed from the OWASP description.

Just one (1), tens of individuals (5), thousands (9).

### Vulnerability factors

The next set of factors are related to the vulnerability involved, if the vulnerability is already known. The goal here is to estimate the likelihood of the particular vulnerability involved being discovered and exploited. If a vulnerability is not known (although a risk has been identified), the following factors, with the exception of Intrusion Detection, should be weighted to 0.

#### Ease of discovery (weight: 0.25)

This factor describes how easy it is for attackers to discover the exploit.

Due to the speed at which knowledge of exploits spread on the Internet, the importance of this factor is reduced, as it is less important if it is easier or harder to discover.

Practically impossible (1); difficult but possible (5); automated tools available (9).

#### Ease of exploit (weight: 0.25)

This factor describes how easy it is to perform the exploit and to obtain access to functionalities or data, but also in terms of how much effort must be spent to be able to take advantage of the functionalities or data.

The importance of this factor is reduced because it should be assumed that if an exploit exists, then automated solutions will appear over time for exploiting it. For this reason, the factor has less impact.

Theoretical (1), difficult but possible (5), automated tools available (9).

#### Awareness (weight: 0.25)

This factor describes how aware potential attackers are of this exploit.

Its importance is reduced due to the speed with which exploits and tools utilizing them are propagated. This means that if only a few attackers are aware of the exploit today, this will change rapidly.

Unknown (1), hidden (5), public knowledge (9)

#### Intrusion detection (weight: 1)

This factor describes the capability of detecting if the exploit is utilized, i.e. in terms of logging access.

Active detection in application (1), logged and reviewed (5), not logged (9).

## Factors for estimating impact

When considering the impact of a successful attack, it is important to realize that there are two kinds of impacts. The first is the “technical impact” on the application, the data it uses, and the functions it provides. The other is the “business impact” on the business and company operating the application.

Ultimately, the business impact is more important. However, you may not have access to all the information required to figure out the business consequences of a successful exploit. In this case, providing as much detail about the technical risk will enable the appropriate business representative to make a decision about the business risk.

Again, each factor has a set of options, and each option has an impact rating from 0 to 9 associated with it. We will use these numbers later to estimate the overall impact.

### Technical impact factors

The goal of the technical impact factors is to estimate the magnitude of the impact on the system if the vulnerabilities associated to a given risk were to be exploited.

#### Loss of privacy (weight: 1)

This factor describes how accurately the functionalities or data exposed via the exploit allow an attacker to track users or their actions. Note this is a new factor not defined in OWASP.

Minimal non-sensitive data disclosed (1), minimal critical data disclosed to extensive non-sensitive data disclosed (5), personal information that is directly linked to an individual (9).

#### Loss of accountability (weight: 0.5)

This factor describes to what extent the actions of an attacker are traceable.

Its importance is reduced because it is of less importance to be able to identify the individual attacker.

Fully traceable (1); possibly traceable (5); completely anonymous (9).

### Business Impact Factors

The business impact stems from the technical impact, but requires a deep understanding of what is important to the company running the application. In general, you should be aiming to support your risks with business impact, particularly if your audience is executive level. The business risk is what justifies investment in fixing security problems.

The factors below are common areas for many businesses, but this area is even more unique to a company than the factors related to threat agent, vulnerability, and technical impact.

#### Financial damage (weight: 0.5)

This factor describes the financial impact of the exploit, i.e. in terms of how much the business or business model is financially affected.

The importance of this factor is reduced because the financial impact will vary for different cases, and for this reason should not be a determining factor of the risk evaluation. Furthermore, this methodology focuses on a general IoT ecosystem.

Less than the cost to fix the vulnerability (1), some effect on annual profit (5), bankruptcy (9).

#### Reputation damage (weight: 0.5)

This factor describes the damage to the reputation of the business based on the exploit.

The importance of this factor is reduced because influence on reputation will have different impact on different businesses, and again this methodology focuses on a general IoT ecosystem. For this reason, this factor should not be a determining factor of the risk evaluation.

Minimal damage (1); loss of goodwill (5); brand damage (9).

#### Privacy violation scale (weight: 1)

This factor describes the scale of devices and/or users affected in an exploit, i.e. how much personally identifiable information have been released about entities or individuals.

Minimal scale (1); few hundreds (5); millions (9).

# Privacy analysis of the BIGIoT pilots

The BIGIoT use cases serve the purpose of demonstrating how the developed technological solution can achieve the BIGIoT overall goal of establishing syntactic and semantic interoperability for smart object platforms. They are aimed at showing how a service or application provider can offer their assets on top of different smart object platforms through the achieved interoperability. With such a purpose, three large-scale regional pilots (Northern Germany – NG, Barcelona – BCN and Piedmont – PIE). The different use cases addressed in the three pilots have been classified into eight use case clusters (see Table 8) that are detailed in Deliverable 2.2.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Cluster** | **Pilot** | | |
|  | **NG** | **BCN** | **PIE** |
| Smart Parking |  |  |  |
| Smart Traffic Management |  |  |  |
| Public Transport Optimization |  |  |  |
| Healthy Bike Navigation |  |  |  |
| Smart Bike Sharing |  |  |  |
| Incentive-based Green Route Planning |  |  |  |
| Multi-modal Route Optimizer |  |  |  |
| Smart Charging |  |  |  |
| *Blue highlighting indicates relevant use case clusters for the specific pilots.* | | | | |

Table 8 - Use case clusters.

It is important to note that from the end user point of view, in a typical use case, all the functionalities are accessed with a single BIGIoT application; however, behind the scenes this app is consuming data from several services, which in turn consume data provided by different platforms.

Services are an important abstraction layer in BIGIoT because they allow code re-utilization and simplify the process of building BIGIoT applications. For example, a service can aggregate data acquired from different platforms and then, present a unified dataset to apps. Another service could manage the history of data, allowing the user to access to past data (data not currently available in the source platform). Another service could create forecasts from data (acquiring data from a platform or from another service). Finally, another interesting use could be a service that anonymises an underlying dataset. This could allow lower levels of ASVS security in the upper layer (app).

With BIGIoT use case clusters shared between the three pilots, we conduct a security and privacy analysis of the different platforms and services involved regardless of the specific pilot.

The risk assessment has been conducted following the methodology in Section 4. After some tuning and discussion, the array of weights agreed for the risk assessment of the pilots is detailed in Table 9. Some of the factors have weights less than one because they have been considered to be less relevant for the BIGIoT pilot scenarios.

|  |  |  |
| --- | --- | --- |
|  | Factor | Weight |
| Likelihood | Skill level | 1 |
| Motive | 0.5 |
| Opportunity | 1 |
| Size | 1 |
| Ease of discovery | 0.25 |
| Ease of exploit | 0.25 |
| Awareness | 0.25 |
| Intrusion detection | 1 |
| Technical  impact | Loss of privacy | 1 |
| Loss of accountability | 0.5 |
| Business  impact | Financial damage | 0.5 |
| Reputation damage | 0.5 |
| Privacy violation scale | 1 |

Table 9 - Array of weights for the BIGIoT risk assessment.

The security and privacy analysis has been an on-going process and new services and applications have been appearing and/or are modified. Consequently, to date only a subset of the services/platforms has been addressed. For every analysed service or platform an overview of such an analysis has been generated. As the entire process of the analysis was already explained in D3.3b, for the sake of clarity and shortness, the details about the performed risk assessment can only be found in “ANNEX A. Risk Assessment” and the Deliverable 3.3b. In the version c of this deliverable (D3.3c), we just provide the new analysis of privacy KPIs, which were initiated as a joint work with Task 5.4 and Deliverable 5.4b.

## Key Performance Indicators (KPI)

During the development lifecycle of the BIGIoT pilots, the following KPIs have been taken into account regarding the privacy of the data, services, applications and platforms. We have evaluated the KPI in three iterations of the analysis:

* 1st iteration: updated in month 20 (July 2017)
* 2nd iteration: updated in month 29 (May 2018)
* 3rd iteration: updated in month 31 (July 2018)

### Weighted Privacy Risk Trend (WPRT)

A Weighted Privacy Risk Trend (WPRT) is a business-based representation of privacy risk from vetted BIGIoT platform, service and application security vulnerabilities over a specified time-period, or repeated iterations of application development.

The state of the art of this KPI is often defined with a formula similar to (1)

where *t* is the time/date when the risk was evaluated, is the type of risk, is the weight assigned to the type of risk *i*, and the number of detected vulnerabilities (often also called *defects* on the literature) with risk *i*. The is a factor that accounts for the business impact (the higher, the worse) that would also be necessary to compute.

In BIGIoT, however, we use a slight variation of this indicator. The BIGIoT risk rating methodology (RRM), presented in deliverable D3.3b, allows to get a more granular weight, also accounting for the business impact, for every specific vulnerability and not just for the categories critical, high, medium and low. The result of the BIGIoT RRM when applied to a given risk is two scores indeed: a technical risk value, *tecScore* in (2), and a business risk value, *busScore* in (3). Consequently, the way of measuring WPRT in BIGIoT is expressed as in (4), where and are the weights assigned to the technical score and the business score, respectively.

After a discussion, for the application of this WPRT to the BIGIoT pilots, it has been decided to use and based on the following reasoning:

1) Mainly companies take decision based upon business strategies and not technical ones. Therefore, the WPRT should clearly denote this fact and be the main factor.

2) The business score provided by the BIGIoT RRM does not account for the loss of reputation due to frequent technical vulnerabilities. A company that is prone to have technical vulnerabilities will soon be seen as non-trustable and that will surely affect its business. Addressing technical risks is therefore not only a good practice but also a way to reduce indirect business risks. Therefore, the weight of technical risks cannot be negligible and should be also important.

Moreover, for every vulnerability the decided weights have been: wnote = 0, wlow = 1/7, wmedium = 2/7, whigh = 4/7, wcritical = 8/7. Besides “note” vulnerabilities which are not taken into account, weights are assigned so that they are doubled with every risk category. The decision has been to put special focus on the most critical vulnerabilities: one critical vulnerability detected raises the KPI over 1, which is intuitively a threshold that shouldn’t be passed; the same result could be obtained with 2 high vulnerabilities, 4 medium, 8 low, or a combination of the previous.

Based on the results of the analysis provided in ANNEX A. Risk Assessment, after three iteration of the privacy risk analysis the computed WPRT is:

Figure 15. Weighted Privacy Risk Trend (WPRT)

As expected the WPRT KPI has decreased after the first iteration as a consequence of the application of the BIGIoT RRM and the valuable feedback provided to the pilot developers. No changes were detected between the second and third iteration of the analysis. While the WPRT is now below 1, this team believes that the expected value should get in the future lower than 0.5; that is to say, not a single high or critical vulnerability detected.

### Critical|High|Medium|Low|Weighted Vulnerabilities (CV, HV, MV, LV and WV)

These KPIs measure the evolution of the amount of vulnerabilities detected for a given. If they raise over time, it denotes that the software development is not performing properly (probably not enough testing or premature releases of software). In such a case, actions should be taken to enhance the quality of the development. If actions are appropriate, the KPI will soon lower down.

For a greater detail, this KPI is split into five that can be represented together over time in a same figure:

1. **Critical Vulnerabilities (CV)**: *CV*(Δ*t*) is the number of critical-risk vulnerabilities detected during time interval Δt.
2. **High Vulnerabilities (HV)**: *HV*(Δ*t*) is the number of high-risk vulnerabilities detected during time interval Δt.
3. **Medium Vulnerabilities (MV)**: *MV*(Δ*t*) is the number of medium-risk vulnerabilities detected during time interval Δt.
4. **Low Vulnerabilities (LV)**: *LV*(Δ*t*) is the number of low-risk vulnerabilities detected during time interval Δt.
5. **Weighted Vulnerabilities (WV)**: *WV*(Δ*t*) the sum of vulnerabilities weighted according to their risk: critical, high, medium or low. It is computes as in (3).

For the latter case, within the BIGIoT project the following weights were agreed: , applying the same reasoning as for the WPRT.

Figure 16. Technical and business vulnerabilities

Both in terms of technical and business vulnerabilities the evolution of these KPIs goes on the right track. We can easily note that, after the second iteration, the WV KPI is below one in terms of technical vulnerabilities and well below one regarding the business ones; although, once again, the target should be to be below 0.5 in a near future.

### Defect Remediation Window (DRW)

This KPI measures the length of time from when a vetted application/service/platform security defect (vulnerability) is identified until it is verified closed. It involves tracking security vulnerabilities in development, testing and production environments. As a result, it is also an indicator of the level of cooperation between the development, quality assurance and operation teams.

Since no changes were detected from iteration 2 to iteration 3 in terms of privacy, we have only collected the time needed to fix the vulnerabilities from iteration 1 to iteration 3.

Figure 17. Total and average Defect Remediation Window (DRW) in hours

As it is shown in Figure 17. Total and average Defect Remediation Window (DRW)Figure 17, and more in detail in ANNEX A. Risk Assessment, the DRW is in average approximately 3 days and 19 hours and the total efforts in hours needed to fix the detected vulnerabilities has been 450 hours. Obviously the amount of samples is small (only the BIGIoT pilot services/platforms are analysed) and thus the representativeness of the results. Ideally, with enough samples, the results should show that the higher the risk is, the more responsive the developing team is for fixing the vulnerability. In any case, the obtained results denote a good interaction between the security and privacy team and the BIGIoT pilots’ developers since vulnerabilities has been fixed in a short amount of time.

# Conclusions & Outlook

This deliverable builds up on two earlier versions. D3.3a defined the general security requirements for BIGIoT, studied best practices for privacy in IoT ecosystems [29], and specified a security design for the BIGIoT interface and marketplace. D3.3b described the BIGIoT Risk Rating Methodology (RRM), performed a security analysis of the implementation of the BIGIoT interface and marketplace, as well as a security and privacy analysis of the BIGIoT pilots. This is now the final version of the D3.3 deliverable and concludes the outcomes of Task 3.3 on security and privacy measures for the BIGIoT project.

This D3.3c deliverable contains three main lines of research regarding the potential of the blockchain for the BIGIoT scenario: (1) a testbed implementation of BIGIoT on the Ethereum distributed ledger, (2) a federation of BIGIoT marketplaces backed up and empowered by a distributed ledger, and (3) a fair data-exchange protocol backed by a set of SCs running on a DLT. Further, a new iteration of the privacy analysis is presented and privacy related KPIs are defined and evaluated.

In detail, the five sections above have discussed the following contents:

In Section 1, an extension of the BIGIoT architecture and in particular the Marketplace with the usage of the Ethereum blockchain technology is described. Thereby, Offerings have been stored on the blockchain as smart contracts and providers and consumers interact with these offerings through dedicated nodes that have direct access to the blockchain.

In Section 2, the usage of the blockchain technology is demonstrated to federate IoT data marketplaces. The different marketplaces are setup on a shared storage that enables sharing of offering descriptions. Thereby, important feature is the immutable and auditable contracts that can be shared across marketplace boundaries. A crypto token provides a transparent payment solution for the trading of IoT resources.

In Section 3, we introduce a data exchange protocol that uses the blockchain to allow a consumer to access or buy a subset of a provider’s data offering. The protocol is able to protect both the consumer and the provider from misconduct by the opposing party.

Section 4 provides an update on the privacy risk assessment methodology previously presented in D3.3b. A risk rating methodology (RRM) for the software development lifecycle is presented and has been published in [38].

Section 5 follows the presented RRM methodology and conducts a privacy risk assessment of the BIGIoT services in the pilots.

Beyond BIGIoT, the know-how acquired by the team during the development of this task has allowed the task members to contribute to new project proposals. The acquired know-how on blockchain, as well as on authentication and authorization flows could be contributed to new H2020 proposals.

# References

|  |  |
| --- | --- |
| [1] | Ethereum, “Ethereum white paper: A next-generation smart contract and decentralized application platform,” 2014. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper. [Accessed 7 August 2018]. |
| [2] | N. Szabo, “Formalizing and securing relationships on public networks,” *First Monday,* vol. 2, September 2017. |
| [3] | K. W. Mortensen and H. H. Rasmussen, “Design, Development and Evaluation of BIG IoT Functionalities on the Ethereum Blockchain,” 4. semester, Networks and Distributed Systems, Master thesis, 2018. |
| [4] | IEEE Computer Society, “The 17th IEEE International Symposium on Network Computing and Applications (NCA 2018),” 2018. [Online]. Available: http://www.ieee-nca.org/2018/. |
| [5] | M. Lorenz, K. Cinibulk, W. Schwarzott, A. Broering and A. Zappa, “Deliverable 4.1b: Design of marketplace,” 2017. |
| [6] | Ethereum, “Solidity,” 2018. [Online]. Available: http://solidity.readthedocs.io/en/v0.4.24/. |
| [7] | V. Buterin, “On slow and fast block times,” 13 September 2015. [Online]. Available: https://blog.ethereum.org/2015/09/14/on-slow-and-fast-block-times/. [Accessed 7 August 2018]. |
| [8] | Ethereum, “Geth,” 2018. [Online]. Available: https://github.com/ethereum/go-ethereum/wiki/geth. |
| [9] | A. P. Ozisik, G. Bissias and B. N. Levine, “Estimation of miner hash rates and consensus on blockchains (draft),” *CoRR,* vol. abs/1707.00082, 2017. |
| [10] | R. Bowden, H. P. Keeler, A. E. Krzesinski and P. G. Taylor, “Block arrivals in the bitcoin blockchain,” *CoRR,* vol. abs/1801.07447, 2018. |
| [11] | A. M. Sukhov, Y. N. Kuznetsova, A. K. Pervitsky and A. A. Galtsev, “Generating function for network delay,” *CoRR,* vol. abs/1003.0190, 2010. |
| [12] | C. Natoli and V. Gramoli, “The balance attack or why forkable blockchains are ill-suited for consortium,” in *47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2017. |
| [13] | A. Bröring, S. Schmid, C.-K. Schindhelm, A. Khelil, S. Käbisch, D. Kramer, D. Le Phuoc, J. Mitic, D. Anicic and E. Teniente, “Enabling IoT ecosystems through platform interoperability,” *IEEE Software,* vol. 34, no. 1, pp. 54-61, 2017. |
| [14] | Ethersphere, “SWARM,” [Online]. Available: https://github.com/ethersphere/swarm. [Accessed 6 August 2018]. |
| [15] | BigchainDB GmbH, “BigchainDB,” [Online]. Available: https://www.bigchaindb.com/. [Accessed 6 August 2018]. |
| [16] | IPFS, “IPFS (the InterPlanetary File System),” [Online]. Available: https://ipfs.io/. [Accessed 6 August 2018]. |
| [17] | D. Reed, M. Sporny, D. Longley, C. Allen, R. Grant and M. Sabadello, “Decentralized Identifiers (DIDs),” July 2018. [Online]. Available: https://w3c-ccg.github.io/did-spec/. [Accessed 20 August 2018]. |
| [18] | N. Bitansky, R. Canetti, A. Chiesa and E. Tromer, “From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again,” in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS 12)*, Cambridge, Massachusetts, 2012. |
| [19] | L. Lamport, R. Shostak and M. Pease, “The Byzantine Generals Problem,” *ACM Transactions on Programming Languages and Systems (TOPLAS),* vol. 4, no. 3, pp. 382-401, 1982. |
| [20] | S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf. [Accessed 7 August 2018]. |
| [21] | S. King and S. Nadal, “PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake,” 19 August 2012. [Online]. Available: https://peercoin.net/assets/paper/peercoin-paper.pdf. [Accessed 7 August 2018]. |
| [22] | Ethereum, “What is the "nothing at stake" problem and how can it be fixed?,” [Online]. Available: https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQs#what-is-the-nothing-at-stake-problem-and-how-can-it-be-fixed. [Accessed 7 August 2018]. |
| [23] | M. Castro and B. Liskov, “Practical Byzantine fault tolerance,” in *Third Symposium on Operating Systems Design and Implementation (OSDI 99)*, New Orleans, USA, 1999. |
| [24] | Open Source (initially contributed by Digital Asset and IBM), “Hyperledger Fabric,” The Linux Foundation, [Online]. Available: https://www.hyperledger.org/projects/fabric. [Accessed 6 August 2018]. |
| [25] | Streamr, “Streamr,” [Online]. Available: https://www.streamr.com. [Accessed 6 August 2018]. |
| [26] | L. D. W. Thomas and A. Leiponen, “Big data commercialization,” *IEEE Engineering Management Review,* vol. 44, no. 2, pp. 74-90, 2016. |
| [27] | Z. Feng and M. Iansiti, “Entry into platform-based markets,” *Strategic Management Journal,* vol. 33, no. 1, pp. 88-106, 2012. |
| [28] | R. C. Merkle, “A Digital Signature Based on a Conventional Encryption Function,” in *Advances in Cryptology (CRYPTO '87)*, 1988. |
| [29] | J. Hernández-Serrano, J. L. Muñoz, A. Bröring, Ó. Esparza, L. Mikkelsen, W. Schwarzott, O. León and J. Zibuschka, “On the Road to Secure and Privacy-Preserving IoT Ecosystems,” in *Interoperability and Open-Source Solutions for the Internet of Things: Second International Workshop, InterOSS-IoT 2016, Held in Conjunction with IoT 2016.*, Stuttgart (Germany), 2016. |
| [30] | European Parliament, Council of the European Union, “Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC,” April 2016. [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:32016R0679. [Accessed 6 August 2018]. |
| [31] | Ponemon Institute and IBM Security, “2017 Cost of a Data Breach Study,” June 2017. [Online]. Available: https://www.ibm.com/security/data-breach. [Accessed 6 August 2018]. |
| [32] | A. Jacobsson, M. Boldt and B. Carlsson, “A risk analysis of a smart home automation system,” *Future Generation Computer Systems,* vol. 56, pp. 719-733, 2016. |
| [33] | R. M. Savola, P. Savolainen, A. Evesti, H. Abie and M. Sihvonen, “Risk-driven security metrics development for an e-health IoT application,” in *IEEE Information Security for South Africa (ISSA), 2015*, 2015. |
| [34] | H. Tai, A. Celesti, M. Fazio, M. Villari and A. Puliafito, “An integrated system for advanced water risk management based on cloud computing and IoT,” in *IEEE 2nd World Symposium on Web Applications and Networking (WSWAN)*, 2015. |
| [35] | A. Arabo, I. Brown and F. El-Moussa, “Privacy in the Age of Mobility and Smart Devices in Smart Homes,” in *International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, 2012. |
| [36] | D. Kozlov, J. Veijalainen and Y. Ali, “Security and Privacy Threats in IoT Architectures,” in *Proceedings of the 7th International Conference on Body Area Networks (BodyNets '12)*, 2012. |
| [37] | R. H. Weber, “Accountability in the Internet of Things,” *Computer Law and Security Review,* vol. 27, no. 2, pp. 133-138, 2011. |
| [38] | J. Hernández-Serrano, J. L. Munoz, O. León, L. Mikkelsen, H. P. Schwefel and A. Bröring, “Privacy risk analysis in the IoT domain,” in *3rd International Workshop on Interoperability & Open Source Solutions for the Internet of Things (InterOSS-IoT 2018) in conjunction with: Global Internet of Things Summit (GIoTS 2018)*, Bilbao, Spain, 2018. |
| [39] | R. S. Ross, “Guide for Conducting Risk Assessments,” 17 September 2012. [Online]. Available: https://www.nist.gov/publications/guide-conducting-risk-assessments. [Accessed 12 April 2018]. |
| [40] | R. Caralli, J. Stevens, L. Young and W. Wilson, “Introducing OCTAVE Allegro: Improving the Information Security Risk Assessment Process,” 2007. |
| [41] | OWASP, “Risk Rating Methodology,” 17 September 2014. [Online]. Available: https://www.owasp.org/index.php/OWASP\_Risk\_Rating\_Methodology. [Accessed 20 July 2018]. |

# ANNEX A. Risk Assessment

This is Annex A of Deliverable 3.3c - Security and Privacy Design for Smart Objects. This Annex is an excel spreadsheet to help testers with the analysis of privacy risks with the BIGIoT risk rating methodology and to evaluate privacy KPIs.

The risk assessment has been conducted following the methodology in Section 3. The array of weights initially set on the template is detailed in Table 9, but obviously could be tuned and recomputation of risks will be automatic. Remember that there is quite a lot of uncertainty in these estimates and that these factors are intended to help the tester arrive at a sensible result.

The template can be downloaded from [http://big-iot.eu/download/d3-3c-annex-a-riskassessment/#](http://big-iot.eu/download/d3-3c-annex-a-riskassessment/).

1. Notice that a merkle proof also includes the merkle root hash. [↑](#footnote-ref-1)