mF2C

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

D3.1 Security and privacy aspects for the mF2C Controller Block (IT-1)

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

Work Package	**WP3, mF2C Controller block design and implementation**
Due Date:	*M6*
Submission Date:	*30/06/2017*
Version:	*0.8*
Status	*Final*
Author(s):	*Jens Jensen (STFC), Roi Sucasas (ATOS), Eva Marin Tordera (UPC), Antonio Salis (ENG)*
Reviewer(s)	*Admela Jukan (TUBS)* *Toni Cortes (BSC)*

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)	

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	24/05/2017	ToC and first draft	Jens Jensen (STFC)
0.2	25/05/2017	Imported and tidies the security policy	Jens Jensen (STFC)
0.3	13/06/2017	Fixed tables of sec. 3, AC description	Roi Sucasas (ATOS), Eva Marin Tordera (UPC)
0.4	14/06/2017	Comments and edits	Antonio Salis (ENG)
0.5	15/06/2017	Edits, reorg., exec. Sum., conclusions	Jens Jensen (STFC)
0.6	22/06/2017	First internal review, addressing reviewers' comments	Toni Cortes (BSC), Admela Jukan (TUBS), Jens Jensen, Shirley Crompton (STFC), Xavi Masip (UPC)
0.7	23/06/2017	Challenges, actions. Updated executive summary	Jens Jensen (STFC)
0.8	29/06/2017	Final version and quality check	Lara Lopez (ATOS)

Table of Contents

List of figures

List of tables

Executive Summary

An mF2C infrastructure must implement security, and we propose here a security policy comprising three levels of data protection: **Public** for data requiring no special protection, **Protected** for data which needs to be integrity protected but is not confidential, and **Private** for data which needs both integrity and confidentiality protection. The perhaps best way of implementing this is to tag data with the required security level – as determined by the sender (or owner) of the data.

This deliverable discusses how the security requirements for the architectural layers that were identified in D2.4 implement and support the security policy. Likewise, each of the functional blocks identified for agents map to requirements which also build on the policy. For example, the requirement for service discovery is linked to the Protected category, as services are usually not secret, but endpoints need to be integrity protected in order to prevent an attacker from impersonating a service, or from maliciously advertising their own services.

This deliverable highlights the need for multipurpose messages, that can carry (say) confidential information along with public, or confidential information from several recipients – there are several such protocols but we are not aware of any used in an IoT scenario. Related challenges arise when lightweight devices – or devices from outside the control of mF2C – communicate data into mF2C; in this case, mF2C must infer ownership and protection requirements. Moreover, other interesting challenges arise when data is processed; if Private data belonging to different individuals is processed, who owns the result? Further challenges arise from the mobility of devices – e.g. if the same device fulfils different roles in mF2C at different times, and from the potentially limited capacity of devices, which may require supplementary services, e.g. that the recipient of data takes (some) responsibility for the policy, or for providing loosely coupled security services to support agents in the fog.

This deliverable highlights these challenges, which are challenges for both the use cases and for the implementation of mF2C in general.

1. Introduction

1.1 Introduction

Whenever a system is designed for implementing data security, it is essential to define a security policy. The policy should guide implementers towards thinking about which bits of data need securing, and how to secure them. One cannot just apply the highest security level to all data, because some data – and metadata – needs to be readable by all participants, and some does not; and applying the highest level of protection can also be too costly, and impair usability.

Conversely, a security policy should be simple enough – and clear enough – that it can be implemented by developers and tested by security analysts.

Specifically, the DoW called for "guarantee security in a dynamic scenario consisting in a large set of heterogeneous devices with very different characteristics, considering aspects such as confidentiality, database protection, while maintaining user privacy and authentication, and messages integrity. " Thus, while we need to focus on mF2C's first software release (known as "IT-1"), the deliverable will have some content which points beyond IT-1. However, it is likely that the security policy will need updating for IT-2, once we have gained experience with it in practice. Core principles will remain, such as the requirement to be secure by design, but additional security features may be required for IT-2; see section 4.

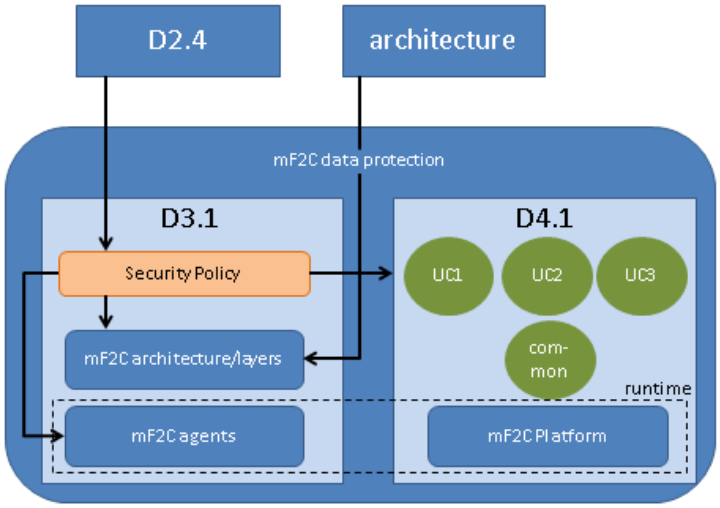

Figure 1: relations to other deliverables

Figure 1 illustrates the relations between this deliverable and the other security deliverables.

The structure of this deliverable is as follows:

- Section 1 (this section) describes the context and content of the deliverable
- Section 2 describes the security policy.
- Section 3.1 lists how the security policy applies to the security requirements (identified in D2.4) for the different mF2C layers.
- Section 3.2 lists the mapping of the security requirements (again by layer) against the expected agent functionalities and discusses the implementation in the agent controller.
- Section 4 discusses the security considerations behind the decisions taken for the first release.
- Section 5 summarises the challenges, the conclusion, and the relevant actions.

1.2 Purpose

The objective of this deliverable is to define a security policy for mF2C IT-1, and to outline the application of it to the components of the architecture, namely the layers in the architecture and the agents operating within it.

1.3 Glossary of Acronyms

Acronym	Definition
AC	Agent Controller

ACL	Access Control List
API	Application Programming Interface
DDoS	Distributed Denial of Service
DoW	Description of Work
GDPR	Generic Data Protection Regulation (European Union)
IaaS	Infrastructure as a Service
IETF	Internet Engineering Task Force
IT-x	The mF2C software release foreseen in the DoW
MQTT	Message Queue Telemetry Transport protocol
RFC	Request For Comments
SDN	Software Defined Network
S/MIME	Secure Multipurpose Internet Mail Extensions
XML	eXtensible Markup Language

Table 1. Acronyms

2. Security Policy

The key words MUST, SHOULD, etc., are to be understood as in RFC 2119 [1]. Specifically, "MUST" means that the feature is an absolute requirement of the specification, "MUST NOT" means that the feature is absolutely prohibited; "SHOULD" means that if the feature is absent one must fully understand the consequences of not providing it, "SHOULD NOT" means one must understand the consequence of providing it. "MAY" is used for optional features.

2.1 mF2C Data Protection Categories

There shall be three data security levels:

- **Public** - accessible to anyone (not necessarily *published* but there is no harm in any participant seeing it.)
- **Protected** – used for data with integrity requirements; may be accessible to any participant in mF2C, but MUST NOT be modified by unauthorised parties.
- **Private** - confidentiality is important, access control is necessary (Private also includes the integrity protection of Protected.)

The general intention is that Private data is personal or commercial (or both, as in accounting), is owned by someone and is access controlled. Protected data is (meta)data which is used for the operations of mF2C, e.g. publishing services and endpoints; the data is not secret, but anyone interfering with it might impersonate a legitimate service or publish a rogue service. Finally, public data is public, e.g. information about the service or aggregated and anonymised statistics.

mF2C Security Policy

Scope

For the purposes of this policy, we consider *two* architectural layers: a fog/cloud layer which is capable of implementing security controls, and a lower "edge" layer which in general is not able to implement the same controls, due to processing capabilities, bandwidth limitations, etc.

This policy applies to data:

- Stored in fog/cloud, or moved or copied between mF2C services in fog/cloud across non-private networks;
- Copied or moved from fog/cloud to edge or from edge to fog/cloud.
- That leaves the mF2C infrastructure.

Policy

- Every entity in mF2C SHOULD have a unique and persistent identity.
- Given a "blob" of data, it MUST be possible to algorithmically determine its protection category.
- All non-Public data must have metadata associated with it in fog/cloud; ideally as an encapsulation (i.e. a data payload in some wrap/protocol), so the metadata moves with the data.
- Metadata of Private data must record the owner's identity and the extent (including time interval) of the owner's consent for use of the data.
    - Metadata of Private data SHOULD contain access control lists
    - Unless the entity is the owner, or a service running on behalf of the owner, an entity MUST NOT be able to add or remove identities/roles to/from the access control list
    - Any entity on the ACL may access the data, but MUST NOT use the data for a purpose for which consent has not been given.
    - Any entity not on the ACL MUST NOT access the data unless
        - It is an aggregating or monitoring service that anonymises or pseudonymises the data; or
        - Consent to this has been given by the owner; or
        - mF2C is legally required to release the data
- Protected and Private data must contain information for integrity check. This information MAY be cryptographic (i.e. protecting against malicious modification as opposed to accidental corruption.)
- Private data MUST NOT be copied or moved to the edge, or moved or copied within the edge, unless
    - the recipient device, or owner of this device, is on the access control list; or
    - consent has been granted by the owner

3. mF2C system security

In deliverable D2.4, [1] we collected all the security requirements of system with IoT, fog and cloud. However, our intention in this section is to provide a holistic view of the security requirements of mF2C systems with three hierarchical layers, as it is shown in Figure 1, and to investigate the application of the security policy to an infrastructure conforming to the architecture. We recap briefly the architecture; for full details see D2.6.

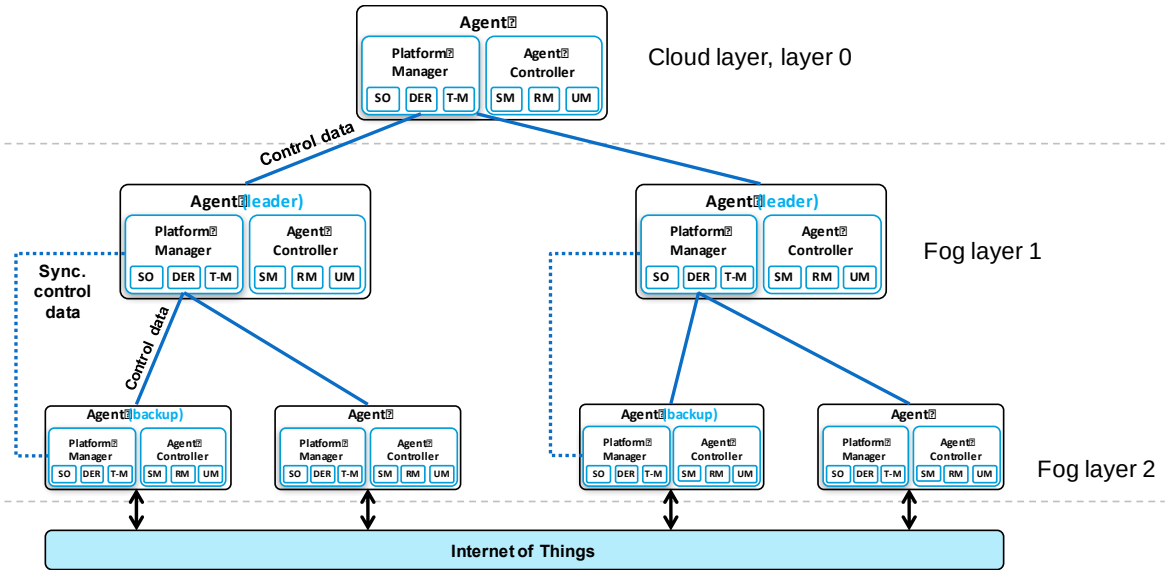

Figure 2 mF2C architecture

The mF2C architecture deploys distributed agents as leader/worker to manage the whole mF2C system. The architecture includes 3 layers (layer 0 = cloud, layer 1= agent (leader), and layer2= edge/agents).

We discuss the security requirements and applications of the security policy in two sections; the first focuses on the layers themselves, and the second one on the agents and their (expected) functionalities.

3.1 Security Requirements – general

The following sections list the security requirements and functionalities by layer, indicating how they must align to the security policy. These requirements were identified in D2.4.

3.1.1. Security requirements in cloud (layer 0)

The requirements for cloud (layer 0) security were identified in D2.4:

- **Secure Storage (Private/Protected/Public):** It must be possible to determine the level of protection offered by a cloud storage service (e.g. does it encrypt data at rest.)
- **Authentication and authorization (Private):** All the agents (layer 1) must be authenticated and take authorization from agent in layer 0. Note that credentials are Private, as are the access control lists, as discussed above. However, the list of authenticated participants may not need to be Private.
- **Key distribution and management (Private/Protected):** An agent in layer 0 must provide keys for the agents in layer1. A well-defined strategy for key management, revocation and updates must be defined. Again, credentials are obviously Private as they are assigned to individual entities, but revocation lists (and typically other public updates) must be Protected.

- **Identity management (Private/Protected):** The unique identities required by the policy may need to be kept private to not disclose any information about agents, or they may need to be advertised to others for discovery (e.g. service providers in the fog). Credentials (such as private keys or shared keys) will need to be Private.
- **Security policies (Protected):** An agent (layer 0) should define well-structure policies for the agent's layer 1. It is Protected since it must not be modified by unauthorised parties; but must be publicly readable and verifiable.
- **Logging protection mechanism (Private):** Logs can be used to manage usage, incidents, and accounting, but can also be used to determine what a person is doing. Thus, in general, logging information is classified as Private.
- **Access controls (Private):** The leader agent in layer 0 should define a well-secure access control for layer 1 agents to prevent any malicious or fake agent accesses to the cloud. Since access control protects Private data, it is Private.
- **Trust (Protected/Public):** It must be possible for a participant to establish trust – to its satisfaction – through Public information (e.g. locating information about providers) and Protected (service endpoint credentials).
- **Data security (Private/Protected/Public):** All the data processing, aggregation, storing, and data sent in secure channels must, of course, be protected according to the policy (e.g. Private data encrypted at rest and in flight.) It is the responsibility of the owner/creator of data to define the policy required for the data; it is the responsibility of the entity *storing* and *transmitting* the data to ensure, to the best of its ability, that the protection policy is honoured.
- **Application programming interface security (Private/Protected/Public):** Agent (in layer 0) APIs provide all infrastructure, platform and software services level communication with agents in layer 1. In other words, also control data must be protected according to policy: critical control data Private, advertised services Protected, and general public information remains Public.
- **Web application security (Private/Protected):** Agent in layer 0 that hosts or offers hosting of web services (e.g. for payment services) must implement secure web quality to avoid attackers getting information in critical applications such as banking (Private), or for an attacker to impersonate a service (Protected).
- **Federation of security among multi clouds (Private/Protected/Public):** although multicloud is seen as out of scope for IT-1, when multiple clouds are federated some services from different clouds are needed, the security requirements for agent (layer 0) in different systems must be federated. In particular, data protection levels must be honoured across clouds.
- **Heterogeneity (Private/Protected/Public):** services may be provided by a multitude of protocols (web, web services, MQTT, etc.) or cloud services (IoT-specific services, general storage blobs/buckets/queues, ad-hoc IaaS services, etc.); these MUST all be implemented to honour the security policy.
- **Integrity (Protected):** Integrity refers not data but also to system integrity. Virtual machine images and images on running systems must be integrity protected appropriately. In the former case only authorised patches may be applied (keeping the image relatively fresh), in the latter, intrusion detection protects against unauthorised modification of system services.
- **Confidentiality and privacy (Private):** Access must be restricted to the agents that are authorized to view the data: this is done either by having the agent's unique identity on the access control list, or by some equivalent (or better) means.
- **Availability (Protected/Public):** High availability is usually achieved through replicating data, and is largely concerned with metadata required for the operations of the infrastructure, e.g. information services. Private data SHOULD NOT be replicated as it becomes harder to protect.

- **Non-repudiation (Private/Protected):** Agents cannot deny to be source or receiver of information. Also Protected data SHOULD include origin authentication. (This one is new; it was omitted from the table in D2.4.)

3.1.2. Security requirements for agents in fog (layer 1)

Some of the security requirements identified for layer 1 (see Table 3 in D2.4, p.51) are similar to those of layer 0; in this list we therefore highlight whether and how they differ from those of layer 0.

- **Security Management (Private/Protected/Public):** like layer 0, layer 1 agents are responsible for enforcing the security policy. Mobile agents, in particular, must be careful with the connections they make; in general, the sender is responsible for ensuring that the data policy is enforced.
    - Authentication and authorisation (Private): As in layer 0.
    - Access controls (Private): As in layer 0.
    - Data protection (Private/Protected/Public: As in layer 0.
    - Identity management (Private/Protected): As in layer 0.
- **Secure communication, secure gateway (Private/Protected/Public):** As in layer 0. Note that layer 1 agents are expected to act as gateways for less capable layer 2 devices (and for devices communicating over private channels) to layer 0 or to other parties in layer 1, so must be able to infer the required security policy and apply this protection policies intelligently.
    - Intrusion detection and prevention (Protected): As in layer 0.
    - Integrity (Protected): As in layer 0.
    - Privacy and confidentiality (Private/Protected/Public): As in layer 0.
    - Availability (Protected/Public): As in layer 0.
    - Non-repudiation (Private/Protected): As in layer 0.

3.1.3. Security requirements in fog/edge (layer 2)

Here we list the applications of the security policy to the requirements of layer 2 (D2.4, Table 3, p.51), highlighting where they differ from the requirements in layer 1. Note that this list only covers the requirements; the implementation in layer 2 may still be different from that in layer 1.

- **Authentication and authorisation (Private):** Layer 2 messages may not be protected through message level security, in which case they will need other types of protection (e.g. dedicated physical links, or something device-specific.)
- **Access controls (Private):** For less capable devices (in terms of processing, storage, and bandwidth) communicating with layer 1, enough information must be passed to layer 1 devices to enable the receiver to infer the access control required for the data.
- **Secure Bootstrapping and Mobility (Private/Protected):** Onboarding a device into mF2C may require getting keys onto it (Private), and/or getting software/apps or agents onto it (Protected). The device may not be deployed by mF2C but may be owned and operated by end users: thus, users should have incentives to not just participate in mF2C but also to honour the mF2C security policy (conversely, users must also feel they can trust mF2C.)
- **Data security (Private/Protected/Public):** Level 2 devices may not have been designed for/with mF2C in mind (e.g. mobile phone), care must be taken when initially connecting such devices. In particular, they violate the principle that the sender be responsible for the enforcement of the mF2C data security policy; so the onus falls on the recipient.
- Level 2 devices may, as permitted by the policy, rely on physical protection of data, both at rest (stored) and in flight (communicated).
- **Identity management (Private):** distributed agents in layer 2 must have unique identities to be recognizable. A secure identity management must be defined and implemented for agents.

- In case of device cloning, two different fog nodes will authenticate to “the same identity.” Thus, the cloning process should avoid copying secrets across and the cloned device should obtain its own fresh identity.
- In other words, cloning SHOULD NOT copy private data.
- However, devices can also be cloned maliciously, and it may make sense to have an agent at fog level that cross-checks identities of nodes at edge to discover such cases.
- The fact that devices can be cloned maliciously strengthens the requirement for unique identities. If devices can naturally share identities, it becomes harder to distinguish authorised clones from malicious clones.
- Integrity (Protected): As layer 1.
- Availability (Protected/Public): As layer 1.
- Non-repudiation (Private/Protected): As in layer 1.

3.2 Mapping Security requirements to functionalities

In addition to the general security requirements, listed in the previous section, the tables below list how the security requirements in the different layers map to functionalities expected from the agents. For example, in **Error! Reference source not found.**, Discovery is a functionality expected for ayer 0 services (i.e. a client discovers services available to it). Discovery is expected to require Authentication and Authorisation (due to the need for the client to determine that it is talking to a genuine service, as opposed to an imposter), but is not expected to require Logging (as discoveries are going on all the time, but are not that sensitive.) Logging discovery queries is not prohibited, but it is not expected to be required for IT-1[1].

3.2.1. Tables of functionality/requirement by layer

The tables below list the security requirements for each of the functional blocks foreseen for mF2C. However, at the time of writing it is not yet clear which of these functional blocks can be included in IT-1 and which must remain for IT-2, so we have included all the functional blocks in the tables.

[1] It would make sense to log DDoS attacks against the discovery service and any mitigating actions, as well as other exceptional events

functional blocks	Security requirements in layer 0																	
	Logging protection mechanism	Authentication And authorization	Key distribution and management	Security Policies	Identity management	Access controls	Secure Communication	Data Security	Web application security	Integrity	Availability	Confidentiality	API security	Non-repudiation	Intrusion detection	Intrusion prevention	Federation of security among different clouds	Heterogeneity
Discovery		✓	✓	✓	✓	✓	✓			✓	✓	✓			✓			
Policies		✓		✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓
Identification	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
Categorization				✓	✓			✓		✓	✓	✓			✓	✓		
Monitoring				✓			✓	✓		✓	✓	✓			✓	✓		
Data Management						✓	✓	✓		✓	✓	✓		✓	✓	✓	✓	✓
Categorization																		
Mapping																		
Allocation																		
QoS providing				✓					✓	✓	✓	✓		✓				
Service Runtime																		
Profiling						✓	✓	✓		✓	✓	✓		✓				
SLA						✓	✓			✓	✓	✓		✓				
Sharing Model						✓	✓	✓		✓	✓	✓		✓				

Table 2. Security requirements in Layer 0 agents

functional blocks	Security requirements in layer 1											
	Authentication	Key distribution and management	Identity management	Access controls	Secure Communication	Data Security	Integrity	Availability	Confidentiality And privacy	Non-repudiation	Intrusion detection	Intrusion prevention
Discovery	✓			✓	✓		✓	✓	✓		✓	
Policies	✓		✓	✓	✓		✓	✓	✓	✓	✓	✓
Identification	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓
Categorization			✓			✓	✓	✓	✓		✓	✓
Monitoring					✓	✓	✓	✓	✓		✓	✓
Data Management				✓	✓	✓	✓	✓	✓	✓	✓	✓
Categorization												
Mapping												
Allocation												
QoS providing							✓	✓	✓	✓		
Service Runtime												
Profiling				✓	✓	✓	✓	✓	✓	✓		
SLA				✓	✓		✓	✓	✓	✓		
Sharing Model				✓	✓	✓	✓	✓	✓	✓		

Table 3. Security requirements in Layer 1 agents

functional blocks	Security requirements in layer 2												
	Authentication	Key distribution and management	Identity management	Access controls	Secure Communication	Data Security	Integrity	Availability	Confidentiality And privacy	Non-repudiation	Intrusion detection	Intrusion prevention	Secure Mobility
Discovery	✓			✓	✓		✓	✓	✓		✓		✓
Policies	✓		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
Identification	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
Categorization			✓		✓	✓	✓	✓	✓		✓	✓	
Monitoring					✓	✓	✓	✓	✓		✓	✓	
Data Management				✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Categorization													
Mapping													
Allocation													
QoS providing	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Service Runtime													
Profiling				✓	✓	✓	✓	✓	✓	✓			
SLA				✓	✓		✓	✓	✓	✓			
Sharing Model				✓	✓	✓	✓	✓	✓	✓			

Table 4. Security requirements in Layer 2 agents

4. Discussion

4.1 Data tagging and security levels – design choices

In section 2 we have defined three security levels, Public, Protected, and Private. In this section, we briefly look at some of the design choices underlying this decision; the purpose of doing so is partly explanatory, and partly to highlight potential future directions.

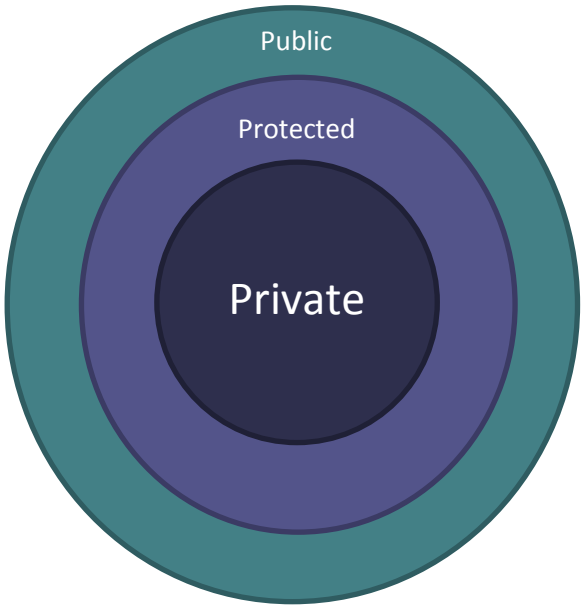

Figure 3: data protection levels

First, we note that the levels are incremental, and not exclusive; see **Error! Reference source ot found.**. The figure is meant to illustrate that Private data has all the protection features of Protected, specifically that Private data is also integrity protected. Likewise, Protected data has the security features of Public data. Alternatively, we could have tagged data with individual confidentiality and integrity requirements, but this would have led to more complexity. The current approach makes it easier to channel data appropriately, e.g. a service which is designed to handle private data can also be used for protected data.

One can ask: why not have only two levels, Public and Private, and use Private to implement Protected? After all, Private already provides integrity checks. In terms of *policy,* it is required that Protected be generally readable – e.g. for publishing a certificate or a web services endpoint, so it is genuinely a different category from Private. However, the *implementation* would be free to implement Protected as a Private communication with a null cipher, and just use the checksumming features of secure sockets.

Another related question is whether we should go further and implement multilateral security. This means that we don't just have Private data but more than one level of privacy, say treating medical data as more confidential than commercial data. There may be more than one Protected level as well, with a higher level protecting against malicious modifications, whereas lower ones protect only against accidental modifications.

For IT-1, we have decided not to implement multilateral security; each protection will have only a single level. However, with the protection levels designed as in **Error! Reference source not found.**, e have not ruled it out; it will be possible later, if needed, to introduce different protection levels.

In practical terms, it is possible that a single message may contain Protected data from different origins, or may contain several chunks of Private data, each owned by a different individual, or both Protected and Private data (e.g. hiding sensitive data in a record), but it would still be advantageous to keep it as a single message. This approach is akin to XML security where the XML messages can contain different types of protection [3], or S/MIME [4].

4.2 Data processing

Another aspect of the security policy is *data processing*. The classic case is processing of personal (and thus Private) data from several persons by a trusted entity; the processed data may no longer

be sensitive (e.g. aggregated information from a large number of individuals) or it may still be sensitive but may be owned by someone else (e.g. airport security). If users need to give explicit consent to the processing (see 4.3), the process can become cumbersome, and explicit consent is not needed for all types of processing (but these needs to be clarified). Similarly, it should be clear to the owner of the data what they can expect; as discussed in D2.4, ideally, the owner can see how the data was processed.

For any type of processing of data, particularly processing that *lowers* the protection level and/or changes ownership of Private data, a supplementary policy MUST be defined to specify when and how data can be reclassified.

Conversely, there may be a need for data to become more restricted; a user's device may be publishing Public data (i.e. potentially shared with other devices), but in a certain context this data could be considered sensitive.

D4.1 discusses the use cases, and highlights cases where processing may require change of security level.

4.3 GDPR

No sensitive/personal data processing discussion can avoid mentioning the GDPR as a particular challenge (see D2.4 section 2.4.1, p.21, and Annex 7 on p. 96).

The security policy helps implement the GDPR by:

- Ensuring that all personal and/or sensitive data is classified appropriately as Private (see D4.1 for the specifics of each use case.)
- Requiring that Private data be identifiable as Private at any time within the mF2C infrastructure, with a clearly defined owner, with suitable levels of confidentiality protection, and with an access control list which is honoured for all types of access except those covered by processing as described in section 4.2 (such as anonymisation and aggregation).

However, the security policy is not sufficient to meet all the requirements of the GDPR, and additional work will be needed.

For example, if we record consent as metadata, it must be taken into account when processing data (section 4.2), but it must be stored also when data is at rest. If consent can be revoked at a later time, or the right to be forgotten is exercised, something needs to be invoked on the data at rest, or the action needs to be remembered and acted upon next time the data is accessed.

4.4 Agents and Agent Controller

Section 3.2 tabulates expected mappings of agent *functionality* against security *requirements*. However, agents are not necessarily isolated entities and they, and their security functionality, will be implemented/supported and deployed through an agent controller.

The security features can be implemented in the controller's functional blocks directly, or in the controller itself. The advantage is that each controller will have all the functionality it might need, at the cost of increasing the size and possibly the computational requirements of the controller (because all calculations are done locally.)

Alternatively, one can take a more loosely coupled approach with locally available providers of security functionality. Here, security services would be available locally to nearby agents in what we here call a "control-area unit", i.e. a security service which is factored out of the agents. This is illustrated in Figure 4. It will obviously require some sort of (local) connectivity, but could also itself provide connectivity to other services, or coordination between units.

Note that this should not be seen as a modification of the architecture, but rather a concomitant service unit run alongside the agents within layer 1. In simple code terms, the question is whether security functionality is provided by a library linked into the agent, or by a nearby web service (or similar.)

This approach has already been proposed [2], building on a novel leader/follower solution, which uses a set of distributed "control-area units" along with a centralized controller located in a cloud. By deploying distributed control-area units as a middleware between fog and clouds, we can provide a convenient coordinated hierarchical security in all layers. The strategy uses a centralized controller in cloud (layer 0 in mF2C) and distributed control-area units to provide the security requirements in distributed fogs (layer 1 and layer 2 in mF2C). The distributed control-area units, in registration and initialization phases, get their authentication and authorization from a centralized controller in the cloud to provide security to their corresponding fogs.

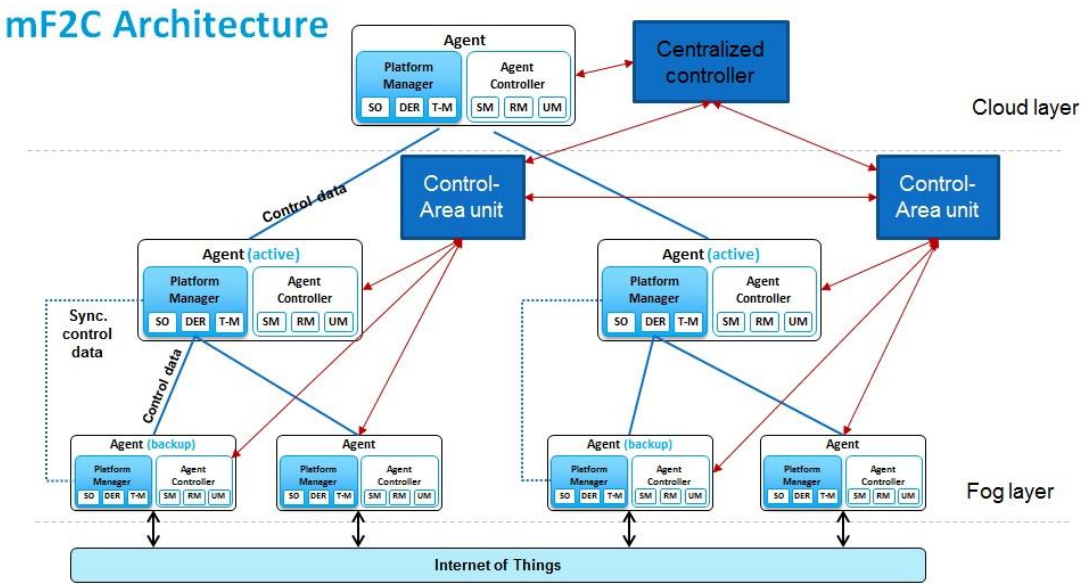

Figure 4: Secure architecture for the mF2C system

5. Conclusions, Challenges and Actions

5.1 Conclusion

This deliverable has described the security policy for mF2C (section 2) and the associated (data) protection levels, Public, Protected, and Private. We further discussed in section 3.1 how the security requirements identified for each layer in D2.4 build on the protection levels.

Section 3.2 takes the agent functional blocks that the agents and/or agent controller are expected to provide and maps them in tables against the security requirements. We propose an area-local "control-area unit" to provide shared security functionality for the agents/controller.

- The security policy (section 2) must be honoured throughout the mF2C infrastructure.
- The policy may be amended from time to time, as we gain experience with it, so some level of flexibility would be useful.
- For components, services, and agents in each layer, the relevant functional blocks can be looked up in the tables in section 3.2. Software developers implementing these functionalities are expected to take the associated security requirements into account in their implementation (and testing).
- For implementers of security, each of the security requirements must also honour the security policy. This mapping is described in section 3.1.

5.2 Challenges

The analysis in this deliverable has identified some particular challenges for mF2C (in no particular order):

1. How does one do multi-protection messages, i.e. where a message can contain data owned by different entities, or some parts are Private and others not?
2. While the sender is normally held responsible for ensuring that adequate controls are implemented for data that it communicates to others (or stores), this may not be possible in all cases for devices not controlled by mF2C or for less capable devices; thus the recipient must take some of the responsibility for ensuring the protection.
3. In particular, layer 1 devices that receive data from such level 2 devices are likely to have to be designed to act as secure gateways for clouds and other fogs. It follows that they must be able to infer and apply the appropriate protection for the data they receive (and relay).
4. Processing of data is particularly interesting, since data can be anonymised and/or aggregated. And even if it is not, how shared ownership of processed data is determined (i.e. processing Private data which is owned by several people?) What sort of policy can be defined and applied in an automated way?
5. Another general challenge is the mobility of fog/edge devices in implementing the security policy; if devices carry data with them, they must be careful how they connect and communicate with new participants, particularly when challenges 2 and 3 are taken into account – i.e. a lightweight layer 2 device communicates non-Public data into a foreign mF2C layer 1 but relies on the recipient to apply the security policy. (In terms of the mF2C use cases, a person may have an mF2C app on their mobile phone which connects both to their boat and to an airport smart hub – albeit not at the same time.)
6. "Outsourcing" security features to local agents (section 4.4) should be explored, particularly when it is useful or beneficial to offload work from agents. Obviously the security implications should also be understood (tests are described in D2.4).

5.3 Actions

The main actions arise from the challenges: to try to determine appropriate technologies for solving these challenges. For example, as mentioned XML [3] and S/MIME [4] provide the kind of

security we need, but are likely too cumbersome for edge devices; conversely, we do not wish to reinvent the wheel, so should be working with existing standards as much as possible.

ID	Description	Action	Priority (IT-1)
1a	Multi-security messages	Survey existing protocols and test	SHOULD
1b	Data tagging with metadata, providing data origin authentication and checksums for Protected, and access control and ownership for Private	Proposed implementation (tested on relevant devices!)	MUST
2a	Receiver-applied security policies	Identify situations (based on D4.1) in which this case arises (see 5.2.2)	MUST
2b	Receiver-applied security policies	Propose tools and languages for determining whether data is Public, Protected, or Private	MUST
2c	Receiver-applied security policies	Test policy implementation	MUST
3	Receiver-applied security policies for Private data	Extend policy (in 2.2) to infer owner and access control list for Private data	SHOULD
4a	Data processing	Identify situations in which the data processing case arises (D4.1).	MUST
4b	Data processing	Identify rules for determining security level and ownership (if applicable) for derived data	SHOULD
4c	Data processing	Indentify potential langauges for deriving protection requirements	SHOULD
5a	Mobile device	Identify situations in the use cases where this situation (2.5.5) can occur	MUST
5b	Mobile device	Discuss possible remedies for 5a	COULD
6a	Implementing agent security functionality	Identify opportunities for “outsourcing” security functionality to local units (section 4.4)	SHOULD
6b	Implementing agent security functionality	Assess security implications of outsourcing	MUST

References

[1] S. Bradner, “Key words for use in RFCs to Indicate Requirement Levels,” RFC 2119 (1997), http://www.rfc-edictor.org/rfc/rfc2119.txt, DOI:10.17487/RFC2119.

[2] D. D2.4. [Online]. Available: http://www.mf2c-project.eu/d2-4-m4/.

[3] “W3C XML Security Working Group,” 2008-2016. [Online]. Available: https://www.w3.org/2008/xmlsec/.

[4] S. M. S. C. N. F. J Galvin, “Security Multiparts for MIME: Multipart/Signed and Multipart/Encripted, RFC 1847,” Oct. 1995. [Online]. Available: Http://www.rfc-editor.org/rfc/rfc/1847.txt, DOI:10.17487/RFC1847.

[5] V. S. X. M.-B. E. M.-T. J. G. R. D. S. Kahvazadeh, “Securing combined Fog-to-Cloud system Through SDN Approach,” *Crosscloud, Serbia,* 2017.