Project number: 653454
H2020-DS-2014-1

Project start: October 1, 2015
Project duration: 3 years

CREDENTIAL

Secure Cloud Identity Wallet

D2.2

System Security Requirements, Risk and Threat Analysis

Document Identification	
Due date	April 1, 2016
Submission date	March 31, 2016
Revision	1.0

Related WP	WP2	**Dissemination Level**	PU
Lead Participant	AIT	**Lead Author**	Andreas Happe (AIT)
Contributing Beneficiaries	AIT, ATOS, FOKUS, TUG	**Related Deliverables**	D2.3, D2.5, D2.6

Abstract: This document defines and discusses the non-functional security requirements for the core components developed within CREDENTIAL. Because of the early stage of the project, it mainly focuses on generic security requirements and only gives pilot specific requirements for certain aspects. In particular, the current requirements cover software development, lifecycle and deployment aspects, communication and user-management facets, as well as generic server- and client-side security concerns. Furthermore, a preliminary risk assessment for the CREDENTIAL wallet is performed.

Usability, privacy, and functional requirements are out of scope of this document.

This document is issued within the CREDENTIAL project. This project has received funding from the European Union's Horizon 2020 Programme under grant agreement no. 653454.

Executive Summary

On a high level, the central goal of the CREDENTIAL project is to develop a privacy-preserving data sharing platform (wallet) with integrated identity provider (IdP), which can be used to share authenticated data without the wallet learning any of the user's personal information. The functionality and added value of these services will be showcased by concrete pilots from the domains of eGovernment, eHealth, and eBusiness.

A central task that needs to be performed before and in parallel to the actual implementation and development of such a service is to precisely specify the requirements the system has to fulfill. This requirement engineering process is necessary to achieve best possible acceptance by all stakeholders, but also to identify and resolve potentially conflicting requirements posed by different stakeholders.

The requirement analysis is consists of the assessment of functional and non-functional requirements. This document focuses on the *non-functional security requirements* of the developed core components, ranging from software architecture requirements over deployment and life-cycle management, via communication, user-management, and logging, through to server- and client-specific requirements. Because of the very early stage of the project, this document mainly focuses on generic requirements that are not specific to the CREDENTIAL pilots, but apply to many web applications. However, certain design decisions have already been made and are addressed in this document. Also, wherever possible, requirements, inherently coming from the very own approach of CREDENTIAL and going beyond those of existing IdP and data sharing platforms, are discussed.

Besides the specification of non-functional security requirements, we give a precise description of the considered adversary model.

Furthermore, we describe the security risk management approach that is used to evaluate actual risks in CREDENTIAL. We here follow a STRIDE&DREAD approach. Also, a preliminary high-level risk assessment is given and already considered in the requirements engineering process.

Differentiation to other deliverables. This document focuses on non-functional security requirements. Functional requirements on the Cloud Identity Wallet can be found in D2.3 ("Cloud Identity Wallet Requirements"), which in particular focuses on data exchange, authenticity, and secure re-encryption, but also on technical requirements like compatibility with existing services. Legal, socio-economic, privacy, and usability requirements are discussed in D2.6 ("User Centric Privacy and Usability Requirements").

Finally, a second iteration of the document at hand will be given through D2.5. This enhanced document will in particular cover use-case and pilot specific security requirements, which we will infer from D2.1 ("Scenarios and Use-Cases") and D6.1 ("Pilot Use Case Specification"). Also, based on the concrete pilot scenarios, the actual security assessment will be performed in D2.5.

Document information

Contributors

Name	Partner
Andreas Abraham	TUG
Andreas Happe	AIT
Aleksandar Hudic	AIT
Stephan Krenn	AIT
Nicolás Notario McDonnell	ATOS
Christoph Striecks	AIT
Florian Thiemer	FOKUS

Reviewers

Name	Partner
Tobias Pulls	KAU
Andrea Migliavacca	LISPA
Bernd Zwattendorfer	SIC

History

0.1	2015-11-24	Andreas Happe	Initial commit and first ideas
0.2	2016-01-19	Andreas Happe	Updated draft, first structure
0.3	2016-02-10	Andreas Happe	Added first concrete requirements
0.4	2016-02-12	Andreas Happe	Collected further concrete requirements
0.5	2016-02-22	Aleksandar Hudic	Relevant requirements standards added
0.6	2016-02-23	Nicolás Notario McDonnell	Added section about risk evaluation
0.7	2016-02-26	Andreas Abraham	Added adversary model
0.8	2016-03-07	Andreas Abraham Andreas Happe	Added some requirement category overviews
0.9	2016-03-11	Stephan Krenn	Category overviews; differentiation to other requirement documents in CREDENTIAL
0.10	2016-03-14	Andreas Happe Stephan Krenn	Added use-case specifications; added lifecycle, TLS, server-side requirements
0.11	2016-03-16	Andreas Happe Stephan Krenn Nicolás Notario McDonnell	Template for requirements; added user-management requirements; addressed comments from 1st KAU-review
0.12	2016-03-18	Andreas Abraham Andreas Happe Stephan Krenn Florian Thiemer	Added executive summary and abstract; added requirements for logging and service isolation; adversary model

0.13	2016-03-22	Andreas Abraham Andreas Happe Aleksandar Hudic Stephan Krenn Florian Thiemer	Added access rights, communication, service isolation, and cryptographic requirements; added reference architecture; added some floating text
0.14	2016-03-25	Andreas Abraham Andreas Happe Stephan Krenn Christoph Striecks	Added server side requirements; internal review; work on requirements standards mapping; addressed comments from 2nd KAU-review
0.15	2016-03-29	Andreas Happe Aleksandar Hudic Stephan Krenn Nicolás Notario McDonnell	Amended multiple requirements; work on requirements standards mapping; added risk assessment demonstration
0.16	2016-03-30	Andreas Happe Aleksandar Hudic Stephan Krenn	Addressed reviewer comments from LISPA; copy-editing
1.0	2016-03-31	Andreas Happe Stephan Krenn	Addressed reviewer comments from SIC; final proof-reading; editorial details

Table of Contents

List of Figures

List of Tables

List of Acronyms

AEAD	Authenticated Encryption with Associated Data
AIDE	Advanced Intrusion Detection Environment
API	Application Programming Interface
CP	Cloud Provider
CUMULUS	Certification Infrastructure for Multi-Layer Cloud Services
DFD	Data Flow Diagram
ENISA	European Union Agency for Network and Information Security
EXIF	Exchangeable Image File Format
HSM	Hardware Security Module
IdP	Identity Provider
IP	Internet Protocol (address)
ISO	International Organization for Standardization
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP Secure
NIST	National Institute of Standards and Technology
OWASP	Open Web Application Security Project
PIN	Personal Identification Number
TLS	Transport Layer Security
XSS	Cross-Site Scripting

Glossary of Terms

DREAD
Approach developed by Microsoft to prioritize risks, depending on **D**amage potential, **R**eliability, **E**xploitability, **A**ffected users, and **D**iscoverability.

STRIDE
Threat modeling technique developed by Microsoft; considered thread categories are **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service, and **E**levation of privilege.

1 Introduction

Requirements engineering's main goal is to detect, analyze, and document the different, possibly conflicting, requirements that a system must fulfill in order to maximize the acceptance by its stakeholders—ranging from end users over potential investors through to governmental bodies. Requirements engineering is a task that not only has to be performed before, but also in parallel to the development process of the actual system. This is because during the implementation, some requirements may turn out to be missing, overly restrictive, or might have to be changed, e.g., due to changing legal regulations.

Complementary to the requirements analysis, risk analysis is the process of identifying and analyzing the dangers of adverse events to individuals and businesses. The assessment of the identified risks may lead to the identification of additional (security) requirements, while the compliance with these requirements in turn reduces the negative impact of potential threats.

In this document, we focus on the analysis of generic non-functional security requirements, as well as on a definition of the risk assessment process used within CREDENTIAL and performing a high-level risk assessment. This is because of the early stage of the project where precise (functional) specifications of the CREDENTIAL tools are not yet available. Therefore, pilot-specific requirements and also a concrete risk assessment can not be given. Consequently, this document should be seen as a basis for the development of the single components within the CREDENTIAL process, which will further be extended in parallel to the pilot specification and delivered in form of a second iteration by D2.5.

Finally, we want to note that this document is about defining security requirements, and analyzing security threats. However, it does not focus on organizational requirements in case a security incident happens despite of having all these specifications in place. Such incident response procedures have to be defined on a case-to-case basis, and cannot be covered by a generic requirements catalogue.

1.1 Requirements in CREDENTIAL

Within CREDENTIAL, multiple deliverables are concerned with requirements elicitation. To help the reader in finding the appropriate document, and to delimit the single documents, we next briefly discuss these deliverables.

D2.2 & D2.5 ("System Security Requirements, Risk and Threat Analysis—1st and 2nd iteration") are two iterations of the same deliverable. They cover the non-functional security requirements for the CREDENTIAL core components. Furthermore, they contain a subsequent security analysis from the perspective of different stakeholders following a STRIDE&DREAD approach. These documents cover technical areas such as general security aspects, infrastructure resiliency, and assurance of security properties.

In particular, the first iteration, i.e., D2.2, mainly contains generic security requirements. Building upon these requirements and on the precise specification of the pilot use-cases from **D2.1** ("Scenarios and Use-Cases") and **D6.1** ("Pilot Use Case Specification"), D2.5

then also gives use-case specific security requirements and analyses. Special attention will be paid to the definition of the requirements related to the cross-layer, end-to-end security. This is necessary due to the active participation of peripheral devices such as smart-phones in the end-to-end composition of the cloud service and the exchange of confidential data across different domains.

D2.3 ("Cloud Identity Wallet Requirements") covers functional requirements for the CREDENTIAL Cloud Identity Wallet architecture, and reflects the end-user and business user needs. In particular, D2.3 in details covers aspects like authentication, secure data re-encryption, data authenticity, and data exchange. Furthermore, technical requirements addressing existing interfaces of service providers, or organizational requirements for secure cloud solutions in real production environments are analyzed in this document.

D2.6 ("User Centric Privacy and Usability Requirements") is concerned with legal, socio-economic, privacy, as well as usability requirements. These requirements are not only collected through literature and law studies, but also through workshops and face-to-face meetings with domain experts from the fields of users' rights, privacy, and usability.

1.2 Document Structure

The remainder of this document is structured as follows.

In Section 2, we give an overview of the involved parties in CREDENTIAL, a description of the intended pilot scenarios, and a specification of the adversary model considered in this project.

In Section 3, we explain the concepts of STRIDE&DREAD that are used for evaluating risks in CREDENTIAL. Also, a preliminary high-level risk assessment is performed there.

Then, in Section 4, based on the risk assessment, we specify the considered categories of (non-functional) security requirements and also define the boundaries of this document. The concrete lists of requirements are then given in Section 5, grouped by the categories identified before.

We finally give an outlook on future work in Section 6.

The main body of this document is complemented by the following appendices. In Appendix A, we detail the components of generic cloud systems to obtain a common language. In Appendix B, we discuss the most relevant related requirement standards, which were also used to identify the categories and requirements presented in this document. We also show how our requirement categories can be mapped to those of chosen other standards. A small set of functional requirements that was identified during the work on this document and which influenced the design of the non-functional requirements is finally given in Appendix C. Finally, Appendix D contains an overview table of all requirements defined in this document for easy reference.

2 System Modeling and Pilot Scenarios

In this section, we detail some generic modeling which is required for a meaningful security analysis. We therefore first give a high-level description of the functional model of the components developed within CREDENTIAL and introduce the involved stakeholders. We then specify a generic setup for the pilots to be implemented. Finally, we define the adversary model that we are considering within CREDENTIAL.

2.1 High-Level Functional Model and Stakeholders

The CREDENTIAL system consists of a set of involved stakeholders, whose interactions and tasks we summarize in the following. An graphical representation of this model can be seen in Figure 1.

CREDENTIAL assumes a client-server computation model where clients always communicate with server-side services. This model allows easier server-side control of data access privileges. The CREDENTIAL *wallet* allows *users* to securely share personal data among each other and other interested parties. Therefore, a user uses a *device* (e.g., mobile phone, laptop) to execute a *client application* (e.g., smart phone app). This client application interacts with its counterpart on the server side (e.g., online service) maintained by some *application hoster* (e.g., service provider), which in turn requests certain information from the wallet (e.g., to grant the user access to the confidential data owned by a potentially different user).

The data wallet decouples data owners from users accessing their data. After the data owner has given data access-rights on a fine granular level, approved users can access this data. In contrast to traditional solutions, the data owner can be offline during the latter operations. In addition, data confidentiality is upheld at all times, i.e., data is never decrypted on the server.

A special case of information is dedicated identity data. Within the wallet, this data is managed by the internal *identity provider*. This service selectively discloses identity data to service providers, thus the functional workflow is very similar[1] to the wallet's data operations. After an initial authentication of the user at the identity provider, this component allows a user to authenticate itself to other service providers similar to existing single sign-on systems. This is similar to the standard model for identity management by Bertino and Takahashi [BT10].

Concrete components will be detailed during the project's runtime and will be incorporated in future revisions of this document. Furthermore, necessary details of the single components will be introduced in this document whenever necessary.

2.2 Pilot-Scenarios

Within CREDENTIAL, we consider three concrete instantiations of the above model, coming from three different real-world scenarios.

[1]We assume, that the implementation of the identity provider API will be sufficiently different compared to the "normal" data API to warrant a distinct Identity Provider component.

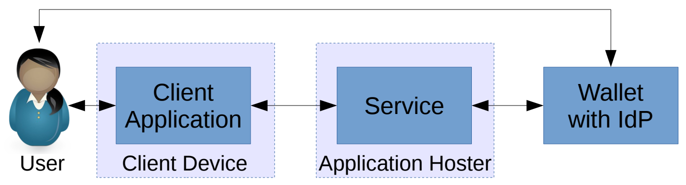


Figure 1: High-Level Functional Model

eGovernment. The eGovernment pilot considers citizens who wan to remotely pay taxes or request financial support from his local tax office. For instance, the pilot considers a citizen of country A living abroad in another country B, who needs to pay local taxes in country B. Now, he can use his electronic identity card of country A to securely and strongly authenticate himself to the tax portal of country B, potentially using STORK [STO16] and eIDAS [eID15] to perform this cross-border authentication.

The CREDENTIAL platform is now used to host authentic personal data that goes beyond the data that is stored on the national eID card. For instance, such data might include pay slips or certificates of registration. The user can now grant the tax authority of country B access to this data. As granting access rights can also be done for documents that will be added to the wallet in the future, the user can easily file certain required documents later without having to contact the tax authority again, but by simply uploading the data to the CREDENTIAL wallet.

eHealth. The eHealth pilot is concerned with a data sharing platform between patients, doctors, and further parties, in particular in the context of Type 2 Diabetes. Namely, the developed components will allow patients to record their health data (blood sugar level, weight, blood pressure, etc.) using external mobile devices. The data measured on these devices will be collected by a CREDENTIAL eHealth mobile app, which remotely stores this data in the CREDENTIAL wallet. The user can then define who is allowed to access which parts of this medical data, to share specific parts of the measurements, e.g., with the family doctor, diabetologist, nutritionist, or personal trainer. Based on the data they see, they can then provide recommendations back to the user.

Because of the confidentiality of medical data, it is of prime importance that only legitimate users are able to access a user's data. Furthermore, because of the potential consequences of wrong recommendations, the authenticity and integrity needs to be guaranteed.

eBusiness. The eBusiness pilot realizes the classical single sign-on (SSO) functionality. Furthermore, the CREDENTIAL system is used to remotely unlock a hardware security module (HSM) for digitally signing documents. That is, the HSM with the secret signing key is under the physical control of a certified and trusted service provider. To digitally sign a document, the (hash of the) document is sent to the HSM through the certified provider. Traditionally, the user now would have to enter a short PIN as well as some one-time pad (OTP) sent to his mobile phone in order to trigger the signature process. Using CREDENTIAL, this low-entropy PIN can be replaced by a long and secure string: the HSM—through the service provider—requests this string directly from the CREDENTIAL wallet, after the user has instructed the wallet to share this information with the provider. In this way, the security level of this remote signature procedure can be increased without affecting usability.

2.3 Generic Pilot Setup

The mentioned pilots will be partially implemented during the CREDENTIAL project. Based upon the pilots' descriptions, expected functionalities were identified. This section introduces an imagined generic setup which implements these functionalities.

While the final prototype implementation might differ from this early setup, their commonalities allow us to derive requirements during this early project phase. During the project's lifetime—and before the second iteration of this document—this mental model will be replaced with the concrete realization. During this transformation, existing requirements will be adapted and new requirements added.

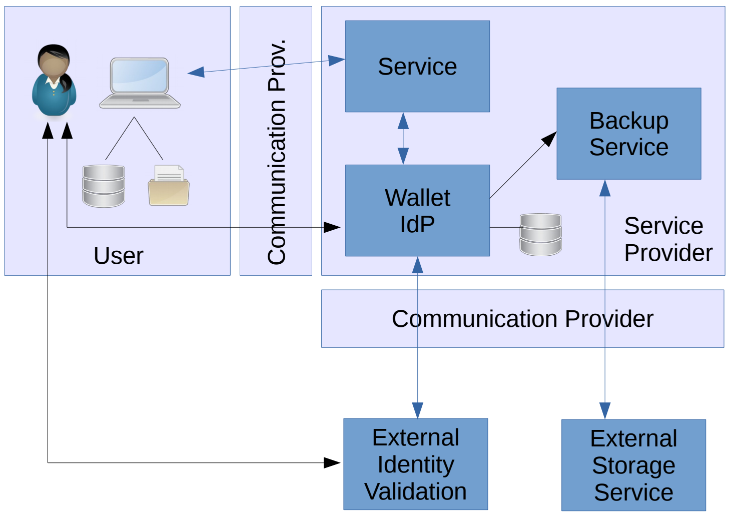


Figure 2: Generic Pilot Setup

We foresee the following large components shown in Figure 2:

Users use different **Clients** to interact with the system. A single user might use multiple different clients simultaneously. We limit clients to three different classes: desktop applications, mobile applications and web-browser based rich web-clients. Clients can persist data (files, caches, databases) locally—this is the main reason for adding rich web-clients and their client-side storage options as separate client.

The **wallet** stores sensitive data on behalf of the client. It additionally provides **identity services** to third-parties. The wallet needs means to validate its users' identity. To achieve this, external validation services (e.g., a passport office) can be integrated. The validation itself can be implemented in different ways, we differentiate between a wallet- and user-triggered validation process. The first model, the wallet directly communicates with the **external validation service**. In the latter model, the user requests that the external validation service gives him proof about his identity and then forwards this proof to the wallet service.

Services provide functionality to clients and depend upon the identity management provided by the wallet service. Not all services are triggered by end-users. For example a backup service has to be run periodically to create external backups. In this case there must be some way of extracting the backup data from the wallet and storing it on external media while upholding the user data's privacy and security.

All entities (e.g., services or clients) can store local data (files, cache, databases). All communication between entities (e.g., client applications or external storage providers) is performed over potential hostile territory. Communication providers are assumed to be potentially malicious.

Services reside on logically distinct machines. On a physical level, a service might be realized within a virtual machine and be placed with other virtual machines on the same host.

2.4 Threats, Attacks, and Adversaries

According to the Cambridge dictionary a threat is "a suggestion that something unpleasant or violent will happen". For a computer system, a threat is the potential for the occurrence of an harmful event. An attack is an example of the latter: it is a concrete action with the intention of inflicting harm against a computer system.

Adversary models are used to specify an attacker's capabilities. For CREDENTIAL, two adversary models are of high importance: honest-but-curious and malicious/Byzantine. Within the former model, an attacker is assumed to follow the precise protocol specifications, i.e., it does not deviate from the behavior of an honest party. Rather, it tries to infer information from all the data it can gather, typically in particular comprising all network communication. Within the latter model, the attacker can be active and behave arbitrarily maliciously; in particular, it can thus alter messages or directly attack network components.

The adversary considered in CREDENTIAL is a combination of the above two types. That is, certain components of the system are assumed to be honest-but-curious. However, the project strives to achieve higher security by considering threats and stronger attackers where reasonable, such as for communication. We also strive to minimise and distribute trust wherever possible. In the following we specify the trust assumptions made for the single components.

Wallet. The wallet service is honest-but-curious. Note here that certain assumptions have to be made for the wallet, e.g., to guarantee availability of the service, as the wallet could otherwise easily exclude all (or subsets of) users from the service—however, this would be irrational behavior for a cloud provider. Furthermore, when aiming for privacy, we have to make "non-collusion assumptions" between the wallet and the service providers: by collaborating with the intended and legitimate receiver of a user's data, the wallet could otherwise trivially learn this data as well.

Wherever possible, we will furthermore aim for concrete instantiations of (cryptographic) primitives in which users can detect incorrect behavior of the wallet, or where such a deviation of the protocol (e.g., structured sampling of intended random numbers) on the wallet's side cannot cause any privacy problems for the user. By doing so, we can significantly reduce the necessary trust compared to a vanilla honest-but-curious model.

Services. We assume that services accessing the wallet are semi-trusted, i.e., after they have been authorized by the wallet owner, we trust that they correctly execute their intended role on a semantical level. This is because we cannot validate the service operations' semantics (e.g., detect maliciously altered patient data given by a medical professional). However, we assume that the service might behave maliciously otherwise, and might in particular try to access other parts of the wallet for which it was not granted access rights (e.g., a malicious medical professional might try to access legal documents within the wallet). This model also suits real-world services: they will be contributed by third-parties and the wallet cannot reason about their security or privacy requirements.

Validation services. The external validation service is an external entity and again must be assumed to be semi-trusted. That is, we trust that it performs its validation tasks (e.g., physical identity checks) correctly, but we protect CREDENTIAL against other attacks originating from the external validation service.

Users. We assume users to be semi-trusted: the data owner won't maliciously access her own data but might be interested into other user's wallet data.

Applications. The user utilizes client applications to interface CREDENTIAL's services. Of those we assume mobile clients to be fully trusted due to their corresponding platform's security restrictions (sandboxing, application signing). Browser and Desktop applications are trusted but part of an dynamical application environment. We expect, that the user's system might contain malicious code.

More precisely, we will make those assumptions when theoretically evaluating and proving certain security aspects of the CREDENTIAL wallet and IdP. In order to allow for making those assumptions (which is made in virtually all theoretical papers), we also concentrate on giving requirements that need to be satisfied to ensure a secure software development process. Indeed, a significant fraction of the requirements stated in this document is necessary in order to fulfill those assumptions.

A deployed service seldom operates as a monolithic block—for example, when using an Amazon-compatible deployment model, processing might be performed by multiple EC2 computing instances while storage is delegated to a separate S3 or database instance. This separation of storage and processing is even more prominent in container-based settings (e.g., Docker). To allow requirements to include this deployment detail we differentiate between Storage Providers (providing long-term storage) and Service Providers (processing service or application service). This distinction is used to clarify our requirements' scope.

3 Risk Evaluation

According to the Security Risk Management Body of Knowledge [TJ09],

> *a security risk is any event that could result in the compromise of organizational assets. The unauthorized use, loss, damage, disclosure, or modification of organizational assets for the profit, personal interest, or political interests of individuals, groups, or other entities constitutes a compromise of the asset, and includes the risk of harm to people.*

For the main purposes of this deliverable and in the context of our security risk analysis, CREDENTIAL is focused in Information Security risks, where the main asset to be protected is information. However, it must be taken into account that information security risks have deep implications and cannot be untangled from other types of risk such as privacy, safety or financial; for instance, stealing your bank credentials creates a financial risk.

The European Union Agency for Network and Information Security (ENISA) devoted in 2006 a significant amount of effort to compile the most prevalent risk assessment/management methodologies and published its results in its "Survey of existing Risk Management and Risk Assessment" [ENI06]. It analyzed a list of 13 methodologies comparing them according to several criteria such as the availability of supporting tools, price, language and the level of degree of fulfillment of the methodology regarding the different phases of risk management: identification, analysis, evaluation, assessment, treatment, acceptance and communication. Most of the analyzed methodologies are still relevant, however, many new ones have appeared since 2006. The same criteria could, and should, be used to determine the most adequate risk management methodology for any given project or organization.

After reviewing the different analysis methodologies mentioned in ENISA's report and the review of others (not necessarily security-specific), we've identified some common tasks within a risk management process that CREDENTIAL's approach should address:

- Identify what is to be protected: security objectives and assets
- Identify threats to these assets
- Prioritize the threats according to their likelihood and potential impact
- Address priority threats by establishing some measures
- Continuously monitor the assets and the efficiency of such measures

CREDENTIAL's security risk management approach will follow a generic risk management process, with the steps above described and supported by STRIDE and DREAD.

3.1 STRIDE

STRIDE is a threat modeling technique, developed by Microsoft that classifies threats according to the exploitation class. Each of STRIDE's threat classes matches a corresponding security property as shown in Table 1.

Security property	Threat class	Example
Authentication	**S**poofing	Using another's account to access the system
Integrity	**T**ampering	Modifying data at rest (e.g., account balance)
Confirmation	**R**epudiation	Someone can deny performing some specific action (e.g., no secure signature available)
Confidentiality	**I**nformation Disclosure	Someone can access other users stored data
Availability	**D**enial of Service	The system denies access to valid users during a DoS attack
Authorization	**E**levation of Privilege	A user becomes a super user, being able to tear down the whole system

Table 1: STRIDE threat categories and examples related to security properties

As it is shown in Figure 3, STRIDE's iterative process can be described in terms of four different steps:

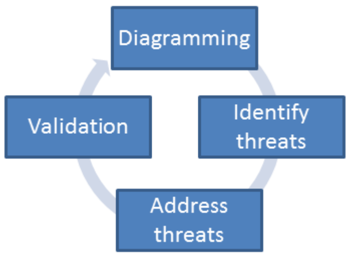


Figure 3: The four steps of STRIDE

3.1.1 Diagramming

STRIDE process starts with a visual description of the system depicting its entities, processes, data stores, data flows and trust boundaries. While STRIDE strongly suggests the usage of Data Flow Diagrams (DFDs), it may be the case that, due to the project's scope and relationship with

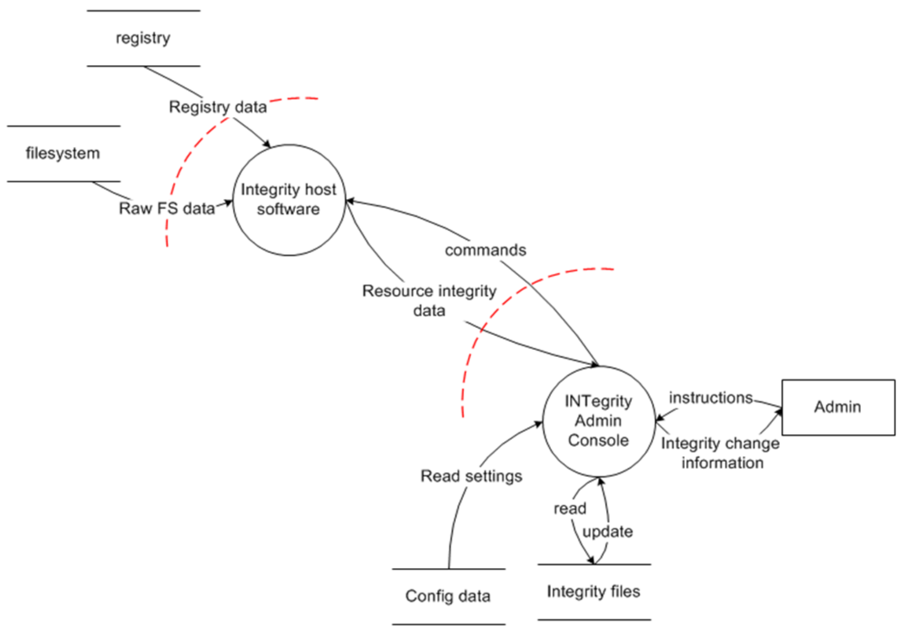


Figure 4: Example of a Level 1 DFD [Mic08]

previous and further work, an alternative diagramming standard (e.g., UML) may be chosen (see [JJ10]). The level of detail of the visual description must be enough that:

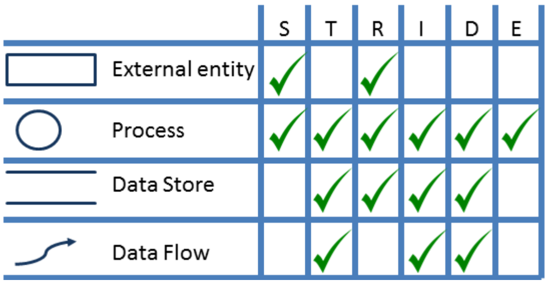
- processes do not cross trust boundaries
- security impact of the design is clear
- the processes have a single responsibility

Normally, when using DFDs, it is not necessary to go beyond a level 2 DFD. Figure 4 shows an example of a level 1 DFD of an imaginary system.

3.1.2 Identify Threats

Once the system description is finalised, all elements of the diagram must be assessed in terms of STRIDE threats categories, seeking to discover relevant threats. Notice that all categories do not apply to all type of DFD artifacts, only the ones that appear in Figure 5 must be considered:

This task can be supported by using tools such as Microsoft's Threat Modeling one [Mic16] which already contains a large list of types of processes, entities, flows, boundaries, etc. linked to relevant threats that will be automatically identified as the system's diagram is created.



	S	T	R	I	D	E
External entity	✓		✓			
Process	✓	✓	✓	✓	✓	✓
Data Store		✓	✓	✓	✓	
Data Flow		✓		✓	✓	

Figure 5: STRIDE threat categories' table

3.1.3 Address Threats

Once the threats have been identified, they must be addressed by:

- redesigning the system
- apply mitigations (existing solutions that are known to reduce the level of risk)
- invent mitigation (new solutions that are expected to reduce the level of risk)
- last, and whenever the circumstances do not allow to mitigate the risk, the risks can be accepted. In such cases, they must be adequately documented.

3.1.4 Validation

The developed threat model must be validated to ensure that:

- The visual description of the system matches its final implementation
- All threats are clearly identified and traceable to specific actions (mitigations, redesign or acceptance)
- All threats and mitigation measures are adequately described

3.2 DREAD

DREAD was also developed as a tool within Microsoft's team in order to prioritize the risks discovered by using STRIDE. Namely, while STRIDE describes a process that identifies and manages risks, it does not address the general risk management task of prioritizing the risks. This is a key step of the process, as it is often the case where mitigation measures can create some new risks or the mitigation measures of two different risks are incompatible. Also, systems usually have a limited scope in terms of time or funding. Risks have to prioritized so a trade-off process can determine which will be the actual mitigations to be put in place.

DREAD characterizes five different aspects of threats which try to capture the impact and its probability of happening:

- **D**amage potential: How much is the asset affected?
- **R**eliability: How easy is it to reproduce the attack?
- **E**xploitability: How easy it it to launch the attack?
- **A**ffected users: How many people will be impacted?
- **D**iscoverability: How easy it is for attackers to discover the threat?

Each of the aspects must be ranked with a number between 0 and 9. The final level of risk is the mean value of all the aspects.

There are other options for risk assessment such as the one proposed by CNIL's privacy risk management [CNI12], NIST's Special Publication 800-30 [SGF02], OWASP's risk rating methodology [OWA16b] but in the end, they can all be reduced to a function of severity and likelihood.

3.3 CREDENTIAL's Risk Evaluation Process

By using a generic process supported by these tools (STRIDE and DREAD), CREDENTIAL will be addressing all the relevant steps of risk management process. ENISA describes in its Annex II of [ENI06] a common structure to evaluate each of the assessed methodologies. If we were to extend their assessment with the described management process supported by STRIDE and DREAD using the criteria specified by ENISA in its risk management comparison report, it would obtain the following values and would position itself among the best candidates:

- Risk identification: 3/3 (It describes detailed risk identification activities)
- Risk Analysis: 3/3 (It assesses the likelihood and negative consequences of the materialization of the threats)
- Risk Evaluation: 3/3 (The risks are evaluated taking into account different stakeholder perspectives)
- Risk assessment: 3/3 (STRIDE tool automatically helps to identify risks)
- Risk treatment: 3/3 (STRIDE tool provides suggestions to address risks)
- Risk acceptance: 3/3 (This risk management process allows to accept risks, as long as they are reasonable and documented)
- Risk communication: 3/3 (The output of the process can be used to communicate risks, those addressed and those accepted)
- Languages: English

- Price (method only): free
- Size of organization: all
- Skills needed: basic level
- License: No license required
- Certification: No license required
- Dedicated support tools: Microsoft' Threat Risk modeling Tool

While it is true that STRIDE and DREAD have been criticized by sectors of the security community and their own authors admit that both "were developed with any real academic rigor, and from a scientific standpoint, neither of them tend to hold up very well" [LeB07], it is also true that there are prevalent practices and are encouraged by renown organizations in the security domain such as ENISA and OWASP. ENISA used STRIDE for its App Store risk assessment [DH11] and OWASP recommends "Microsoft's threat modeling process because it works well for addressing the unique challenges facing web application security and is simple to learn and adopt by designers, developers, code reviewers, and the quality assurance team" [OWA16d].

The security risk analysis performed in this task is complementary to the privacy risk analysis that will also be performed within CREDENTIAL. Using STRIDE and DREAD with DFDs for the security risk analysis is complementary to using LINDDUN, STRIDE's privacy-specific risk assessment methodology and which is being considered to be used within the project. Both methodologies could leverage the same visual representation of the system and privacy and security risks could be easily compared. PRIPARE, a FP7 Coordination and Support Action that developed a security and privacy by design methodology specifically mentions STRIDE and LINDDUN as adequate tools for performing risk analysis during the design of secure systems [GMT+15].

The next iteration of this deliverable will provide more details on the methodology's and will perform the actual assessment. For this deliverable, an initial and partial assessment has been performed (see section 3.4.) in an initial version of CREDENTIAL's Level 1 DFD. This assessment will only consider a reduced set of the threats automatically detected by Microsoft's threat modeling tool, however, it will help to demonstrate how the final assessment will by conducted, when a more detailed DFD (level 2) will be used and additional threats will be considered.

3.4 CREDENTIAL's Risk Assessment Demonstration

In the following we show how the above approach can be deployed to CREDENTIAL. This section mainly has an illustration purpose—a more detailed analysis based on a more fine granular DFD will be performed once the concrete use-cases have been defined in D2.1 and D6.1.

3.4.1 Step 1: Diagramming

An initial version of a Level 1 DFD of CREDENTIAL has been developed in order to understand what are the assets to be protected and to enable the identification of specific threats:

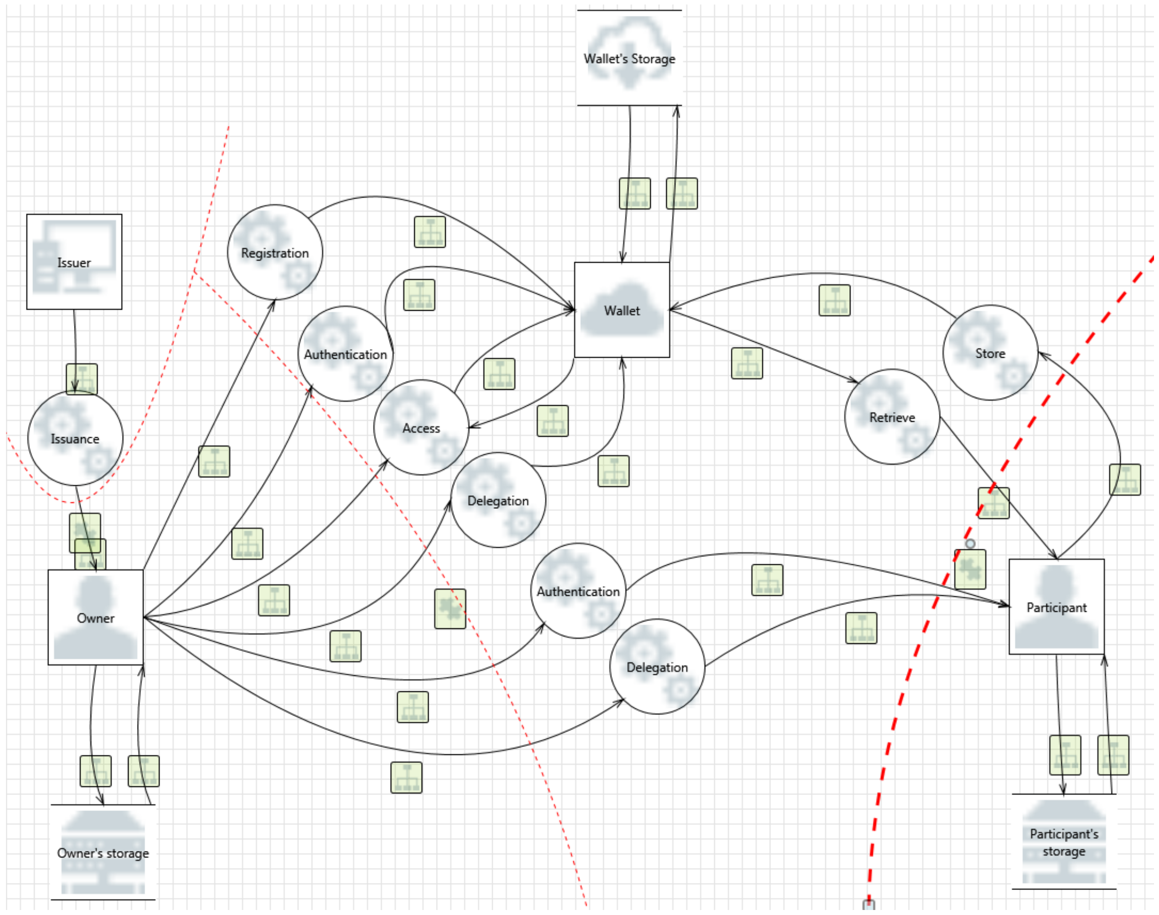


Figure 6: CREDENTIAL's Wallet DFD - Level 1

The DFD shows four main trust zones (delimited by red dotted lines in Figure 6):

- The owner's trust zone, which represents what the owner (of a CREDENTIAL wallet's account) is in control of (a computer, a web browser, a smart phone, etc.)
- The participant's trust zone, which represents other CREDENTIAL participants (third party services or other owners) and their systems.
- The issuer trust zone, which represents the systems of entities capable of issuing owner's personal data (e.g., an attribute provider)
- The wallet's trust zone, which represents the main system, including the processes and storage that allows participants to manage and share their data.

3.4.2 Step 2: Threat Identification

Microsoft's threat modeling tool, in base to the DFD shown before (Figure 6) identifies 104 different threats, categorized into the six threat STRIDE categories. As this section's purpose is to demonstrate the risk management process, only a few threats are included in the following table. These threats have been chosen to show the casuistry of categories and of the mitigation options.

ID	Category	Name	Description	Comments
1	Spoofing	Spoofing of Source Data Store Wallet's Storage	Wallet's Storage may be spoofed by an attacker and this may lead to incorrect data delivered to Wallet. Consider using a standard authentication mechanism to identify the source data store.	Mutual authentication mechanisms could easily avoid, this risk
2	Information Disclosure	Weak Access Control for a Resource	Improper data protection of Wallet's Storage can, allow an attacker to read information not intended for disclosure. Review, authorization settings.	This threat will be avoided or minimized by using, re-encryption technologies and by using authorization controls
22	Repudiation	Potential Data Repudiation by Access (process)	Access (process) claims that it did not receive data, from a source outside the trust boundary	It will be considered the use of logging or auditing, to record the source, time, and summary of the received data

ID	Category	Name	Description	Comments
23	Tampering	Potential Lack of Input Validation for Access	Data flowing across may be tampered with by an, attacker. This may lead to a denial of service attack against Access or an, elevation of privilege attack against Access or an information disclosure by, Access. Failure to verify that input is as expected is a root cause of a very, large number of exploitable issues	All inputs will be verified for correctness using an, approved list input validation approach.
26	Denial Of Service	Potential Process Crash or Stop for Access	Access process crashes, halts, stops or runs slowly; in all cases violating an availability metric.	High availability techniques will be applied to address, this threat
32	Elevation Of Privilege	Elevation Using Impersonation	Authentication process may be able to impersonate, the context of Owner in order to gain additional privilege	Authentication process will include mechanisms (e.g., two-factor) to ensure only the owner has access to its data

Table 2: Subset of identified threats with DFD - Level 1

3.4.3 Step 3: Threat Prioritization

As described in the generic process, the third step is to prioritize the threats according to their likelihood and potential impact. For that, it will be used the DREAD scheme. The following table shows the application of the rates in the five categories specified by DREAD:

ID	Name	Damage potential	Reliability	Exploit-ability	Affected users	Discover-ability	Total
1	Spoofing of Source Data Store Wallet's Storage	**5** Individual user data is compromised or affected	**3** IP Spoofing or DNS poisoning could be enough for this attack	**3** Logic access to the Wallet internal network	**10** All users	**4** Knowledge of the internal wallet workings	**5.0**

ID	Name	Damage potential	Reliability	Exploit-ability	Affected users	Discover-ability	Total
2	Weak Access Control for a Resource	**5** Individual user data is compromised or affected	**8** If the access control is weak this attack could be easily reproduced	**10** Just a web browser	**10** All users	**5** Can figure it out by guessing	**7.6**
22	Potential Data Repudiation by Access (process)	**1** No individual data is compromised but the trust on the system is	**8** If there are no appropriate logging or auditing means this threat can be reproduced always	**5** Not directly exploitable, it requires additional attack	**10** All users	**2** Hard to discover internal logging or auditing issues	**5.2**
23	Potential Lack of Input Validation for Access	**10** The lack of validation controls enables code injection attacks which could lead to data destruction	**10** No appropriate validation controls makes this attack to be fully reliable	**10** Just a web browser (enough, e.g., for SQL Injection attacks)	**10** All Users	**9** Details of faults like this are already in the public domain and can be easily discovered using a search engine	**9.8**
26	Potential Process Crash or Stop for Access	**4** Access to the data is prevented but it is not compromised	– It depends on the exact attack performed (e.g., DoS)	– It depends on the exact attack performed (e.g., DoS)	**10** All users	**9** Details of faults like this are already in the public	**7.7**
32	Elevation Using Impersonation	**10** An owner being able to impersonate (e.g., a system admin) may enable to destroy all users data	**8** If the access control is weak this attack could be easily reproduced	**4** Knowledge of the internal wallet workings	**10** All users	**2** Only with custom or advanced tools this attack would be possible	**6.4**

Table 3: DREAD values for identified threats

Note that threats 23, 26 and 2 are the ones with highest DREAD value and should be prioritized.

3.4.4 Step 4: Measure Establishment

The following requirements will be established to address the already identified and prioritized threats:

- The Wallet Storage and the Wallet process, which accesses the storage, must be securely and mutually identified by the means of HTTPS connections with the usage of certificates issued by the organization responsible of operating CREDENTIAL. This minimizes the reliability, exploitability, and discoverability of threat 1. For further information see Section 4.5.
- All user inputs to the system must be validated in order to prevent common SQL Injection or XSS attacks. This minimizes the reliability, exploitability, and discoverability of threats 32, 23 and 2. For further information see Section 4.7.
- A two-factor authentication process is required to access owner's and/or administration's functionalities of the Wallet. This minimizes the reliability, exploitability, and discoverability of threats 32, 23 and 2. For further information see Section 4.6.
- All accesses to the system must be logged for posterior auditing (if required). The log will include the IP address, the time and the operation performed by the owner. For further information see Section 4.4.

3.4.5 Step 5: Risk Monitoring

Once a final list of security requirements has been established, the following step would include the following activities:

- Discussing the trade-offs of these security requirements with other aspects of the system. That is, the logging requirement may conflict privacy requirements.
- Ensuring that the agreed requirements are implemented
- Periodically reviewing that the actual measures are effective according to the advances in security and attacker's capabilities.

As initially discussed, this is just a demonstration of how the process will be fully applied once the system's DFD is complete (with more detail). Existing requirements from other stakeholders and from other points of view (legal, privacy) and technology choices will modify the list of threats, rendering some of them already addressed and others non-applicable. Other requirements or technological choices will also add new threats to the list.

4 Requirement Categories

Based upon the generic pilot setup and the adversary model, potential threats and requirements were identified using a top-down approach. The starting point was "Software Architecture" which is concerned with the overall structure of the CREDENTIAL system as well as with the process that was utilized to create this structure. Our architecture consists of multiple components, each of which has a separate requirement category ("Client-side Concerns", "Server-side Concerns"). Naturally, communication between components is also of concern for security requirements ("Communication-Related Requirements"). A separate set of requirements is primarily concerned with operational matters during runtime ("Service Isolation", "Lifecycle Management"). In addition, we have identified cross-cutting requirements valid for multiple components ("Cryptographic Requirements", "Logging Requirements").

The CREDENTIAL project places high the importance upon privacy and usability. Due to this reason privacy and usability requirements have been elevated and are handled in distinct deliverables. A border case are functional requirements: those are also handled within a separate document scheduled after this document.

State of the art standards, best practices, and research projects that outline security based requirements were consulted during the inception of CREDENTIAL's requirement categories. Based upon this review, the ISO 27001 Information Security Management Standard and the OWASP Software Architecture Verification Standard were decided to have the highest impact for CREDENTIAL—both, because of their technical scope and their practical international relevance (for instance, ISO 27001 is often referred to as the de facto standard for information security management). Details regarding the mapping of our categories to OWASP's Application Security Verification and ISO 27001 Information security management standard can be found in Appendix B.

Two iterations of this document are planned during the CREDENTIAL project. The first version focuses upon non-functional requirements. This suits the project plan which introduces use-cases (which are the base for functional requirements) after the initial version of this document. We expect that the second iteration will focus more on requirements derived from specific requirements. In addition, some of the requirements within this document will be transferred to related deliverables as D2.3.

4.1 Software Architecture

Software architecture describes the high-level structure of a software project, its inception step as well as means of documenting the chosen architecture [CGB+02]. As architecture greatly influences the final implementation, security and privacy concerns must be part of the initial architecture discussion.

The overall system consists of multiple components, each of which consists of multiple interfaces. They must conform to basic object-oriented architecture standards, e.g., SOLID-principles [Mar09] (single responsibility, open-closed, Liskov substitution, interface segregation and dependency inversion). Each component and interface must be documented in a way that allows external personnel to reason about the software's functionality.

The combination of accordance to design standards and documentation will allow for automated software testing and thus aid both productivity as well as software quality.

4.2 Life-cycle Management

Software life-cycle management concerns itself with deployment, monitoring and disposal of software and hardware artifacts.

On the software side, it contains requirements about the deployment step, i.e., who is allowed to install new software, how is this automated and which log information is created during deployment. There are security and privacy concerns arise for the deployment activity itself as well as for artifacts created during deployment (i.e., sensitive information within backups created during deployments).

During the software's lifetime, it has to be monitored for runtime exceptions and intrusions. The monitoring software itself must not introduce additional security or privacy problems into the overall system.

For the disposal phase, both hardware and software are involved. Sensitive data must be destroyed before hardware is decommissioned.

4.3 Service Isolation

Services by their very nature process and entail various types of information therefore requiring, even depending on, mutual interaction and information exchange with other services. Depending on the criticality of a service and sensitivity of related information, proper actions have to be taken to fulfill the requirements within respect to security of services deployed in cloud environments.

Although, cloud service utilization model offers plethora of features for processing, scaling, maintaining availability and fast provisioning, the lack of control and transparency unfortunately only amplify the security challenges. Hence, this category puts in the main focus service deployment in cloud-based environments with regards to security. Current approaches that attempt to suppress and mitigate those challenges by supporting service isolation via containerized or virtualized environments, encrypting information or services, considering legal and geographical deployment restrictions, and other service related mechanisms have to be supported by predefined set of requirements.

We highlight the importance to define the requirements that support secure service communication, isolation or collocation across all layers in the cloud (service, virtual and physical) to prevent major losses and outages of service or related information. Hence, an extensive analysis and elicitation of the corresponding requirements for secure service deployment is required. We detail these requirements in Section 5.3.

4.4 Logging and Reporting

Continuous monitoring of systems, services, and information in cloud based systems is of crucial importance. One needs to be able to detect any violation with respect to integrity, availability, or confidentiality of the logged information in order to provide reliable reports. Therefore, for reliable monitoring information mechanisms should be set in place that guarantee that there are neither inconsistencies nor loss of information. This becomes especially challenging when it comes to sensitive information that is being logged, as one has to prevent that this information is not being exposed to unauthorized internal and external entities.

On the one hand, we therefore need to require that a sufficient amount of information is being logged. On the other hand, requirements guaranteeing that logged information is reliably decommissioned once it is no longer needed, and access to the logging data needs to be restricted. Furthermore, the requirements need to capture the tradeoff of generating reliable reports without exposing sensitive information.

We detail the requirements and corresponding methodologies to suppress the above mentioned challenges in Section 5.4.

4.5 Communication-Related Requirements

All communication between CREDENTIAL's server-components and user clients is performed over inherently untrusted channels. As communication providers might permit unauthorized access to communication data or modify messages, we have to protect the communication between the client and server application.

Transport level security requirements focus on the exchange of information. To fulfill these requirements, different security mechanisms have to be utilized such as transport layer security (TLS) connections, perfect forward secrecy (PFS) and authentication encryption with associated data (AEAD) schemes. All transport level security related requirements are described in detail in Section 5.5.1.

How the user is notified by the system is explained in the user notification part in Section 5.5.2. The methods should be secure depending on the delivered content and fulfilled requirements.

4.6 User- and Session-Management

User-management is concerned with authentication, authorization and access control, and session management aspects.

Authentication is the process of verifying that a user accessing the service is actually the one that he claims to be. In particular, the type of login mechanisms (passwords, two factor authentication, etc.) are specified here. Out of scope of CREDENTIAL, this aspect would also cover physical checks, e.g., at local authorities, to guarantee the authenticity of the user's personal data before issuing initial digital credentials. On the other hand, authorization specifies what a user is allowed to do, e.g., which files is a user allowed to access or modify. For instance, one needs to ensure that no user can access any data for which it does not have appropriate access rights.

Complementary to these aspects, this category also covers session management. This ranges from the correct choice of valid session identifies, over counter-measures against hijacking of active sessions, through to time-outs in case of inactivity.

The detailed specification of the requirements can be found in Section 5.6.

4.7 Server-Side Concerns

CREDENTIAL employs a traditional client-server architecture. We assume, that servers will provide a web as well as an HTTP API.

Two out of the Top 3 OWASP security risks are related to input validation and injection attacks (XSS, injection), security requirements will reflect upon that.

Special consideration is placed for protection of data at rest. Malicious insiders and incidents with stored backup data are common occurrences. To prevent data loss, encryption at multiple levels will be required.

The full list of server-side requirements can be found in Section 5.7.

4.8 Client-Side Concerns

This category focuses on the client-side concerns which consist of a generic part valid for all clients, followed by client-specific requirements. Within CREDENTIAL there will be desktop clients, mobile applications and web-based clients.

The security and privacy of the user's data must be maintained. In particular, the focus here is split into different parts – how sensitive data is stored, what kind of data can be cached and how the cache's security and privacy can be provided. Within the client-specific part detailed requirements for our concrete clients will be given.

The specific requirements related to mobile clients can be found in Section 5.8.1. Methodology on how to store data on the mobile device, what information is being utilized and shown. Authentication related requirements are detailed as well.

4.9 Cryptographic Requirements

Because of the central role of advanced cryptography within CREDENTIAL, the respective requirements are clustered in a dedicated category. This category mainly focuses on the correct generation and handling of cryptographic key material, and the correct choice of cryptographic primitives, yet does not define specific algorithms to not be overly restrictive.

We note that side channel security is out of scope of the CREDENTIAL project, and is subject to dedicated large scale projects such as, e.g., SAFEcrypto [SAF16].

The concrete list of requirements can be found in Section 5.9.

4.10 Categories Beyond the Scope of this Document

As already mentioned earlier, this document is concerned with the collection and analysis of non-functional security requirements. These are grouped in the nine categories introduced so far. In the following, we now describe some more potentially interesting categories. For each of these requirements, we either provide references to the corresponding documents within CREDENTIAL, or we explicitly define the boundaries of the requirements considered within the project.

4.10.1 Functional Requirements

Functional requirements define the operation of a system (i.e., "How is a system supposed to be?"); non-functional requirements define specific functions that a system offers to users (i.e., "What is a system supposed to do?"). This is problematic for the CREDENTIAL project, as typical non-functional requirements such as privacy or usability are within the project's scope and thus functional requirements.

Functional requirements and non-functional requirements influence each other. We therefore give a minimal set of functional requirements that impacted the requirements listed in this document in Appendix C. All other functional requirements can be found in D2.3.

4.10.2 Privacy Requirements

Informally, the main ambition of CREDENTIAL is the development of a privacy preserving data sharing platform. In particular, this means that the provider of this platform should not gain access to any information stored by a user. However, also other levels of privacy are conceivable. For instance, leaking the access pattern for a specific file might leak information to the provider. Even more, a user might be interested in even hiding from the provider whether or not he has stored any data in the service.

Because of the broad spectrum of potential privacy levels and requirements, and because of the key importance of privacy within CREDENTIAL, these requirements are treated in a separate document, cf. D2.6.

4.10.3 Usability Requirements

Simply put, usability specifies how easy the system must be to use. That is, the system should be easy to learn, in the best case even for an inexperienced user. Furthermore, it should be efficient to use at least for frequent users, and easy to remember for casual users [LY98]. Usability also covers the efficiency of applications on the intended computing platforms.

Usability is also concerned with factors like understandability. This means that users should be able to understand what is actually happening in the system and what is different to similar solutions, a property which is of prime importance if one wants to convince users to migrate from an existing solution to a functionality-wise similar privacy-preserving alternative.

Without good usability, the created software prototypes will not be integrated into real-life workflows. To highlight the importance of this area, usability-related requirements are defined in a separate deliverable. The interested reader is referred to D2.6 ("User Centric Privacy and Usability Requirements") for a detailed discussion of usability requirements.

4.10.4 Legal Requirements

A successful long-term deployment of any service always requires that certain legal requirements are satisfied, where these requirements can vary significantly from country to country. For instance, different countries may require different information to be logged due to data preservation regulations. Such requirements are out of scope of this document. Legal requirements related to privacy will be discussed in D2.6.

4.10.5 Physical Security

CREDENTIAL's focus lies on software and network protocols. Physical access security (i.e., an attacker that has physical access to the server or client hardware and can thus introduce hardware backdoors) is thus out of scope. Also, physically securing service providers, etc. is not considered in this project.

4.10.6 Protection of Intellectual Property

Service providers granting users access to their services typically have the economic need to protect certain parts of their intellectual property. For instance, this might be the case for the concrete implementation of certain features or the internal structure of data centers. Therefore, appropriate requirements might be posed, such as the impossibility to efficiently reverse-engineer certain binaries. Such strategies are out of scope of the CREDENTIAL project, but are subject to specific other research and development activities, e.g., [Tor15].

5 Security Requirements

In the following, we now formalize the non-functional security requirements for the different categories discussed in Section 4. As discussed earlier, during the first iteration we focus on generic non-functional security requirements; more pilot-specific requirements will be included in subsequent versions of this document, depending on D2.1 and D6.1.

As requirements are derived from different domains (including legal, usability, security, privacy as well as pilot-specific requirements), single requirements might conflict or even contradict each other. If this occurs, conflicting requirements will be noted. After all requirements have been noted and a risk analysis has been performed upon them, concrete requirements for the design and implementation phase of CREDENTIAL will be selected on a case-to-case basis.

Requirement Definition Key Words

We follow best practices in the specification of our requirements. In particular, we follow the recommendations of the Network Security Group concerning the semantics of key words [Bra97]:

MUST, REQUIRED, SHALL. These words indicate an absolute requirement for the specification.

MUST NOT, SHALL NOT. These words indicate an absolute prohibition for the specification.

SHOULD, RECOMMENDED. These words indicate that there may be valid reasons to ignore a requirement under certain circumstances. However, the implication of ignoring the recommendation should be well understood and carefully weighed before proceeding.

SHOULD NOT, NOT RECOMMENDED. These words indicate that under certain circumstances a particular behavior may be useful or even necessary. However, the full implications of this decision should be understood and carefully weighted.

MAY, OPTIONAL. These words indicate truly optional requirements. Depending on the circumstances, the person in charge is free to decide whether or not to address the specific requirement. However, interoperability with systems satisfying other sets of optional requirements MUST be guaranteed.

Requirement Definition Template

All requirements are presented in the following unified way specified next:

Title of requirement	
REFERENCE ID	Unique identifier of the requirement, which will be preserved also in future revisions of this document to have a unique reference; e.g., ARC-MIN-COMPS. Each ID consists of a three-letter acronym of the category, followed by an at most 12-letter acronym for the requirement.
AREA	Specifies which areas of CREDENTIAL are concerned by this requirement. Possible values are "Client" and "Service" for all client- or service-specific requirements. Please note that "client" is used for the client-application itself. A special case is "wallet": these requirements are for the wallet-service. As the wallet is itself a service, all "service"-level requirements are automatically mandatory for wallets too. "Environment" is used for all environment requirements, e.g., infrastructure or mobile-client requirements.
RESPONSIBLE	Describes who is responsible for the requirement. Possible values are "architecture" if it is a high-level requirement that concerns multiple components. "Implementation"-level requirements are software developer centric and mostly target a single component. If multiple components are selected (through "area") those requirements are not interdependent. "Administration"-level requirements should be configurable at runtime. A higher-level requirement implies that lower levels will heed the higher-level's decision, e.g., an "architecture"-level requirement (formulated through the architectural design provided by the software architects) will be heeded by software developers during implementation. Ideally only a single entity is responsible for a requirement but due to the early development stage this is not always possible, e.g., responsibility for transport-level requirements are dependent upon the chosen technology. If HTTPS is used for transport, this is mostly a configuration setting thus "administrative". On the other hand, if a custom socket-based transport is chosen, the implementation (or software developer) is responsible for fulfilling this requirement. We assume all redundant responsibilities will be solved before the second iteration of this document.
DESCRIPTION	Detailed description of the requirement and the underlying rationals
CONFLICTING	Here, potentially conflicting or contradictory requirements are listed. This block will in particular become relevant once all usability and privacy requirements have been defined in D2.6.

In addition to the key words defined above (e.g., MUST, SHOULD, MAY), we support the reader by using the following color coding: absolute requirements and prohibitions are marked **red**, recommendations are denoted in **yellow**, and truly optional requirements are labeled in **green**.

5.1 Software Architecture

The main focus of this section is the overall architecture and the corresponding software development workflow.

Number of Components SHOULD be minimal	
REFERENCE ID	ARC-MIN-COMPS
AREA	Client, Service
RESPONSIBLE	Architecture
DESCRIPTION	The overall architecture should consist of the minimal amount of components needed to fulfill the pilot use cases. This minimizes feature creep and results in a smaller and thus better understandable and testable source code base.

Components MUST conform to the single responsibility principle	
REFERENCE ID	ARC-SING-RESP-PR
AREA	Client, Service
RESPONSIBLE	Architecture
DESCRIPTION	Each component should fulfill a single responsibility. This mandates the creation of multiple components, each of which is responsible for a single aspect or feature of the overall architecture. This allows to test each component for failures and security problems. In addition, this aids modular software design and results in reusable components.

Interfaces MUST be documented	
REFERENCE ID	ARC-DOC-INTERF
AREA	Client, Service
RESPONSIBLE	Architecture
DESCRIPTION	Each provided interface must be fully documented, i.e., each offered operation and its parameters and return value.

Interfaces MUST be minimal	
REFERENCE ID	ARC-MIN-INTERF
AREA	Client, Service
RESPONSIBLE	Architecture
DESCRIPTION	Interfaces must only contain functions needed to achieve their responsibility. If additional functions are added, they must be placed into a new (or existing fitting) interface.

Software development MUST follow a security-aware life-cycle	
REFERENCE ID	ARC-SEC-AWARE
AREA	Client, Service
RESPONSIBLE	Implementation
DESCRIPTION	The whole product life-cycle must follow a security- and privacy-aware life-cycle, e.g., Microsoft Security Development Life-Cycle. This includes activities such as security requirements, creation and evaluation of secure coding practices and security/privacy by design.

ACL rules MUST match between different layers	
REFERENCE ID	ARC-MATCHING-ACL
AREA	Client, Service
RESPONSIBLE	Architecture
DESCRIPTION	Server side implementation and presentation layer representations of access control rules must match.

Source code MUST comply to secure coding standards	
REFERENCE ID	ARC-CODING-STND
AREA	Client, Service
RESPONSIBLE	Implementation
DESCRIPTION	All generated or written source code must comply with secure coding standards (e.g., OWASP Secure Coding Practices [OWA10] or SEI CERT Coding Standards [CER16]). Compliance should be validated using automated tools during the continuous software validation step.

Continuous software testing MUST be performed	
REFERENCE ID	ARC-TESTING
AREA	Client, Service
RESPONSIBLE	Implementation
DESCRIPTION	Written software should be tested using techniques such as unit testing. Those tests must be periodically run over the current source code.

Third party software dependencies (e.g., libraries) MUST be identified and documented	
REFERENCE ID	ARC-LIBRARIES
AREA	Client, Service
RESPONSIBLE	Implementation
DESCRIPTION	Newly written software always uses existing software components and systems. Those might include third-party libraries, operating systems, infrastructure software, management software, etc. All dependencies to external software and software libraries must be documented and monitored. This is essential to allow for periodic security-update checks.

An clean overall architecture MUST be enforced	
REFERENCE ID	ARC-CLEAN-ARCH
AREA	Client, Service
RESPONSIBLE	Architecture
DESCRIPTION	CREDENTIAL must follow an overall architecture with a clear separation between data, controller, and view components. This allows for easily testable components and thus improves security.

Information MUST be classified	
REFERENCE ID	ARC-INF-CLASSIFY
AREA	Client
RESPONSIBLE	Implementation
DESCRIPTION	To ensure that each information receives the appropriate level of protection, information MUST be classified according, e.g., to regulatory or contractual requirements.

5.2 Lifecycle Management

Lifecycle management's scope is mostly within the administrative runtime domain and concerns itself with software component configuration, deployment and maintenance.

The lifecycle process MUST be defined

REFERENCE ID	LIF-DEF-PROCESS
AREA	Environment
RESPONSIBLE	Architecture
DESCRIPTION	The lifecycle process must be documented. This includes workflow information about who is allowed to change configuration (i.e., is allowed to install or alter software or its configuration), who has administrative access to production machines and when direct administrative access is allowed.

Interactions with the lifecycle process MUST be logged

REFERENCE ID	LIF-LOG-INTERACT
AREA	Environment
RESPONSIBLE	Administration, Implementation
DESCRIPTION	For instance, an interaction with the lifecycle process are "update service configuration" or "restart service" by an allowed actor. All processes and services must be logged in line with the following requirements LGR-LOG-LAWS and LGR-LOG-MIN-INFO and should be audited by a third-party later.

Configuration management MUST be performed

REFERENCE ID	LIF-CONFIG-MGMT
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	Each newly deployed software defines a new configuration. The different configurations existing during the product's lifetime must be documented. This is essential for debugging and security analysis (i.e., post-mortem) purposes.

It SHOULD be possible to revert to old configurations	
REFERENCE ID	LIF-CONFIG-REV
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	It must be possible to revert older software versions. For example, after a system breach this can be utilized to create a test-environment that duplicates the breached environment.

New configurations MUST be tested prior to Deployment	
REFERENCE ID	LIF-CONFIG-TEST
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	Before a new version is deployed, all test cases must be successfully finished.

Deployed Services MUST be monitored for availability	
REFERENCE ID	LIF-MON-AVAIL
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	The availability of deployed services must be monitored, in case of an error, the workflow defined within lifecycle management must be triggered.

Deployed Services MUST be monitored for security breaches	
REFERENCE ID	LIF-MON-SEC-BREA
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	Deployed services should be periodically checked for security breaches.

Backups MUST maintain data privacy	
REFERENCE ID	LIF-BK-PRIVACY
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	Secure backups must be periodically performed. Backup data should remain private even if it is stored on external untrusted cloud providers.

Backups MUST be integrity-protected	
REFERENCE ID	LIF-BK-INTEGRITY
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	If a backup has been created, it must be able to be reconstructed. It should be possible to test the integrity of a backup without performing the reconstruction operation.

Data MUST be securely deleted before server disposal	
REFERENCE ID	LIF-DEL-BEF-DISP
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	If a server gets out of commission, all data stored on it must be securely deleted.

Administrative functions and security configuration settings MUST be logged	
REFERENCE ID	LIF-LOG-ADMIN
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	Configuration files must be logged and any unexpected modifications should be reported, e.g., by using AIDE (Advanced Intrusion Detection Environment) to detect configuration changes.

5.3 Service Isolation

While software conceptually runs on distinct hosts, deployed software might run on the same host but separated within multiple containers or virtual machines. The requirements in this section ensure that services cannot maliciously interact with other deployed services.

Services MUST be isolated from each other	
Reference ID	SRI-SER-ISOLATE
Area	Environment
Responsible	Administration
Description	Services are the basic unit creating functional components. Each service includes exactly enough functionality to fulfill a related functional requirement (single responsibility principle, cf. ARC-SING-RESP-PR). During execution/runtime multiple services might be deployed on the same physical host. Each deployed service must be isolated from other services running on the same physical host. Typical technologies used for this isolation can be virtual machines, containers, or jailroots.

Communication between services MUST be restricted to minimum required	
Reference ID	SRI-COM-MINIMAL
Area	Service, Environment
Responsible	Architecture, Implementation, Administration
Description	All external communication links (e.g., inbound and outbound communication via ports) must be restricted according to the least privilege principle. Implement least privilege, restrict users to only the functionality, data and system information that is required to perform their tasks.

Service's privileges MUST be restricted	
Reference ID	SRI-SER-PRIVIL
Area	Environment
Responsible	Administration
Description	Service's privileges must be restricted to minimum. This reduces attack surface and prevents unauthorized access to other resources/services. This is the enforcement of the least privilege principle.

Service resources SHOULD be limited	
REFERENCE ID	SRI-SER-RESOURCE
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	Service's resources (CPU, memory, storage, network bandwidth) must be restricted to be within feasible limits. This should prevent overcommitment of available resources.

Service environment MUST be monitored	
REFERENCE ID	SRI-SER-ENV
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	Anomaly detection system must be set in place to detect any deviations from standard behaviour defined in SRI-SER-RESOURCE. Each incident must be logged through the logging component.

Service MUST always fail secure	
REFERENCE ID	SRI-SER-SEC-FAIL
AREA	Service
RESPONSIBLE	Implementation
DESCRIPTION	In case of a failure during runtime a service must fail securely in order to prevent any malicious privilege escalation.

Composite service behaviour model MUST be defined	
REFERENCE ID	SRI-COMPSER-DEF
AREA	Service
RESPONSIBLE	Architecture
DESCRIPTION	Composite services combine calls to other services to provide a new service interface. For each composite service the expected calls to other services have to be documented.

Composite service behaviour model MUST be monitored	
REFERENCE ID	SRI-COMPSER-MON
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	Anomaly detection system must be set in place to detect any deviations from standard behaviour defined in SRI-COMPSER-DEF.

5.4 Logging and Reporting

Providing adequate log information is important for system debugging and auditing. As log data includes much user-related information, its security and privacy is of high importance for the overall system security and privacy.

An Independent storage location for logging data MUST be configured	
REFERENCE ID	LGR-LOG-STORAGE
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	An independent storage location from the service being monitored must be used to avoid disk space conflicts.

Log information MUST be restricted with strong access control mechanisms	
REFERENCE ID	LGR-LOG-ACCESS
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	All logging information should be protected from unauthorized access, e.g., with utilizing encryption or access control. Access should be given only to the admins.

Consistency of logged information MUST be verified during its life cycle	
REFERENCE ID	LGR-LOG-CONSIST
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	Logged information must be verified during both transport and storage time to detect any alteration, e.g., consistency checking by AIDE, signatures, or cryptographic hash functions.

Sensitive information MUST be excluded from the error output	
REFERENCE ID	LGR-OUT-SENS-INF
AREA	Service, Client
RESPONSIBLE	Implementation
DESCRIPTION	No sensitive information (e.g., users personal data) should be included within the error output, and if it is not possible to exclude sensitive information from log files, then access to those log files must be subject to access verification.

Error handlers MUST NOT display debugging or stack trace information	
REFERENCE ID	LGR-OUT-DEB-INF
AREA	Service, Client
RESPONSIBLE	Implementation
DESCRIPTION	Error handlers that do not display debugging or stack trace information should be used.

Apparent tampering events of any sensitive data MUST be logged	
REFERENCE ID	LGR-LOG-TAMPER
AREA	Service, Client
RESPONSIBLE	Implementation, Administration
DESCRIPTION	Unexpected changes of the data during its life cycle must be logged.

Log information retention period MUST be defined	
REFERENCE ID	LGR-LOG-RET-PER
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	Time period after which logged data can be destroyed.

Log information MUST be securely discarded	
REFERENCE ID	LGR-LOG-DELETE
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	After the retention period was reached, log information must be securely discarded.

Logging procedures MUST oblige national and EU legal directives	
REFERENCE ID	LGR-LOG-LAWS
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	Logging processes for acquiring information where provider's or customer's private information is involved must to be in line with national and EU laws.

Periodic reporting MUST be established	
REFERENCE ID	LGR-REP-PERIODS
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	Periodic reporting must be established depending on the criticality level of a service, i.e., annual, monthly, weekly, daily, or hourly report period should be defined with respect to the criticality of a service.

Periodic report form MUST be defined	
REFERENCE ID	LGR-REP-FORM
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	Periodic reports must be provided in an electronic form and as a hard copy.

Minimum log event information set MUST be defined	
REFERENCE ID	LGR-LOG-MIN-INFO
AREA	Client, Service
RESPONSIBLE	Implementation
DESCRIPTION	Log Event Data: This should include the following: 1. Time stamp from a trusted system component 2. Severity rating for each event 3. Tagging of security relevant events, if they are mixed with other log entries 4. Identity of the account/user that caused the event 5. Source IP address associated with the request 6. Event outcome (success or failure) 7. Description of the event

Cryptographic module failures MUST be logged	
REFERENCE ID	LGR-LOG-CRYPTO
AREA	Client, Service
RESPONSIBLE	Implementation, Administration
DESCRIPTION	Any failures by cryptographic modules must be logged in line with the following requirements LGR-LOG-LAWS and LGR-LOG-MIN-INFO.

Logging guidelines SHOULD be documented	
REFERENCE ID	LGR-LOG-DOCUMENT
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	Logging guidelines should be documented. A proper documentation can increase the maintainability of the system.

Location CAN be logged	
REFERENCE ID	LGR-LOG-LOCATION
AREA	Environment
RESPONSIBLE	Implementation, Administration
DESCRIPTION	The location of where the system access has been performed can be logged, e.g., utilizing the mobile device.

Logging CAN be customizable	
REFERENCE ID	LGR-LOG-CUSTOM
AREA	Environment
RESPONSIBLE	Implementation, Administration
DESCRIPTION	The logging level can be customizable. The system administrator can customize the logging, which can be used for maintenance or finding system problems.

Logging language SHOULD be standardized	
REFERENCE ID	LGR-LOG-LANGUAGE
AREA	Client, Service
RESPONSIBLE	Implementation
DESCRIPTION	The language used for the logging should altogether be the same in the system such as English.

5.5 Communication-Related Requirements

Users (or rather their software clients) communicate with the wallet and other services. Thus, the confidentiality and integrity of communication is of high importance for CREDENTIAL.

Communication data MUST be kept confidential

REFERENCE ID	COM-CONFIDENTIAL
AREA	Service, Client
RESPONSIBLE	Architecture, Implementation
DESCRIPTION	Communication may include sensitive data. To prevent data leaks all communication must be performed through private channels. The requirements within “Transport Level Security” further detail secure channels.

Communication data’s integrity MUST be protected

REFERENCE ID	COM-INTEGRITY
AREA	Service, Client
RESPONSIBLE	Architecture, Implementation
DESCRIPTION	In addition to COM-CONFIDENTIAL the integrity of communication data must also be protected. Otherwise, an attacker could maliciously alter communication and thus exploit potential flaws at the communication endpoints. The requirements within “Transport Level Security” further detail secure channels.

Communication partners MUST be authenticated

REFERENCE ID	COM-AUTHENTICITY
AREA	Service, Client
RESPONSIBLE	Architecture, Implementation
DESCRIPTION	Communication partners must have the means of identifying and validating their communication partner’s identity. Clients must have means to verify that they are communication with authentic service providers and/or wallets. The requirements within “Transport Level Security” further detail authentic channels.

Communication sessions MUST be logged	
REFERENCE ID	COM-LOG-SESSIONS
AREA	Service
RESPONSIBLE	Implementation
DESCRIPTION	Communication sessions data must be logged in line with the following requirements LGR-LOG-LAWS and LGR-LOG-MIN-INFO.

Communication failures MUST be logged	
REFERENCE ID	COM-LOG-FAILURES
AREA	Service
RESPONSIBLE	Implementation
DESCRIPTION	Failure in communication must be logged in line with the following requirements LGR-LOG-LAWS and LGR-LOG-MIN-INFO.

5.5.1 Transport Level Security

Transport Level Security concerns itself with the low-level implementation of transport directives.

All Communication MUST utilize TLS	
REFERENCE ID	COM-TLS-USE
AREA	Service, Client
RESPONSIBLE	Architecture, Implementation, Administration
DESCRIPTION	All communication MUST be encrypted and authenticated utilizing Transport Layer Security (TLS) v1.2 or higher. The encrypted connection has to be adequately configured so that it is not possible to create a weak connection.

Connections MUST support Perfect Forward Secrecy	
REFERENCE ID	COM-TLS-PFS
AREA	Service, Client
RESPONSIBLE	Architecture, Implementation, Administration
DESCRIPTION	Perfect Forward Secrecy is a property of communication protocols which protects confidentiality of communication from compromise of long-term key material in the future. Each newly created session is based on a new generated key pair. That is, if an attacker is able to compromise the key material of one session, he must not be able to use this to break the confidentiality of previous sessions.

Desktop/Mobile Clients SHOULD use certificate pinning	
REFERENCE ID	COM-TLS-CERT-CLI
AREA	Client
RESPONSIBLE	Implementation
DESCRIPTION	Certificate pinning is a security mechanism where the client certificate is being attached to the client application. This concept is a possibility to significantly decrease the probability of a successful man in the middle attack. Certificate whitelisting of all valid certificates is used in this concept.

Web-Browser/Servers SHOULD use certificate pinning	
REFERENCE ID	COM-TLS-CERT-SER
AREA	Service
RESPONSIBLE	Implementation, Administration
DESCRIPTION	Webserver should utilize HTTP Strict Transport Security (HSTS) to implement certificate pinning.

Utilized Ciphers SHOULD be AEAD	
REFERENCE ID	COM-TLS-AEAD
AREA	Service, Client
RESPONSIBLE	Implementation, Administration
DESCRIPTION	Authenticated Encryption with Associated Data (AEAD) is a combination of encryption together with integrity and authenticity verification. Therefore, AEAD scheme sustains the confidentiality, integrity and authenticity of the data processed.

5.5.2 User Notifications

Asynchronous user notifications (e.g., Emails or Mobile Device Notifications) are common in modern applications. Those may contain sensitive information and thus are candidates for our requirements.

Sensitive data SHALL NOT be sent via email to the user

REFERENCE ID	COM-UN-NO-EMAIL
AREA	Service
RESPONSIBLE	Implementation
DESCRIPTION	User notification sent via email SHALL NOT contain sensitive information such as credentials as plaintext. Otherwise, sensitive data could easily be disclosed, leading to severe security impacts.

Email-based user notifications SHOULD be signed and encrypted

REFERENCE ID	COM-UN-ENC-MAIL
AREA	Service, Client
RESPONSIBLE	Implementation
DESCRIPTION	It is RECOMMENDED that if the user gets notification emails, those emails SHOULD be signed and encrypted. It is recommended because sensitive data SHALL NOT be sent via email anyway.

Sensitive data MUST NOT be displayed within notifications

REFERENCE ID	COM-UN-APP-NOTIF
AREA	Client
RESPONSIBLE	Implementation
DESCRIPTION	Client Applications have multiple means of providing user notification (e.g., iOS Notification Center, Google Notification Services). Notifications are commonly handed over to a vendor-specific notification platform which in turn forwards the notification. Thus, notifications MUST NOT contain sensitive information.

5.6 User- and Session-Management

As already discussed in Section 4.6, this category is mainly concerned with authentication, authorization, and session management. While certain requirements are very generic and apply to many web applications, other requirements are specific to data sharing platforms and identity providers.

The requirements listed in this section were partially inspired by [OWA16c].

Multi-factor authentication SHOULD be implemented	
REFERENCE ID	USM-2FACTOR-AUTH
AREA	Service
RESPONSIBLE	Architecture, Implementation
DESCRIPTION	Multi-factor authentication guarantees significantly higher security guarantees than, e.g., username/password authentication. The potential usability drawback should be accepted.

There SHOULD be no direct interaction between server application and IdP	
REFERENCE ID	USM-COM-SERV-IDP
AREA	Wallet, Service
RESPONSIBLE	Architecture
DESCRIPTION	In case of a direct interaction, the IdP component would learn the application visited by the user, thereby potentially posing a severe privacy problem.

The IdP SHOULD not be able to identify the application used by the user	
REFERENCE ID	USM-HIDE-APP
AREA	Client, Wallet
RESPONSIBLE	Architecture
DESCRIPTION	Identifying the application used by the user might provide meta data to the IdP, which could be used to obtain a detailed behavior pattern of the user. This might lead to severe privacy problems.

User authentications MUST have an expiration date or expiration event	
REFERENCE ID	USM-UAUTHENT-EXP
AREA	Client, Wallet
RESPONSIBLE	Implementation
DESCRIPTION	User authorisation credentials should either expire, e.g., one year after they were issues, or upon a specific event such as upon reaching a certain age. The former guarantees that an adversary does not have unlimited time to use a stolen/lost credential, the latter guarantees that credentials cannot be abused by the legitimate owners, either.

Granted authorizations MUST have an expiration date or an expiration event	
REFERENCE ID	USM-UAUTORIZ-EXP
AREA	Client, Wallet
RESPONSIBLE	Implementation
DESCRIPTION	For reasons similar to USM-UAUTHENT-EXP, granted authorizations should automatically expire and require a re-confirmation by the data owner.

For each access to restricted resources, the user SHOULD be authenticated	
REFERENCE ID	USM-REQ-AUTHENT
AREA	Service
RESPONSIBLE	Implementation
DESCRIPTION	In general, access to restricted resources SHOULD always require a preceding authentication. However, for privacy reasons it might sometimes be preferable to not let the user authenticate himself, but only let the user prove to the server that he is allowed to access the resource, without revealing its full identity. The trade-off between privacy and the impossibility of logging accesses to a restricted resource needs to be weighted carefully against each other.

For each access to restricted resources, the user MUST have the corresponding authorizations	
REFERENCE ID	USM-REQ-AUTHORI
AREA	Service
RESPONSIBLE	Implementation
DESCRIPTION	This is the core aspect of authorization. No user must be able to access data which is neither public, nor owned by the user himself, nor was explicitly shared with this specific user.

User authentication credentials MUST be revokable	
REFERENCE ID	USM-CREDS-REV
AREA	Wallet
RESPONSIBLE	Architecture, Administration
DESCRIPTION	In case of theft of authentication data or upon abuse, authentication credentials need to be revokable by the user and/or the wallet.

Granted authorizations MUST be revokable	
REFERENCE ID	USM-AUTHOR-REV
AREA	Wallet
RESPONSIBLE	Architecture, Administration
DESCRIPTION	Users must be able to withdraw authorizations provided to other stakeholders of the system.

User enumeration MUST NOT be possible	
REFERENCE ID	USM-USER-ENUM
AREA	Wallet
RESPONSIBLE	Implementation
DESCRIPTION	User enumeration allows an adversary to check for which usernames there exists accounts on a system. To avoid this attack vector, in particular unsuccessful login attempts for existing and non-existing user names MUST NOT be distinguishable. Also, upon registration and password recovery, it MUST NOT be leaked whether an account for this username or email address exists on the system. Instead, generic error messages MUST be used.

Logs of access to user data SHOULD be visible to the user	
REFERENCE ID	USM-LOG-ACCESS
AREA	Client, Wallet
RESPONSIBLE	Architecture
DESCRIPTION	A user SHOULD have the possibility to check when and which data was accessed by whom. This is important to prevent misuse of access rights by other users.

Unsuccessful login attempts and access attempts SHOULD be logged	
REFERENCE ID	USM-LOG-FAILS
AREA	Wallet
RESPONSIBLE	Implementation, Administration
DESCRIPTION	A user SHOULD be able to see when unprivileged users tried to access his account or his data.

Anomaly detection to detect irregular login attempts MAY be implemented	
REFERENCE ID	USM-ANOMALY
AREA	Wallet
RESPONSIBLE	Administration
DESCRIPTION	The wallet MAY inform the user, e.g., upon login attempts from untypical geographic locations. In this case, the wallet MAY also request the user for additional authentication data to ensure that the legitimate owner of the account is accessing.

Logins SHOULD be throttled after too many failed attempts	
REFERENCE ID	USM-THROTTELING
AREA	Wallet
RESPONSIBLE	Implementation
DESCRIPTION	To prevent brute force attacks on the user's login credentials (e.g., password), login attempts should be throttled after a small number of invalid attempts.

The IdP SHOULD be able to suspend/ban specific users	
REFERENCE ID	USM-SUSPEND
AREA	Wallet
RESPONSIBLE	Architecture
DESCRIPTION	For instance, if a user is regularly trying to access other users' data without having the authorization to do so, the CREDENTIAL wallet SHOULD be able to suspend the user from the service.

5.6.1 Requirements to Session Management

A session is a sequence of activities and messages being sent between a client and a server. Session management is the process of keeping track of all these activities. In the following, we specify non-functional security requirements specific to this process. We note that these requirements are not only important in the context of CREDENTIAL, but also need to be considered in virtually any web application.

Secure session IDs MUST be used	
REFERENCE ID	USM-SID-SECURE
AREA	Service
RESPONSIBLE	Implementation
DESCRIPTION	Secure session IDs SHOULD not be fingerprinted, i.e., they SHOULD not reveal the technologies used by the web application. Session IDs MUST be long enough to disable an adversary from brute forcing session IDs when searching for existing valid ones. For the same reason, session IDs also need to have high entropy (at least 128 bit) in order to prevent guessing attacks. Therefore, a cryptographically secure (pseudo) random generator MUST be deployed. Finally, the content of the session ID MUST NOT reveal any information to an attacker.

Session IDs MUST be fresh	
REFERENCE ID	USM-SID-FRESH
AREA	Service
RESPONSIBLE	Implementation
DESCRIPTION	A fresh session ID MUST be computed upon every login.

Session ID guessing and brute forcing attacks SHOULD be detected	
REFERENCE ID	USM-SID-GUESSING
AREA	Service, Client
RESPONSIBLE	Implementation
DESCRIPTION	If an adversary tries to brute force or guess a valid session ID, many sequential requests using different session IDs will be received by the application within a short time period, potentially from a small set of IP addresses. If the attacker tries to break the unpredictability of session IDs, he also needs to send numerous requests. Both scenarios SHOULD be detected by the server, and countermeasures (e.g., blocking the IP address) SHOULD be taken. The threshold for an acceptable frequency needs to be define on a per use-case basis.

Session timeouts in case of inactivity SHOULD be implemented	
REFERENCE ID	USM-SID-TO-IDLE
AREA	Service
RESPONSIBLE	Implementation
DESCRIPTION	In case of inactivity, a session should automatically be closed and invalidated. This reduced an adversary's chances to hijack a given session, e.g., by guessing its ID. Note that it is crucial to enforce this timeout on the server's side, as an attacker might be able to manipulate the timing on the client's side.

Absolute session timeouts SHOULD be implemented	
REFERENCE ID	USM-SID-TO-ABS
AREA	Service, Client
RESPONSIBLE	Implementation
DESCRIPTION	Sessions should automatically be ended by the server after predefined time periods, and require new logins. In this way, the adversary's capabilities after a successful hijack of a session can be limited.

Sessions SHOULD automatically be ended upon pre-defined events	
REFERENCE ID	USM-SID-LOGOUT
AREA	Service, Client
RESPONSIBLE	Implementation
DESCRIPTION	A session SHOULD be terminated upon certain pre-defined events, which may depend on the concrete circumstances. In particular, it is recommended that to capture termination events (e.g., closing a browser), and terminate the session by emulating an active logout by the user. For high-security data, also changing to another application might cause a session termination.

5.7 Server-Specific Requirements

Services will process and store sensitive user data. A compromised service can tamper its own user data or try to access sensitive data of other services. Thus, CREDENTIAL puts special focus upon securing server-side services.

All input MUST be checked for common injection vectors	
REFERENCE ID	SER-CHECK-INPUT
AREA	Service
RESPONSIBLE	Implementation
DESCRIPTION	According to the OWASP Top 10 missing input validation is one of the most common vulnerabilities. Depending upon the chosen technology stack input must be validated against XSS, SQLi, XML, LFI, RFI attack vectors. Validate all client provided data before processing, including all parameters, URLs and HTTP header content (e.g., Cookie names and values). Be sure to include automated post backs from JavaScript, Flash or other embedded code.

User-generated files with sensitive content MUST be encrypted	
REFERENCE ID	SER-ENC-USER-DAT
AREA	Service
RESPONSIBLE	Architecture
DESCRIPTION	If users can upload data, this data's confidentiality must be provided. At least data at rest should be encrypted. Do not store passwords, connection strings or other sensitive information in clear text or in any non-cryptographically secure manner on the client side. This includes embedding in insecure formats like: MS viewstate, Adobe flash or compiled code.

A single user MUST be deletable (legal requirement)	
REFERENCE ID	SER-USER-DELETE
AREA	Service
RESPONSIBLE	Architecture
DESCRIPTION	Legislation can mandate that personally identifiable information must be deleted upon user's request. It thus must be possible to delete all personal data of a given user identity.

Input/Output validation SHOULD be centralized

REFERENCE ID	SER-VALIDATE-IO
AREA	Service
RESPONSIBLE	Implementation
DESCRIPTION	There should be a centralized input validation and output sanitation routine for all operations.

Encoding MUST be validated

REFERENCE ID	SER-VALIDATE-ENC
AREA	Service
RESPONSIBLE	Implementation
DESCRIPTION	Verify that header values in both requests and responses contain only ASCII characters.

Rate-limits SHOULD be introduced

REFERENCE ID	SER-RATE-LIMITS
AREA	Service
RESPONSIBLE	Implementation
DESCRIPTION	Limit the number of transactions a single user or device can perform in a given period of time. The transactions/time should be above the actual business requirement, but low enough to deter automated attacks.

Temporary Files SHOULD be encrypted

REFERENCE ID	SER-ENC-TEMPS
AREA	Service
RESPONSIBLE	Implementation
DESCRIPTION	Protect all cached or temporary copies of sensitive data stored on the server from unauthorized access and purge those temporary working files a soon as they are no longer required.

Unnecessary documentation SHOULD be removed

REFERENCE ID	SER-DEL-DOCU
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	Remove unnecessary application and system documentation as this can reveal useful information to attackers. Remove comments in user accessible production code that may reveal backend system or other sensitive information.

Software and Libraries MUST use latest patch level

REFERENCE ID	SER-UPDATE-SW
AREA	Service, Environment
RESPONSIBLE	Administration, Implementation
DESCRIPTION	Ensure servers, frameworks and system components are running the latest approved version. Ensure servers, frameworks and system components have all patches issued for the version in use.

Uploaded Files MUST be stored outside the web-context

REFERENCE ID	SER-UPL-STORE
AREA	Service
RESPONSIBLE	Implementation, Administration
DESCRIPTION	Do not save files in the same web context as the application. Files should either go to the content server or in the database.

Uploaded Files MUST be verified

REFERENCE ID	SER-UPL-VERIF
AREA	Service
RESPONSIBLE	Implementation
DESCRIPTION	Verify that files obtained from semi-trusted sources are validated to be of the expected file type and scanned by antivirus scanners to prevent upload of known malicious content.

Uploaded Data MUST NOT be executed	
REFERENCE ID	SER-UPL-NO-EXEC
AREA	Service
RESPONSIBLE	Implementation
DESCRIPTION	Untrusted data must not be used within inclusion, class loader or reflection capabilities (e.g., EXIF data) to prevent local/remote file inclusion. The application must not executed executed data from semi-trusted sources.

5.8 Client-Side Concerns

User will primarily interact with CREDENTIAL through client applications. This places high security requirements upon the corresponding client applications as they are uniquely placed to loose sensitive user data.

Sensitive data SHALL NOT be included in the installation package	
REFERENCE ID	CLI-INS-SENS-DAT
AREA	Client
RESPONSIBLE	Implementation
DESCRIPTION	The installation package SHALL NOT contain sensitive data to prevent the loss of sensitive data at this point. In this context, sensitive data can be for example key material or access credentials. Sensitive data in the installation package are easily to disclose, therefore, this would be a vulnerability.

Integrity and authenticity of the installation package MUST be provided	
REFERENCE ID	CLI-INS-AUTH-INT
AREA	Environment, Client
RESPONSIBLE	Implementation
DESCRIPTION	The installation package must be signed by a trusted party. Clients must verify that 1) the signature belongs to a trusted party, 2) no additional software was added and 3) the signature has not been altered.

Sensitive data MUST NOT be stored in insecure application caches	
REFERENCE ID	CLI-DATA-CACHE
AREA	Environment, Client
RESPONSIBLE	Implementation
DESCRIPTION	Sensitive data MUST NOT be stored in insecure application caches such as the web browser cache. Application caches are an inappropriate place to store sensitive data, therefore, this should be avoided.

Sensitive Data MUST be stored encrypted	
REFERENCE ID	CLI-DATA-ENC
AREA	Client
RESPONSIBLE	Architecture, Implementation
DESCRIPTION	All sensitive Data MUST be encrypted prior to storage. For example, iOS devices should use SQLcipher, desktop applications should use standard cryptographic libraries to provide encryption services.

Secure storage SHOULD be used if it is offered by the platform	
REFERENCE ID	CLI-SEC-STORAGE
AREA	Environment, Client
RESPONSIBLE	Implementation
DESCRIPTION	If offered by the platform and supported by the operating system, secure storage SHOULD be used to store, in particular, sensitive data. For example on iOS devices, iOS's secure storage facilities (keychain with hardware support) should be utilized.

Client SHOULD use trusted execution	
REFERENCE ID	CLI-TRUSTED-EXEC
AREA	Environment, Client
RESPONSIBLE	Implementation
DESCRIPTION	The client SHOULD use the trusted execution environment (TEE), for security important parts of the client application, if this is offered by the hardware. TEE offers a secure and isolated execution environment that protects the data confidentiality and integrity. Moreover, the code being processed inside this environment is protected as well. Parts of the clients application, which requires the highest security SHOULD be considered to be run in the TEE, if this is possible in a efficient manner.

Operating system and system libraries MUST be up to date	
REFERENCE ID	CLI-OS-UPDATES
AREA	Environment
RESPONSIBLE	Implementation, Administration
DESCRIPTION	In order to keep the security in our system as high as possible, the operating system as well as mandatory libraries MUST use a maintained version (with security updates). Newly available software updates SHOULD be updated as soon as possible.

Clients SHOULD use secure backup	
REFERENCE ID	CLI-SEC-BACKUP
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	All backup data must be encrypted before it is transferred to external storage providers. Cloud-based backups, where the client-side encryption cannot be validated (e.g., default iOS or Android backup) MUST be disabled.

5.8.1 Specific Requirements due to Mobile Clients

Mobile clients operate applications within defined application execution environments (also called sandboxes). This allows for additional security requirements that can increase the overall system security. Note, that all "Client-Side Concerns" are still valid for mobile clients.

Mobile devices SHOULD use authentication	
REFERENCE ID	CLI-MOB-AUTH-DEV
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	The user SHOULD use TouchID with a longer passcode in lieu of a 4-digit PIN. It is recommended that the user sets a secure passcode which improves security like unauthorized access.

Users MUST authenticate when opening the application	
REFERENCE ID	CLI-MOB-AUTH-APP
AREA	Client
RESPONSIBLE	Implementation
DESCRIPTION	The user MUST authenticate himself when opening the application (by default). This authentication mechanism can vary from device to device.

No sensitive information SHOULD be shown in the application switcher preview	
REFERENCE ID	CLI-MOB-APP-SWIT
AREA	Client
RESPONSIBLE	Implementation
DESCRIPTION	The application switcher for example on iOS and Android devices, displays a preview of the application. This screenshot SHOULD be blacked out or a thumbnail SHOULD be used instead to prevent sensitive information from being shown.

5.9 Cryptographic Requirements

In the following, we define cryptographic requirements. Even though these requirements are mainly generic, they are of prime importance for securely realizing the vision of CREDENTIAL. Indeed, numerous real-world incidents show that not following these recommendations can have severe negative impacts.

Some of the following requirements have been inspired by [OWA16a].

Key sizes MUST be sufficiently large

REFERENCE ID	CRY-KEY-SIZES
AREA	Client, Service
RESPONSIBLE	Implementation
DESCRIPTION	Key sizes for each cryptographic primitive must be chosen such that security can be guaranteed for the intended time period. For recommendations, see, e.g., [Gir16].

Passwords MUST NOT be stored in the plain

REFERENCE ID	CRY-PW-PLAIN
AREA	Service
RESPONSIBLE	Implementation
DESCRIPTION	Passwords stored in the plain pose a huge security risk, e.g., in the case of a data breach.

Stored passwords MUST be salted

REFERENCE ID	CRY-PW-SALT
AREA	Service
RESPONSIBLE	Implementation
DESCRIPTION	In case of a data breach, passwords can often be recovered from unsalted hashes, e.g., using rainbow tables. The used salt does not need to be secret and can be stored together with the username and the hashed and salted password.

Dedicated hashing algorithms SHOULD be used for hashing passwords	
REFERENCE ID	CRY-PW-HASH
AREA	Service
RESPONSIBLE	Implementation
DESCRIPTION	Even for salted password hashes, brute force attacks are often too powerful and can be used to recover the password. Therefore, dedicated hash algorithms developed for hashing passwords should be used, e.g., scrypt [DGK+12]

Fresh and long salts MUST be used for every password	
REFERENCE ID	CRY-SALT-FRESH
AREA	Service
RESPONSIBLE	Implementation
DESCRIPTION	If too short salts are used, an attacker can pre-compute rainbow tables for every single salt, thereby violating security. On the other hand, even if long salts are reused, an attacker can decide - from the hash value – whether two passwords of different users are the same or different. This would, e.g., allow for the use of lookup tables to recover the password.

Private keys SHOULD be stored in secure environments	
REFERENCE ID	CRY-KEY-SEC-STOR
AREA	Client, Service
RESPONSIBLE	Implementation, Architecture
DESCRIPTION	The cryptographic key material should not be accessible, e.g., by malware, but only by legitimate processes.

Private keys SHOULD never be cached	
REFERENCE ID	CRY-KEY-CACHE
AREA	Client, Service
RESPONSIBLE	Implementation
DESCRIPTION	Experimental results show that cryptographic keys can efficiently be located in the cache. A direct mitigation strategy is to not cache sensitive key material.

Private keys MUST be stored separately from encrypted data	
REFERENCE ID	CRY-DATA-KEY-SEP
AREA	Client, Service
RESPONSIBLE	Architecture
DESCRIPTION	In case of a data breach, the risk of leaking the decryption key together with the encrypted data must be minimized.

Cryptographic keys SHOULD NOT be reused for different purposes	
REFERENCE ID	CRY-KEY-SEPARAT
AREA	Client, Service
RESPONSIBLE	Implementation
DESCRIPTION	Reusing cryptographic keys for different purposes (e.g., signing and encryption) requires profound expertise, and must not be done for most instantiations. Also using the same keys in different contexts poses the risk to harm multiple services if the key is leaked, similar to the case of reused passwords.

Secure random number generators MUST be used	
REFERENCE ID	CRY-PRG-SECURE
AREA	Environment
RESPONSIBLE	Administration
DESCRIPTION	The used random numbers for all cryptographic purposes (e.g., initialization vectors for encryption, cryptographic keys, etc.) MUST be generated in a cryptographically secure fashion. It is recommended to use certified generators only, and/or to test their randomness using, e.g., the NIST RNG Test Tool to assess the quality of the generated random numbers.

Secure instantiations of primitives MUST be used	
REFERENCE ID	CRY-INST-SECURE
AREA	Client, Service
RESPONSIBLE	Implementation
DESCRIPTION	Only recommended instantiations should be used. For instance, for hash functions SHA-2 or SHA-3 is recommended as a hash function, while the usage of MD5 for cryptographic purposes is prohibited. Recommended algorithms can be found, e.g., in [ENI14].

Secure implementations of primitives MUST be used

REFERENCE ID	CRY-IMPL-SECURE
AREA	Environment
RESPONSIBLE	Administration, Implementation
DESCRIPTION	Secure and well-tested implementations MUST be used in order to avoid implementation or installation flaws.

Deployed implementations SHOULD guarantee side-channel resistance

REFERENCE ID	CRY-IMPL-SIDECHA
AREA	Environment
RESPONSIBLE	Administration, Implementation
DESCRIPTION	Side channels attacks do not attack the specification of a cryptographic primitive. Rather, they are based on any information that can be obtained from the physical implementation such as energy consumption, computing time, or radiation. For many implementations there exist sophisticated attacks. Avoid side channel vulnerabilities requires special attention and training of the software developer.

6 Conclusion and Future Work

In this document we performed a preliminary risk assessment for CREDENTIAL based on a high level DFD coming from the currently envisioned functioning of the single components. Based on this analysis, and based on a review of the ISO and OWASP security requirement catalogues, we collected approximately 120 security requirements, most of which are generic and also apply to many other secure software development projects. This preliminary requirements catalogue will be used as a basis for specifying and designing the overall architecture of CREDENTIAL. It will later be extended and enhanced (through D2.5) based on the precise definition of the pilots and scenarios implemented within CREDENTIAL.

In the following, we give a short overview of planned future work extending the results of the current document.

Pilot- and use-case specific security requirements. At the time of writing this document, the precise specification of the CREDENTIAL pilots was still ongoing. Therefore, the current version of the document mainly focuses on generic non-functional security requirement. A main extension of this document will therefore be given by adding pilot specific security requirements in D2.6. This extension will consider the detailed specification of the CREDENTIAL pilots and use-cases presented in D2.1 and D6.1. In particular, these specific requirements will cover aspects coming from the usage of potentially insecure mobile devices, or the exchange of confidential data across different domains with different security guarantees.

Additional requirements might also result from concrete design choices made during the implementation process of the different CREDENTIAL components.

Requirement verification questions. The current version of the document mainly lists a broad range of security requirements, including potential risks if they are not satisfied. To increase the usability of this document for the reader, we plan to further add verification and falsification questions at least to those requirements with the most significant impacts in case of security breaches.

By verification questions, we mean questions for sufficient conditions: if these questions can all be answered by “yes” by a software engineer, system administrator, etc., he can be assured that the requirement is indeed satisfied by his system. On the other hand, falsification questions ask for necessary conditions: if any of these question is answered by “yes”, the responsible person knows that the requirement is definitely not satisfied by his system, and thus there is urgent need for action.

Relation to existing standard. Related to the validation questions, we are considering to add a mapping from our security requirements catalogue to existing standards and vice versa. In contrast to Appendix B, this mapping would be done on a requirements level (not on a category level), at least for a chosen set of central requirements. This will simplify the transition from one catalogue to another one, as one does no longer need to re-evaluate every single requirement because of clearly communicated implications between the different standards.

List of References

[AI13] ANSI and ISA. Security for industrial automation and control systems terminology, concepts, and models. ANSI/ISA 62443 series 62443, International Society for Automation (ISA) and American National Standards Institute (ANSI), 2013.

[Bra97] Scott Bradner. Key words for use in RFCs to Indicate Requirement Levels. Network Working Group, Request for Comments: 2119, 1997.

[BT10] Elisa Bertino and Kenji Takahashi. *Identity Management: Concepts, Technologies, and Systems*. Artech House, 2010.

[CER16] CERT. SEI CERT Coding Standards. https://www.securecoding.cert.org/, last accessed: 2016/03/25, 2016.

[CGB+02] Paul Clements, David Garlan, Len Bass, Judith Stafford, Robert Nord, James Ivers, and Reed Little. *Documenting software architectures: views and beyond*. Pearson Education, 2002.

[CNI12] CNIL. Methodology for Privacy Risk Management. Technical report, Commission Nationale de l'Informatique et des Liberté, 2012.

[DGK+12] Markus Dürmuth, Tim Güneysu, Markus Kasper, Christof Paar, Tolga Yalçin, and Ralf Zimmermann. Evaluation of Standardized Password-Based Key Derivation against Parallel Processing Platforms. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *ESORICS*, volume 7459 of *LNCS*, pages 716–733. Springer, 2012.

[DH11] Marnix Dekker and Giles Hogben. Appstore security, 5 lines of defence against malware. Technical report, ENISA - European Union Agency for Network and Information Security, 2011.

[eID15] eIDAS. eIDAS Technical Specifications v1.0. https://joinup.ec.europa.eu/software/cefeid/document/eidas-technical-specifications-v10, last accessed: 2016/03/31, 2015.

[ENI06] ENISA. Risk management: Implementation principles and inventories for risk management/risk assessment methods and tools. Technical report, ENISA - European Union Agency for Network and Information Security, 2006.

[ENI14] ENISA. Algorithms, key size and parameters report – 2014. Technical report, ENISA - European Union Agency for Network and Information Security, 2014.

[Gir16] Damien Giry. Bluekrypt | cryptographic key length recommendation. https://www.keylength.com/, 2016.

[GMT+15] Alberto Crespo García, Nicolás Notario McDonnell, Carmela Troncoso, Daniel Le Métayer, Inga Kroener, David Wright, José María del Álamo, and Yod Samuel Martín. Privacy and Security-by-design Methodology. PRIPARE – *PReparing Industry to Privacy-by-design by supporting its Application in REsearch*. Deliverable D1.2, 2015.

[ISO13] ISO. ISO/IEC 27001 - Information technology, Security techniques, Information security management systems and Requirements. ISO 27000 family 27001, International Organization for Standardization (ISO), 2013.

[JJ10] Michael N. Johnstone and Michael N. Johnstone. Threat Modeling with Stride and UML. In *Australian Information Security Management Conference*, pages 18–27, 2010.

[LeB07] David LeBlanc. DREADful. https://blogs.msdn.microsoft.com/david_leblanc/, last accessed: 2016/03/31, 2007.

[LTM$^+$11] Fang Liu, Jin Tong, Jian Mao, Robert Bohn, John Messina, Lee Badger, and Dawn Leaf. NIST Cloud Computing Reference Architecture. *NIST special publication*, 500-292, 2011.

[LY98] Søren Lauesen and Houman Younessi. Six Styles for Usability Requirements. In Eric Dubois, Andreas L. Opdahl, and Klaus Pohl, editors, *Requirements Engineering: Foundation for Software Quality – REFSQ 1998*, pages 155–166. Presses Universitaires de Namur, 1998.

[Mar09] Robert C. Martin. *Clean Code - a Handbook of Agile Software Craftsmanship*. Prentice Hall, 2009.

[Mic08] Microsoft. Introduction to Microsoft Security Development Lifecycle (SDL), 2008.

[Mic16] Microsoft. Microsoft threat modeling tool 2016. https://www.microsoft.com/en-us/download/details.aspx?id=49168, 2016.

[NIS03] SP NIST. Nist 800-53 - recommended security controls for federal information systems. Technical Report NIST 800-53, National Institute of Standards and Technology, 2003.

[OWA10] OWASP. Secure Coding Practices Quick Reference Guide. https://www.owasp.org/, 2010.

[OWA16a] OWASP. Cryptographic Storage Cheat Sheet. https://www.owasp.org/, last accessed: 2016/03/22, 2016.

[OWA16b] OWASP. Risk Rating Methodology. https://www.owasp.org/, last accessed: 2016/03/25, 2016.

[OWA16c] OWASP. Session Management Cheat Sheet. https://www.owasp.org/, last accessed: 2016/03/16, 2016.

[OWA16d] OWASP. Threat Risk Modeling. https://www.owasp.org/, last accessed: 2016/03/25, 2016.

[SAF16] SAFEcrypto. Secure Architectures of Future Emerging Cryptography. http://www.safecrypto.eu/, ICT-644729, 2016.

[SGF02] Gary Stoneburner, Alice Y. Goguen, and Alexis Feringa. Sp 800-30. risk management guide for information technology systems. Technical report, Gaithersburg, MD, United States, 2002.

[STO16] STORK. Secure idenTity acrOss boRders linKed 2.0. https://www.eid-stork2.eu/, INFSO-ICT-PSP-297263, 2016.

[TJ09] Julian Talbot and Miles Jakeman. *Security Risk Management Body of Knowledge*. John Wiley and Sons Ltd, 1 edition, 2009.

[Tor15] Jacob Torrey. HARES: Hardened Anti-Reverse Engineering System. Technical report, Assured Information Security, Inc., 2015.

A Low-Level Cloud Reference Architecture

To avoid any potential ambiguities concerning the semantics of certain terms, we next briefly summarize the National Institute of Standards and Technology (NIST) Cloud Computing Architecture [LTM+11], shown in Figure 7, as the reference architecture for this work. The NIST conceptual reference model presents an high-level overview of the cloud computing reference architecture model that identifies major actors, their activities and functions. There are five major actors utilized by the NIST model: *cloud consumer*, *cloud provider*, *cloud carrier*, *cloud auditor* and *cloud broker*.

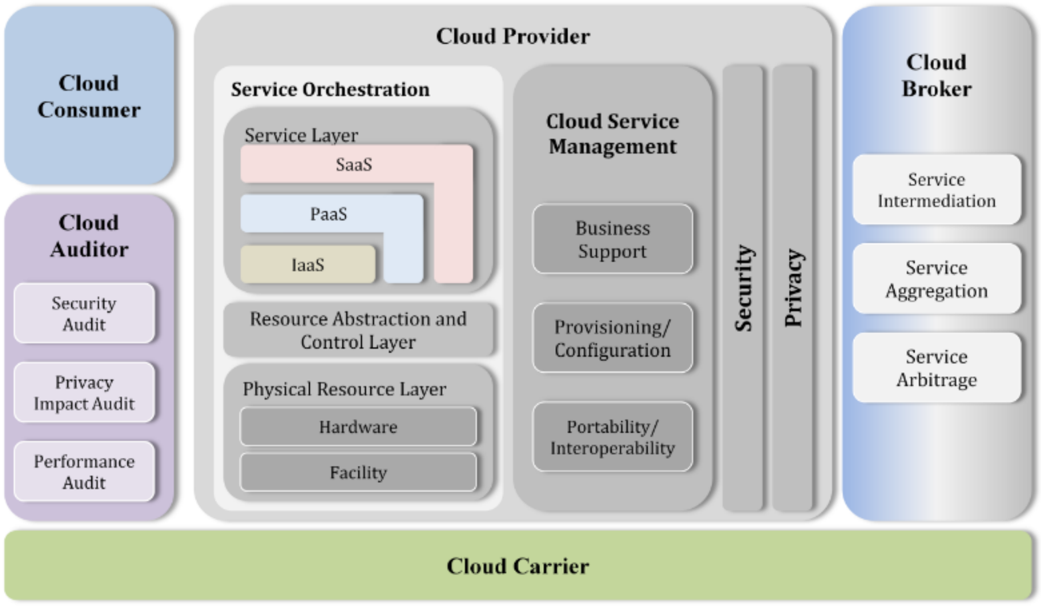


Figure 7: NIST Cloud Computing Reference Architecture

Cloud Provider is a person, organization, or entity that acquires, manages and provisions the computing infrastructure required for utilizing services by the following five principles: *on demand self-service deployment and maintenance*, *ubiquitous network access*, *resource pooling*, *rapid elasticity*, and *measured service*.

Cloud Consumer is the principal stakeholder that takes the leverage of the unique service delivery model offered by a cloud provider. Due to their unique roles, cloud consumer and cloud provider present the most essential actors of the cloud reference architecture. Furthermore, cloud provider and cloud consumer formalize their relationship via the Service Level Agreement (SLA) used to specify the technical performance requirements that a cloud provider has to provide.

Cloud Carrier is a third party, person, organization, or entity, which acts as an intermediary for providing connectivity and transport the services offered by a cloud provider to a cloud consumers or even other cloud providers. The business relationship of a cloud provider and cloud carrier is also formalized an SLA to ensure that cloud carrier consistently providers its services.

Cloud Auditor is a third party, person, organization, or entity, that can perform an independent examination of both cloud service and cloud infrastructure with the intent to express an opinion thereon. Audits are commonly performed to verify conformance to standards through extensive reviews of objective evidence with regards to security controls, privacy impact, or performance. Security audits perform assessment with respect to security controls to determine to which extent security mechanism are implemented to protects information and services of both cloud provider and cloud consumer.

Cloud Broker is a third party, person, organization, or entity that manages the use, performance and delivery of cloud services between variety of cloud providers for building a hybrid cloud model for utilizing services.

B Related Requirements Standards

We evaluate and outline the most relevant standards, research projects, best practices and guidelines that share the common ground for defining security related requirements. The contribution of the mentioned initiatives differs depending on the perspective and motivation for defining particular requirements. Therefore, we use those requirements as the foundation and source for our requirement elicitation. In our final deliverable **D2.5** finalize our requirement elicitation by tailoring our requirements to the use case scenarios defined in the pilot use-cases from **D2.1** (“Scenarios and Use-Cases”) and **D6.1** (“Pilot Use Case Specification”). Among the evaluated standards and best practices we highlight ISO 27001 and OWASP Software Architecture Verification Standard as the most prominent and widely accepted standards. The ISO/IEC 27001, together with its successor ISO/IEC 27002 that provides more detailed description of security controls, is the most comprehensive framework used for evaluating Information Security Management System compliance. The Software Architecture Verification Standard from OWASP is more focused on application security that provides at the same time a security metric and secure development guideline. Essentially, both standards provide a strong focus on security related requirements that we use as a strong basis for elicitation of our requirements with respect to security.

B.1 ISO/IEC 27001/2 - Information technology, Security techniques, Information security management systems and Requirements

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) in ISO/IEC 27001 [ISO13] provides variety of controls that are defined as requirements for establishing, implementing, maintaining, and continually improving an information security management system. In addition, the standard contains requirements for the assessment and treatment of information security risks tailored to the needs of the organization. These requirements are defined to support security assessment and evaluation of information security management systems (ISMS).

ISO 27001	CREDENTIAL
Information Security Policies	Software Architecture, Service Isolation
Organization of information security	Software Architecture, Service Isolation, Client-Side Concerns
Human resource security	**OUT OF SCOPE**
Asset management	**OUT OF SCOPE**
Access control	Service Isolation, Client-Side Concerns, Server-Specific Requirements

ISO 27001	CREDENTIAL
Cryptography	Cryptographic Requirements
Physical and environmental security	**OUT OF SCOPE**
Operations security	Software Architecture, Logging and Reporting, Lifecycle Management, Service Isolation
Communications security	Communication-Related Requirements
System acquisition, development and maintenance	Client-Side Concerns, Lifecycle Management
Supplier relationships	**OUT OF SCOPE**
Information security incident management	**OUT OF SCOPE**
Information security aspects of business continuity management	Lifecycle Management
Compliance	**OUT OF SCOPE**

Table 4: Mapping between ISO 27001 and CREDENTIAL

ISO 27001 Control Categories	**Objective**
Information security policies	Provide management direction and support for information security in accordance to business requirements and relevant laws and regulations.
Organization of information security	Offer a management framework to initiate and control the implementation and operation of information security within the organization.
Human resource security	Ensure that both employees and contractors are aware and understand how to fulfill their information security responsibility, are suitable for the roles for which they are considered, and to protect the organization's interests as part of the process of changing or terminating employment
Asset management	Identify organizational assets and define appropriate protection responsibilities, ensure that information receives an appropriate level of protection in accordance with its importance to the organization, and prevent unauthorized disclosure, modification, removal or destruction of information stored on media.

ISO 27001 Control Categories	Objective
Access control	Limit access to information and information processing facilities, ensure authorized user access and to prevent unauthorized access to systems and services, make users accountable for safeguarding their authentication information, and prevent unauthorized access to systems and applications.
Cryptography	To ensure proper and effective use of cryptography, to protect the confidentiality, authenticity and/or integrity of information
Physical and environmental security	Prevent unauthorized physical access, damage and interference to the organization's information and information processing facilities, and mitigating loss, damage, theft or compromise of assets and interruption to the organization's operations
Operations security	Ensure correct and secure operations of information processing facilities, malware protection, data loss, events recording and generating audit trails, and ensuring integrity.
Communications security	To ensure the protection of information internally and during the transport across external networks and its supporting information processing facilities.
System acquisition, development and maintenance	Ensure that information security is an integral part of information systems across the entire lifecycle that includes the requirements for information systems which provide services over public networks, and ensures that information security is designed and implemented within the development lifecycle of information systems.
Supplier relationships	Ensure protection of the organization's assets that are accessible by suppliers
Information security incident management	Ensure consistent and effective approach to the management of information security incidents, including communication on security events and weaknesses
Information security aspects of business continuity management	Information security continuity shall be embedded in the organization's business continuity management systems and support availability of information processing facilities.

ISO 27001 Control Categories	Objective
Compliance	Ensure that breaches of legal, statutory, regulatory or contractual obligations related to information security and of any security requirements are avoided, and that information security is implemented and operated in accordance with the organizational policies and procedures.

Table 5: ISO 27001 categories description

B.2 OWASP Software Architecture Verification Standard 3.0

The OWASP[2] is a community-driven framework of holistic security requirements and controls for (web and mobile) application development.

The standard defines three different Application Security Verification Levels, ranging from 1 (“all software should comply”) to 3 (“most critical applications with sensitive data”). Each level contains a list of security-specific requirements. In addition the requirements have been grouped into categories, each of which concerns itself with “control objectives”. A list of the categories and their corresponding control objectives can be found in Table 7.

A comparison between the Software Architecture Verification Standard 3.0, Level 3, requirement domains and the requirement categories used within CREDENTIAL can be seen in Table 6. While a direct 1:1 mapping between categories is not easily possible, the covered requirements should be matching. The first iteration of CREDENTIAL’s security requirements do lack low-level implementation specific and some high-level business use-case specific requirements. Those will be added during the second iteration as the corresponding use-case documents will only be available during that iteration.

Appendix D of the OWASP standard maps ASVS to PCI-DSS 3.0.

OWASP	CREDENTIAL
V1. Architecture, design and threat modeling	Software Architecture, Service Isolation
V2. Authentication	Communication-Related Requirements, User- and Session-Management, Specific Requirements due to Mobile Clients
V3. Session management	Communication-Related Requirements, User- and Session-Management
V4. Access control	Service Isolation, Client-Side Concerns

[2] Software Architecture Verification Standard - https://www.owasp.org/images/6/67/OWASPApplicationSecurityVerificationStandard3.0.pdf

OWASP	CREDENTIAL
V5. Malicious input handling	Server-Specific Requirements
V7. Cryptography at rest	Cryptographic Requirements
V8. Error handling and logging	Logging and Reporting
V9. Data protection	Cryptographic Requirements, Communication-Related Requirements
V10. Communications	Service Isolation, Communication-Related Requirements
V11. HTTP security configuration	Server-Specific Requirements
V13. Malicious input handling verification requirements	Service Isolation, Communication-Related Requirements, Server-Specific Requirements
V15. Business logic verification requirements	**OUT OF SCOPE**
V16. File and resources	Communication-Related Requirements, User- and Session-Management, Specific Requirements due to Mobile Clients, Cryptographic Requirements
V17. Mobile	Specific Requirements due to Mobile Clients
V18. Web services	Communication-Related Requirements, Client-Side Concerns
V19. Configuration	Lifecycle Management

Table 6: Mapping between OWASP and CREDENTIAL

OWASP	Control Objective
V1. Architecture, design and threat modeling	L1: Components of the application are identified and have a reason for being in the app L2: The architecture has been defined and the code adheres to the architecture L3: The architecture and design is in place, in use, and effective
V2. Authentication	Verifies the digital identity of the sender of a communication Ensures that only those authorized are able to authenticate and credentials are transported in a secure manner
V3. Session management	Sessions are unique to each individual and cannot be guessed or shared Sessions are invalidated when no longer required and timed out during periods of inactivity

OWASP	Control Objective
V4. Access control	Persons accessing resources holds valid credentials to do so Users are associated with a well-defined set of roles and privileges Role and permission metadata is protected from replay or tampering
V5. Malicious input handling	All input is validated to be correct and fit for the intended purpose Data from an external entity or client should never be trusted and should be handled accordingly
V7. Cryptography at rest	That all cryptographic modules fail in a secure manner and that errors are handled correctly That a suitable random number generator is used when randomness is required That access to keys is managed in a secure way.
V8. Error handling and logging	Not collecting or logging sensitive information if not specifically required. Ensuring all logged information is handled securely and protected as per its data classification. Ensuring that logs are not forever, but have an absolute lifetime that is as short as possible.
V9. Data protection	Confidentiality: Data must be protected from unauthorized observation or disclosure both in transit and when stored. Integrity: Data should be protected being maliciously created, altered or deleted by unauthorized attackers Availability: Data should be available to authorized users as required.
V10. Communications	That TLS is used where sensitive data is transmitted That strong algorithms and ciphers are used at all times
V11. HTTP security configuration	The application server is suitable hardened from a default configuration. HTTP responses contain a safe character set in the content type header
V13. Malicious controls	Detected malicious activity is handled securely and properly as to not affect the rest of the application
V15. Business logic	The business logic flow is sequential and in order
V16. File and resources	Untrusted file data should be handled accordingly and in a secure manner obtained from untrusted sources are stored outside the webroot and limited permissions

OWASP	Control Objective
V17. Mobile	Any server side controls, such as an API or Web Service, should have the same level of security controls in place as found on the device itself. Sensitive information assets stored on the device should be done so in a secure manner All sensitive data transmitted from the device should be done so with transport layer security in mind.
V18. Web services	Adequate authentication, session management and authorization of all web services Input validation of all parameters that transmit from a lower to a higher trust level Basic interoperability of SOAP web services layer to promote API use
V19. Configuration	Lifecycle Management Service Isolation Communication-Reqlted Requirements

Table 7: OWASP Architecture Verification Standard 3.0 Categories

B.3 NIST SP 800-53 R4 Security and Privacy Controls for Federal Information Systems and Organizations

The National Institute of Standards and Technology (NIST) provides a comprehensive set of safeguards and countermeasures for organizations and information systems in NIST SP 800-53 publication [NIS03]. The security controls NIST SP 800-53 are designed first of all to be technology independent and in line with applicable federal laws, executive orders, directives, policies, regulations, standards, and guidelines. In addition, each individual security control is defined by a set of requirements that are often repeated across multiple security controls. Furthermore, these requirements are defined based upon the functional behavior of a system and serve as the foundation of the development of assessment methods and procedures for determining security control effectiveness. In Table 8 there are come categories like Security Assessment And Authorization or Physical And Environmental Protection that we marked as out of scope due to their focus on organizational processes or physical security which are beyond the scope of our current work.

NIST 800-53	CREDENTIAL
Access Control	Service Isolation, Client-Side Concerns
Awareness And Training	**OUT OF SCOPE**

NIST 800-53	CREDENTIAL
Audit And Accountability Policy And Procedures	Logging and Reporting
Security Assessment And Authorization	**OUT OF SCOPE**
Configuration Management Policy And Procedures	Lifecycle Management
Contingency Planning	**OUT OF SCOPE**
Identification And Authentication	Communication-Related Requirements, User- and Session-Management, Specific Requirements due to Mobile Clients
Incident Response	Service Isolation
Maintenance	**OUT OF SCOPE**
Media Protection	Cryptographic Requirements
Physical And Environmental Protection	**OUT OF SCOPE**
Planning	**OUT OF SCOPE**
Personnel Security	**OUT OF SCOPE**
Risk Assessment	Section 3
System And Services Acquisition	Service Isolation
System And Communications Protection	Service Isolation, Communication-Related Requirements
System And Information Integrity	Cryptographic Requirements, Communication-Related Requirements
Program Management	**OUT OF SCOPE**
Authority and Purpose	**OUT OF SCOPE**
Accountability, Audit, and Risk Management	Deliverable D2.6 - Logging and Reporting
Data Quality And Integrity	Cryptographic Requirements
Data Minimization And Retention	Logging and Reporting
Individual Participation And Redress	**OUT OF SCOPE**
Security	Deliverable D2.2
Transparency	**OUT OF SCOPE**
Use Limitation	**OUT OF SCOPE**

Table 8: Mapping between NIST SP 800-53 R4 and CREDENTIAL

B.4 ANSI/ISA-62443 Security for Industrial Automation and Control Systems

The International Society for Automation (ISA) and American National Standards Institute (ANSI) published series of standards, ANSI/ISA-62443 [AI13], clustered across four domains (General, Policies and Procedures, System, and Component) that define procedures for implementing electronically secure Industrial Automation and Control Systems (IACS). The General domain is focused on the terminology, concepts, models, system security metrics for and lifecycles. Policies and Procedures provided within the second domain are used for implementation, installation, and maintenance of IACS cyber security management systems. Furthermore, the System domain of the standard series details security technology, levels and requirements. In the final domain, the components, product development and technical security requirements are defined.

ANSI/ISA-62443	CREDENTIAL
Identification and authentication control	Communication-Related Requirements, User- and Session-Management, Specific Requirements due to Mobile Clients, Logging and Reporting
Use control	Logging and Reporting, Communication-Related Requirements, User- and Session-Management, Specific Requirements due to Mobile Clients
System integrity	Communication-Related Requirements, Specific Requirements due to Mobile Clients
Data confidentiality	Server-Specific Requirements
Restricted data flow	Service Isolation, Communication-Related Requirements
Resource availability	Service Isolation

Table 9: Mapping between ANSI/ISA-62443 and CREDENTIAL

B.5 CUMULUS

CUMULUS (Certification infrastructure for multi-layer cloud services)[3] is an FP7 framework research project focused on designing and developing solutions for certification of security properties of infrastructure (IaaS), platform (PaaS), and software application layer (SaaS) services in cloud. Within the project, an extensive set of security related properties is elicited and clustered

[3] Certification infrastructure for multi-layer cloud services EU research project - http://www.cumulus-project.eu/

across 16 domains where each individual property is used as requirement to be fulfilled in order to certify the service. These security properties share common ground with the non-functional requirements due to the fact that they are being used to validate the behavior of a system or a service. We marked Organizational, privacy and legal requirements as out of the scope given that they will be addressed by other CREDENTIAL deliverables.

CUMULUS	CREDENTIAL
Application & Interface Security	Service Isolation, Client-Side Concerns
Infrastructure & Virtualization Security	Service Isolation, Client-Side Concerns
Interoperability & Portability	**OUT OF SCOPE**
Security Incident Management, E - Discovery & Cloud Forensics	**OUT OF SCOPE**
Identity & Access Management	Service Isolation, Client-Side Concerns, Server-Specific Requirements
Encryption & Key Management	Cryptographic Requirements
Governance & Risk Management	Section 3
Supply Chain Management, Transparency & Accountability	**OUT OF SCOPE**
Data Security & Information Lifecycle Management	Server-Specific Requirements, Cryptographic Requirements, Communication-Related Requirements
Mobile Security	Specific Requirements due to Mobile Clients
Legal & Standards Compliance	**OUT OF SCOPE**
Data Security & Information Lifecycle Management	Deliverable D2.6
Threat & Vulnerability Management	Software Architecture
Datacenter Security	**OUT OF SCOPE**
Business Continuity Management & Operational Resilience	**OUT OF SCOPE**
Change Control & Configuration Management	Lifecycle Management
Human Resources Security	**OUT OF SCOPE**

Table 10: Mapping between CUMULUS and CREDENTIAL

C Functional Requirements

During the non-functional requirements analysis process, we discovered multiple functional requirements. After collaboration with the corresponding requirements engineering team, we created a small set of functional requirements and included them within this appendix for reference purposes. Those requirements were needed to better reason about the resulting overall system and aided during the requirement collection process.

We assume that these functional requirements will be included in the corresponding functional requirement documents and thus be removed from the next iteration of this non-functional requirement document.

C.1 Wallet Access Management

The requirements in this section describe the elementary operations provided by the CREDENTIAL wallet. Each of them will be part of the project's pilots.

A user MUST be able to grant access rights on his personal resources in the wallet to other users	
REFERENCE ID	ACC-GRANT-RIGHTS
AREA	Wallet
RESPONSIBLE	Architecture
DESCRIPTION	The user can allow other users to access his personal wallet, e.g., a patient allows his doctor to read data from his electronic health record.

Every access to a resource in the wallet MUST be written in an Audit-Log	
REFERENCE ID	ACC-LOG-ACCESS
AREA	Wallet
RESPONSIBLE	Implementation
DESCRIPTION	The wallet has to provide an audit log in case of storing and processing sensitive data.

The Wallet SHOULD encrypt the Audit-Log only for the user himself	
REFERENCE ID	ACC-LOG-ENCRYPT
AREA	Wallet
RESPONSIBLE	Architecture
DESCRIPTION	In case of sensitive data, e.g., medical data, the Audit-Log may not be stored unencrypted.

A user with delegate access rights SHOULD be able to give access rights to another user on a resource where he is not the owner	
REFERENCE ID	ACC-GRANT-DELEG
AREA	Wallet
RESPONSIBLE	Architecture
DESCRIPTION	For example a patient gives his doctor full access on his medical data in the cloud.

A user with read access rights on a resource MUST be able to read data from the Wallet	
REFERENCE ID	ACC-ACCESS-READ
AREA	Wallet
RESPONSIBLE	Architecture
DESCRIPTION	Only users with read access on a specific resource are able to read this data set from the Wallet.

A user with write access rights on a resource MUST be able to append or update data on the Wallet	
REFERENCE ID	ACC-ACCESS-WRITE
AREA	Wallet
RESPONSIBLE	Architecture
DESCRIPTION	Write access includes update, create and append operations.

A user MUST be able to append data to a personal wallet	
REFERENCE ID	ACC-ADD-DATA
AREA	Wallet
RESPONSIBLE	Architecture
DESCRIPTION	In case of continuous data, for example measured data over time, the Wallet MUST support appending data to an existing record.

On a request the Wallet SHOULD process data for the requester if he has read access rights	
REFERENCE ID	ACC-DATA-PROCESS
AREA	Wallet
RESPONSIBLE	Architecture
DESCRIPTION	For example the requester is a service provider and requests an Identity Assertion. The service provider needs read access on the Identity Attributes to let the Wallet issue an Identity Assertion.

D Overview Table

The following table provides an overview over all requirements collected in Section 5. It aims at providing an easy reference for developers and administrators of the different components within CREDENTIAL.

ID	Requirement	Environment	Client	Wallet	Service	Architecture	Implementation	Administration
ACC-GRANT-RIGHTS	A user MUST be able to grant access rights on his personal resources in the wallet to other users			x		x		
ACC-LOG-ACCESS	Every access to a resource in the wallet MUST be written in an Audit-Log			x			x	
ACC-LOG-ENCRYPT	The Wallet SHOULD encrypt the Audit-Log only for the user himself			x		x		
ACC-GRANT-DELEG	A user with delegate access rights SHOULD be able to give access rights to another user on a resource where he is not the owner			x		x		
ACC-ACCESS-READ	A user with read access rights on a resource MUST be able to read data from the Wallet			x		x		
ACC-ACCESS-WRITE	A user with write access rights on a resource MUST be able to append or update data on the Wallet			x		x		
ACC-ADD-DATA	A user MUST be able to append data to a personal wallet			x		x		
ACC-DATA-PROCESS	On a request the Wallet SHOULD process data for the requester if he has read access rights			x		x		
ARC-MIN-COMPS	Number of Components SHOULD be minimal		x		x	x		
ARC-SING-RESP-PR	Components MUST conform to the single responsibility principle		x		x	x		
ARC-DOC-INTERF	Interfaces MUST be documented		x		x	x		
ARC-MIN-INTERF	Interfaces MUST be minimal		x		x	x		
ARC-SEC-AWARE	Software development MUST follow a security-aware life-cycle		x		x		x	
ARC-MATCHING-ACL	ACL rules MUST match between different layers		x		x	x		
ARC-CODING-STND	Source code MUST comply to secure coding standards		x		x		x	
ARC-TESTING	Continuous software testing MUST be performed		x		x		x	
ARC-LIBRARIES	Third party software dependencies (e.g., libraries) MUST be identified and documented		x		x		x	
ARC-CLEAN-ARCH	An clean overall architecture MUST be enforced		x		x	x		

ID	Requirement	Environment	Client	Wallet	Service	Architecture	Implementation	Administration
ARC-INF-CLASSIFY	Information MUST be classified		x				x	
CLI-INS-SENS-DAT	Sensitive data SHALL NOT be included in the installation package		x				x	
CLI-INS-AUTH-INT	Integrity and authenticity of the installation package MUST be provided	x	x				x	
CLI-DATA-CACHE	Sensitive data MUST NOT be stored in insecure application caches	x	x				x	
CLI-DATA-ENC	Sensitive Data MUST be stored encrypted		x			x	x	
CLI-SEC-STORAGE	Secure storage SHOULD be used if it is offered by the platform	x	x				x	
CLI-TRUSTED-EXEC	Client SHOULD use trusted execution	x	x				x	
CLI-OS-UPDATES	Operating system and system libraries MUST be up to date	x					x	x
CLI-SEC-BACKUP	Clients SHOULD use secure backup	x						x
CLI-MOB-AUTH-DEV	Mobile devices SHOULD use authentication	x						x
CLI-MOB-AUTH-APP	Users MUST authenticate when opening the application		x				x	
CLI-MOB-APP-SWIT	No sensitive information SHOULD be shown in the application switcher preview		x				x	
COM-CONFIDENTIAL	Communication data MUST be kept confidential		x		x	x	x	
COM-INTEGRITY	Communication data's integrity MUST be protected		x		x	x	x	
COM-AUTHENTICITY	Communication partners MUST be authenticated		x		x	x	x	
COM-LOG-SESSIONS	Communication sessions MUST be logged				x		x	
COM-LOG-FAILURES	Communication failures MUST be logged				x		x	
CRY-KEY-SIZES	Key sizes MUST be sufficiently large		x		x		x	
CRY-PW-PLAIN	Passwords MUST NOT be stored in the plain				x		x	
CRY-PW-SALT	Stored passwords MUST be salted				x		x	
CRY-PW-HASH	Dedicated hashing algorithms SHOULD be used for hashing passwords				x		x	
CRY-SALT-FRESH	Fresh and long salts MUST be used for every password				x		x	
CRY-KEY-SEC-STOR	Private keys SHOULD be stored in secure environments		x		x	x	x	
CRY-KEY-CACHE	Private keys SHOULD never be cached		x		x		x	
CRY-DATA-KEY-SEP	Private keys MUST be stored separately from encrypted data		x		x	x		

ID	Requirement	Environment	Client	Wallet	Service	Architecture	Implementation	Administration
CRY-KEY-SEPARAT	Cryptographic keys SHOULD NOT be reused for different purposes		x		x		x	
CRY-PRG-SECURE	Secure random number generators MUST be used	x						x
CRY-INST-SECURE	Secure instantiations of primitives MUST be used		x		x		x	
CRY-IMPL-SECURE	Secure implementations of primitives MUST be used	x					x	x
CRY-IMPL-SIDECHA	Deployed implementations SHOULD guarantee side-channel resistance	x					x	x
LIF-DEF-PROCESS	The lifecycle process MUST be defined	x				x		
LIF-LOG-INTERACT	Interactions with the lifecycle process MUST be logged	x					x	x
LIF-CONFIG-MGMT	Configuration management MUST be performed	x						x
LIF-CONFIG-REV	It SHOULD be possible to revert to old configurations	x						x
LIF-CONFIG-TEST	New configurations MUST be tested prior to Deployment	x						x
LIF-MON-AVAIL	Deployed Services MUST be monitored for availability	x						x
LIF-MON-SEC-BREA	Deployed Services MUST be monitored for security breaches	x						x
LIF-BK-PRIVACY	Backups MUST maintain data privacy	x						x
LIF-BK-INTEGRITY	Backups MUST be integrity-protected	x						x
LIF-DEL-BEF-DISP	Data MUST be securely deleted before server disposal	x						x
LIF-LOG-ADMIN	Administrative functions and security configuration settings MUST be logged	x						x
LGR-LOG-STORAGE	An Independent storage location for logging data MUST be configured	x						x
LGR-LOG-ACCESS	Log information MUST be restricted with strong access control mechanisms	x						x
LGR-LOG-CONSIST	Consistency of logged information MUST be verified during its life cycle	x						x
LGR-OUT-SENS-INF	Sensitive information MUST be excluded from the error output		x		x		x	
LGR-OUT-DEB-INF	Error handlers MUST NOT display debugging or stack trace information		x		x		x	

ID	Requirement	Environment	Client	Wallet	Service	Architecture	Implementation	Administration
LGR-LOG-TAMPER	Apparent tampering events of any sensitive data MUST be logged		x		x		x	x
LGR-LOG-RET-PER	Log information retention period MUST be defined	x						x
LGR-LOG-DELETE	Log information MUST be securely discarded	x						x
LGR-LOG-LAWS	Logging procedures MUST oblige national and EU legal directives	x						x
LGR-REP-PERIODS	Periodic reporting MUST be established	x						x
LGR-REP-FORM	Periodic report form MUST be defined	x						x
LGR-LOG-MIN-INFO	Minimum log event information set MUST be defined		x		x		x	
LGR-LOG-CRYPTO	Cryptographic module failures MUST be logged		x		x		x	x
LGR-LOG-DOCUMENT	Logging guidelines SHOULD be documented	x						x
LGR-LOG-LOCATION	Location CAN be logged	x					x	x
LGR-LOG-CUSTOM	Logging CAN be customizable	x					x	x
LGR-LOG-LANGUAGE	Logging language SHOULD be standardized		x		x		x	
SER-CHECK-INPUT	All input MUST be checked for common injection vectors				x		x	
SER-ENC-USER-DAT	User-generated files with sensitive content MUST be encrypted				x	x		
SER-USER-DELETE	A single user MUST be deletable (legal requirement)				x	x		
SER-VALIDATE-IO	Input/Output validation SHOULD be centralized				x		x	
SER-VALIDATE-ENC	Encoding MUST be validated				x		x	
SER-RATE-LIMITS	Rate-limits SHOULD be introduced				x		x	
SER-ENC-TEMPS	Temporary Files SHOULD be encrypted				x		x	
SER-DEL-DOCU	Unnecessary documentation SHOULD be removed	x						x
SER-UPDATE-SW	Software and Libraries MUST use latest patch level	x			x		x	x
SER-UPL-STORE	Uploaded Files MUST be stored outside the web-context				x		x	x
SER-UPL-VERIF	Uploaded Files MUST be verified				x		x	
SER-UPL-NO-EXEC	Uploaded Data MUST NOT be executed				x		x	
SRI-SER-ISOLATE	Services MUST be isolated from each other	x						x
SRI-COM-MINIMAL	Communication between services MUST be restricted to minimum required	x			x	x	x	x
SRI-SER-PRIVIL	Service's privileges MUST be restricted	x						x

ID	Requirement	Environment	Client	Wallet	Service	Architecture	Implementation	Administration
SRI-SER-RESOURCE	Service resources SHOULD be limited	x						x
SRI-SER-ENV	Service environment MUST be monitored	x						x
SRI-SER-SEC-FAIL	Service MUST always fail secure				x		x	
SRI-COMPSER-DEF	Composite service behaviour model MUST be defined				x	x		
SRI-COMPSER-MON	Composite service behaviour model MUST be monitored	x						x
USM-SID-SECURE	Secure session IDs MUST be used				x		x	
USM-SID-FRESH	Session IDs MUST be fresh				x		x	
USM-SID-GUESSING	Session ID guessing and brute forcing attacks SHOULD be detected		x		x		x	
USM-SID-TO-IDLE	Session timeouts in case of inactivity SHOULD be implemented				x		x	
USM-SID-TO-ABS	Absolute session timeouts SHOULD be implemented		x		x		x	
USM-SID-LOGOUT	Sessions SHOULD automatically be ended upon pre-defined events		x		x		x	
COM-TLS-USE	All Communication MUST utilize TLS		x		x	x	x	x
COM-TLS-PFS	Connections MUST support Perfect Forward Secrecy		x		x	x	x	x
COM-TLS-CERT-CLI	Desktop/Mobile Clients SHOULD use certificate pinning		x				x	
COM-TLS-CERT-SER	Web-Browser/Servers SHOULD use certificate pinning				x		x	x
COM-TLS-AEAD	Utilized Ciphers SHOULD be AEAD		x		x		x	x
USM-2FACTOR-AUTH	Multi-factor authentication SHOULD be implemented				x	x	x	
USM-COM-SERV-IDP	There SHOULD be no direct interaction between server application and IdP			x	x	x		
USM-HIDE-APP	The IdP SHOULD not be able to identify the application used by the user		x	x		x		
USM-UAUTHENT-EXP	User authentications MUST have an expiration date or expiration event		x	x			x	
USM-UAUTORIZ-EXP	Granted authorizations MUST have an expiration date or an expiration event		x	x			x	
USM-REQ-AUTHENT	For each access to restricted resources, the user SHOULD be authenticated				x		x	
USM-REQ-AUTHORI	For each access to restricted resources, the user MUST have the corresponding authorizations				x		x	

ID	Requirement	Environment	Client	Wallet	Service	Architecture	Implementation	Administration
USM-CREDS-REV	User authentication credentials MUST be revokable			x		x		x
USM-AUTHOR-REV	Granted authorizations MUST be revokable			x		x		x
USM-USER-ENUM	User enumeration MUST NOT be possible			x			x	
USM-LOG-ACCESS	Logs of access to user data SHOULD be visible to the user		x	x		x		
USM-LOG-FAILS	Unsuccessful login attempts and access attempts SHOULD be logged			x			x	x
USM-ANOMALY	Anomaly detection to detect irregular login attempts MAY be implemented			x				x
USM-THROTTELING	Logins SHOULD be throttled after too many failed attempts			x			x	
USM-SUSPEND	The IdP SHOULD be able to suspend/ban specific users			x		x		
COM-UN-NO-EMAIL	Sensitive data SHALL NOT be sent via email to the user				x		x	
COM-UN-ENC-MAIL	Email-based user notifications SHOULD be signed and encrypted		x		x		x	
COM-UN-APP-NOTIF	Sensitive data MUST NOT be displayed within notifications		x				x	