Ref. Ares(2020)2467449 - 10/05/2020

LOCARD

Approved

DELIVERABLE

D3.5 Reference Architecture

Project Acronym:	LOCARD	
Project title:	Lawful evidence collecting and continuity platform development	
Grant Agreement No.	832735	
Website:	http://locard.eu/	
Contact:	info@locard.eu	
Version:	1.0	
Date:	30 April 2020	
Responsible Partner:	ARC	
Contributing Partners:	MOT, NRS, ICO, IMC, EEMA, TID, KEMEA	
Reviewers:	Felicitas Hetzelt Georgios Spathoulas	
Dissemination Level:	Public	X
	Confidential – only consortium members and European Commission Services	
	Classified Information: RESTREINT UE (Commission Decision 2015/444/EC)	

Revision History

Revision	Date	Author	Organization	Description
0.1	01/03/2020	Fran Casino	ARC	Initial draft
0.2	12/04/2020	Fran Casino, Constantinos Patsakis, Nykos Lykousas, Christoforos Ntantogian, Argyris Chrysanthou, Jon Shamah, Steve Kimsley, Vaggelis Pallis, Miltiadis Anastasiadis, Nykos Lyrakis, Fotis Kouretas, Yiannis Perros	ARC, NRS, MOT, ICO, IMC, EEMA	Working document v1
0.3			ARC, NRS, MOT, ICO, IMC, EEMA, KEMEA, TID	Working document v2
0.4			ARC, NRS, MOT, ICO, IMC, EEMA, KEMEA, TID	Working document v3
0.5			ARC, TID, NRS	Revision comments from TID, NRS
0.6			ARC, KEMEA, MOT	Input from KEMEA, MOT
0.7			ARC, NRS, MOT, ICO, IMC, EEMA, KEMEA, TID	Version for review
0.8	30/04/2020		ARC, MOT	Address comments from 1st review iteration
0.9				Address comments from 2nd review iteration
1.0		Final version, Fran Casino, Constantinos Patsakis	ARC	Final version

Every effort has been made to ensure that all statements and information contained herein are accurate, however the LOCARD Project Partners accept no liability for any error or omission in the same.

Table of Contents

Approved

Approved

Approved

List of Tables

List of Figures

Approved

1 Executive Summary

This document provides a detailed technical overview of the LOCARD system's high-level architectural design and functionality, and serves as a blueprint for the design and implementation of all LOCARD components and their corresponding integration in the whole system.

Based on the consolidation of the information and guidelines provided in the LOCARD use cases, end user requirements and system requirements documents (all of them belonging to WP3), system design is outlined and implementation goals are stated. Moreover, the integration of the tools described in the research outcomes of WP4 are also outlined.

High-level architecture is specified, including descriptions of major building blocks and containers of the LOCARD system. The components of the LOCARD architecture are specified in detail, including description of the system software modules relevant technologies for implementation and associated business processes, major interfaces, relationships between the components and guidelines for their deployment. To support generation of work package level software specification documents, implementation responsibilities of the partners are clearly specified. Partner task leadership, requirements and specifications are cross-referenced to enable a comprehensive view of the implementation work assignment and responsibilities.

Approved

2 Introduction

This document provides a detailed architecture of the LOCARD system as well as a consolidated list of technical specifications and workflows for each of the identified modules. These specifications define the project's implementation goals and will serve as baseline for the LOCARD technical activities.

This document presents the required implementation tasks and the schema of inter-module communication, including:

- Description of system software modules
- Business processes
- High-level architecture
- Major interfaces and relationship between components
- Work assignments

This is not a complete software specification of the LOCARD platform. Rather, it consolidates the information provided by the use cases in D3.1 and D3.2, and by the user and system requirement documents D3.3 and D3.4 into a unified system design with clear implementation goals. Moreover, it integrates some implementation guidelines for research modules that will be developed in WP5.

It must be mentioned that the creation of technical software specifications in LOCARD is intended to be a continuous process. This deliverable is the first version of the conceptual architecture (the next will be D3.8) that identifies the functional parts of the platform and specifies the dependencies and interoperability between LOCARD components.

Approved

3 High-Level Architecture

This section presents the high-level architecture of the platform, including descriptions of the conceptual and logical architecture, the high level modules and the system blocks. It also provides the guidelines for deployment.

3.1 Conceptual overview

In a global perspective, different entities with different jurisdictions and procedural mechanisms may join LOCARD. The specific case within a jurisdiction can be seen in the left side of Figure 1. For each entity (for the sake of clarity, we will use "entity" and "end user" interchangeably along the document), we have different users which may have different clearance level and will be allowed to use the LOCARD portal according to local policies and identity management. Moreover, each entity may have one or more blockchain nodes (i.e. to increase the security and the performance of the system on premise) or none at all (e.g. in the case of defendant lawyers, in which the procedures to lend data about the investigation may vary between jurisdictions). Nevertheless, the use of a special multi-tenant node will allow the interaction with the blockchain for these users that have clearance yet cannot host a node themselves. Note that the blockchain nodes can be virtual machines running in premises or in the cloud. The relationships and clearance of each entity will be set in the jurisdictional policies, and therefore may vary.

An **Area Coordinator** comparable entity is assigned to each jurisdiction , which will manage the creation of credentials for specific entities as well as the crowdsource intelligence input cases, distributing such information accordingly between the rest of the entities under its area.

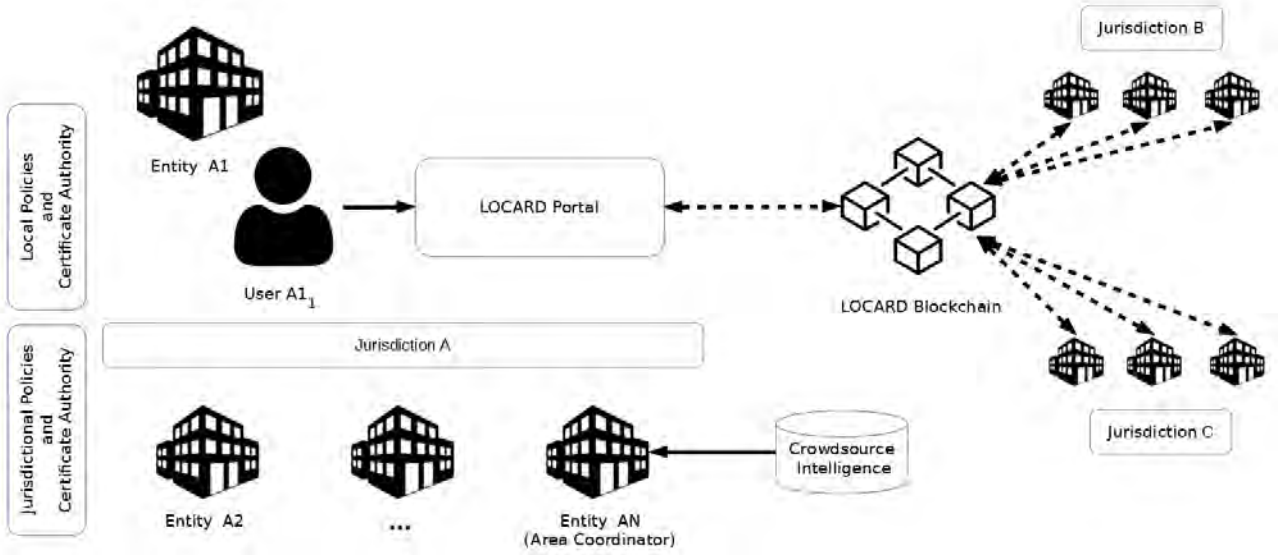

Figure 1: Conceptual overview of the LOCARD system.

3.2 Logical architecture

LOCARD is based on a web multi-tier architecture. Through this architecture, the system hardware and software components are organized into individual modules that can be independently updated or changed (modular system organization). This way, the system is flexible and can adjust to dynamic changes of the operational environment as well as future technological changes of software and hardware.

The proposed system will be deployed in a multi-tier architecture, which includes the following basic tiers:

- **The Web / Presentation / Client Tier** which includes the interaction environment for the end users and constitutes the data presentation layer of system. This specific tier will be deployed using: Java spring boot framework[1], React[2], and NodeJS[3].
- **The Application Tier** which comprises:
    - The business logic of the system which is offered through Java spring framework, NodeJS.
    - The central infrastructure for identity management, authentication, authorization and security of applications and systems and the related infrastructure that is required.
    - Several logically separate containers (some of the applications will be deployed in docker containers[4]) that are implemented in dedicated tasks and subtasks. Each container comprises multiple modules each of which provides a specific functionality. Each module consists of multiple components; each component implements a subset of the module's business logic.
- **The Integration layer** is based on Rabbitmq[5] ESB, which is a lightweight Java-based enterprise service bus (ESB) and integration platform that allows developers to connect applications together quickly and easily, enabling them to exchange data. Rabbitmq ESB enables easy integration of existing systems, regardless of the different technologies that the applications use, including JMS, Web Services, JDBC, HTTP, and more. The key advantage of an ESB is that it allows different applications to communicate with each other by acting as a transit system for carrying data between applications within the enterprise or across the Internet.
- **The Data Tier** is composed of several databases. MariaDB which is characterized by its high efficiency, availability, security and manageability. In this Tier, a Database Server is proposed to be deployed for User Accounts, Policies and LogDB (the latter being used for the crowdsource intelligence module). It relates to storage systems and to information management regardless of whether this relates to transactional data, master data, or data of the offered system databases. The subsystems/modules of the Application Tier will be able to share the common data models and the common data infrastructure. to each corresponding database. Should the testing or implementation phases determine, that processing capabilities of the system are insufficient, other technologies such as mongoDB will be evaluated.

1 https://spring.io/projects/spring-boot

2 https://reactjs.org/

3 https://nodejs.org/en/

4 https://www.docker.com/

5 https://www.rabbitmq.com

Approved

Blockchain storage resides on each one of the nodes. The technology used for that is LevelDB which is a fast key-value storage library that provides an ordered mapping from string keys to string values.

Data Tier also contains a TiDB open source NewSQL distributed database in order to efficiently store case metadata. Therefore, TiDB offers high availability and strong consistency for these data as it uses Raft consensus algorithms and replicates data across nodes. In addition it is compatible to MySQL which offers a coherent interface to perform queries and transactions.

LevelDB will also be used to guarantee the integrity of the evidence which will be stored on a file storage system, which will be accessible through SFTP and an API. Finally, the Intelligence Engine database will temporarily store some correlations discovered by analysing the LOCARD data as well as data in external repositories.

The rest of the architecture is complemented with the following Tiers:

- **Enterprise Security:** It is part of the security infrastructure. This security component will be shared for the entire architecture and will treat all issues regarding user access, automated provision of user rights, data encryption, data security and system security. Moreover, a dedicated module for Trusted Execution will also be part of this tier and is described in Section 4.13.

In order to support user management and authentication subsystem, Oauth2 with MariaDB connectivity will be used. Therefore, the implementation of the user management will be based on Oauth2 to streamline the user login process and to automate administrative tasks such as creating users and assigning them to groups. The proposed product includes, but is not limited to, the following functionalities:

  - Option of storing enhanced user related information
  - Assignment of each user to user groups according to their accessible services
  - Assignment of user groups to proxy access lists

- **Enterprise Management:** The enterprise management allows administrators to supervise the operation of all architecture tiers under the common environment and perform administrative tasks.

The proposed system architecture is based on international standards and is open to many platforms and technologies. Specifically, the standards adhered to are:

- J2EE
- Web Services support
- MariaDB Relational Database system
- Data exchange between different data sources with the use of Web Services (https, rest)
- Support of Java Spring Boot, Node.JS and React
- Improved data access
- Improved security: components that belong to the Application/Business Tier can be secured using a centralized infrastructure. Access to each component can be given or denied separately, facilitating the work of administrators
- Simplified access to external resources

Due to their stability, inherent security and open source nature, we propose CentOS and Ubuntu as the operating system environment of the system infrastructure. Nonetheless, any operating system that can run a Java Virtual Machine (JVM) can be used. The system will be implemented on Java spring boot

Approved

framework technology and will make use of multiple modern widely-used open-source Java frameworks. Figure 2[6] presents the logical architecture of the proposed system.

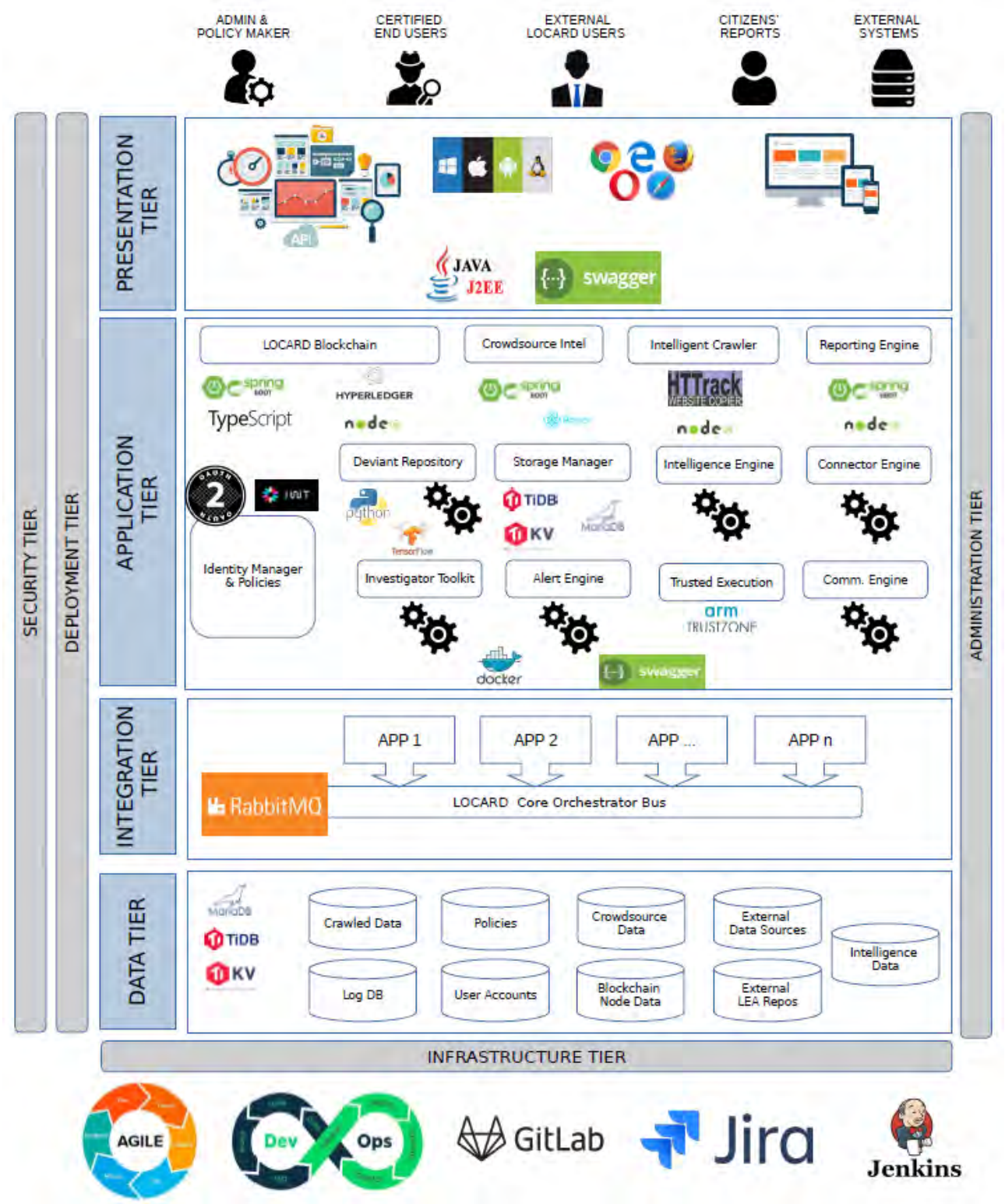

Figure 2: Logical architecture of LOCARD.

6 For a PDF version with more resolution access to https://owncloud.locard.eu/index.php/s/L0VBb5yJxzOHqAN

3.3 LOCARD System blocks

All platform containers and services interact through the LOCARD Core Orchestration Bus, which manages and orchestrates the module interactions and communications. The overall design consideration for the system blocks account for internal and external security risks[7] such as injection flaws[8], denial-of-service[9], insecure communications , information leakage, replay attacks[10] and insufficient authentication. Therefore, APIs, workflows and internal logic will be implemented using high security standards and guidelines and appropriate protection mechanisms. Finally, the use of blockchain and its inherent features[11] provide a further layer of protection and availability, enhancing the robustness of the overall system. As also documented in D3.4, the main system blocks are the LOCARD portal (which contains critical modules such as the ID management, the Storage Manager and the Core Orchestrator Bus), and the LOCARD Blockchain.

3.3.1 LOCARD Portal

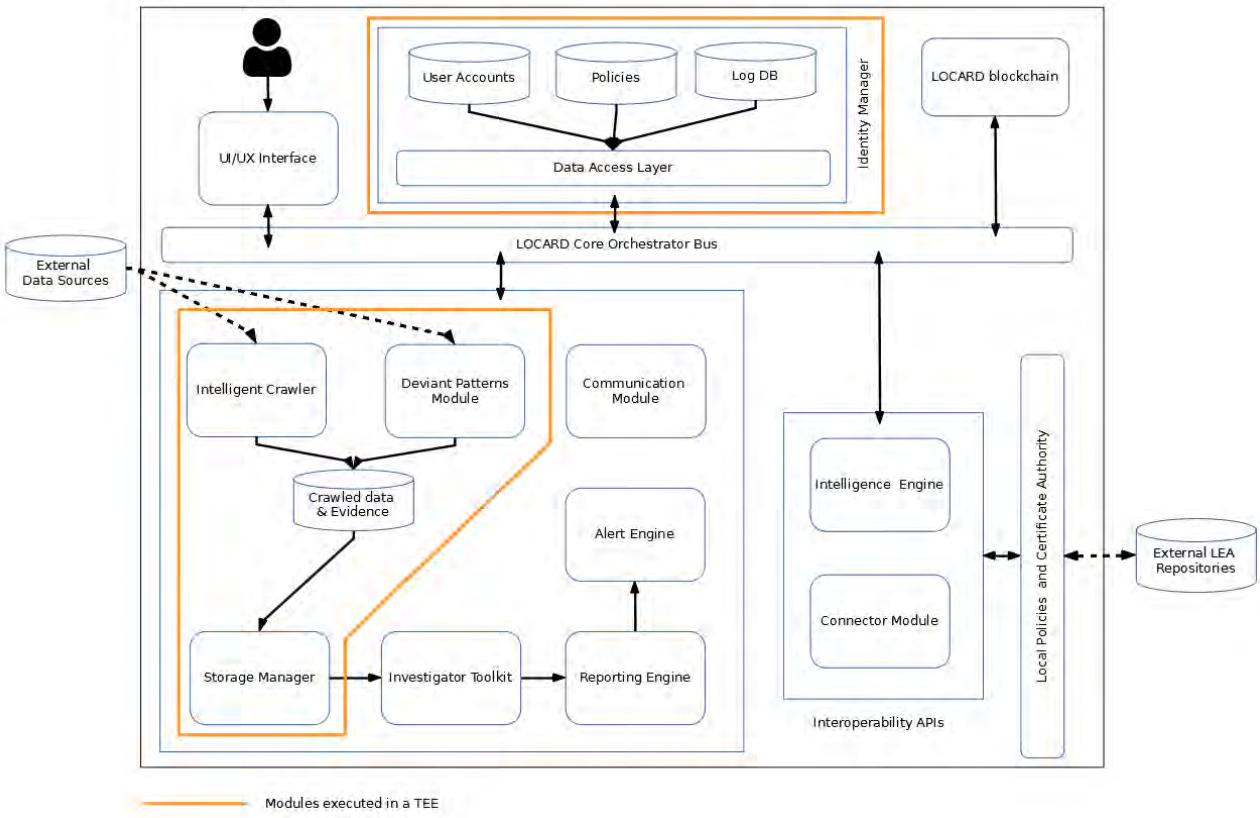

Figure 3: LOCARD Portal components overview.

The LOCARD portal integrates the different modules of the LOCARD system, each with its corresponding internal databases and policies, as depicted in Figure 3. A detailed description of each module will be provided in Section 4. In general, data crawled from external sources is stored in the corresponding protected database (according to the local policies). This operation, as well as the identity management interactions, are executed under a trusted execution environment. Thereafter, the data stored, namely evidence, will be processed and analysed by the investigator toolkit, which contain several forensic tools.

7 https://www.owasp.org/index.php/Category:OWASP_Best_Practices:_Use_of_Web_Application_Firewalls

8 https://www.owasp.org/index.php/Injection_Flaws

9 https://www.owasp.org/index.php/Denial_of_Service

10 https://en.wikipedia.org/wiki/Replay_attack

11 Fran Casino, Thomas K. Dasaklis, Constantinos Patsakis, A systematic literature review of blockchain-based applications: Current status, classification and open issues, Telematics and Informatics, Volume 36, 2019, Pages 55-81

Consequently, a report will be generated in each case, which will be used to notify the investigator(s) involved in the corresponding investigation via the alert engine.

In parallel, the findings reported to the blockchain, when made available and stored in the public channel, will be collected and used by the intelligence engine, to help in ongoing and future investigations. The latter will be done by correlating such data with internal databases as well as external repositories. Note that, in the scope of the project, the use of external repositories will not be enabled, yet the system will have the potential to enable such connection. Moreover, in regard to the testing and validation, **only synthetic data will be used**. More details about this can be found in D5.1. Finally, the interactions of the users with the portal will be logged internally for audit purposes. From these interactions, the relevant ones will be logged into the blockchain to achieve forensic soundness and verifiable chain of custody, in order to be used in a court of law. In addition, the communication engine will be used in the case that end users want to share information about investigations. The aim of this module is to enable the communication and simplify the investigations of distinct partners, yet those will have to exchange data or evidence out of LOCARD platform, if required. Note that data exchange is out of the scope of LOCARD, yet this interaction can be logged into the blockchain accordingly.

3.3.2 Blockchain

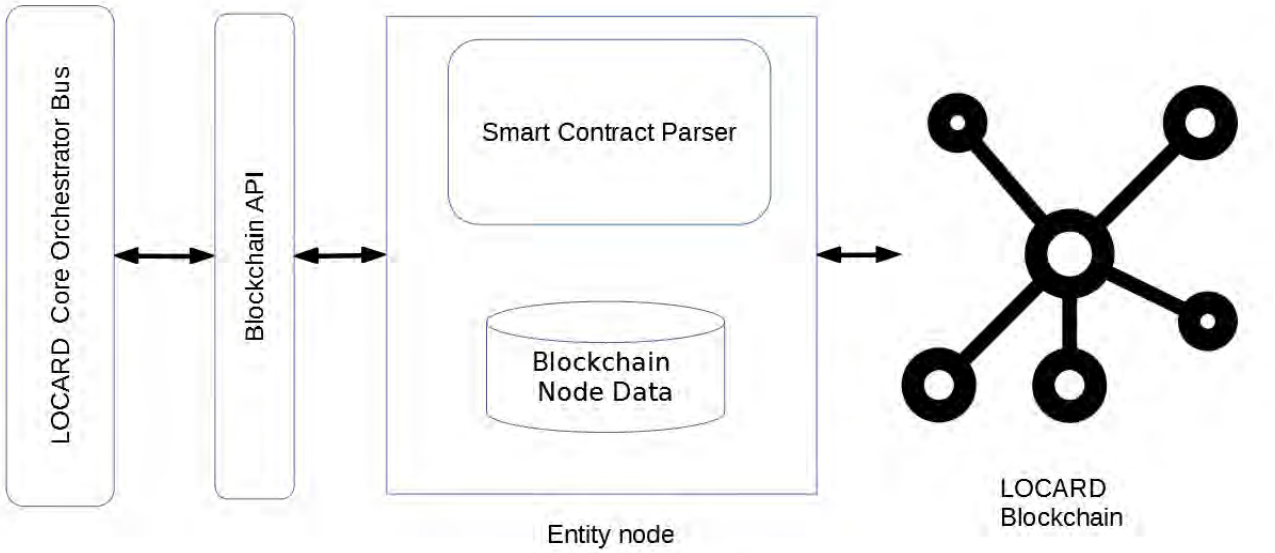

Figure 4: High level modular overview of the interaction between the LOCARD portal and the blockchain node of an entity.

The use of blockchain technologies to secure the integrity and verifiability of data is one of the key aspects of LOCARD. In this regard, a set of forensic tasks will be audited and logged into the blockchain by means of smart contracts (e.g. new evidence acquisition, new analysis performed on data, new report created), constituting a sound audit trail of events. In addition, the immutability of blockchain guarantees that the chain of custody is preserved. Every eligible end user (i.e. as determined by the eligibility policies declared at a jurisdiction level) will run and maintain one or more blockchain nodes, as depicted in Figure 4. Note that smart contracts and their definitions will be designed under each jurisdictional policy, and thus, may vary.

Despite the fact that LOCARD fosters collaboration to enhance cybercrime investigation, the need for privacy as well as different restriction levels hinder the possibility of sharing data with arbitrary LOCARD users. Therefore, end users will share data in a private channel (i.e. a channel is a private data and communication context) only with the users who have the corresponding clearance to view such contents. Nevertheless, if users want to share specific data with all LOCARD end users, this can be performed through

the public channel. This public channel is used to share metadata information (metadata refers to high level summaries or keywords, without disclosing any personal data or investigation details) to enable, for example, further collaboration if different users are investigating the same or similar matters (e.g. a specific type of device or malware). The latter will also enable the intelligence engine to cross-check information between different cases and enable further collaboration through the communication engine and the other standard procedures already available.

3.4 LOCARD System Flow

In order to provide a more comprehensive description of the forensic flow of LOCARD, we provide a step by step example of its functionality, according to the timeline of possible events. The forensic analysis and investigation flows followed in LOCARD correspond to standard practices, such as the ones recommended by Interpol[12]. We assume that the system has been deployed in a specific entity which has permission to host a node of the LOCARD blockchain, and that a specific user has been successfully registered into the system.

The main flow of LOCARD considers the following steps:

1. Case opening (receive request, register case)
2. Data acquisition via crawler or standalone forensic tools (register evidence)
3. Data analysis with forensic tools (analyze evidence)
4. Data cross checking with intelligence (this can be also performed before step 3) (optional)
5. Communication and collaboration (optional)
6. Report creation

A detailed representation of the LOCARD flow is depicted in Figure 5. The interactions with the system will be recorded in the blockchain (represented with dashed lines in Figure 5) to provide a sound forensic chain of custody and event track. Moreover, given a specific investigation, when a new input/interaction is registered, the users related to such investigation will be alerted via email. The specific functionalities of the modules are described in Section 4.

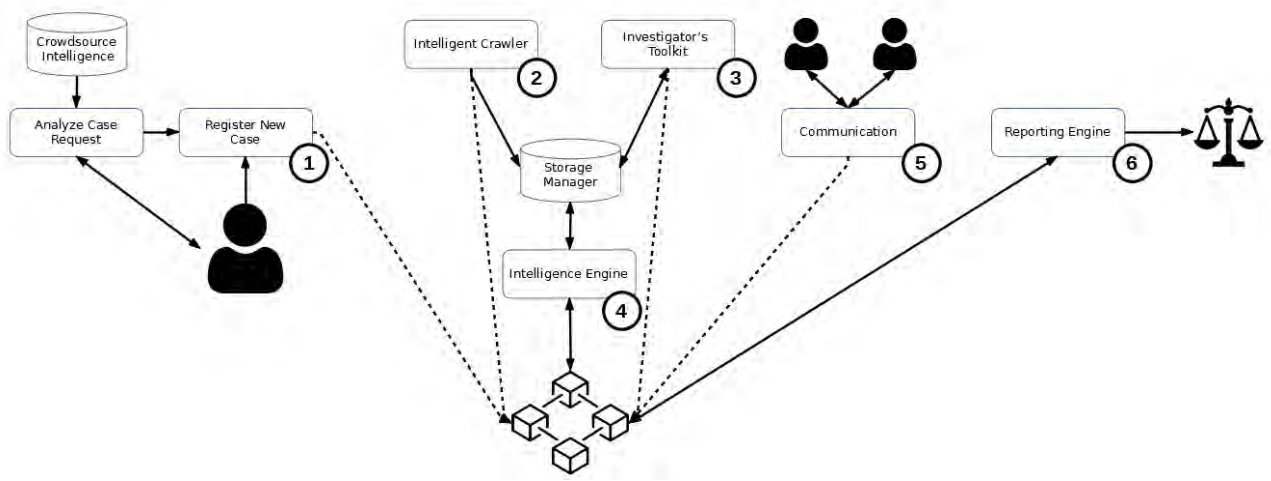

Figure 5: LOCARD forensic flow overview.

12 https://www.interpol.int/content/download/13501/file/INTERPOL_DFL_GlobalGuidelinesDigitalForensicsLaboratory.pdf

3.5 Physical Architecture

The infrastructure required for LOCARD development will be hosted on a secure data center of Motivian which is certified with ISO 27001 and ISO 9001. Regarding the physical architecture and hardware requirements of the proposed system, the following setup and requirements are recommended in order to ensure the high availability and performance of the system:

- 2VMs (under ESX)
- Each VM powered by 2 x Intel Xeon SP Gold 16-Core @ 2.10GHz
- 3.84 TB NVMe SSD Datacenter Edition
- 96 GB DDR4 ECC RAM
- Linux Ubuntu 18.x, Linux CentOS
- VMWare vSphere License.
- VPN connections to all technical partners.

Note that the above specifications comply with the system requirements described in D3.4.

4. Description of functional components

In what follows, we provide a functional and technical description of all the LOCARD modules, as well as their interactions. In the latter case, we omit the ID management and the LOCARD Core Orchestrator Bus, since they are related with all modules (with only few exceptions).

4.1 The Intelligent Crawler

4.1.1 Functional description

The LOCARD crawler provides an interface to facilitate evidence collection from the Internet. From the existing crawler software technologies, we opt to use HTTrack[13]. In the former case, the project will use the httrack web crawler, which is a free open source crawler, for indexing a particular domain. HTTrack allows you to download a World Wide Web site from the Internet to a local directory, recursively crawling all directories, storing HTML, images, and other files from the server to your computer. In other words, HTTrack arranges the original site's relative link-structure locally, so that the "mirrored" website can be accessed by a browser; and can be browsed as if the user was viewing it online. HTTrack can also update an existing mirrored site, and resume interrupted downloads. HTTrack stores its results by default in the local storage but this can be controlled with the help of the -O flag in the HTTrack executable. In this way any location can be used as a storage back-end including network drive or services like SFTP. In addition, HTTrack offers a programming interface which it allows easy integration with the main system. Mainly, we will use its capability to expose its functionality through the use of shell scripts. Therefore, HTTrack can be used as a third-party program in batch files by creating appropriate shell scripts. Amongst others, HTTrack's offers features such as selecting specific files, search for keywords in all mirrored HTML files, and count the number of keywords in a website.

Crowdsource intelligence inputs can be used by the Intelligent Crawler to search for malicious content in specific sites. Such collected evidence will then be stored and processed by the Investigator's Toolkit and/or the Deviant Pattern Repository accordingly. Note that the corresponding interactions will be logged into the blockchain, while alerts will be raised when new evidence is collected.

13https://www.httrack.com/

Approved

4.1.2 Component diagram

The main components of the Intelligent Crawled are depicted in Figure 6.

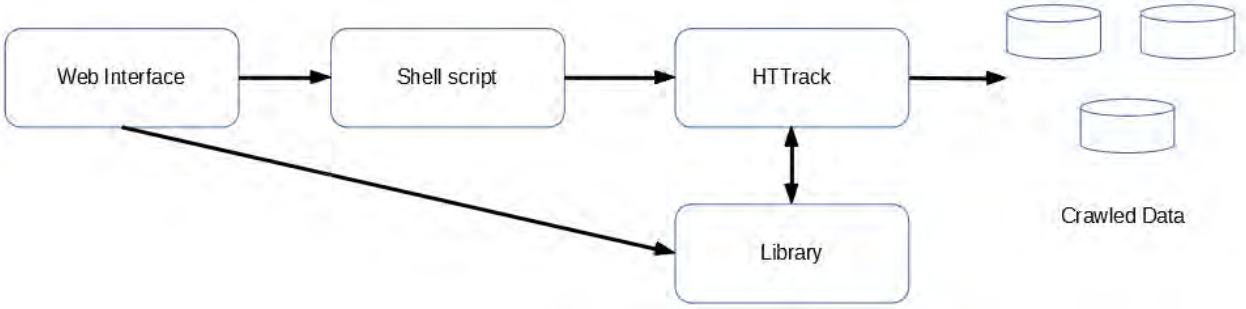

Figure 6: Overview of the main flow of the component.

4.1.3 Interaction between modules and related workflows

The Intelligent Crawler mainly interacts with the following modules:

- Crowdsource Intelligence
- Storage Manager
- Investigator Toolkit
- Alert Engine
- LOCARD Blockchain
- LOCARD UI/UX Interface and Portal
- LOCARD TEE
- LOCARD Deviant Patterns Repository

4.1.4 Design and architecture goals and guidelines

LOCARD's Intelligent Crawler module aims to provide a robust and dynamic evidence collection framework, facilitating and automating as much as possible the work of the investigators.

4.2 The Investigator's toolkit

4.2.1 Functional description

This is a set of offline forensic tools whose output is automatically bridged with the Storage manager. The tools allow investigators to perform their usual data acquisition and analysis procedures. However, in this case the tools' output is stored along with the evidences. Moreover, the credentials of the user are also linked to such forensic output to enable auditing as well as committing each investigator to his findings. These outputs will be files or sets of files and summary reports when eligible. The latter will be processed by the Reporting Engine in order to create the final documentation of a case.

The toolkit will include forensics tools that will be developed/adopted considering the research conducted in LOCARD. Table 1 shows the information collected during the research findings phase in regard to forensic tools suitable for LOCARD, as those are mentioned in Deliverable 4.7. Note that, since our aim is to avoid overheads in the system with tools which perform very similar forensic analysis, the final tools to be

Approved

included will be decided during the development phase. In addition, we will consider the addition of new relevant tools that may appear in the state of the art.

Tools	Topic	Link to description
Mobile Pentest, JIT MF toolkit, ASAINT, SPECK, Pushbullet	Mobile forensics	Section 5.1.3 in D4.7
VM snapshot, Hypershot, VM introspection, Linux Test Project, Forenvisor, kumodd, VMlog auditor, Frost	Cloud forensics	Section 5.2.3 in D4.7
LOCARD Trajectory Sampling and its internal add-ons	Streaming forensics	Section 5.3.3 in D4.7
Ad-hoc implementation based on AI & ML	Social forensics	Section 4.9 in this document

Table 1: Overview of the main tools of the component.

In terms of permissions, the Investigator's toolkit will enable access to a subset of forensic tools as well as the proper connection with the Storage Manager to retrieve only data from the specific cases assigned to a user. In addition, the interactions will be stored in the blockchain and the corresponding alerts will be sent only to the users assigned to a specific case.

The tools will access the API or the SFTP of the storage manager, depending on the location of the files, to collect and store data if needed or to access the actual evidence that is going to be analysed.

Approved

4.2.2 Component diagram

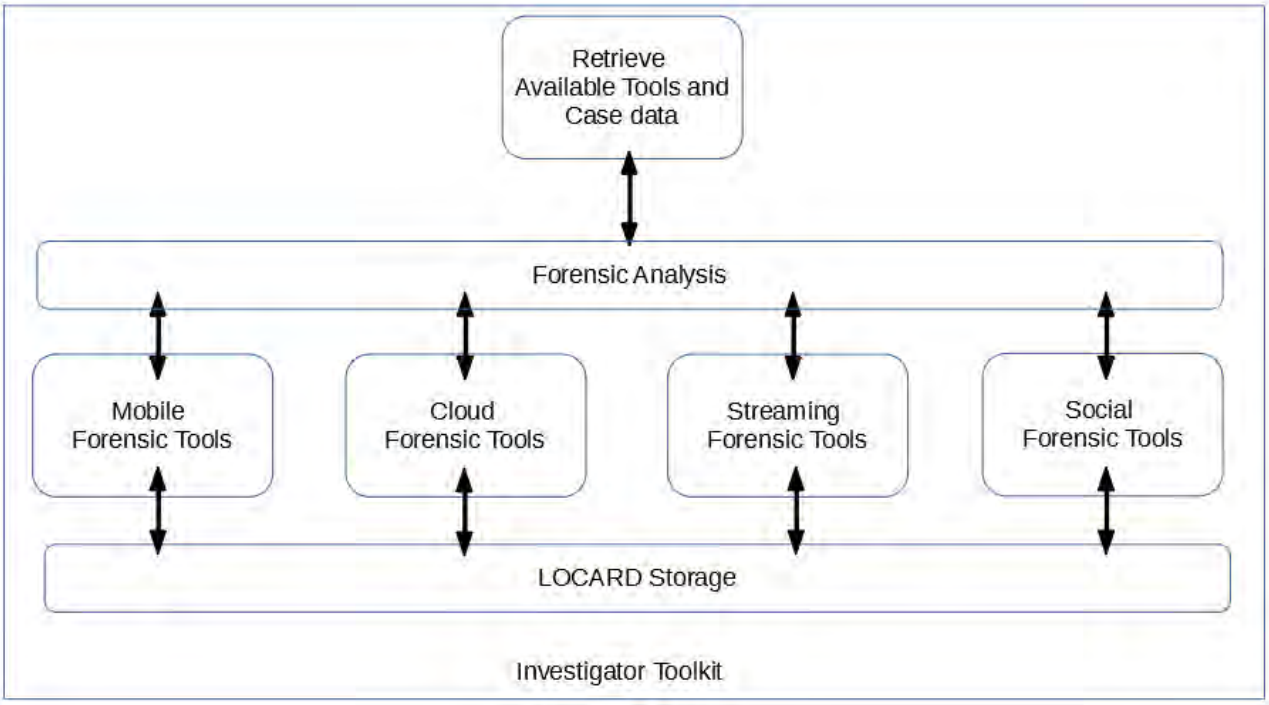

Figure 7: Overview of the main flow of the component.

4.2.3 Interaction between modules and related workflows

The Investigator's Toolkit interacts with the following modules:

- Alert Engine
- Storage Manager
- Intelligent Crawler
- Deviant Patterns Repository
- LOCARD Reporting Engine
- LOCARD Blockchain
- LOCARD UI/UX Interface and Portal
- LOCARD TEE

4.2.4 Design and architecture goals and guidelines

The Investigator's Toolkit is designed to be an adaptable resource of forensic tools to be used by the investigators. During the project, we assume the integration of several tools related with the topics stated in the user case definitions in D3.1 and D3.2, which are also described in more detail in D4.7. This module aims to incorporate more tools to analyse other cybercrime contexts in the future.

Approv

4.3 The Communication Engine

4.3.1 Functional description

As required by many end users, the possibility of communication between LOCARD users is a desirable feature. Due to the ease of use, the intrusiveness balance, the cost, and the fast interaction capabilities, the electronic mail technology will be used for such communication. Therefore, the LOCARD portal will incorporate the possibility to send an email to a specific user.

The main use of this communication is to request further details of a specific evidence, report, or case from other LOCARD users. The Identities of the users will be managed by the ID management module, and the requirement that these are anonymized will be discussed with the end users and assessed during the project. In addition, we assume that these interactions can be logged in the blockchain if desirable, yet this will be discussed during the implementation phase.

During the initialization of the BPM nodes:

- web service end-points to accept encrypted messages are set
- Each node broadcasts its directory to the other BPM nodes, web services and encryption keys, hence routing of messages for each BPM node will take place directly and encrypted from BPM to BPM node.

It is important to note that, although LOCARD will enhance the collaboration between end users, the exchange of evidence between end users that did not collaborate in a specific investigation is a procedure out of scope of LOCARD, and thus, the corresponding external protocols will be used by each end user to execute that task.

4.3.2 Component diagram

Figure 8 shows a graphical representation of the main components of this module.

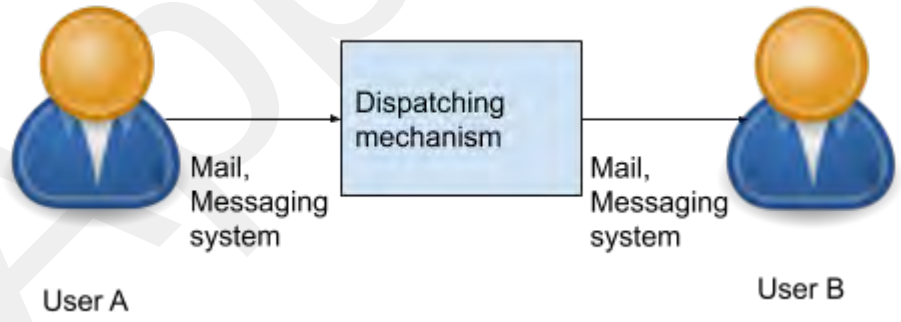

Figure 8: Overview of the main flow of the component.

4.3.3 Interaction between modules and related workflows

- LOCARD UI/UX Interface and Portal
- LOCARD Blockchain

4.3.4 Design and architecture goals and guidelines

The design of an easy-to-use communication system in the LOCARD system will enable seamless collaboration between users, which is one of the main aims of the project.

Approved

4.4 Crowdsource Intelligence

4.4.1 Functional description

This module is mainly focused on collecting intelligence from the general public / citizens. To this end, citizens - after registering - will be able to report content such as abuse over a social network (public feeds) or illegal streaming content (child pornography, IP infringement etc.). To allow general public / citizen reporting, a web form will be designed and accessed through the main LOCARD portal which will incorporate user authentication mechanisms to prevent service abuse, spam and DoS attacks. Once the proper information is received, the information will be forwarded to the LEA Area Coordinator (see Section 3.1) that will manage the information and distribute it accordingly or assign it to other end users. This minimizes the amount of time for reporting an incident and processing it for the corresponding entities. The type of information initially collected by the module will need to be sufficient to be assessed and will need to be consistent with GDPR and other local jurisdictional requirements. The exact details are outside of the scope of the LOCARD project, but in practice will need to be acceptable by all participating jurisdictions, and so may need to be centrally managed.

4.4.2 Component diagram

The functional description is presented graphically on the figure below.

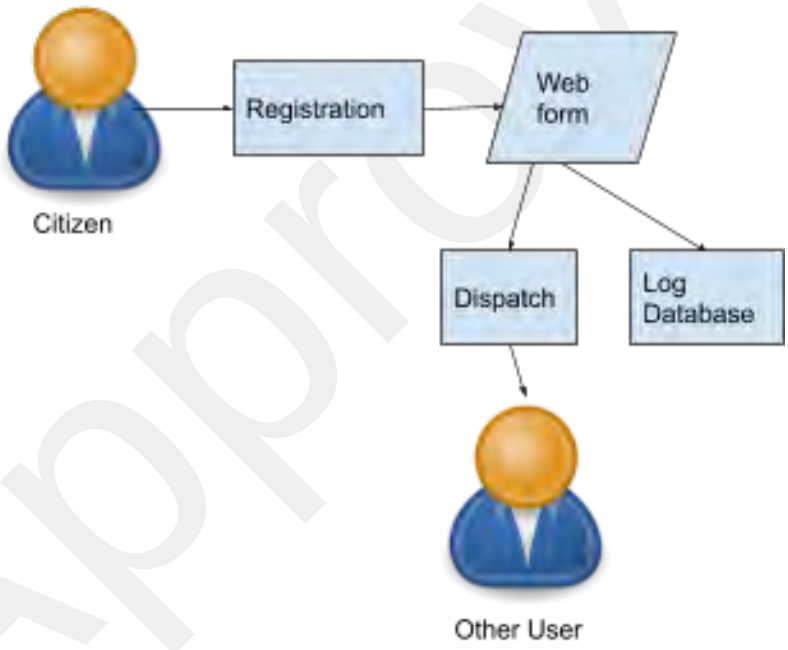

Figure 9: Overview of the main flow of the component.

4.4.3 Interaction between modules and related workflows

The crowdsource intelligence inputs are stored - for auditing and archiving purposes - , and are then forwarded to the Area Coordinator entity of each jurisdiction. Next, the input provided by citizens will be analysed to discard irrelevant information. Thereafter, the cases will be revised by a local user or transferred to another entity. From this point on, a new investigation will be opened in the LOCARD system and a new case ID will be assigned, as described in Section 3.4.

Approved

4.4.4 Design and architecture goals and guidelines

The aim of this tool is to incorporate the intelligence of users, and thus, fight cybercrime with more assets. This collaboration will foster easy to use web forms, which can be built and adapted to be compliant with each corresponding jurisdiction.

4.5 Storage Manager

4.5.1 Functional description

This module consists of several databases which store different types of data. Apart from the evidence storage (which will be different according to each end user requirements) as well as its proper access mechanism (API or SFTP service), this module will be used for giving users the ability to store metadata of cases/evidence. Users will select either to store this data on their own premises offline or in this distributed storage, consisting of nodes hosted in all end user's premises. In order to do that we will use TiKV/TiDB[14] technology which is the most popular, open source distributed transactional key-value database. Data is distributed across TiKV instances via the Raft consensus algorithm, which is based on the so-called Raft paper[15]. The idea behind that is that every organisation will host a node in their own premises and all these nodes will be interconnected . Data will be stored across regions and jurisdictions and all nodes will be participating in the consensus algorithm. Users can interact with storage instances through APIs provided by each instance. Each instance contains its own RocksDB[16] database as well as a raft orderer that defines the raft groups between instances and how RAFT consensus is implemented between regions[17]. This functionality is orchestrated by TiKV placement driver, the cluster manager of TiKV, which periodically checks replication constraints to balance load and data automatically across nodes and regions in a process called auto-sharding. When a Node starts, the metadata for the Node, Store, and Region is recorded into the Placement Driver, while the status of each Region and Store is regularly reported to the PD. On top of all TiKV instances there is a TiDB server that operates as a MySQL translator to these TiKV nodes and connects to their API Connectors to retrieve data accordingly.

4.5.2 Component diagram

Figure 10 shows a graphical representation of the storage manager and how this is going to be distributed and shared among participants of the blockchain network. Following this architecture we achieve high availability and consistency of data . The network can be easily expanded in the future as it is horizontally scalable.

In regard to the evidence storage, each end user can have a different access mechanism (API, SFTP), as previously stated. Therefore, further details about this will be described during the implementation phase.

4.5.3 Interaction between modules and related workflows

Access to each node will be handled by the Identity Manager of LOCARD which will define roles and rights on reading and writing data.

---

14 https://tikv.org/

15 https://raft.github.io/raft.pdf

16 https://rocksdb.org/

17 https://raft.github.io/

Approved

The LOCARD Portal will connect to the LOCARD Core Orchestrator Bus for making any calls both to the blockchain manager and to the distributed storage. So after making a call, the corresponding APIs will connect by JDBC Drivers to the TiDB server and then along with the Placement Driver will make the transaction across TiKV nodes. In general all transactions between modules and Storage Manager can be handled through the API so that access control and connectivity are secured.

- Intelligence Engine
- LOCARD Reporting Engine
- LOCARD Blockchain
- LOCARD UI/UX Interface and Portal
- Intelligent Crawler
- Investigator's Toolkit
- LOCARD TEE
- LOCARD Deviant Pattern Repository

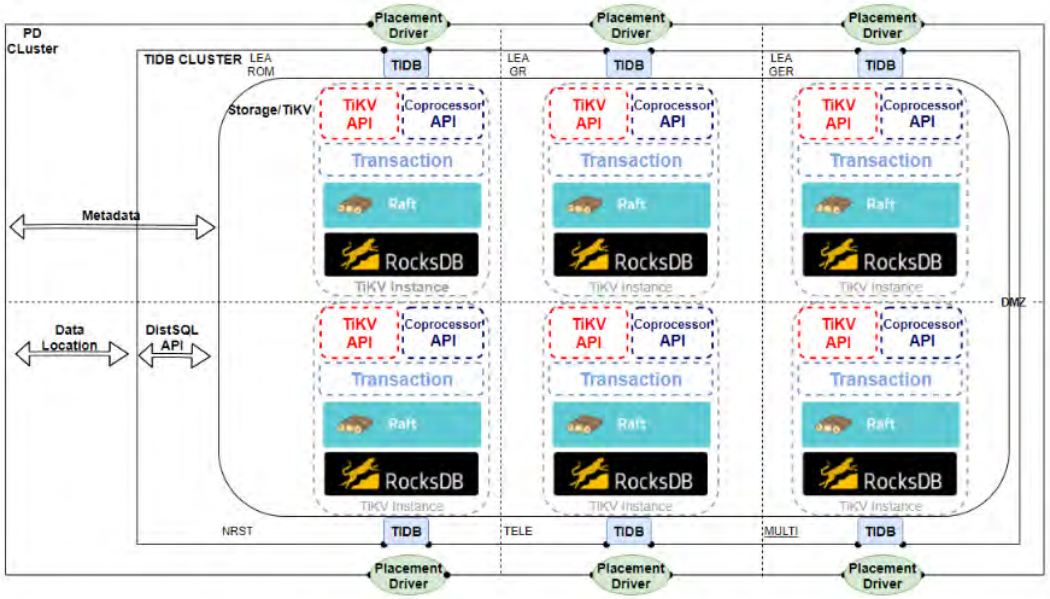

Figure 10: Overview of TiDB and TiKV interactions

4.5.4 Design and architecture goals and guidelines

A Storage Manager implemented in TiKV/TiDB technology offers to LOCARD a consistent distributed database for storing securely metadata of evidence as well as other data also. While each end user is hosting his own node with data, immutability, availability and trust is maximized. As it is open source, it already has a large community around it and many production environments live. TiKV excels at handling large volumes of data by supporting petabyte scale deployments spanning up to trillions of rows.

4.6 Blockchain Manager

4.6.1 Functional description

This module is in charge of managing the LOCARD blockchain network and issuing the smart contracts that will enable end user's interaction and describe the specific actions that a user can issue on the blockchain. The goal of the Blockchain Manager is to initialise a federated permissioned blockchain system, manage the users and their transactions and cater to the automatic generation of the Smart Contracts. In this regard,

LOCARD will allow the participating entities to use a predefined set of Smart Contract as skeletons that will meet the end-user requirements. In terms of Blockchain technology, we plan to use a federated permissioned blockchain platform which is mature enough and can offer advanced functionalities. This project will use the Hyperledger implementation which is publicly available by the Linux Foundation[18]. The blockchain manager functionality will consist of the creation of the network, the deployment of the smart contracts and the offering of the interface trough wich the users can interact with the network.

The blockchain network

There are multiple platforms that can be used to set-up a blockchain network. Because of the requirements imposed by the LOCARD context we have opted for implementing a distributed private and trusted network by using the Hyperledger Fabric blockchain implementation. The blockchain network will contain 6 different blockchain nodes, one for each organization, and one as a multi-tenant node where public users can interact with the network. Individual shell scripts will be built in order to bring each node up and set-up the blockchain network. Moreover, these scripts will install/instantiate the appropriate smart contracts to the channels (the private data contexts only visible by a predefined set of users) and set up the required endorsement policies. Nodes will communicate with each other by specific peer to peer connection that occurs through channels that interconnect the peers from different organizations, giving them the option to keep their communication private from peers of other organizations. Through setting up the the appropriate endorsement policies, all peers participating in a channel can also participate in the consensus procedure and secure the validity of the transactions. Additionally, all nodes store a copy of the data representing the state of the smart contracts, thus this state cannot be tampered with.

Blockchain nodes will be connected by channels. One basic channel will be created where all organizations will participate. There, transactions will be visible to all participants. Data segregation can be done through the Smart Contract access control. Because this general channel will not cover all the needs, regarding privacy, new channels will be previously created for every group of participants when a new collaboration is started. This way, a user can select to examine a case with another participant in their private channel where all transactions are not visible to other participants.

Ownership and fine granularity of data are two of the key concepts in Hyperledger Fabric in addition to immutability and trust. Every user will be enrolled by his organisation and interact with the blockchain by his private key. This will be his identity inside the network and at the same time will define specific attributes that will be used to decide allowed actions/view on data.

The smart contract

A smart contract will define the operations that users can issue to the blockchain; it will define how the user will interact with the blockchain to meet the user requirements. Namely, the smart contract will feature a number of methods that will be called by a user based according to the user's permissions. The functionality of the smart contract is not yet defined but one will be able to store new information in the blockchain, edit existing information or conduct functionalities according to each case. Moreover, the smart contract will need to vary from jurisdiction to jurisdiction and nation to nation as each may have different rules/standards. In addition, there needs to be some method whereby the files shared between two sites, the higher standard of the two is the one that is adhered to.

Smart contracts will be created using the IBM blockchain extension for Visual Studio Code which allows the easy creation of a smart contract (with a basic functionality), in Typescript, Java or Go technology. In this project we are going to use the Typescript technology and the produced smart contract will be installed on the created blockchain network

Smart Contract Functionality

18https://www.linuxfoundation.org/projects/case-studies/hyperledger/

A smart contract determines how the data record that is stored in LOCARD will be created and accessed. The control will be granular in that each data record stored on LOCARD must have its own set of rules defined in the executed smart contract. Smart contracts operate under a set of conditions that the users agree to ensure access to the data. When those conditions are met, the terms of the agreement are automatically executed (carried out), and the data is made available in the predetermined manner and under the predetermined restrictions.

Examples of data centric control can be:

- "This data entry can only be viewed by a user with level 'Inspector' or higher."
- "This data entry must be anonymised in all cases, when viewed outside of the originating jurisdiction"
- "This data entry must not be viewed in XYZ Member States"

These controls can be complex or simple and form the basis of the smart contract. It is likely that there will be standard schemas established to make it as easy as possible to store a data record in LOCARD. The use of schemas reduces the number of variables to be controlled and thus stops a proliferation of fields and conditions being created. However, there should be a simple interface to shield the end user from any complexity.

The Smart Contract must be executed in a trusted manner. That means the conditions of the schema must be verified by a mutually trusted third party, under consideration that this will be cross-border and cross jurisdictional. Without this trust, there can be no confidence that the smart contract is valid, and the entire methodology is usable.

Whilst the process of establishing trusted third parties, and indeed other critical procedures are not in the critical path of the LOCARD architecture, these issues (outlined in D4.7 and D4.3 Sections 8 onward) must be resolved before the third iteration of the architecture, and will probably consist of a governance and procedures rather than code. Nonetheless, they will be crucial to uptake of the LOCARD platform by LEAs and other involved entities.

The smart contract must interact with the trusted third party for execution:

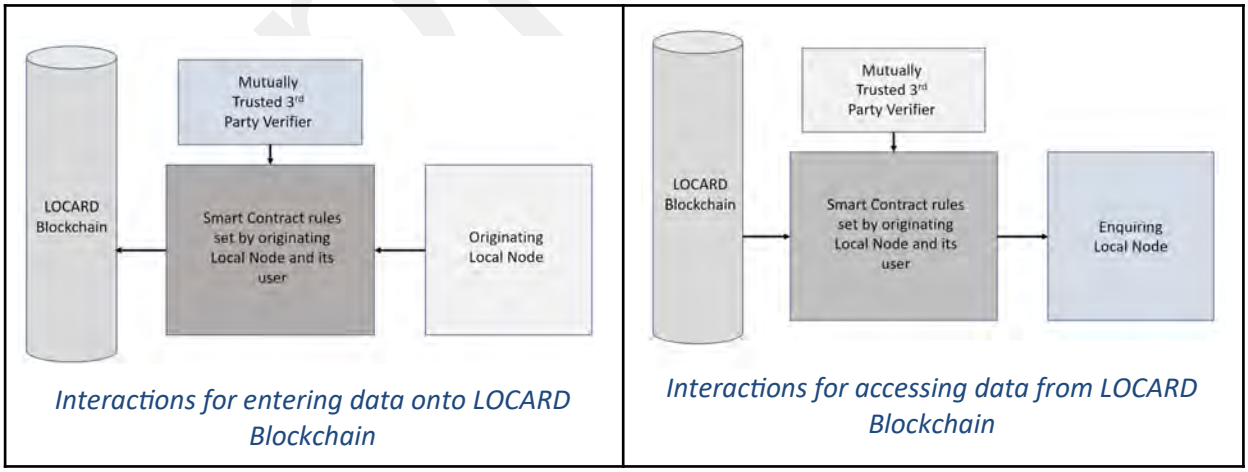

Figure 11: Verification and deployment of a LOCARD smart contract.

The smart contract will be invoked by the Motivian BPM tool as REST services. Namely the smart contract functionality will be exposed as a series of REST web services, which will be interfaced by the Motivian BPM tool. The smart contract functionality is defined as a series of methods that are exposed by the smart

contract and can be called by a specific user based on the user's permissions. These methods will be wrapped as REST web service and will be called by the same user using a web service client.

4.6.2 Component diagram

As explained earlier here is a visual representation of the Blockchain network between the different organisations.

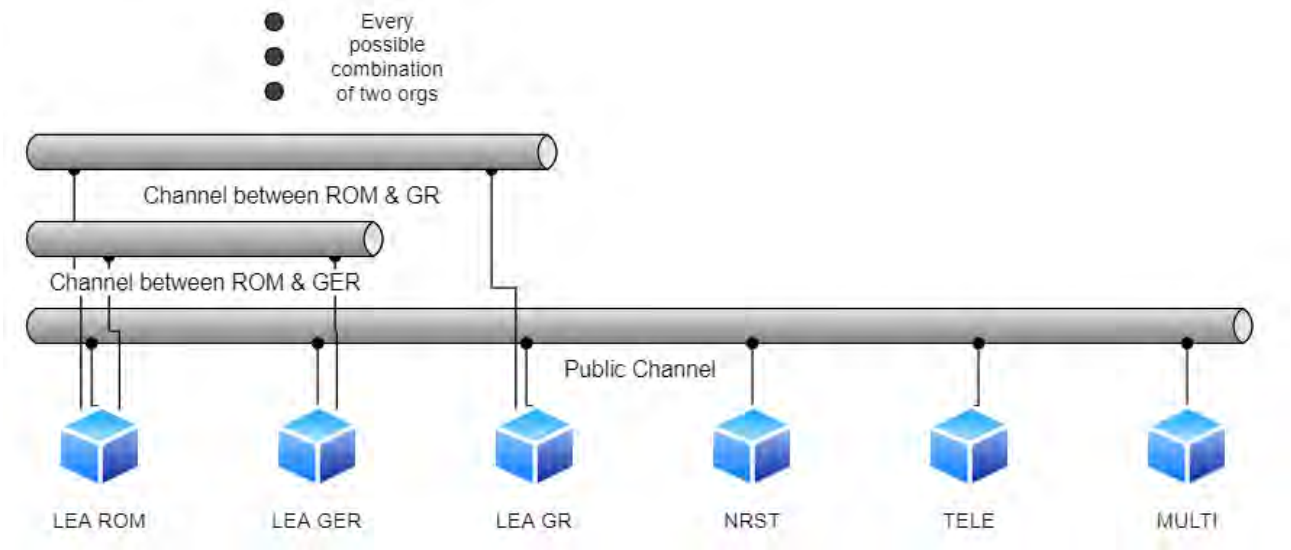

Figure 12: Blockchain network architecture

Each blockchain participates in a channel with its peers. In this setup, every node will contain at least two peers. Peers contribute to all transactions in a channel that are connected. Below is a more detailed schema how two organisations are contributing to the channel's transactions and how applications are interacting with the blockchain network.

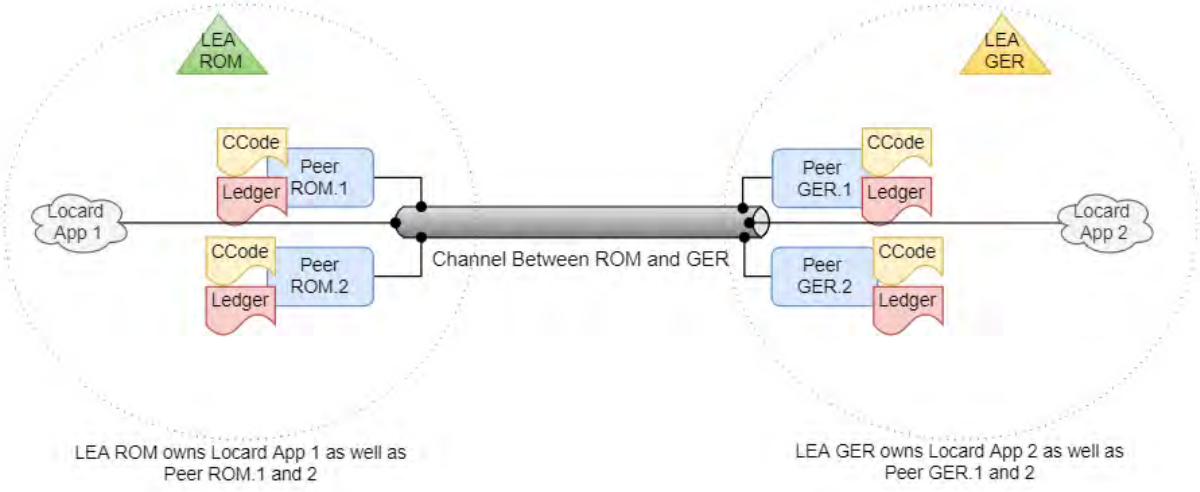

Figure 13: Blockchain node detailed view and participation in channel.

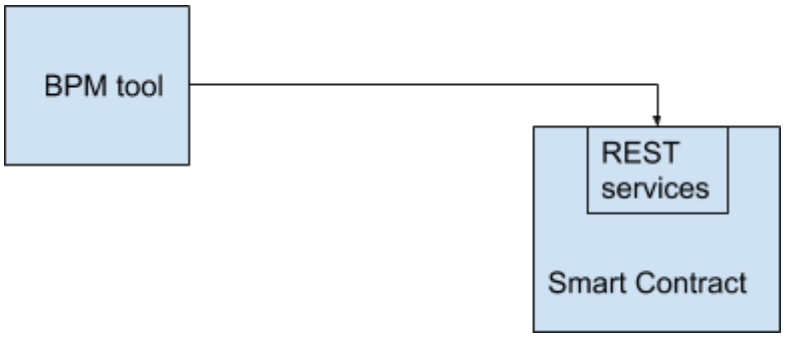

Figure 14: BPM and smart contract access.

4.6.3 Interaction between modules and related workflows

All Interactions between modules of LOCARD and Blockchain network will be achieved through a specific REST API that is going to be developed. This API will invoke/query functions of chaincode in a specific channel. Therefore, through this API, the Blockchain interacts with the following modules.

- Storage Manager
- Intelligence Engine
- LOCARD Reporting Engine
- Communication Engine
- LOCARD UI/UX Interface and Portal
- Intelligent Crawler
- Investigator's Toolkit

4.6.4 Design and architecture goals and guidelines

As far as the blockchain network is concerned, the architecture goals are covered in the functional description.

4.7 User and Identity manager

4.7.1 Functional description

There are multiple instances of identity management involved in the context of LOCARD.

Approved

Requirement	Process	Responsibility
Identification and access management of the end user. This could be dependent on user type, such as Defence solicitor or investigating officer rank etc.	Providing access to the LOCARD Platform	Responsibility of the end user entity and utilising organisation IDAM (such as LEA)
Identification (mapping) of the accessing end user within the LOCARD platform	Providing pseudonymisation and keys within the blockchain, together with appropriate channel/case access. (different pseudonyms for each channel?)	Responsibility of LOCARD Platform, with Channel/Case access rights transferred from end user entity (such as LEA)
Identification and authorisation of trusted third party as part of the smart contract verification	Certification of part of the smart contract schema's condition for execution. Should be signed using an advanced or qualified certificate	Responsibility of high level LOCARD issuing authority trusted by all LOCARD participants for that certification action
Identification and authorisation of jurisdictional or other LOCARD entities	Assorted authentication or signature requirements for non end user entities (such as schema certification which should be secured using signed code)	Responsibility of high level LOCARD issuing authority trusted by all LOCARD participants for that certification action

Table 2: Excerpt of identity management instances.

Possible system components should interact as follows, maintaining a single flexible Identity Management System (IDMS) interacting between varied user entities within a jurisdiction.

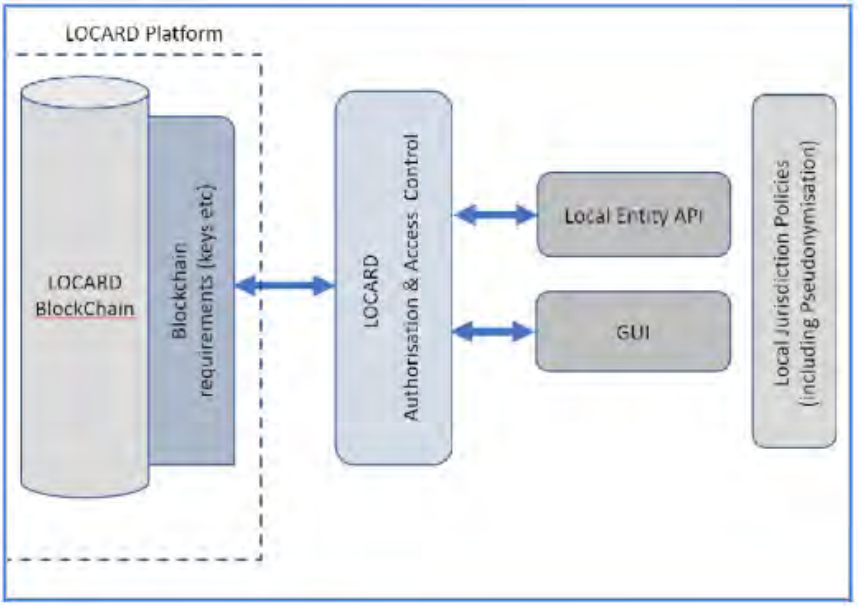

Figure 15: Interactions between ID system components

User Management in LOCARD is going to be implemented by a custom OAUTH2 Service which will define access, rights, roles and segregation of data for every user. This will be achieved by an OAUTH2 Server based on Java and Spring Security. OAUTH2 Server will rely on a Maria DB database which will hold credentials information for every user. When a user logins successfully through this service he will receive an access token. This token, a JSON Web Token (JWT), will be valid for a short period of time and will define the role of a user inside the application. MariaDB will also keep the public keys for each organization's users in Blockchain. Identity Manager will handle access to the app giving the users the ability to have a normal login procedure (username, password). At the same time when a user requests info from Blockchain or does an action to it, in order to accomplish that, the identity manager will retrieve his private key from the database and provide it to the Blockchain API to make any transactions. Like this we create a middle application layer that connects users to their keys in Blockchain. Also we keep sensitive Blockchain key info in the back end protected and not exposed over communication between services. This database/layer will be hosted separately on each organization's premises and will be handled by the defined admin. Admins can create groups or assign roles/access to menus and services by an admin dashboard in UI. Access to Blockchain will be defined by roles and permissions decided in each jurisdiction. Blockchain roles can be in sync with identity manager roles or allocated in an appropriate way.

4.7.2 Component diagram

Figure 16 describes the interaction between the user and the authentication server. This schema illustrates a basic communication for OAUTH2 protocol. After successful authentication, a user will retrieve an access token which will be used as his identity inside the app.

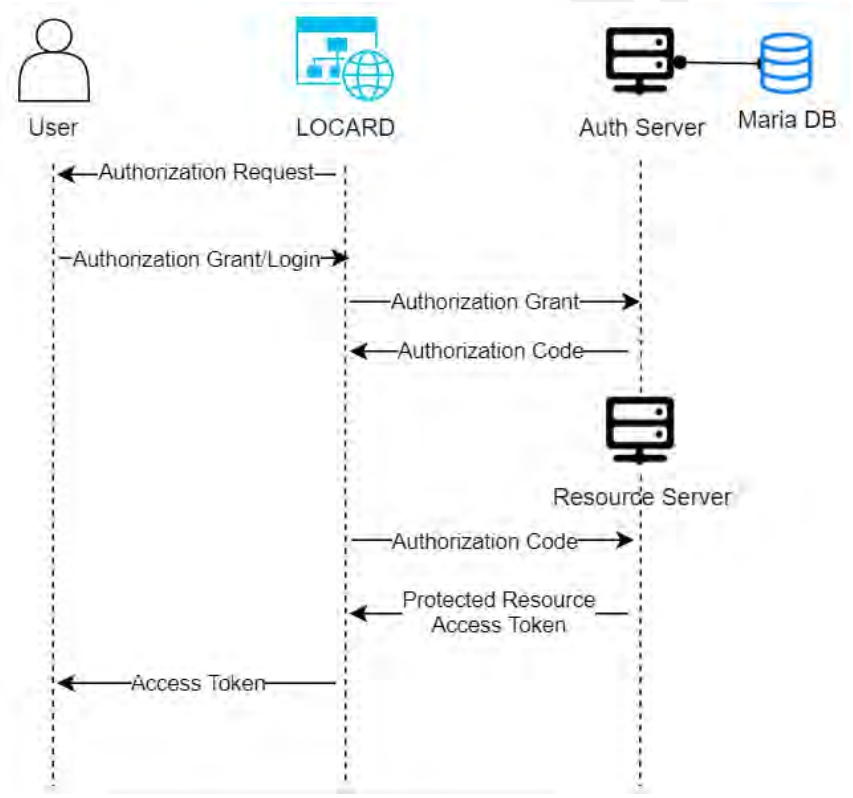

Figure 16: Authentication flow with JWT access token.

After retrieving the access token, the way of making transactions to the blockchain is displayed here.

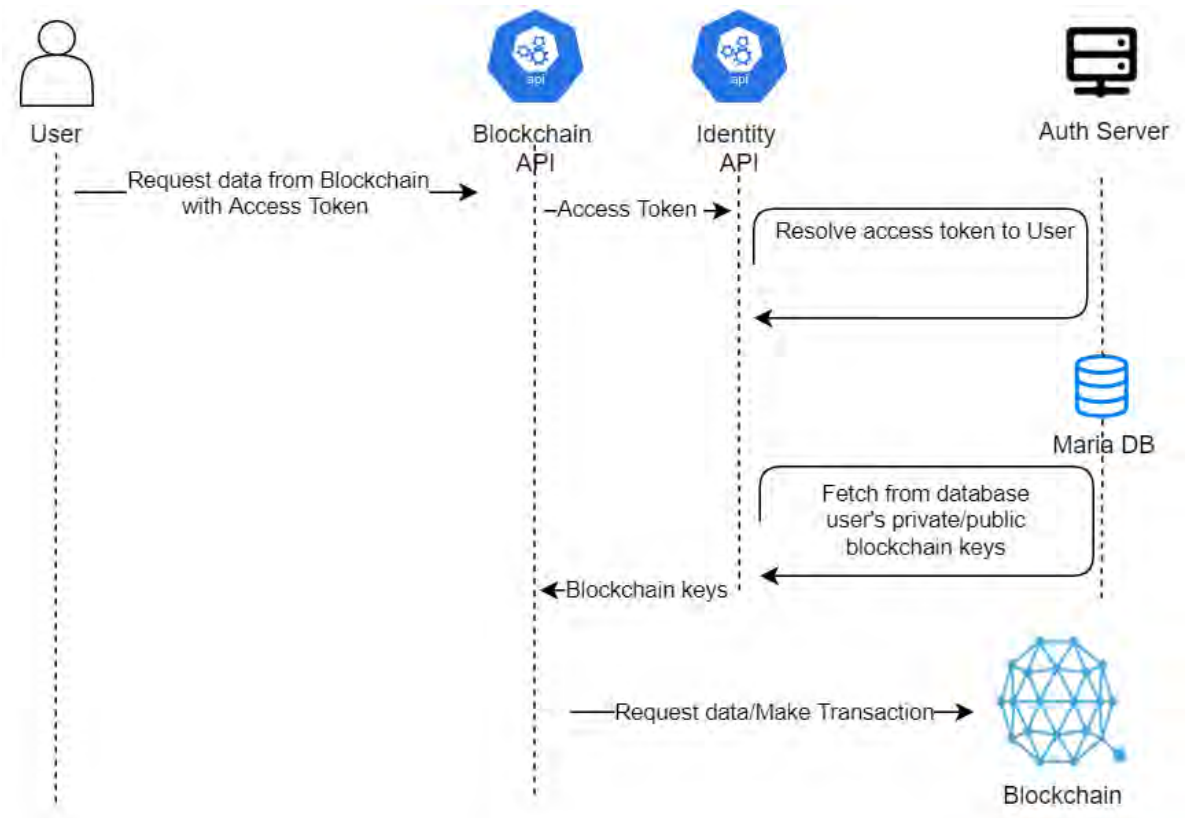

Figure 17: Blockchain authentication flow.

All aforementioned structures are hosted on each organization's premises.

4.7.3 Interaction between modules and related workflows

The ID management module interacts with all of the modules of the architecture.

Identity Manager is vital for LOCARD, as every transaction made by a user to the application needs first to be authenticated by the Identity Manager. Hence, it will be exposed to a REST Service that will delegate accessibility and role management.

4.7.4 Design and architecture goals and guidelines

Some of the main aims and goals of the ID management system are described in the functional section of this module. In general, this module will centralise all authentication services and it is aligned with the use of open source projects to allow the integration of other protocols if required (e.g. OpenID).

4.8 Intelligence Engine

4.8.1 Functional description

The intelligence engine's main aim is to process the data existing in the context of LOCARD to provide intelligence and extract correlations to ease criminal investigation. This module can be divided into several subcomponents and procedural flows as follows:

1. Data gathering: The intelligence engine will collect data from the blockchain through the blockchain API. This procedure will extract the data existing in the public channel of blockchain (i.e. private channels and data will not be accessible). In parallel, the intelligence engine is linked to the connector module to provide further connectivity with external datasets. Nevertheless, although

LOCARD has the potential to correlate data with other sources, in the scope of the project only LOCARD information will be used.

2. Data processing: The data shared in the public channel will have similar structure, yet this is not always possible due to the nature of the information, which includes different operations, hardware descriptions, or high level report outcomes. Therefore, we will classify this data according to its nature and thereafter we will perform an automated correlation with it, in order to find similarities.
3. Similarity computation: There exist many similarity metrics that enable computing how close objects are from each other[19]. In this sense, we will use, semantic analysis libraries (CoreNLP[20], NLP4J[21]) and correlation algorithms.
4. Query/Response: Given a specific information vector or input data, we will query the information stored in the intelligence engine. The intelligence engine will respond with a set of transactions containing related information. Thereafter, the active user can check if this information is relevant and potentially enable further communication with the specific entity to retrieve more data, or touse similar forensic procedures to the ones reported in a previous case. The latter enables a more efficient criminal investigation, reducing the time required to find evidence as well as minimizing personnel costs.

4.8.2 Component diagram

Figure 18 shows the modular flow of and the interactions between the Intelligence Engine and other modules.

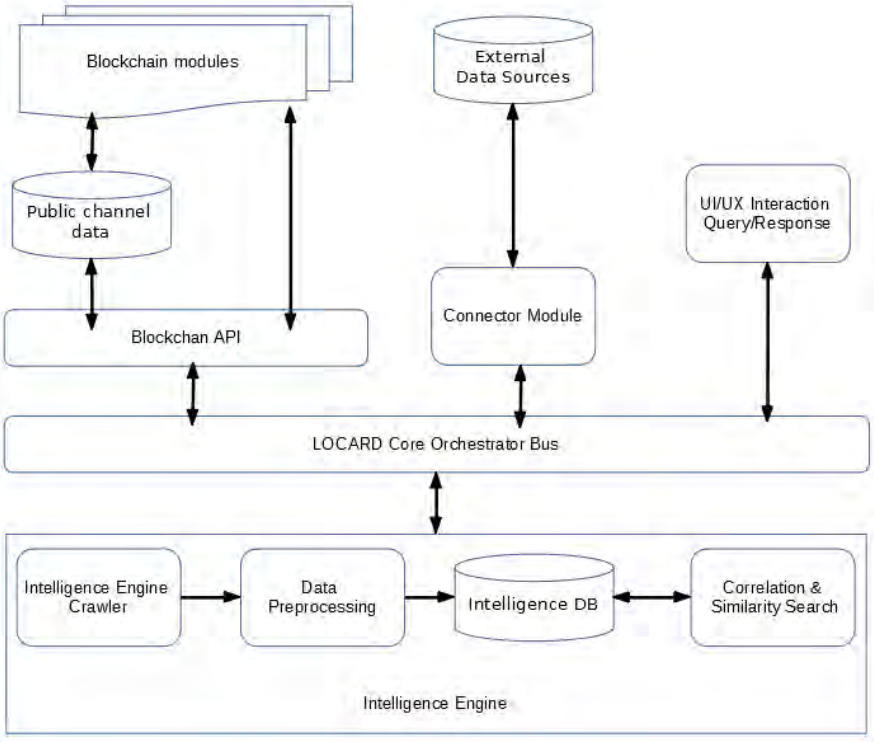

Figure 18: Overview and flow of the component.

19 Irani, J., Pise, N., & Phatak, M. (2016). Clustering techniques and the similarity measures used in clustering: a survey. International Journal of Computer Applications, 134(7), 9-14.
Gomaa, W. H., & Fahmy, A. A. (2013). A survey of text similarity approaches. International Journal of Computer Applications, 68(13), 13-18.

20 Stanford CoreNLP – Natural language software - https://stanfordnlp.github.io/CoreNLP/index.html

21 NLP Toolkit for JVM Languages - https://emorynlp.github.io/nlp4j/

4.8.3 Interaction between modules and related workflows

The main modules interacting with the Intelligence Engine are

- Storage Manager
- Connector Module
- LOCARD Blockchain
- LOCARD UI/UX Interface and Portal

4.8.4 Design and architecture goals and guidelines

The collaboration between different end users and the enhancement of criminal investigation are two of the main aims of LOCARD. In this regard, the knowledge extraction from data related to LOCARD, and the potential to correlate it with other data sources is a mandatory feature. In addition, the integration of communication APIs, which can extract data to be integrated in other local systems is a functional advantage for all end users. In summary, we will provide mechanisms to ease the use of data and knowledge across different system contexts.

4.9 Deviant Patterns Repository

4.9.1 Functional description

This module consists of two main parts. Note that this module can act as a standalone system and therefore, its interaction with the system is now proposed as such. Nevertheless, the integration of this module inside LOCARD will be assessed during the implementation phase. We therefore elaborate a concise explanation of this module due to its specific features and sensitive data management procedures, since it is one of the key research points of LOCARD.

The first task of this module is to monitor social live streaming services and social media to identify deviant behaviors such as cases of child exploitation, grooming of underage users and other predatory acts. To this end, the module will collect multimedia content, text messages and user interactions and analyse them in real time, as they become available, and then push it to the SFTP service. This module is implemented using a fault-tolerant, distributed crawler architecture in Python, leveraging a distributed task queue (e.g. RabbitMQ[22]) for orchestrating and distributing the jobs to multiple workers, in order to ensure the timely and robust collection of data from multiple sources. Accordingly, the collected data will be assessed in terms of deviant/criminal behavior using a flexible analytics pipeline for the different sources. This pipeline will include Machine Learning, Natural Language Processing and Social Network Analysis frameworks such as Tensorflow[23], SpaCy[24], gensim[25] and igraph[26], leveraging the models and datasets stored in Deviant Patterns Repository of the respective LOCARD node. For each type of crawled data, a different analytical

22 https://www.rabbitmq.com/

23 https://www.tensorflow.org/

24 https://spacy.io/

25 https://radimrehurek.com/gensim/

26 https://igraph.org/

Approved

workflow will be executed in order to analyze the collected content for known patterns and characterizing traits of deviant behaviors.

The second part of this module is the Deviant Patterns repository, which will contain curated datasets as well as trained ML models for the detection of deviant behaviors. Specifically, the role of the Deviant Patterns repository module is to safely store "ground-truth" patterns in various forms, reflecting the different aspects of deviant behaviors considered in LOCARD. For instance, such patterns can include text, multimedia and social graphs in annotated form, hashes of known malicious content, trained ML and probabilistic models for assessing if the content collected is deviant or not, and to what degree (i.e. by providing a probabilistic confidence score in a supervised classification setting). Examples include public datasets and natural language processing (NLP) models of sexual grooming/sexually predatory behavior in chat data[27], patterns of social connections indicating the production or consumption of sexually explicit content in social media platforms[28], trained computer vision models for detecting nudity in multimedia content[29], and collections of perceptual hashes[30] of illegal or copyrighted content. Additionally, the content of the repository can be updated with additional data resources and better ML models as they become available, to improve the performance and the accuracy of the module. The content of the Deviant Patterns repository module deployed in each LOCARD node, is subject to many locality and legal constraints. As such it will not be shared with other nodes and strict policies will apply, as stated in the Data Management Plan (in its second iteration M12), as well as in Deliverables 8.4 and 8.12. Moreover, the Deviant Patterns repository will safely expose interfaces/APIs to the Intelligent Crawler module, in order to offer the necessary interoperability for assessing the collected content.

The ML and AI techniques are designed and implemented as a standalone module. The idea is to use specific ad-hoc techniques and have a better control of how data are processed. Nevertheless, we will investigate during the implementation phase the feasibility of incorporating these methods to the investigator's toolkit module. Therefore, the proper links with blockchain and the Alert Engine will be established, as per the rest of the Investigator's Toolkit tools.

4.9.2 Component diagram

In regard to the social networks, we will crawl the social graphs and metadata of specific users which exhibit suspicious or abnormal behavior. In addition, usernames which were reported in the past can also be used a search criteria. Thereafter, this data will be processed and analysed accordingly.

Approved

27 Nikolaos Lykousas, Constantinos Patsakis: Large-scale analysis of grooming in modern social networks, arXiv:2004.08205, 2020

28 Lykousas, N., Gómez, V., & Patsakis, C. (2018, August). Adult content in Social Live Streaming Services: Characterizing deviant users and relationships. In 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM) (pp. 375-382). IEEE.

29 Zhelonkin, D., & Karpov, N. (2019, July). Training Effective Model for Real-Time Detection of NSFW Photos and Drawings. In International Conference on Analysis of Images, Social Networks and Texts (pp. 301-312). Springer, Cham.

30 Bjelland, P. C., Franke, K., & Årnes, A. (2014). Practical use of Approximate Hash Based Matching in digital investigations. Digital Investigation, 11, S18-S26.

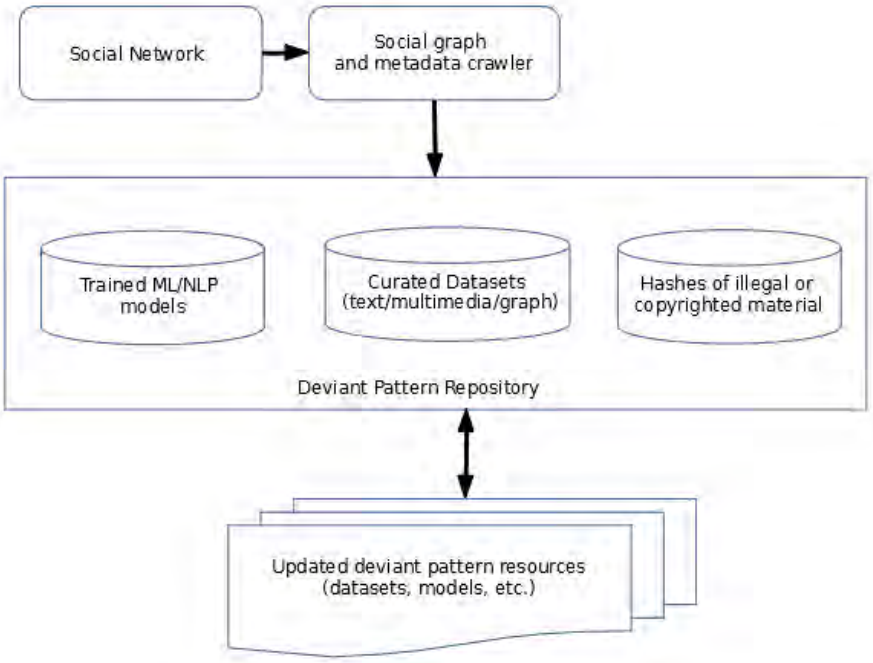

Figure 19: Overview and flow of the component.

4.9.3 Interaction between modules and related workflows

The main modules interacting with the Deviant Patterns Repository module are:

- Storage Manager
- Alert Engine
- Intelligent Crawler
- Investigator's Toolkit
- LOCARD UI/UX Interface and Portal
- LOCARD Blockchain
- LOCARD TEE

4.9.4 Design and architecture goals and guidelines

The Deviant Patterns Repository, as well as its associated data crawling, are one of the research cores of LOCARD to fight specific types of cybercrime. In addition, since ethical concerns arise due to the use of these tools, a careful design, implementation, and integration needs to be performed. In this regard, this module will comply with all the legal and ethical requirements, and the appropriate implementation procedures will ensure the integration of this module with the system by means of DevOps and Agile procedures, as well as the implementation methodologies described in D5.1.

4.10 Connector

4.10.1 Functional description

This module acts as the link between the LOCARD platform and external repositories. More concretely, the intelligence engine, which extracts and correlates information from the LOCARD platform and blockchain,

will be also used to gather information from other sources by means of the connector. This way, more information can be used to correlate LOCARD cases with external investigations. The connector provides a bridge connection via an API that connects to a selected external repository and crawls data given a specific query. Also it provides an API to connect LOCARD with the local system of the end user.

The connection to LOCARD case data for importing and exporting basic information will be performed via a RESTful API exposing numbers, dates, participants, status and other basic metadata but not evidence related data. Therefore, it can be used as a simple import/export tool for bridging data to internal LEA systems, which access will be granted via authorization from UIDMS.

The connection to **external sources** as a data import channel for the Intelligence Engine will be implemented as a web service client consuming information available in several API implementations (RSS, REST, SOAP etc). For example, it can consume public RSS feeds with most wanted persons and save them in an Intelligence Engine dataset to correlate them with active case data.

4.10.2 Component diagram

Figure 20 shows an example of how data can be retried from a RSS feed. A similar strategy will be applied according to each possible input data.

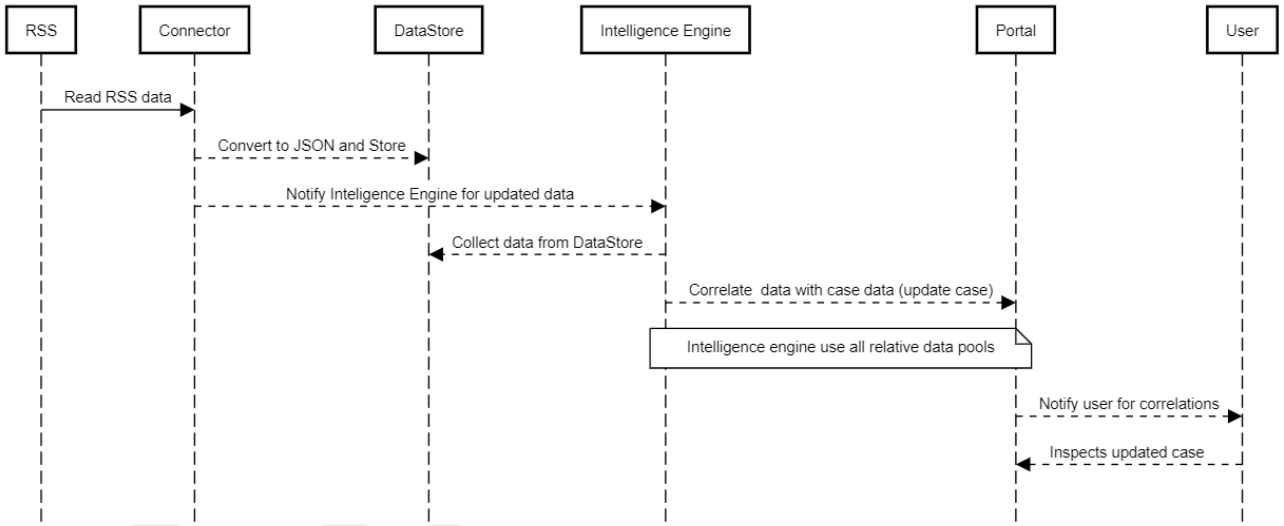

Figure 20: Example of Connector consuming RSS feed and providing data to Intelligence engine for correlation.

4.10.3 Interaction between modules and related workflows

The Connector interacts with the following tools:

- Storage Manager
- Intelligence Engine
- LOCARD UI/UX Interface and Portal

4.10.4 Design and architecture goals and guidelines

The main aim of this module is to expand the possibilities of LOCARD, enhancing collaboration with external data repositories as well as to import/export data from LOCARD to the local systems of the end users.

4.11 Reporting Engine

4.11.1 Functional description

The Reporting Engine's main aim is to collect the data related to a specific case identifier and create an audit log of all the interactions related to it. In addition, this module also interacts with the data related to specific cases in terms of forensic reports and evidence collected, as well as any related crowdsource intelligence. Therefore, the Reporting Engine will also be able to collect this information and append it to create a case report, if necessary.

The main mechanism to create the report will be to extract all the interactions logged into the blockchain, such as user id reporting. In this regard, to (e.g. evaluate the tasks of a user) the ability to create reports at user level is also considered.

The procedure will be a query to the blockchain API. More concretely, a user is able to interact with the blockchain using the exposed functionality in the smart contract. Therefore, the smart contract will expose a series of methods which can be called -based upon user's permissions- either directly or by using a REST wrapper by any REST client - the Motvian BPM tool included. Moreover, the smart contract methods can be invoked directly through a fabric gateway. Fabric gateways are connections to peers participating on Hyperledger Fabric networks. In order to use a gateway, a user needs a valid identity enabling him to transact on the network. This can be achieved with an identity created in the associated wallet. The methods for identity creation can be invoked directly or through a REST web service, which can be queried using a REST client such as the BPM tool. A special REST API will be created for this use case.

The engine will collect all transactions from the blockchain given a case id and will produce a report which will reflect the status for the given case. This will include people assigned to the case, collected evidence etc. The report will be produced for the specific user based on the given access. Thereafter, all transactions will be linked and collected, creating a report document. In addition, more extended information can be added if required, linking the hashes of the transactions with the corresponding forensic outcomes and evidence. Due to the variable runtime of this procedure, it will be performed as an asynchronous task.

4.11.2 Component diagram

Figure 21 describes the connection between the smart contract code and the report generation through the BMP tool.

Approved

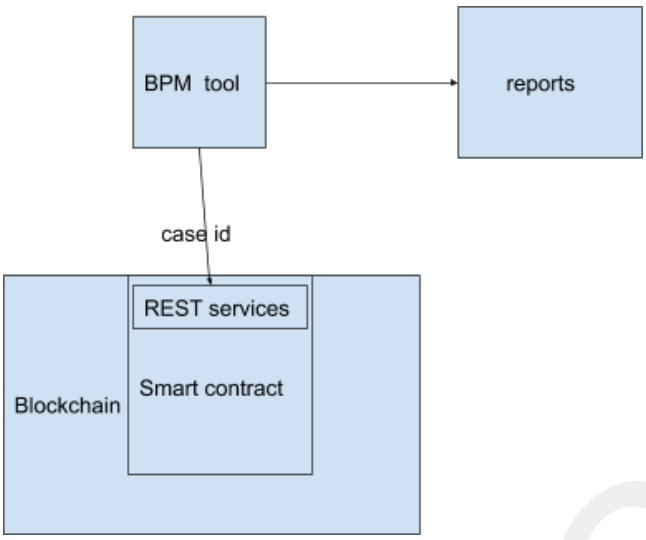

Figure 21: Overview and flow of the component.

4.11.3 Interaction between modules and related workflows

The Reporting Engine intracts with:

- Storage Manager
- LOCARD Blockchain
- LOCARD UI/UX Interface and Portal

4.11.4 Design and architecture goals and guidelines

The Reporting Engine is a key module to enable the validation of all the investigations, as well as to guarantee the integrity and verifiability of the forensic chain of events. Therefore, the reporting engine aims to create a clear and provable summary of the investigation, to be able to present it in court with all guarantees. The reports will follow an understandable schema, so that it can be understood by both technical and non-technical users. Moreover, further guidelines for forensic report creation will be followed to edit the final documents.

4.12 Alert Engine

4.12.1 Functional description

The alert engine is a messaging system that will send a notification under some specific conditions. More concretely, given a new interaction or transaction stored in the blockchain for a specific investigation, the users related with this investigation will receive an alert, specifying that a new event has been logged. These events can be triggered by, for example, the collection of a new evidence or the generation of a new forensic report. In this sense, this module consists of two main components:

- Alerts handler web service
- Smart contracts alert function.

Approved

The Alerts handler is a java web service, which is responsible to accept alerts generated via smart contracts and route them to the messaging interface. The alerts generated by the smart contracts will be defined according to each investigation, but the most general cases have been described in the previous paragraph. The Alert Engine will operate in a similar fashion as the Communication Engine, described in Section 4.3. An email will be sent to inform the users of advances in a case. Only the users with clearance to see any change in a specific investigation will receive notifications.

In order to support messaging functionality BPM flows for messaging will be set up. Apart from the flows, web services will be associated to each BPM installation and a directory of BPM nodes will be kept. More specifically, users of each BPM node will be able to submit messages either to a colleague within organization, or to a colleague related to a case within organization or to other users of LOCARD platform.

The last is the most challenging and will require extra setup and initialization. In this regard, each BPM node will keep a directory of associated BPM nodes with appropriate cypher keys. Each BPM will have a DIRECT communication to others' BPM messaging web service. The messages will be transferred encrypted.

4.12.2 Component diagram

The main operations of the Alert Engine are depicted in Figure 22.

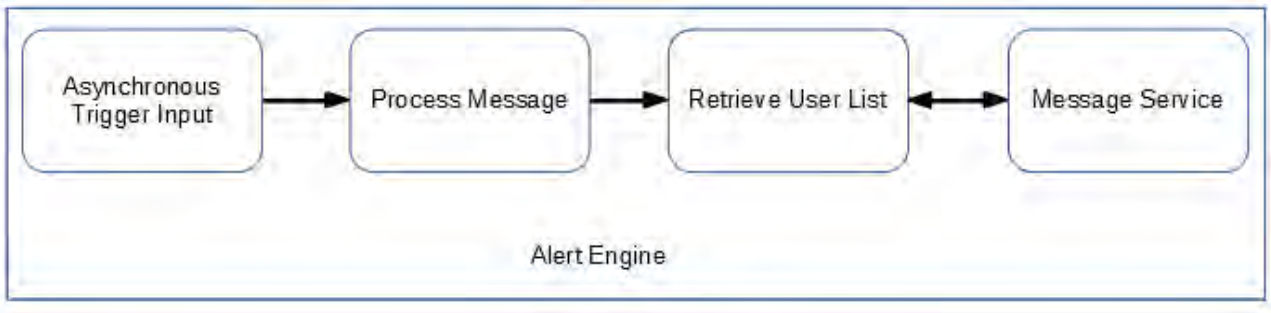

Figure 22: Overview and flow of the component.

4.12.3 Interaction between modules and related workflows

The modules interacting with the Alert Engine are the following:

- Storage Manager
- LOCARD Blockchain
- LOCARD UI/UX Interface and Portal
- Intelligent Crawler and Investigator’s Toolkit

4.12.4 Design and architecture goals and guidelines

The main aim of this component is to maximise the interactions with the system to minimise overheads as well as reducing the overall investigation times, enhancing cybercrime fight and rapid response.

4.13 Trusted Execution Environment

4.13.1 Functional description

This section describes how Trusted Execution Environment (TEE) technology is used in the LOCARD platform. Due to the different interactions of this module with several components (i.e. ID management, secure storage and secure execution of binaries), we elaborate how TEE will be integrated with each corresponding module.

TEEs provide an isolated environment in which specific operations can be offloaded to and executed in a secure manner separated from the possibly compromised normal operating system. Essentially, the TEE is a separate world (equipped with its own virtual CPU and memory) in which these sensitive operations can reside in with minimal capabilities in a need-to-have philosophy. In LOCARD, we are going to align with this practice by provisioning Trusted Applications (TAs) that implement only the required and most sensitive components of the project functionality. Our aim is to develop generic components that can be reused by different LOCARD entities. This way, we will provide a minimal efficient and as-secure-as possible trusted computing component implementation that will be able to provide the essential functionalities. More specifically, we will utilize the secure storage, the cryptographic functionality and also implement a secure execution environment that will check the integrity of an entire binary before it is allowed to be executed. These functionalities are analysed in the following list:

- **Secure Storage:** The secure storage provided by the TEE utilizes measures of cryptography and virtual separation (bit flagging by the memory management unit in order to provide separate address space for the normal and the secure world). We are going to utilize this for securely storing sensitive data, certificates and cryptographic keys that will only exist within the secure world and never leave it. This way we will maintain the highest possible level of security for this sensitive data and prevent attackers from attempting to eavesdropping it if it were to ever leave the secure world.
- **Cryptographic Operations:** The cryptographic functionality provided by the TEE uses internal libraries to implement a wide variety of cryptographic primitives and key generation functions so as to provide a seamless operation of normal world applications that utilize TAs to perform these highly sensitive functions. Additionally, the cryptographic functionality is closely interconnected with the secure storage which further enhances our scheme of having sensitive keys never leaving the secure world and being safely handled within it. Essentially we will provide an API to the normal world which will be able to specify the cryptographic operation to be executed and/or the key ID to be used with shared memory objects and attributes that will allow to essentially provide a drop-in replacement for every cryptographic operation that would otherwise be executed in the normal world application.
- **Secure External Execution:** We will implement a system for the secure integrity checking of core applications that either use or do not use the provided secure storage and cryptographic APIs mentioned before. This will build an added layer of security for core functions of LOCARD such as the identity validation, the authentication process, the digital evidence storage, and any other native binary identified to be of higher sensitivity. The way this function will work is that it will allow the execution of the pre-specified binaries only if they are first checked for their integrity using the TEE. This will be achieved by storing in the secure world the cryptographic hashes for each of these binaries and before their execution the measured hash will be compared with the stored hash and only if they match the execution will be allowed.

The underlying TEE technology will be ARM TrustZone for the implementation and specifically the OP-TEE environment.

Approved

4.13.2 Component diagram

The trusted execution environment component will be composed of a dynamic interface that will be able to serve a variety of security-sensitive functions. As discussed above, the three main functional categories target the secure storage that TEEs can provide, the secure execution of critical cryptographic functions and our own implementation of binary integrity checking that will be used for the authorized execution of said binaries. All of these sub-components have internal dependencies between them and thus can and will utilize functionalities of each other. For example, the cryptographic functionality will utilize the secure storage for storing the cryptographic keys used and the binary integrity checking functionality will also use the secure storage to store a list of authorized binary hashes that are able to run within the normal world. The aim of this design is to provide a dynamic secure environment that can be utilized by any application within the LOCARD project and protect them in their core functionalities while maintaining a balance between functionality and Trusted Computing Base (TCB) minimization. The overall scheme is presented in Figure 23 where each sub-component utilizes functions of each other, and each sub-component can be used by any LOCARD App.

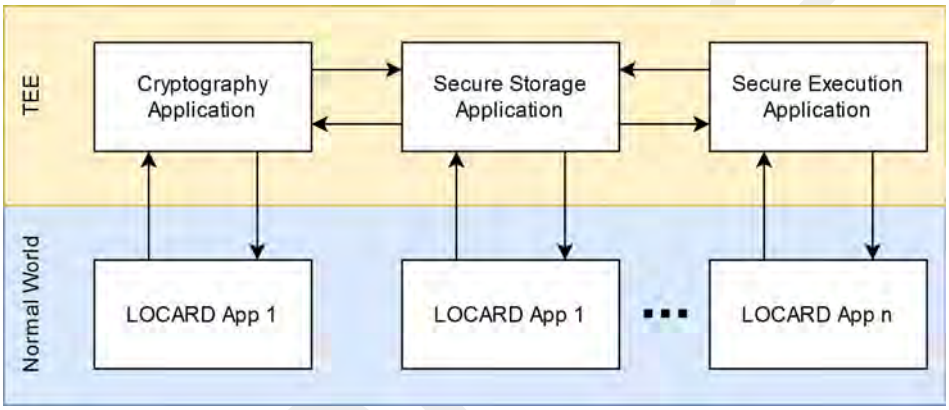

Figure 23: Component overview of TEE module.

The secure storage component application will provide a flexible API for the storage of security critical information which could include authentication data and evidence collected. The main concept behind this, is the utilization of storage identifiers which will allow for easy management. More specifically, a client application needs to first be authenticated and then request the allocation of a single storage slot, the secure storage application will then allocate the requested slot and return to the client application the storage ID. With the storage ID, the client application will be able to store and retrieve data within this slot given that it is always authenticated and that it provides the correct storage ID. The entire procedure is demonstrated in Figure 24. It is of note here that the initial storage allocation can be enriched with additional properties which could for example indicate that a specific data slot is to be written only once, or that it cannot be retrieved to the normal world and can only be utilized in the secure world (such us cryptographic keys used for cryptographic functions executed within the TEE).

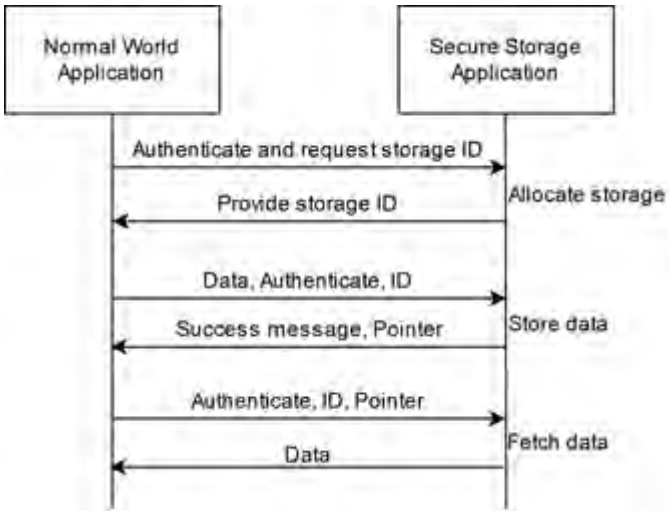

Figure 24: Abstract overview of the secure storage application.

The cryptography application will provide a drop-in replacement for a variety of cryptographic operations that include key generation, encryption-decryption and any other operations supported by the underlying environment which is aligned with the functionality provided by the standardization TEE body GlobalPlatform[31] in the internal API specification. As apparent, this provides a two-sided functionality, one for key creation where the client application will be first authenticated and will be able to either create keys either for inner-TEE cryptography or to create external keys to be exported out of the TEE for other applications. The second part of the cryptographic functionality is composed of the actual cryptographic functions where each function requires an optional key and the appropriate configurations from an authenticated client application. The application will be able to send: (optional) data, (optional) key ID stored in the secure storage, (optional) key stored by the application, cryptographic function ID and cryptographic function configuration. After a correct evocation of the cryptographic function, the TEE will send the success/failure message and the applicable results. This design is depicted in Figure 25.

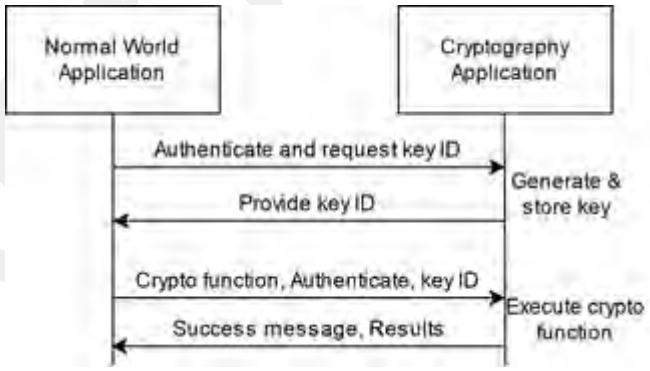

Figure 25: Abstract overview of the TEE Cryptography Application.

The secure execution function will be able to provide a level of assurance behind a normal world application before it is allowed to be executed. A birds eye overview is depicted in Figure 26 where two main functions can be observed: (i) the registration of a trusted application through its calculated hash and (ii) the integrity checking of an already registered application that will allow or disallow the execution of it. The entire procedure will be supported through secure hash functions and possibly asymmetric cryptography for external attestation purposes.

31https://globalplatform.org/technical-committees/trusted-execution-environment-tee-committee/

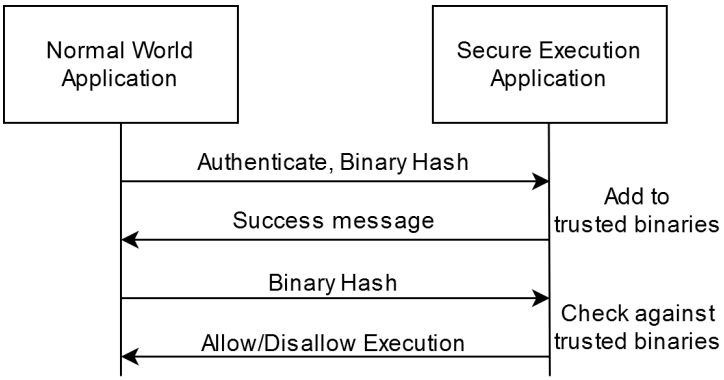

Figure 26: Overview of the secure execution function.

4.13.3 Interaction between modules and related workflows

According to the identified use cases of trusted computing within the project the basic functionalities that need support by this module are the identity validation / authentication process and the secure storage of the collected digital evidence. Given the flexibility of the proposed architecture, it can support these functionalities and any functionality identified in the future, as it will provide drop-in replacement APIs for the required functionality. In summary, the modules interacting with TEE (in addition to the core modules or LOCARD) are:

- Storage Manager
- Intelligent Crawler
- LOCARD UI/UX Interface and Portal

4.13.4 Design and architecture goals and guidelines

The targeted functions are three: (i) cryptography, (ii) storage and (iii) execution integrity. These three functions are common attack surfaces that often are protected through traditional security measures that we aim to enhance by utilizing the security guarantees of TEEs. The cryptography functions that run within the TEE take advantage of the protected and isolated environment that will allow them to be executed securely, minimizing the possibility of inference from malware residing in the normal world. On the other hand, the storage functionality is backed by the secure storage capabilities of the TEE that allows for an isolated and cryptographically secure storage of information. Furthermore, secure storage allows for the protection of the cryptographic keys that never leave the TEE, providing a completely isolated environment for cryptographic operations. Both of the aforementioned benefits apply to the execution integrity checking function that can safely compare the integrity hash values of the running applications, while having a secure place for storing these hashes in the secure storage.

The guidelines we provide for the utilization of the TEE module follow our design and architecture; that is, security, minimality and flexibility. We advise developers to utilize these functionalities only for sensitive operations and data and to avoid overuse of the module as it can reduce its performance and possibly open attack vectors. We aim to provide an API based on which developers can program drop-in replacement libraries for the programming languages that they use. Effectively, we propose a segregated model, in which the development of the described functionalities and the API falls in the development of the TEE component and the library that utilizes the API alongside the application is developed by each application development team (Figure 27).

Approved

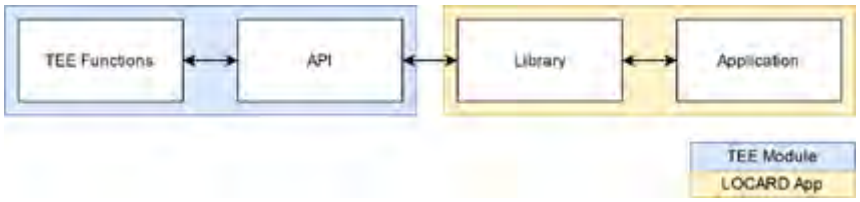

Figure 27: The proposed model of the segregation between TEE and application development.

4.14 LOCARD Core Orchestrator Bus

4.14.1 Functional description

The LOCARD Core Orchestrator Bus (LCOB) is a core module that leverages the internal communication flows of the whole LOCARD system. More concretely, the LCOB can be understood as an Enterprise Service Bus (ESB), which implements a communication system between mutually interacting software applications in a service-oriented architecture. In this sense, an ESB applies the design concept of modern operating systems to independent services running within networks of disparate and independent computers. Like concurrent operating systems, an ESB provides commodity services in addition to adoption, translation and routing of client requests to appropriate answering services.

The main benefits of an ESB in the LOCARD architecture are the following:

- Route messages between services
- Monitor and control routing of message exchange between services
- Resolve contention between communicating service components
- Marshal use of redundant services
- Provide commodity services like event handling, data transformation and mapping, message and event queuing and sequencing, security or exception handling, protocol conversion and enforcing proper quality of communication service.

The main technology of such ESB is RabbitMQ, an open-source message-broker software, providing an asynchronous communication channel. This avoids the necessity of direct interactions between modules enabling the creation of different message topics for internal communication. Therefore, we will be able to broadcast interactions between several modules (e.g. Investigator Toolkit will log interaction in blockchain and also send data to the Reporting Engine), which is mandatory in a system such as LOCARD.

The communication topics will follow a specific format and syntax to refer to the different recipients and modules. Therefore, the information will be processed according to this message contents. For this purpose, simple action messages in JSON format will be defined during the implementation phase.

4.14.2 Component diagram

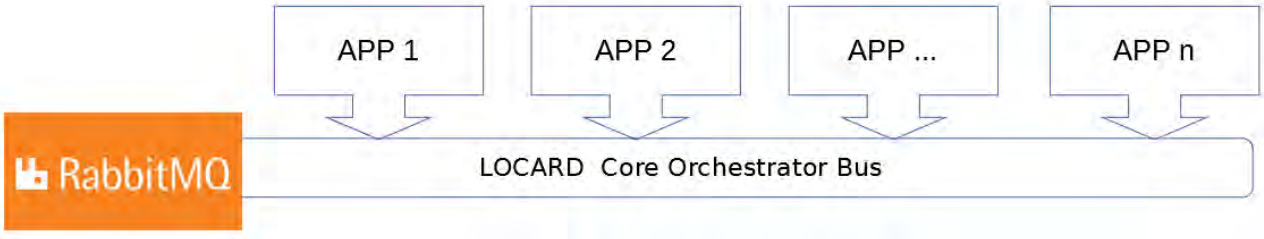

Figure 28: ESB orchestrator in LOCARD.

In Figure 29, each APP corresponds to a service/module in the LOCARD platform. More details and a high overview of the LOCARD components can be found in Section 3.

4.14.3 Interaction between modules and related workflows

This module orchestrates all the rest of modules of the LOCARD platform, including blockchain. The only exception is Crowdsource Intelligence, since this module is considered external in that sense.

4.14.4 Design and architecture goals and guidelines

The main goals are summarised in the benefits of ESB and their features, as previously explained in Section 4.14.1 and in Section 3.2.

4.15 LOCARD Portal - Process Workflow

4.15.1 Functional description

This module comprises of the front end LOCARD Portal through which all modules of the platform will be accessed and also of the Motivian BPM (business process management workflow) tool that will be used to create case scenarios, workflows for all scenarios and workflows in between modules. The BPM tool is accessed through the portal. Both are based on Java spring boot framework technology and will interrelate in order to have a seamless flow of operation for all cases to be designed and developed.

This module is used to define the interactions between various LOCARD modules, in order to automatically create the U/I based on comprehensive form description and to manage the users of the LOCARD Platform. It is able to orchestrate the work done between modules and act as a glue between them, interacting also with the LOCARD core orchestrator bus.

The business process workflow tool consists of four basic functional components, described in the next paragraphs: the Process Designer, the Workflow engine - Process Server, the Application Interface and the Administration tool.

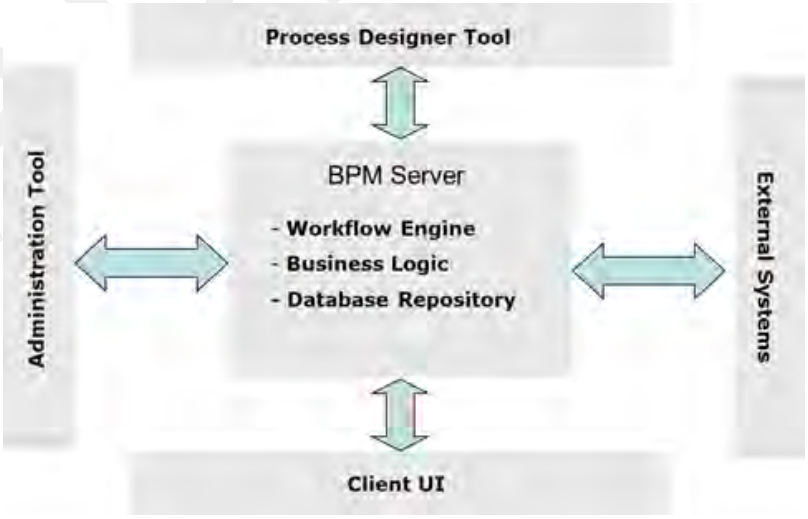

Figure 29: BPM server components.

4.15.1.1 The Process Designer

The Process Designer Server supports graphical design of processes. It features a familiar UI and easy-to-use graphical elements, allowing any business-level person to build processes. From the Process Designer, managers and process builders can view and edit all properties of each process activity, including timing, costs, work assignment rules, data, and applications to be used.

Once the process design is completed, it is registered via XML to the Process Engine and is ready for deployment and execution from the AMA (Administration, Monitoring & Analysis) Console.

4.15.1.2 The Workflow Engine – Process Server

This is the Core Process Engine. It supports multiple processes, sub-processing and exception handling. It operates as an autonomous unit, receiving information from various event channels (web clients, HTTP, APIs, administration interfaces) and invoking services to internal and external systems.

The core functionality is built entirely in Java and can run in any J2EE application server. It deploys what was designed in the previous phase:

- Creates the appropriate application interface (web)
- Manages the different versions of the application
- Processes incoming information and manages data

Moreover, it provides:

- Persistent storage. In production with a Database Server.
- Unicode database supporting multi-language processes.
- Auto-generated web interface for every process that also supports XML-XSL web page customization.
- Robust and autonomous security module that can be redirected to LDAP server.
- Intelligent fail-over.
- Multi-threaded application that increases performance metrics.
- Powerful graphical expression builder.

4.15.1.3 The Application Interface

The Application/User Interface enables users to view relevant process information and offers an easy-to-use point and click interface with a work area to perform work assigned to them.

The UI is automatically generated based on the process model and the process/activity/user properties. The work-items produced by the core Service are passed through the presentation layer. Being an autonomous unit, the presentation layer can bind the work-items to HTML pages, XML data sources or any other format through the Java API supported. The user interface is based on XML/XSL and Java Server Pages (JSP).

4.15.1.4 The AMA Administration Console (Administration, Monitoring and Analysis)

AMA is the Administration and Analysis module of the Process Server. In particular, AMA stands for Administration, Monitoring and Analysis, as described in the paragraphs that follow:

Administration

From the AMA Console, system administrators can view and modify process definitions and rules, manage different versions of processes, specify user privileges and permissions, manage scheduling, monitor security, and perform other system maintenance.

Approved

Designated system administrators have system overriding capabilities to take special actions necessary for handling external events or responding to business-level needs of an organization (e.g. temporarily assign a process to a specific employee in case the designated process user is absent).

Monitoring

To help identify and solve problems in processes, the Monitoring module provides consolidated and detailed information on the status of all process activities, whether they are running, inactive, or terminated. Through a tabular view, Process Monitoring displays process activities and their properties in columns (which can be sorted), offering the ability to expose only relevant activities to more quickly understand a situation.

Process activity properties include process category, type, participant, time, status, application in charge, and much more. Process owners can also use **graphical chart**s to easily view waiting and execution times of terminated activities and compare them with expected times to help forecast time duration of the final process. It also features an **intelligent alert system** that automatically notifies relevant process owners or business managers about possible problems in business processes and then suggests appropriate action. Process owners simply need t**o set criteria for monitoring. For example, a process owner can specify to receive alerts about "time duration that deviates 100% or** greater than the estimated time" for all activities; after the system detects a certain activity took 300% above the expected average time it immediately delivers notification and recommends investigating the cause. Alerts save business managers time as they review only deviation cases that meet criteria they specify rather than having to constantly scan metrics looking for problems.

Analysis

Through comprehensive reports and visuals, the Process Analysis module provides a full range of metrics for all process activities to help managers optimize business processes. It measures a variety of data including user and resource utilization, actual vs. expected work time, process path probability, process status, and much more. Graphs and charts provide easy-to-understand feedback, helping process owners quickly identify ongoing problems, such as bottlenecks in process flows, and refine processes accordingly.

Case Creation

This module will also enable the case creation functionality based on processes to be designed and built using the BPM tool. In this sense, we have two possibilities or scenarios for a new case creation, as shown in Section 3.4. Scenario 1 is related with the information collected from the Crowdsource Engine. Therefore, such information will be analysed by the corresponding LOCARD user and, if admissible, a new case will be created from such interaction. Scenario 2 is where a case is created as a direct input from an investigator.

In both cases, the LOCARD Interface module will enable the creation of such case, specifying the details (e.g. case ID, timestamp, context), and the users involved (which will remain private and only known by the investigators related to such case).

Case creation scenario 1

When the LOCARD Crowdsource Engine identifies that there is sufficient intelligence to warrant an investigation or when a citizen uses the LOCARD web interface to register and report an incident.

In this scenario LOCARD will automatically create a case file or 'container' and allocate all the information identified by the LOCARD Crowdsource Engine to this container. The creation of this container will also generate a case Unique Reference Identifier (URI). To avoid any third party being able to determine any statistical information or gain any intelligence about this or other LOCARD activity, this identifier should be randomly generated and non-sequential.

Approved

LOCARD will now determine which is the most appropriate jurisdiction to allocate the case to, and having done so, will send the container and its URI to the LEA in that location. The LEA will become the 'end user'.

LOCARD will have an established interface with the 'end user' enabling the container to automatically enter the end user's crime reporting system. It will be allocated a second URI by the end user at this point to enable it to be worked upon within the end user's own system.

The 'end user' will have a filter in place (probably human). This filter will analyse each LOCARD case container received. This screening process will result in the container being either

1) flagged for further investigation OR
2) simply close it if the case does meet local[32] required thresholds for investigation.

Case Creation Scenario 2

Where an Investigating Officer (IO) identifies the need to create a case the process will be similar to the previous scenario. There are several reasons where this could occur, such as evidence discovered during another investigation or incident, or a complaint from a member of the public.

In this scenario the IO will create a container within their own investigation system and the container will be given a 'local' URI.

Through the interface established for scenario 1, LOCARD will automatically be notified of this container having been created locally. LOCARD will note this container's creation and create a LOCARD URI. This will enable the other LOCARD process to be applied to this container.

Intelligence

In addition to the above processes for Case Creation, intelligence agencies such as Europol, may also wish to monitor all such containers so that a wider intelligence picture can be established.

This will enable these intelligence services to identify trends and identify anomalies that would warrant further investigation or the issuing of alerts to end users.

Therefore LOCARD will include an interface for intelligence agencies to link to, should there be such a need, and 'end users' agree that this data may be shared.

Required Information

The containers should include the following information:

- Time & date of case creation
- Reasons why the case has been allocated to a particular jurisdiction (e.g. location information of ISP etc.)
- The most likely offence category that is likely to be contained within the case, e.g. sexual exploitation. If this is not immediately obvious, the case should be marked 'To Be Determined'
- Details of the digital evidence, such as size and format (s).

4.15.2 Component diagram

Figure 29 shows the Motivian BPM components and interconnections.

32 This screening/filter is done with most reported crimes before resources are allocated to them. There will be varying investigation priorities from one LEA to another and therefore what is investigated in one jurisdiction may not be a priority in another and will be discounted there.

Approved

4.15.3 Interaction between modules and related workflows

The LOCARD Portal integrates all modules and it orchestrates the messages between the different components of LOCARD by means of the LOCARD Core Orchestrator Bus.

4.15.4 Design and architecture goals and guidelines

The design goals and guidelines of this module are summarised in the functional section as well as in the Annex.

Approved

5 Task assignments, technologies and cross-linked information

The following table summarises, for each module, the leaders, participants, and key technologies used. Note that the development of each module will be supervised by the corresponding partners during the project's lifetime.

Module	Responsible Partner	Collaborating Partners	Keyword Technologies
Crowdsource Intelligence	MOT	KEMEA	Java spring framework, React, MariaDB,
The Intelligent Crawler	ICO	NRS/ MOT/ ARC	HTTrack, ad-hoc crawler for social networks, nodejs
Investigator's Toolkit	All		Ad-hoc implementations, technologies table stated in Section 4.2
Storage manager	NRS	ICO/ MOT	TidB, TikV, MariaDB
Blockchain Manager	NRS	ICO/ MOT	Set Up of Blockchain Network (NRS) Smart Contract Development(MOT) API Connectivity (ICO) hyperledger, java spring framework, nodejs
User and Identity Manager	NRS	ICO/ MOT/ ARC	Oauth2 and Maria DB
Intelligence Engine	IMC	ARC	AI methods, blockchain queries to the blockchain manager. NLP libraries and CBR libraries
Deviant Patterns repository	ARC	NRS/ MOT	AI methods and libraries, ad-hoc social media crawler
Connector	IMC		Proof of concept. Bridge using API of the public service to collect data and relate them to case data in Portal Database.
Reporting Engine	MOT	All	Java spring framework, React, nodejs
Alert Engine	MOT		BPM Mailing system
Communication Engine	MOT	ICO	BPM Mailing system
Trusted Execution Environment	ARC	MOT	ARM TrustZone
LOCARD Core Orchestrator Bus	All		ESB. RabbitMQ
LOCARD interface and portal	MOT	IMC	Java Spring boot, Nodejs, React, Swagger

Table 3: Summary of technologies.

6 Conclusions

This deliverable describes the LOCARD architecture, its components, and main technologies. It provides a comprehensive overview of the forensic flows and provides the implementation details at a modular level as well as at integration level. For each module, we provide a description of its functionality, operations and main technologies. Finally, this document also reports what partners are responsible for each ask. The architecture document is a living document and will be updated according to the issues and needs that will be faced during the development as well as during testing and validation phase. The next iteration of this deliverable will be D3.8 on month 36, and will reflect all of these updates. Nevertheless, no major changes are expected during the project lifetime.

Approved

7 Annex

7.1 Atomic Design Principles and Methodology for UI-UX

Design Methodology

As the craft of Web design continues to evolve, we're recognizing the need to develop thoughtful design systems, rather than creating simple collections of web pages. A lot has been said about creating design systems, and much of it focuses on establishing foundations for colour, typography, grids, texture and the like. This type of thinking is certainly important, but on the other side these aspects of design are and will always be subjective. Our method examines the components of our interfaces, and how we can construct design systems in a more methodical way.

In searching for inspiration and parallels, we kept coming back to chemistry. The thought is that all matter (whether solid, liquid, gas, simple, complex, etc.) is composed of atoms. Those atomic units bond together to form molecules, which in turn combine into more complex organisms to ultimately create all matter in our universe. Similarly, interfaces are made up of smaller components. This means we can break entire interfaces down into fundamental building blocks and work up from there. That's the basic gist of atomic design.

What is Atomic Design?

Atomic design is a methodology for creating design systems. There are five distinct levels in atomic design:

- Atoms
- Molecules
- Organisms
- Templates
- Pages

Figure 30: Atomic design components.

Atoms

Atoms are the basic building blocks of matter. Applied to web interfaces, atoms are our HTML tags, such as a form label, an input or a button.

Atoms can also include more abstract elements like colour palettes, fonts and even more invisible aspects of an interface like animations.

Like atoms in nature they're fairly abstract and often not terribly useful on their own. However, they're good as a reference in the context of a pattern library as you can see all your global styles laid out at a glance.

Molecules

Molecules are groups of atoms bonded together and are the smallest fundamental units of a compound. These molecules take on their own properties and serve as the backbone of our design systems.

For example, a form label, input or button aren't too useful by themselves, but combine them together as a form and now they can actually do something together.

Building up to molecules from atoms encourages a "do one thing and do it well" mentality. While molecules can be complex, as a rule of thumb they are relatively simple combinations of atoms built for reuse.

Organisms

Molecules give us some building blocks to work with, and we can now combine them together to form organisms. Organisms are groups of molecules joined together to form a relatively complex, distinct section of an interface. We're starting to get increasingly concrete. Someone might not be terribly interested in the molecules of a design system, but with organisms we can see the final interface beginning to take shape.

Organisms can consist of similar and/or different molecule types. For example, a masthead organism might consist of diverse components like a logo, primary navigation, search form, and list of social media channels. But a "product grid" organism might consist of the same molecule (possibly containing a product image, product title and price) repeated over and over again.

Building up from molecules to organisms encourages creating standalone, portable, reusable components.

Templates

At the template stage, we break our chemistry analogy to get into language that makes more sense to our clients and our final output. Templates consist mostly of groups of organisms stitched together to form pages. It's here where we start to see the design coming together and start seeing things like layout in action.

Templates are very concrete and provide context to all these relatively abstract molecules and organisms. Templates are also where clients start seeing the final design in place. Working with this methodology, templates begin their life as HTML wireframes, but over time increase fidelity to ultimately become the final deliverable.

Pages

Pages are specific instances of templates. Here, placeholder content is replaced with real representative content to give an accurate depiction of what a user will ultimately see. Pages are the highest level of fidelity and because they're the most tangible, it's typically where most people in the process spend most of their time and what most reviews revolve around. The page stage is essential as it's where we test the effectiveness of the design system. Viewing everything in context allows us to loop back to modify our molecules, organisms, and templates to better address the real context of the design.

Approved

Pages are also the place to test variations in templates. For example, you might want to articulate what a headline containing 40 characters looks like, but also demonstrate what 340 characters looks like. What does it look like when a user has one item in their shopping cart versus 10 items with a discount code applied? Again, these specific instances influence how we loop back through and construct our system.

Why Atomic Design

In a lot of ways, this is how we've been doing things all along, even if we haven't been consciously thinking about it in this specific way. Atomic design provides a clear methodology for crafting design systems. Clients and team members are able to better appreciate the concept of design systems by actually seeing the steps laid out in front of them.

It also gives us the ability to traverse from abstract to concrete. Because of this, we can create systems that promote consistency and scalability while simultaneously showing things in their final context. And by assembling rather than deconstructing, we're crafting a system right out of the gate instead of cherry picking patterns after the fact.

Usability

"One picture is worth a thousand words" is an old adage meaning that some complex issues can be better presented with just a single image than with words. People and especially children are also able to absorb large amounts of data quickly through visualization. However there are multiple ways to visualize data and it is important that the visualization is usable and useful for the user.

Since the main goal of the data visualization is to communicate quantitative information clearly and effectively to the user, it is important to keep it simple enough. This means that the visualization should contain only the data that is needed in particular situation. It is also important to understand how people tend to interpret visual images. The key principles followed are:

- Proximity: When items are placed in close proximity, people tend to assume that they are in the same group. Items that are further apart are seen as unrelated or less related.

- Similarity: When items look the same, people perceive them to be of same type. People naturally assume that objects with similar characteristics (similar colour, similar shapes, similar sizes, similar orientation etc.), are related.

- Closure: People's eyes tend to add any missing pieces to a familiar shape.

- Continuation: If people perceive objects as moving to a certain direction, they see them as continuing to move that way.

- Figure/ground: People perceive objects of a graphic as figures (distinct elements of focus) or as ground (the background on which the figure rest). Foreground objects are important and background objects less important. Thus it is important to have enough contrast between the foreground and background.

When planning the data visualization, it is important to rely on those components that users are already familiar with. Thus, it is important to know the context of use and users' workflow, what kind of data visualizations they are used to use. It is also important to use familiar symbols, icons and colours and to be consistent with these.

Colours should be used in visualizations very carefully. They might make a boring graphic look pretty, but they really need to be handled with care. Use colours sparingly and do not use distracting colours. If possible, use a few colours in your visualization. Colours must be double-checked for the colour blind.

Approved

There are some tools which can be used to simulate the effect of different types of colour blindness (e.g. Coblis and Vischeck).

The software platform will be very user-friendly and supportive for all involved actors and especially for children who are the prime actors, as well as time-saving if possible. The best is, if most interactions are self-explaining and appear analogous to frequently used tools of every-day life (e.g. smartphone/tablet etc.). Therefore, the system:

- Provides explanations that are concise, multilingual and understandable for actors like children and teachers who might not be computer literate.

- Have only a few buttons that are more or less self-explaining like “Apps” on a smartphone.

- Visualize as much as possible with illustrative but clear and abstract images and use as less numerical classifications as possible.

- Have a simple and clearly structured surface with necessary but not irrelevant information for the first view. By continuing in different sub-datasets, it must be possible to get more detailed information in a graduated manner to understand the LOCARD full-fledged framework platform.

Approved