Towards an Open, Secure, Decentralized and Coordinated Fog-to-Cloud Management Ecosystem

D3.2 Security and Privacy Aspects for Agent Controller (IT-2)

Project Number	**730929**
Start Date	**01/01/2017**
Duration	**36 months**
Topic	**ICT-06-2016 - Cloud Computing**

Work Package	**WP3, Agent Controller block design and implementation**
Due Date:	*M25*
Submission Date:	*06/02/2019*
Version:	*1.0*
Status	*Final*
Author(s):	*Jens Jensen (STFC)*
Reviewer(s)	*Toni Cortes (BSC)* *Jasenka Dizdarevic (TUBS)*

Keywords
Security, Privacy, Agent Controller, Architecture

Project co-funded by the European Commission within the Seventh Framework Programme		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission)	
RE	Restricted to a group specified by the consortium (including the Commission)	
CO	Confidential, only for members of the consortium (including the Commission)	

This document is issues within the frame and for the purpose of the mF2C project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730929. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	*10/10/2018*	*Initial upload*	*J Jensen (STFC)*
0.2	*15/01/2019*	*Added micro-agent specific security content*	*Francisco Carpio (TUBS)*
0.3	*16/01/2019*	*Overall contribution*	*S Crompton (STFC-UKRI)*
0.4	*22/01/2019*	*Version for internal review*	*J Jensen, S Crompton (STFC)*
0.5	*28/01/2019*	*Addressing comments from BSC*	*J Jensen (STFC)*
0.6	*28/01/2019*	*Review by TUBS*	*J Dizdarevic (TUBS)*
0.7	*31/01/2019*	*Addressed TUBS review*	*J Jensen (STFC)*
1.0	*06/02/2019*	*Quality Check. Deliverable ready for submission*	*María Teresa García (ATOS)*

Table of Contents

List of figures

List of tables

Executive Summary

Many Agent Controller (AC) security components were prototyped for IT-1, even if not all were integrated into the actual release. Building on the experiences with these prototypes, and with the IT-1 release itself, it is now possible to update the plans for the AC security for the rest of the project, taking into account also the feedback from the review. This description of AC security also takes into account the updated requirements in D2.5 [1] and the revised architecture in D2.7 [2]. Specifically, building upon a core transport layer in the fog, the security framework that underpins the agent security is developed, which is then further used to implement the security goals identified by D2.5 [1]. These goals cover the security features of the agent controller that were demonstrated in IT-1, namely discovery, identification and authentication of agents, bootstrapping agent security (connecting them to a fog and giving them credentials), secure communications, and user registration. The goals go beyond previous work by implementing discovery for other protocols, extending bootstrap to multi-fog scenarios, adding authorisation, privacy and GDPR support in user registration, and more. As a part of this process, we investigate supporting technologies that were not integrated in IT-1, or were not investigated at all earlier. These technologies include the emmy Zero-Knowledge prover developed by XLAB, distributed ledger technologies, IPv6 security features (which are "cleaner" than their equivalent features in IPv4), as well as the practicalities of the physical security of edge devices.

It should be clear that the work presented here, and in the associated Platform Manager security deliverable (D4.2 [3]), combines to form a very ambitious workplan, not all of which will be achievable in the remaining year of the project. However, by documenting the full range of the proposed security implementation, it should be possible to then later make informed decisions about the priorities of the individual security tasks. For example, we can decide to implement very minimal fogs, with agents living in essentially only connectivity with each other, and with access only to the Control Area Unit (CAU) gateway and any leader agent(s) that may be present. It will equally be possible to implement a more extensive fog, that enables devices - once authenticated - to access the wider Internet and, through cloud-hosted information hubs, can share information with agents and devices in other fogs. The AC security provides the foundation for building fogs and edge-to-fog and edge-to-cloud applications.

1. Introduction

1.1. Purpose

This deliverable provides a description of the technology and plans for the implementation of the agent controller security in IT-2. The deliverable D4.2 [3] provides the same description for the platform manager. Other relevant deliverables are D2.5 [1] (security and privacy requirements for IT-2) and D2.7 [2] (architecture for IT-2.) In order to avoid overlapping with D2.5 [1] and D2.7 [2], this deliverable, D3.2, and D4.2 [3] focus on the proposed implementation. Action items focus on things that should be investigated and resolved in IT-2 (as opposed to open research questions) and are identified throughout as "ACx" where x is a number and are summarised in the conclusion.

To be clear, the earlier deliverables D3.1 [4] and D4.1 [5] remain valid, so we do not repeat information here that was covered in those two deliverables.

1.2. Structure of the document

The document is structured into 5 sections:

Section 1 (this section) is the introduction.

Section 2 provides a brief overview of the status of AC security in IT-1.

Section 3 describes our background interpretation of a fog, the underlying technologies of the proposed AC security implementation.

Section 4 describes the proposed AC security implementation.

Section 5 provides the conclusion, which also contains a summary of the action points arising from this deliverable.

1.3. Glossary of Acronyms

Acronym	Definition
AC	Agent Controller
API	Application Programming Interface
CA	Certification Authority
CAU	Control Area Unit
GDPR	General Data Protection Regulation
DHCP	Dynamic Host Configuration Protocol
DLT	Distributed Ledger Technology
DNS	Domain Name System
HTTP	Hypertext Transfer Protocol
IIoT	Industrial Internet of Things
IP	Internet Protocol
ISP	Internet Service Provider
IT	ITeration
JSON	JavaScript Object Notation
OSI	Open Systems Interconnection (ISO/IEC 7498-1:1994)
PII	Personally Identifiable Information (GDPR)
PKI	Public Key Infrastructure
PM	Platform Manager
PoC	Proof of Concept
REST	Representational State Transfer
SSID	Service Set Identifier (Wi-fi networks, 802.11 standard)
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol

Acronym	Definition
UDP	User Datagram Protocol

Table 1. Acronyms

2. Agent Controller Security in IT-1

As described in section 2.1 of D2.5 [1], a number of security components were developed for IT-1, and some, but not all, were integrated into the actual IT-1 release. This section briefly recapitulates the status of those components that fall under the scope of the AC, and briefly discusses the limitations identified in IT-1.

2.1. Review of Security software implemented in IT-1

The security software developed for IT-1 was:

1. A portal-based user registration process, requiring users to register with email addresses and create a password. Although the portal was secured with a browser-friendly certificate, there was no further support for privacy or GDPR. This is discussed further in section 4.6.2.
2. A communications model employing JSON security was prototyped, the so-called "AC library." This development is continued, although it now splits into CAU client and CIMI security, as well as facilities for the core communications; the latter being described in section 4.7.2.
3. The bootstrap process assumed Wi-Fi only as a communications protocol, so was able to use the Wi-Fi protocol as a beacon. However, this did not seem to work with the virtual environment that was developed for security testing. We discuss this further in section 4.1.
4. The same security testing environment was essentially a virtualbox type environment, but with support for wireless through OpenWRT[1]. At this stage, it seems most likely we must drop this component for IT-2, due to the significant effort that would be required to finalise the work. Instead, we propose to investigate physical security of devices (section 4.7.4), as this seems more likely to be useful to the use cases.
5. The active firewall demonstration that was developed but not integrated for IT-1 may need to be given a lower priority, as it now depends on developing machine learning for incident detection first, and the firewalling would then be redeveloped as a part of an automated active response to an incident. See D2.5 [1], section 4.2.1 (p.23). Machine learning for security is covered in D4.2 [3], section 4.6.3.
6. emmy was developed already before IT-1, and development is ongoing; the plans for use of emmy are described in section 4.5.1.
7. The Certification Authority (CA) prototyped in IT-1 (supporting two or three PKIs) was not sufficiently stable in its containerised form and has already been re-engineered for IT-2; see section 4.4.

2.2. Security postponed from IT-1

1. Privacy and GDPR support in particular were considered out of scope for IT-1. These are covered in section 4.5.2.
2. Application security - related to the software developed for the use cases - was also considered out of scope. This is discussed in section 4.7.
3. Physical security of devices was considered out of scope not just for IT-1, but potentially for the whole project. However, given industry developments during the mF2C project, it is now feasible for us to do some exploratory work of industry solutions and other options, see section 4.6.4and D2.5 [1] section 2.2.

[1] https://openwrt.org/

3. Security Technologies for mF2C Agent Controller Components IT-2

In this section, we describe the technologies that have been identified as relevant to meeting the security goals for the AC components. The technologies may not be new, but they are described from the point of view of what we have learned from IT-1. The security goals were described in D2.5 [1].

Other related technologies are described in D4.2 [3], those proposed primarily for the PM. As a general principle, the aim is to use open standards with interoperable implementations, as opposed to proprietary or ad-hoc solutions and de-facto "standards."

Before we dig into the technologies, however, it will be useful to quickly recap what the AC is, and what the components of the AC are. Again learning from experiences with IT-1, D2.7 [2] revised the components of the (full) agent; for the reader's convenience, we replicate the diagram here (Fig. 1). Other than the AC components - Resource Manager, User Management - it should be noted that the Security Layer is also covered by this deliverable, because, historically, the CAU client and the AC library have been developed as parts of the AC.

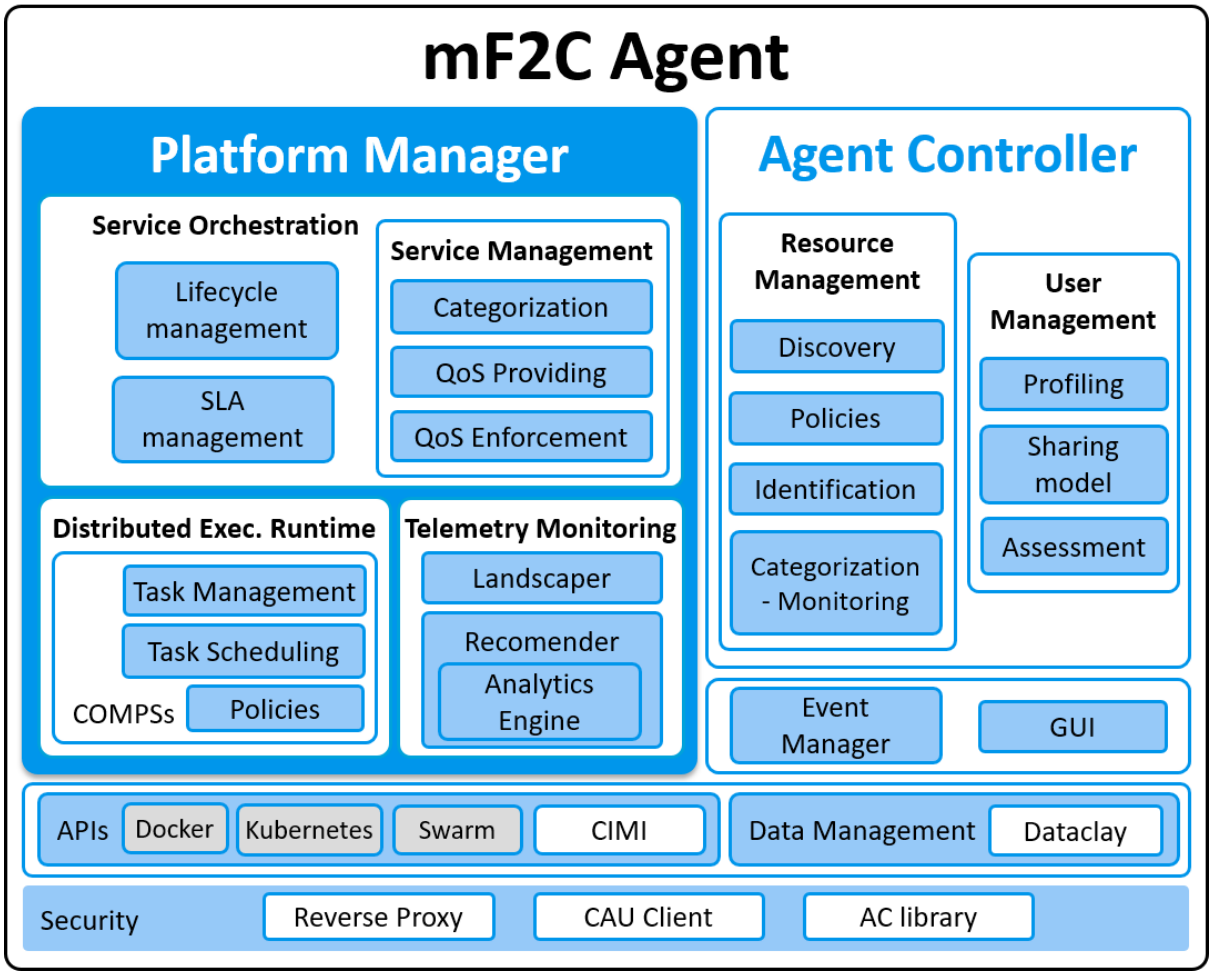

Figure 1 Updated architecture of a full mF2C Agent (see D2.7 [2])

3.1. Securing the Fog

In order to secure "the fog," it is necessary to establish what the minimal fog service is, and how it works. As described in D2.5 [1], a fog is required to have a transport layer (in the sense of the OSI stack [6]; these network layers should not be confused with the mF2C architecture layers), but without direct access to the Internet/cloud for devices that have not authenticated with an mF2C credential

and have been authorised to access Layer 0 services (D2.7 [2], Section 2.1). The transport layer is often (in mF2C) assumed to be Wi-Fi, in order to provide beacon functionality, but for the purposes of this section, we need not constrain the technology to Wi-Fi (see section 4.1 for an updated discussion of the beacon feature of Wi-Fi.)

It follows that there must be a discovery process whereby a new device connects to the transport layer - which must be sufficiently secured to prevent imposters from hijacking connections. The outcome of this process must be a device connected at the transport layer to an established Leader agent. There must also be a bootstrap process, whereby a new device connects to the fog and obtains the relevant credentials (and authorisations) - this is the subject of section 4.3.

The closest analogy of a fog (as outlined here) in everyday terms is a home wifi: a wifi network is available and a new device discovers it, either by SSID (if the SSID is advertised) or by having the SSID configured. As it connects, it goes through a process of authenticating to the wifi router (on a home network typically using a shared secret) and obtaining an IP address. Once connected, the router forwards packets from the device to the Internet, yet the device is typically hosted on a private IP address and is hidden from devices on the Internet; any response to a request from the device is addressed to the router, and the router forwards the response to the intended recipient.

This analogy is useful not just to compare mF2C fog security with a familiar scenario but also to set the challenge for improving on it. The mF2C fog must scale to larger numbers of devices, and must have a connection process with less overhead. To investigate both of these further, the natural question is whether we can simplify the process. For example, a home router always provides a DHCP server, but we claim there is no requirement that a fog must have DHCP (see Annex 1 for details). Similarly, a home router enables the client to add a DNS server address to its network configuration; the DNS server is usually provided by the ISP. However, there is no requirement for DNS either, as other, simpler, discovery mechanisms are available (such as Neighbor Discovery in IPv6). For a more extensive discussion of the network capabilities of a fog transport layer - below the agents - please refer to Annex 1.

It should be emphasised, however, that we do not require that a fog be the minimal possible; we are just seeking to identify the leanest fog possible. A fog deployment may have publicly routable IP addresses, with DHCP and DNS services, and where agents have independent means of connecting to the cloud. Some applications may require these features. The purpose of the discussion outlined here is merely to understand the minimal requirements for mF2C.

Two parts of the analogy remain: authentication and authorisation. Authentication is based on certificates which we discuss in the following section. The bootstrap process, by which the agent obtains a certificate, and the authorisation, are discussed in D4.2 [3], Section 3.4.3.1.

AC1 Investigate IPv6 security features for secure discovery.

3.2. X.509 Certificate and Agent Identity

As demonstrated in IT-1, the fog uses a Public Key Infrastructure (PKI) to manage the identities of the agents and to help bootstrap trust within mF2C. Likewise, certificates are used to authenticate infrastructure services, e.g. CAUs.

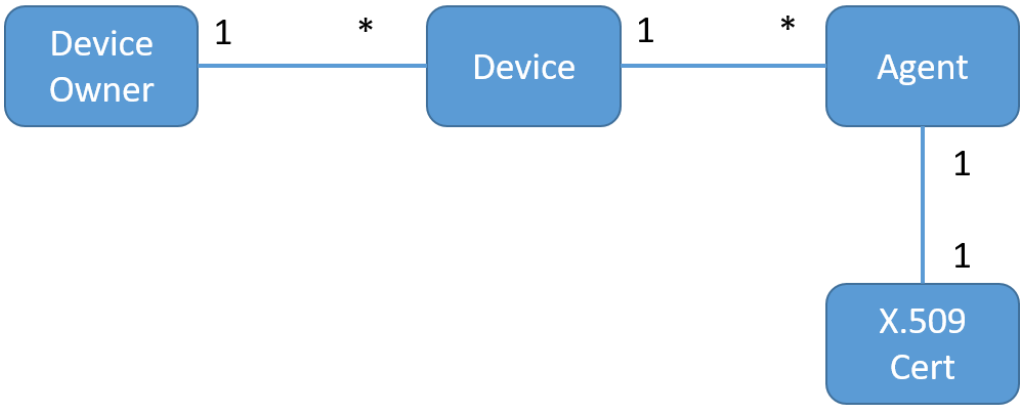

Figure 2 mF2C credential mode

Figure 2 shows a entity-relations model of the device and the Agent credential. The credentials are tied indirectly to the device via an Agent, but not directly to the user principal (Device Owner). The user may own more than one device running mF2C, and each device may run more than one agent. In a scenario when we have two Agents trying to communicate with each other, we need to authenticate the requesting Agent, but not involving the Device Owner[2].

In IT-1, an Agent got issued an X.509 certificate via the CAU by the "untrust" CA (see Section 4.4) upon joining a fog cluster. The certificate is the credential that uniquely identifies an agent in mF2C.

We envisage that each fog cluster led by an mF2C Leader Agent would be supported by a local, geographically proximate CAU (see also Section 4.2) to avoid round-tripping to the cloud CAU for authentication and authorization. Put in another way, each CAU manages a distinct control area and holds the associated security information for the fog clusters within its area. In this way, the hierarchy of CAUs interacting together could share the security information of all agents under the control of the hierarchy (see [7] and section 4.3). The hierarchy of CAUs together maintain a global view of the agents' identities to facilitate trust and to enable agent mobility (between fog instances). If an Agent moves between control areas, it would be possible for the new control area to validate the Agent's X.509 credential through querying the appropriate CAU either directly or via the Cloud CAU.

To reiterate, an individual CAU serves the role of local key manager for the agents that it services. In a highly distributed F2C environment, this set-up facilitates low-latency authorisation decisions when speed is paramount - speed could mean life and death in IoT applications such as health monitors [8] . In D4.2 [3] Section 3.4.3.1, we describe a workflow for securing inter-agent communication in which the CAUs deliver critical identity management and token introspection services. Apart from identity, as per the policy-driven security design proposed in D3.1 [4], the certificate could also be used for encrypting and signing messages (data in flight) between agents to ensure data confidentiality, integrity and authenticity (cf. D3.1 [4]).

In IT1 our focus was to deliver an integrated, functional agent and the hello-world demo stopped at the new agent exchanging (certificate) keys with its leader agent and did not illustrate how the certificates would be used. In IT-2, we move on to exploit the certificate for authenticating and authorising communication between Agents (see D4.2 [3]) and, where appropriate, to secure data in-flight.

AC2 Ensure the agent credential is used as the foundation of all the security goals required of the agent.

[2] There will be occasions where it does make sense to involve the Device Owner; see D4.2, section 4.1.3.

3.3. Blockchain

Blockchain is a Distributed Ledger Technology (DLT) that supports the secure storage of data blocks in a decentralised, peer-to-peer network. DLT's inherent security features have attracted many to consider the technology as a fitting solution for securing the highly distributed fog ecosystem where trust is lacking between peers. In this Section, we review the key security mechanisms in DLT that could be exploited to secure the mF2C platform.

A blockchain may be described as a distributed data store and it was launched in 2009 as the ledger system for the Bitcoin virtual currency. The key advantages of blockchain over other distributed storage systems include:

- immutable, tamper-proof chain fabric
- built-in data confidentiality and integrity through consistent use of cryptography
- a single view of the chain state via consensus mechanisms; which fosters trust and transparency in the transactions
- cost-effective and sustainable; a blockchain is maintained by peers, not a single authority.

A blockchain's immutable fabric is the result of explicitly chaining blocks of data together using their fingerprints. A block in a blockchain is a data container that holds an ordered set of data (transactions). Blockchain uses cryptography to digitally fingerprints the transactions and each block is then chained to the previous one via embedding the hash of the previous block header. This arrangement allows explicit ordering between the blocks and, more importantly, the transactions are appended only, as any changes or deletions would break the digital fingerprint and the consistency of the chain.

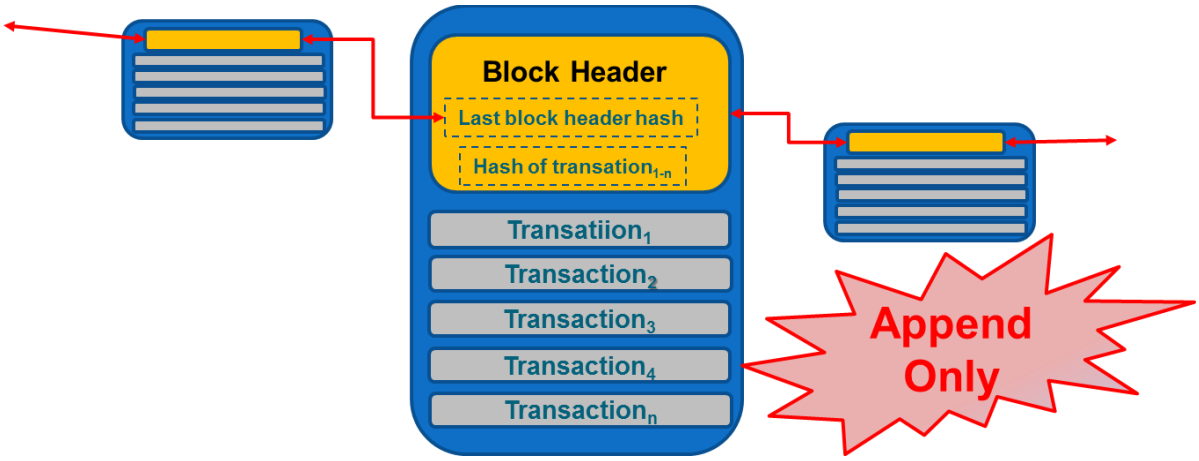

Figure 3 Anatomy of a Blockchain

Since 2013-4, the blockchain technology has evolved rapidly from being a public only transaction ledger operating on the compute-intensive Proof-of-work consensus algorithm into a distributed ledger protocol suitable for the enterprise environment. Unlike the Bitcoin blockchain, DLT can be used to implement private, permissioned blockchains without a cryptocurrency aspect for known registered members. DLT also supports a variety of light-weight consensus protocols, including Proof-of-authority which allows authorised participant/s to validate transactions. DLT is being promoted for automating trusted business processes in various domains, in particular M2M transactions in the supply chains, healthcare, insurance, transportation, voting, contract management and more. For mF2C, we are experimenting with DLT in IT-2 to deliver most of the security features required of the CAU security middleware (see Section 4.2):

- peer network; building in redundancy and availability of security data, such as Agent identities and fog membership
- near-real time replication of security data across the CAU network to ensure availability to support low latency decision making
- safeguarding the confidentiality of security data through access control and cryptography
- protecting the integrity of security data via its tamper-proof fabric
- ensuring data accountability through its immutable data structure
- using automated smart contract to enforce user-defined policies at transaction time, e.g. controlling access, triggering notifications, revoke credentials, etc.

4. Security for Agent Controller - Implementation

This section is the main section of the deliverable: it describes the proposed security implementation including:

- what we want to improve on IT-1
- the items we postponed from IT-1 which are now back in scope,
- mechanisms we have investigated as solutions for IT-2, but may not end up in the release due to resource constraint.

This section covers the components of the AC block, namely, the discovery and leader selection processes, the CAU, the CA service in the cloud, and the CAU client and AC library which, although not part of the AC in Fig. 1, are historically considered as belonging to WP3. Importantly, we need only describe changes that are proposed for IT-2. In IT-1, many components were secured simply by relying on Docker private networks, i.e. the containers were deployed on the same physical host, so the communication between the containers would never leave the host and therefore not need any special security considerations. This practice will continue also in IT-2, and need not be described in details here.

4.1. Leader Discovery and Leader Selection

To enable discovery in mF2C, leaders periodically broadcast custom Wi-Fi beacons. As a result, when a regular (not currently a leader) agent in the vicinity scans for these beacons, it is able to discover the leader/s. As soon as a leader is detected, the authentication process involving the CAU client, the CAU and the (leader) CAU is triggered (see Section 4.2).

The whole discovery process, along with categorization and authentication is represented in Figure 4, slightly modifying the discovery process previously designed for IT-1 and previously reported in D5.1 [9]. The main changes are:

- Periodic scanning process to guarantee a leader is selected when available (message 2).
- Re-scanning process to guarantee a leader is found when the previously selected is down (message 12).
- Two new parameters are added in messages 3.1.1 and 7.2.1, Auth and Conn (boleans) representing whether a device is authenticated and connected to the leader respectively.
- Added message 9 to store the agent information in CIMI.
- Numbering for former messages 9 and 10 are now 10 and 11.

In order to be able to send beacons, an Agent must be acting as a Leader. This process can be initiated in two ways. First, the leader election protocol is the procedure to select a leader when there is no leader in the vicinity or the area is reorganized. Second, the leader protection process is responsible to keep always a leader in the area until there are no more agents capable of being leader.

The leader election is performed following two sets of requirements fixed by policies, where the agents must follow some mandatory characteristics and be sorted by the optional characteristics. This procedure can take place in different agents at the same time when there is not an active leader in the area or triggered by the leader if a reorganization is necessary.

When an agent discovers a leader and enters into the area, or a new leader is selected, the leader protection process is turned on. This involves selecting a backup if the agent is acting as active leader or waiting to be elected if the agent is capable to be a leader. This process requires continuous communication between the backup and the leader to ensure that both are up and running correctly. As soon as the leader is detected as down by the backup agent, a process to promote the backup as the new leader and to select a new backup is started.

AC3 Implement Leader election.

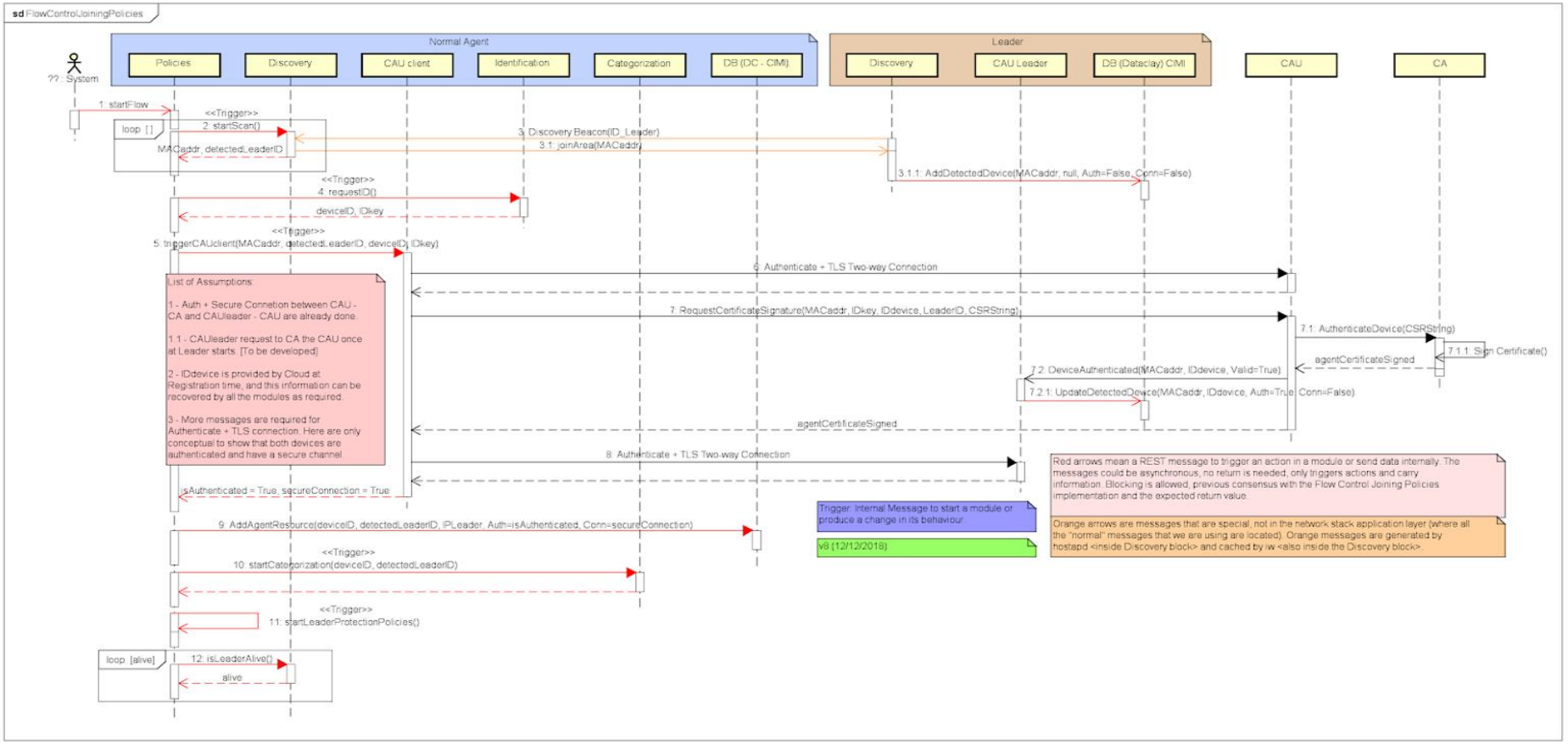

Figure 4 IT-1 Discovery, Authentication (Security) and Categorization Workflow

4.2. Control Area Unit and Certification Authority

Figure 4 shows the decoupled and distributed CAU infrastructure which acts as a smart gateway and bridge between the cloud CA and mF2C agents within the context of the IT1 workflow for providing certificates to agents. The CAU infrastructure is organised hierarchically under a "Cloud CAU" (see D2.7 [2], section 2.5.4, and specifically the diagram). This cloud CAU acts as the master controller and has a global view of the security information managed by its subordinate CAUs. It serves as a fall-back CAU that a candidate Agent or regular Agent can use if it cannot locate a local CAU and if it has internet connectivity; either its own (for the candidate Agent) or provided by mF2C (for the regular Agent). See Section 4.3.1 for a discussion on CAU discovery.

In IT-2, we aim to use the CAU infrastructure as a middleware to provide more functionalities to enhance mf2c security. As mentioned in Section 3.3 above, we are exploring DLT for providing trust in the mf2c infrastructure. We envisage that each distributed CAU would act as the gatekeeper for its corresponding fog area and help new agents to join the mF2C blockchain network. As illustrated in Figure 5 below, each CAU would become a node in the proposed blockchain network. Each node will store Agent identity and fog membership information in its immutable fabric. The chain data is replicated in near-real time to other nodes on the mF2C blockchain network to facilitate identity management, trust establishment, and the confidentiality, integrity and auditability of the security information [10].

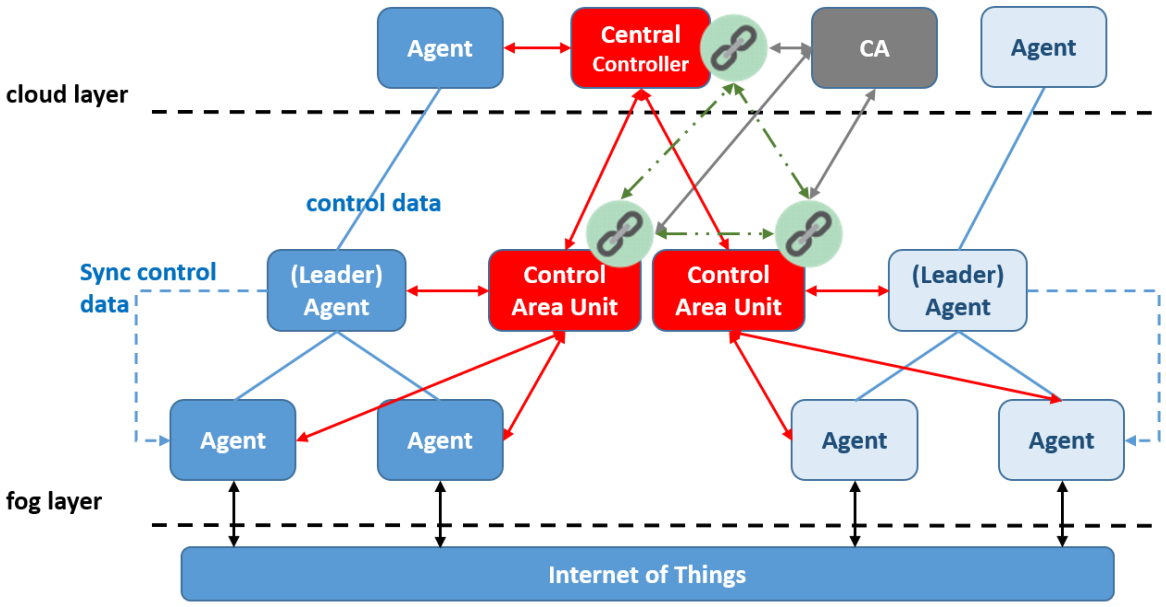

Figure 5 Bolting on blockchain to enhance trust in the mF2C infrastructure

We propose to use IBM Hyperledger Fabric[3] framework to implement the CAU blockchain network. Fabric offers the key features that we require:

- supports private, permissioned blockchain network as we need the CAUs to be identifiable.
- pluggable consensus algorithms, including light-weight Power-of-authority protocol.
- a Membership Service Provider that uses X.509 certificates to identify participants.
- channels for grouping participants (or mF2C fog clusters) to enforce access control and privacy.

[3] https://www.ibm.com/blogs/blockchain/2018/08/hyperledger-fabric-what-you-need-to-know-about-the-framework-that-powers-ibm-blockchain/

- open source product distributed under Apache License, Version 2.0 and has an active user community.

AC4 The CA needs re-engineering for IT-2.

AC5 Implement Cloud CAU synchronisation with fog-local CAUs

4.3. Control Area Unit Client

The CAU Client is an integral Agent block and works as a proxy to the loosely-coupled CAU infrastructure. An Agent invokes the CAU interface locally to access external CAU security functionalities to:

- manage its own identity, e.g. request/renew certificate.
- request security tokens to access resources external to itself, e.g. another Agent, a third-party entity, etc.
- authenticate the identity of another agent requesting access to its resources.
- retrieve the public key of another Agent in the Fog to encrypt message to the the Agent, etc.

As part of the Agent, the CAU Client is trusted by other Agent blocks and they communicate over trusted, private Docker network. In contrast, the CAU is external to the Agent and there are several implementation issues that need to addressed:

- CAU discovery (described in 4.3.1, below)
- ensure that a CAU is trustworthy and is not an imposter (4.3.2)
- safeguarding communication between the CAU Client and the CAU (4.3.3)

It should be noted though these are only relevant for the original (IT-1) CAU deployment without Fabric, as the framework has built-in support for service discovery, trust and security. As it is our plan to take a progressive approach towards developing the CAU, starting from the original version to a more sophisticated version based on Fabric. For the purpose of completeness, we outlined some simple strategies for addressing these concerns.

AC6 Demonstrate CAU client functionality for microagents

4.3.1. Control Area Unit Discovery

In the IT1 workflow, we described the scenario of a new Agent joining an existing fog cluster and the location of the participating components, e.g. CAU, leader Agent, are already known. The CAU was discovered on a fixed IP address which is a perfectly satisfactory option also for IT-2. The local CAU is pre-deployed - without it, there would be no mF2C fog - and it makes sense to use a fixed IP address, particularly if the same address is used in each fog, cf. the discussion in section 3.1. Other discovery methods for local services could be examined, depending on whether IPv6 (as proposed for IT-2) or IPv4 (as used in IT-1) were used, as discussed briefly in section 3.1, but it is not essential if the simpler option works well.

Even a possible CAU in the cloud could be hosted on a fixed IP address provided packets could be routed to it out of the local network (through a local CAU, for example, cf. D2.7 [2] section 5.2.) There are different ways (involving different degrees of complexity) to support cloud CAU discovery. From the simplest, i.e. using a fixed address, and sufficient for a PoC, we could bundle the CAU Client with a default endpoint containing a fully qualified host-name which the cloud CAU needs to support (provided DNS is available, of course). More robust solutions fit for production use would be to

implement a service registry, e.g. Netflix Eureka[4], Apache Zookeeper[5], or an internet/-facing load balancer like Nginx[6], to forward requests to the cloud CAU instance. The distributed CAUs infrastructure are known and discovered through their master in the cloud, i.e., CAU-cloud. In the case the leader is not available for broadcasting, the agent arriving to the area might be leader capable and also have connectivity, therefore, it can discover CAU-cloud to get certificate.

4.3.2. Establishing Trust

We can leverage the mF2C PKI trust model to facilitate trust between the CAU Client and a CAU. In Section 4.4 we describe the two trust hierarchies used in mF2C: untrust and fog. CAUs, as an mF2C infrastructure components are certified by the Fog CA. As long as the CAU client trusts the Fog CA, through the PKI chain of trust model, it would also trust certificates (viz. the subject-named owner of the certificate) issued by the Fog CA or its subordinate CAs. In IT1 we exploit this model and bundle each CAU Client with the public key of the Fog CA which the CAU Client uses to establish a two-way (i.e. authentication required on both the client and server side) SSL handshake with the CAU (see Figure 4).

4.3.3. Securing Communication

In the previous section, we already mentioned using the mF2c X.509 certificates to bootstrap trust between the CAU Client and CAU. The SSL handshake initiates a secure endpoint to endpoint session between the two with messages between the two encrypted with the keys associated with the certificates.

4.4. Certification Authority

In an PKI, the Certification Authority (CA) issues X.509 certificates to certify the ownership of a public key by the named subject in the certificate. Others who trust the CA can also trust the certificate and in cryptography this mainly concerns the ability to rely on the signatures or assertions made about the private key that corresponds to the certified public key. In mF2C, an Agent's X.509 certificate is used for identification and for access control (Authentication and Authorisation) as well for securing message confidentiality, integrity and authenticity. In Section 3.2, we already mentioned that Agent certificates are issued by the "untrust" CA; the term reflects on the fact that mF2c cannot be 100% certain that a potential agent is trustworthy, i.e. they are who they claim they are. We needed, therefore, a separate trust hierarchy for entities that mF2C trusts: the infrastructure components such as the CAU deployed by trusted third parties.

In line with this design, we deployed two CAs: "untrust" and "fog" CAs using the DogTag Certification System[7] for the IT1 demo. The former is responsible for issuing certificates to the mF2c infrastructure components and the latter to the Agents. The two CAs will have different policies and certificate practices. In particular, the fog CA, supporting the infrastructure, can have long-lived (1-3 years) certificates with revocation lists; whereas the "untrust" CA is likely to have shorter lived certificates (say, a week) with no revocation.

In IT-2 we will most likely need a CA-as-a-Service. The reason for this was described in D2.5 [1], sections 3.3.1 and 3.3.4 - that we may ultimately need not just the two PKIs described above but also application-specific PKIs. To support these requirements, we have coded our own simple CA services[8] using the Bouncycastle Crypto Library[9]. This application replaces the Dogtag services mentioned above. We found the Dogtag software to be unreliable when deployed as Docker containers. Each

[4] https://github.com/Netflix/eureka

[5] http://zookeeper.apache.org/

[6] https://www.nginx.com/products/nginx/load-balancing/

[7] https://www.dogtagpki.org/wiki/PKI_Main_Page

[8] https://github.com/mF2C/certauth

[9] http://www.bouncycastle.org/

time the CA service restart, it defaults to sign certificates using a new private key. This means that existing certificates need to be replaced as the issuer's key has changed which is not an ideal situation. Note also D4.2 [3] which identifies a CA in the fog for Use Case 2 (D4.2 [3], Annex 3).

AC4 Re-engineer CA.

4.5. End User Authentication

In this section, we look at the two options for having end users authenticate to an mF2C platform; such an authentication would typically happen to the registration portal as in IT-1, but could also be to an OAuth2 authorisation server (D4.2 [3], section 4.1.3.1). Both of these options have been explored earlier, and we need only describe how they are expected to be changed or applied differently for IT-2.

4.5.1. Zero-knowledge proofs - *emmy*

Management of users and their assets is an important part of large systems. In this section, we introduce the needs for zero-knowledge proof algorithms that, instead of being integrated into mF2C directly, but rather demonstrated inside the use case as a proof of concept that could be expanded to the whole mF2C platform in the future.

Distributed computer systems need to build trust between the devices, applications and users. Trust is a basic value on which a market of resource sharing and paying for resources can be created. In fog-to-cloud, the infrastructure environment is even more complex, comprising of large and powerful clouds and small devices that could be owned and managed by a large group of users and integrated in untrusted physical environments. One part of the trust is maintained with the certification authorities which, to some extent, proves that a device is truly a valid mF2C device. This part covers CA/CAUs that are described in Section 4.2.

The second interesting topic is how to provide a mechanism to prove only a part of information from a device, user or organisation without unveiling the information that is not truly required for finishing the process. This security aspect is also very interesting from the GDPR perspective. For example, in the future of Fog and Cloud computing, a user will pay for resources in mF2C. A particular device that will fulfill user requirements needs a proof of purchase and also related attributes (the amount of resources, etc), but may not need to have access to the full profile of the user. For this and similar problems a mechanism based on zero-knowledge proofs with functional encryption can be adopted. The emmy library[10] implements zero-knowledge proof algorithms and will be used in the second mF2C use-case for anonymously proving the payment of anchoring and docking for boats, without the marina owner having any information about the client. When mF2C will be able to support payments for particular resources, similar mechanisms will be useful for proving the lease of infrastructural resources.

4.5.2. User registration and GDPR implementations

The main GDPR aspects related to mF2C have been recently covered in Deliverable D2.5 [1], section 3.4. This section analyses what GDPR stands for and how GDPR may affect an mF2C deployment. In short, the mF2C project as such must guarantee that the main GDPR pillars will be successfully supported by a potential mF2C deployment, that is:

- Lawfulness, fairness and transparency, ensuring the users' rights about consent and object;
- No repurposing of data in the context of the project, collecting only that data required for the processing purpose;
- Ensure accuracy, up-to-dateness and users' rights to correct data and to be forgotten;

[10] https://github.com/xlab-si/emmy

- Grant confidentially, integrity, accountability and security of all processed data.
- All personal and private user's data must keep private without any leakages (privacy is a must in the mf2c project).

GDPR deals with Personal Identifiable Information (PII), and it is up to the data controller to implement strategies to ensure that sensitive data is not leaked. We need to consider two cases. The first case covers the data collected in a deployment of the mF2C use cases, in particular use cases 2 and 3 where end users interact directly with applications deployed on mF2C. The individual use case "owners" have categorised their data protection requirements in D4.1 [5], section 3, with more details provided in the Annex sections of the same deliverable. As the fundamental data flows of the use cases have not changed since IT-1, it is to be expected that the associated data security requirements have not changed. In this first scenario, the main action points identified are:

- Applications must have a means to adequately protect user data. Although applications are out of scope of this deliverable, applications classify data in the same way as the rest of mF2C and could reuse mF2C services (DataClay in particular, cf. D4.2 [3], section 4.2) or other code or functionalities to implement their data security (cf. section 4.8).
- In particular, message level security should be considered - i.e. the ability to relay messages through a third party, cf. D4.2 [3], Annex 3 (specifically Fig. 13)

The second case refers to the data collected at the user registration process described in D2.7 [2], section 3.5. These data should include all information required to register and profile a new mF2C user, what obviously may turn into highly sensitive data (for example, banking data to manage payments, or profiling data to accommodate specific mF2C demands to mF2C services or execution processes). However, assuming the registration process is handled at cloud level, this sensitive data will not be exposed to lower levels, so removing the need for deploying specific data privacy strategies at these levels. Certainly, we assume cloud to deploy existing strategies to make the required data to be conveyed to lower levels not to be identifiable.

AC7 Implement account deletion and user consent management, including consent withdrawal.

4.6. Edge/fog devices security

This section describes the options for securing devices at both edge and fog levels. As mentioned in the introduction to section 4, many container-to-container communications were (and are still) expected to use private Docker networks, so the first subsection here covers the communication when containers (or, more generally, components) need to be reached over a network (such as a fog). The second subsection looks at an earlier attempt to implement the mF2C security protocol through message-based security, the so-called "AC library." We then look at microagents (cf. D2.7 [2], section 4) from the security point of view, and finally, as an extension of this perspective, and following up on section 2.2.2 of D2.5 [1], we propose an investigation into the physical security of the edge devices themselves.

4.6.1. Reverse Proxy

Due to the decoupled microservice-based architecture in mF2C, multiple services end up running in parallel, and have to communicate between each other and also with the outside world through their own APIs. As mentioned in D5.1 [9], mF2C has adopted RESTful interfaces as the common approach for delivering the components' APIs. This convention works perfectly well when the inter-component communications take place locally, within the agent. But the inter-agent communication this would imply that every microservice would need to publicly expose their APIs in order to be reachable from outside the agent. Such an operation would require securing all the exposed REST interfaces, plus, using non-production and unreserved TCP/UDP ports for splitting the different API endpoints.

To avoid such cumbersome security duplication and in favour of best practices, we've chosen to deploy a reverse proxy at the top of the software stack, which liases and routes the communications to and from outside the agent, to the mF2C components running within.

The chosen implementation is called Traefik[11] and it's a well-known and widely adopted open source project. With it, we can route all the different internal API endpoints into unique path prefixes in the mF2C agent's main endpoint, on port 443 (HTTP protocol over TLS/SSL). Traefik also provides automatic redirection of all HTTP traffic to port 443, which can also be configured to have trusted SSL certificates from known Certificate Authorities like Let'sEncrypt (given that the mF2C agent has a fully qualified DNS).

Finally, Traefik also allows for an easy configuration of security header on a per component-basis, making it simple to maintain testing and production versions of the mF2C Agent software, with more or less security embedded.

4.6.2. Agent Controller Library and API for integration

In D3.1 [4], we defined a data security policy to support data security by design. The policy mandates that data be classified into precisely one of three classes of data:

- Public data needs no special protection
- Protected data is integrity protected, with origin authentication
- Private data is like protected data, but additionally provides confidentiality protection.

To address this requirement, we developed in IT1 a PoC AC library[12] [4] (not demonstrated) for handling message encryption using the keys associated with an Agent's X.509 certificate. Using the library, the sender selects the communication protocol to use and sets the required level of protection as per the data security policy described in D3.1 [4]. The security requirements are bundled as metadata with the message. At the recipient's end, the recipient uses its local security library to process the message; verifying signatures and decrypting private content as required using the recipient's key.

In the IT2 Agent architecture (Figure 1), the library has now been incorporated into the Security block and need to be updated to facilitate integration with the other Agent blocks. We envisaged that the AC library would be used to provide functionalities for signing and en/de-crypt message tokens (see Figure 6).

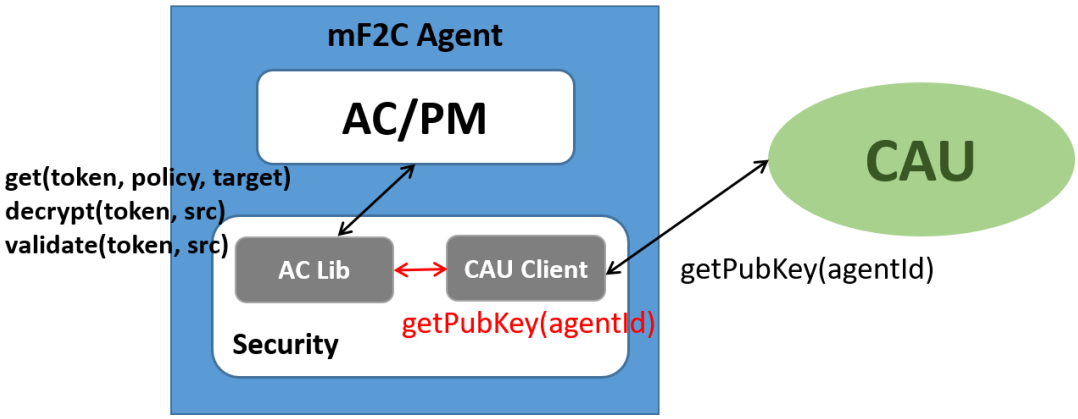

Figure 6 Usage Overview of the AC Library

AC8 Re-engineer AC library to support updated communications requirements - sockets, token based.

[11] https://traefik.io/

[12] https://github.com/mF2C/secLib-java

4.6.3. Microagent specific security

Because of the constrained capabilities of devices running as microagents, some limitations are also considered in terms of security. There are some approaches to extend security capabilities from the agent to the microagent:

1. A pared-down Docker environment is deployed on the less capable device, with containers only for the necessary security modules.
2. The security software is rebuilt for the restricted environment of the device running the microagent.
3. The security code is rewritten/re-engineered for the natural programming environment of the device.
4. Security soft/hardware is used to assist the device. For device capable of there, it could be used to assert device identity and sign communications.

The main objective is to find the way of securing the less capable device in order to secure the received sensor data. In our architecture, the microagent will be responsible of securing this data, since will be the one usually connected to other smaller sensor devices. Therefore, the minimum set of security functionalities for IT-2 in the microagent are:

1. Security bootstrap: as the agent may connect to more than one fog, it could be pre-configured with a certificate rather than bootstrapped through the CAU, but must have the ability to connect and discover the leader.
2. Basic cryptographic functionality: encrypt, decrypt, sign, validate signatures.
3. In particular, microagents are required to be able to handle their own certificate/private key, so they can authenticate to higher layers, and the messages they send can be signed in order to implement data origin authentication.

4.6.4. Physical security

The mF2C platform encompasses cloud and fog resources where some of the hardware is in the physical domain of a particular user. Having this in mind it is impossible to provide full and trustful physical security of the system as a whole, but there are mechanisms to detect physical tampering of particular parts of the system.

Tamper detection relies on sensors that detect whether a device has been opened or modified. For devices in the domain of a company that issues devices with mF2C preinstalled, this could be implemented. For other off-the-shelf devices that were bought by individuals and integrated into mF2C, there is no reliable mechanism to detect tampering.

4.7. Applications

Applications will have the choice to entirely implement their own application security, or to build upon the code provided for the AC security. For the former, it is not for this deliverable to dictate how the work is done, but just to note that to “break out” of the fog, an application will need access to the Internet and the cloud services hosted therein. For mF2C agents, this process is initially gatewayed through the CAU, as an initially untrusted agent cannot be given access to the Internet. However, it would seem likely in Use Cases 2 and 3, that the user would be able to obtain authorisation for their applications to access the Internet. Equally, this process could be implemented based on the technologies developed by mF2C, provided that it is one of the core underlying technologies described in section 3 of this deliverable or D4.2 [3] (i.e. it is a socket, it is a REST web service, or it can implement security based on X.509 certificates directly.)

If using certificates, applications would need a separate PKI to secure their communications: if the applications PKI were compromised, it would not compromise the fog or infrastructure PKI. (Note: by “compromise” we don’t mean that the CA’s key is compromised, but rather that the main threat is

that end entity certificates are compromised and used to maliciously manipulate the system). It is also noted that Use Case 2 requires a CA in the fog, cf. D4.2 [3], Annex 3.

Note that it is in the most general case not necessarily possible to address a device or agent directly in another fog cluster, unless (a) this information is aggregated in the proposed cloud CAU (remembering which agent lives in which fog), and (b) an address translation mechanism is available at the gateway to the fog, presumably facilitated by an agent, that can translate the global address to one that can be routed within the fog. Alternatively, our proposed fog implementation does not preclude a fog implemented with full globally addressable IP addresses and perhaps DNS resolution; in this case, though, it would still be necessary to have a routing mechanism for the message to find the right fog, and it would be necessary to map, but not lose, the agent identity to the routing information, because a device may have only one IP address but may host several agents.

5. Conclusions

The Agent Controller security model builds on standard security technologies, already demonstrated in IT-1, but addressing shortcomings identified in the earlier prototypes and IT-1 itself. Although not all proofs of concept or prototypes were integrated in IT-1, many were sufficient to validate the ideas we now carry forward. At the same time, we must now address the gaps identified and issues postponed from IT-1, such as GDPR and privacy. With this in mind, we now have the ability to build fog infrastructure for the agents, building upon the core transport layer of the fog, with bootstrap mechanisms for agents to establish the security using the methods demonstrated in IT-1, but improved for IT-2. The immediate action items towards this goal, identified throughout this deliverable are, in summary:

#	Description	Explanation	Section
AC1	IPv6 Security	Investigate IPv6 security features for secure discovery	3.1
AC2	Use agent credential	The agent credential should be used as a foundation for all the security goals required of the agent	3.2
AC3	Leader election	Leaders are crucial for the architecture, including security features, so their election is obviously also a security question	4.1
AC4	Re-engineer CA	Based on experiences with the CA in IT-1, work is needed to make it more CA-as-a-service whilst retaining the trust	4.2, 4.4
AC5	Synchronise Cloud CA with Fog CAU	Since the Cloud CAU is proposed in the new architecture, and the CAU, architecturally, is crucial for the security, there is a need to ensure consistency	4.2
AC6	CAU Client for microagents	CAU client is the agent-based part of connecting to a CAU, and must work for microagents	4.3
AC7	GDPR	We need at least account deletion and consent withdrawal	4.5.2
AC8	AC library	Re-engineer to fit existing APIs	4.6.2

Table 2. Action Items

It should be emphasised, however, that there is a lot of work proposed here, and in the D4.2 [3] deliverable, and achieving everything will not be possible. In addition to the above topics, there are also research topics arising from this deliverable. These topics are:

- Use of blockchain distributed ledger for CAU
- Discovery for non Wi-Fi protocol.
- Security of edge devices, both physical and hardware.
- Fog-to-cloud communications for authenticated and authorised agents

References

[1] mF2C Consortium, “D2.5 mF2C Security/Privacy Requirements and Features (IT-2),” Dec 2018. [Online]. Available: https://www.mf2c-project.eu/wp-content/uploads/2019/01/D2.5-Security-and-Privacy-Requirements-for-IT-2-v1.0.pdf.

[2] mF2C Consortium, “D2.7 mF2C Architecture (IT-2),” January 2019. [Online]. Available: https://www.mf2c-project.eu/wp-content/uploads/2019/01/mF2C-D2.7-mF2C-Architecture-IT-2-_v1.0.pdf.

[3] mF2C Consortium, “D4.2 Security and privacy aspects for the mF2C Platform Manager block (IT-2),” January 2019. [Online]. Available: https://www.mf2c-project.eu/wp-content/uploads/2019/02/mF2C-D4.2-Security-and-privacy-aspects-for-the-mF2C-Platform-Manager-IT-2-v1.0.pdf.

[4] mF2C Consortium, “D3.1 Security and privacy aspects for the mF2C Controller Block (IT-1),” June 2017. [Online]. Available: https://www.mf2c-project.eu/wp-content/uploads/2017/06/mF2C-D3.1-Security-and-privacy-aspects-for-the-mF2C-Controller-Block-IT-1.pdf.

[5] mF2C Consortium, “D4.1 Security and privacy aspects for the mF2C Gearbox block (IT-1),” June 2017. [Online]. Available: https://www.mf2c-project.eu/wp-content/uploads/2017/06/mF2C-D4.1-Security-and-privacy-aspects-for-the-mF2C-Gearbox-block-IT-1.pdf.

[6] [Online]. Available: https://www.iso.org/standard/20269.html.

[7] S. Kahvazadeh, V.B. Souza, X Masip-Bruin, E. Marin-Tordera, J. Carcia, R. Diaz, “An SDN-based Architecture for Security Provisioning in Fog-to-Cloud (F2C) Computing Systems,” in *IEEE Future Technologies Conference*, Vancouver, Nov 2017.

[8] S. Kahvazadeh, V.B. Souza, X Masip-Bruin, E. Marin-Tordera, J. Carcia, R. Diaz, “Securing Combined Fog-to-Cloud system Through SDN Approach,,” in *4th Workshop on CrossCloud Infrastructures & Platforms*, April 2017.

[9] mF2C Consortium, “D5.1 mF2C reference architecture (integration IT-1),” May 2018. [Online]. Available: https://www.mf2c-project.eu/wp-content/uploads/2018/06/D5.1-mF2C-reference-architecture-integration-IT-1.pdf.

[10] S. Crompton, J. Jensen, “Towards a Secure and GDPR-compliant Fog-to-Cloud Platform,” in *1st Workshop on Managed Fog-to-Cloud at the 11 IEEE/ACM International Conference on Utility and Cloud Computer (UCC2019)*, Zurich, 17-20 Dec 2018.

Annex 1: What is a Fog?

What is "the fog" at the Link Layer and Network Layer (we use capitalised names to denote layers in the OSI stack)?

- A "fog cluster" is a set of nodes that are Link Layer connected to each other, either directly or through a network router: they look like a local subnet.
    - Example: Ethernet connected devices, connected through a local hub
    - Example: Devices in an airport connected to the same WiFi hub
        - Nodes in fog clusters should be thought of as close, within the reach of direct communication of each other
- Every node has an IP address
    - We discussed whether edge devices can have IP addresses. The general idea is that the very IoT (e.g. sensor) may not be capable of running a Network stack and communicate data over, say, a serial bus. If an edge device is capable, it can have an IP address (e.g. relaying data from a sensor)
        - Example: Arduino Shield
    - Microagents and agents have IP addresses.
- mF2C will need to add logic at the higher OSI layers, e.g. to implement agent to agent communication, or agent to leader communication
    - As far as the Networking Layer is concerned, agents in the same fog cluster can send packets to each other.
    - To reach an agent in another fog cluster, it is necessary to relay messages through services in the cloud. There are two options for doing this.
        - Option one is where Agent 2 (see picture) sends a message to (say) Agent 5. Here there is a need for Agent 2 to address Agent 5 and for a service in the cloud to route the message to the right fog cluster. A Lifecycle service in the cloud could facilitate this. The name of Agent 5 could be its Distinguished Name (in the certificate)
        - Option two is where Agent 5 is subscribed to a messaging service in the cloud and Agent 2 submits a notification of an alert to the same service in the cloud. Agent 2 gets the message either as notification-push, or by polling whether there are any new messages (depending on urgency). Presumably it leaves an acknowledgment for Agent 2. Here, there would be no need of addressing other fog clusters.
- Fog clusters can merge when they become close to each other (i.e. they are able to share a joint Link Layer), and unmerge when they lose connection (it becomes impossible to route packets from some node to another node).
- When they merge, their Leaders have the following options.
    - Each Leader manages its own children and they pass messages to each other to enable communication between their children. Needs horizontal communication between Leaders.
    - Each Leader jointly shares their list of children, so they become duplicates of each other.
        - This needs a process for leaders to share Children with each other,
        - or through the Children (re)discovering local Leaders.
    - One Leader manages/inherits all the children in the cluster.
        - Initially there will be two leaders, and, in a process akin to Leader Selection, one decides to not be Leader (but optionally remain Backup). Again a process of handover of children or rediscovery is needed.
    - Unmerging fog clusters happens when … ? The leader cannot reach some of its children or the children cannot reach one of their leaders?

- In the diagram below, Area 2 is a fog cluster and is Link Layer connected. In other words, Agent 3 would be able to reach the topmost Area-Leader directly.
    - We discussed whether a fog cluster could have another fog cluster connected to it, relaying messages from the subordinate fog cluster to the higher one (through the topmost Leader in the lower fog which needs to be connected to the higher fog). However, general consensus was that this would complicate things too much and we only have ~12 months left to finish the code.
        - However, there may be a use case for dual homed agents, e.g. if a WiFi-based fog needs to connect to an Ethernet based fog. This has to be a Leader in order to facilitate communication between the two halves?
    - Either there is a process for the topmost area leader to hand over the child to a subordinate leader
    - Or there is a process for a child to discover which of all available Leaders is "nearest" or most suitable.
- We are tentatively assuming IPv6 is available at the Networking Layer
    - Which is great for many things, like Neighbor Discovery Protocol (RFC 4861) which includes router discovery, respectively SEcure Neighbor Discovery (RFC 3971, RFC 3756)
    - DHCP may be available (RFC 3315), but we may not need it:
        - We can use Stateless Address Autoconfiguration (RFC 4862, RFC 4941)
        - We can use Cryptographically Generated Addresses (RFC 3972)
        - In particular, it seems preferable to use Link-Local addresses (RFC 4291)
    - We may still need to think about what to do in IPv4 space for legacy environments?
        - Could we have IPv6-capable agents talking to each other over a IPv4 Network router? This may be possible using encapsulation but
- There seems to be no need for DNS

Open Questions

- Communicate to a message queue service in the cloud (possibly subscribing message queues in the fog clusters), or should messages be addressable from the cloud to a node in a fog cluster?
- Details of the merge process: in particular, how leaders hand over children to each other on merge, or, alternatively, what re-triggers the leader/children (re)discovery process after a merge.
- Details of the unmerge process: what triggers it? The action of a child may be different from those of a leader upon an unmerge. If there was only one leader, the remaining nodes need to go through leadership election.
- How to handle intermittent loss of connectivity vs permanent separation
- In a hierarchical-leader scenario (cf Area 2 below), how is the "nearest" or "best" leader selected?

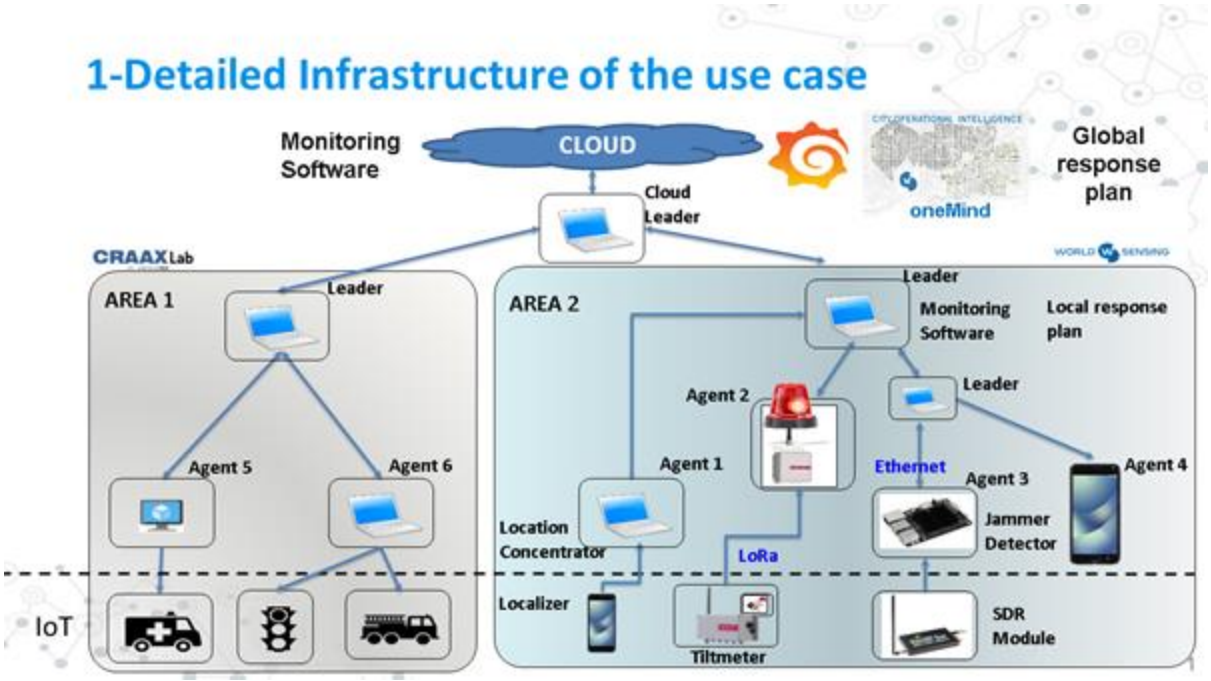

Figure 7 Detailed Infrastructure of the use case