Project number: 653454
H2020-DS-2014-1

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

CREDENTIAL
Secure Cloud Identity Wallet

D4.1

Assessment report on cryptographic technologies, protocols and mechanisms

Document Identification	
Due date	March 31, 2017
Submission date	March 31, 2017
Revision	1.0

Related WP	WP4	**Dissemination Level**	PU
Lead Participant	TUG	**Lead Author**	Felix Hörandner (TUG)
Contributing Beneficiaries	AIT, FOKUS, GUF, TUG, KGH, LISPA, SIC	**Related Deliverables**	D2.2, D2.3, D3.3, D4.2, D4.3, D4.4, D5.1

Abstract: This report contains a detailed assessment and analysis of eligible basic cryptographic technologies, security protocols, and authentication mechanisms with respect to *CREDENTIAL* requirements. It gives an overview of the start of the art in the different areas, evaluates the existing schemes regarding efficiency, security, privacy, usability, etc., and makes concrete suggestions on which technologies to use in the project.

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 has to be performed in order to develop such a data sharing platform is the identification and evaluation of relevant base technologies, which is provided by this deliverable.

First, we concisely introduce relevant technologies, covering different aspects of the generic data sharing system. These aspects include cryptographic technologies, authentication mechanisms, as well as identity and access management protocols and technologies.

We perform assessments based on high-level criteria with a focus on security, privacy, usability and integration effort. These high-level criteria are then mapped to criteria specific to the technologies under evaluation. The criteria are motivated by *CREDENTIAL*'s use cases and requirements.

As a result of the assessment, this deliverable makes concrete recommendations for technologies that should be considered in the design and implementation of the generic data sharing platform as technological basis. Also, this deliverable presents technologies, which could provide additional benefits to *CREDENTIAL*'s envisioned goal and might therefore be of interest for further research in order to apply them.

Besides the generic technologies, we also present an overview of technologies relevant to the pilot use cases in order to facilitate a common understanding of the involved technological ecosystem.

Finally, this deliverable provides detailed descriptions of relevant technologies in the appendix for the curious reader. This appendix also serves as a knowledge base for project participants to acquire information on technologies that are not in their core expertise.

Document information

Editors

Name	E-mail	Organisation
Bernd Zwattendorfer	bernd.zwattendorfer@iaik.tugraz.at	SIC
Felix Hörandner	felix.hoerandner@iaik.tugraz.at	TUG

Contributors

Name	E-mail	Organisation
Stephan Krenn	stephan.krenn@ait.ac.at	AIT
Christoph Striecks	christoph.striecks@ait.ac.at	AIT
Thomas Lorünser	thomas.loruenser@ait.ac.at	AIT
Nicolas Notario McDonnell	nicolas.notario@atos.net	ATOS
Florian Thiemer	florian.thiemer@fokus.fraunhofer.de	FOKUS
Jörg Caumanns	joerg.caumanns@fokus.fraunhofer.de	FOKUS
Jetzabel Serna	jetzabel.serna@m-chair.de	GUF
Andreas Abraham	andreas.abraham@iaik.tugraz.at	TUG
Christof Rabensteiner	christof.rabensteiner@iaik.tugraz.at	TUG
Elias Klughammer	elias@klughammer.com	KGH
Andrea Migliavacca	andrea.migliavacca@cnt.lispa.it	LISPA
Alberto Zanini	alberto.zanini@lispa.it	LISPA
Silvana Mura	silvana.mura@cnt.lispa.it	LISPA
Franco Nieddu	franco.nieddu@iaik.tugraz.at	SIC
Simon Roth	simon.roth@iaik.tugraz.at	SIC
Enrico Francescato	enrico.francescato@infocert.it	ICERT

Reviewers

Name	E-mail	Organisation
Stephan Krenn	stephan.krenn@ait.ac.at	AIT
Florian Thiemer	florian.thiemer@fokus.fraunhofer.de	FOKUS
Elias Klughammer	elias@klughammer.com	KGH

History

Version	Date	Reason/Change	Editor
0.1	14.01.2016	Document creation	Bernd Zwattendorfer
0.2	16.02.2016	Added General Factsheets	Felix Hörandner
0.3	10.03.2016	Added eHealth Factsheets	Florian Thiemer

0.4	18.03.2016	Added eGovernment Factsheets	Andrea Migliavacca
0.5	02.05.2016	Added Technology Details: OpenID Conenct	Felix Hörandner
0.6	02.05.2016	Added Technology Details: FIDO, SAML	Andreas Abraham
0.7	25.05.2016	Added eBusiness Factsheets	Enrico Francescato
0.8	27.05.2016	Added Technology Details: Proxy Re-Encryption	Felix Hörandner
0.9	27.05.2016	Added Technology Details: SQRL	Nicolas Notario
0.10	28.06.2016	Added Technology Details: Malleable Signatures	Andreas Abraham
0.11	28.06.2016	Added Technology Details: OAuth	Felix Hörandner
0.12	01.07.2016	Added Technology Details: UMA, XACML	Andreas Abraham
0.13	01.07.2016	Added Technology Details: PDPs and PORs	Christoph Striecks
0.14	15.07.2016	Added Technology Details: KMIP	Andreas Abraham
0.15	27.07.2016	Added Technology Details: FHE and SE	Christoph Striecks
0.16	27.07.2016	Added Technology Details: secret sharing, pseudonyms, TPASS	Stephan Krenn
0.17	29.07.2016	Added Technology Details: Anonymous credentials	Jetzabel Serna
0.18	29.07.2016	Added details for WS-* description	Florian Thiemer
0.19	01.08.2016	Added e-Health Technology Details	Jörg Caumanns
0.20	01.08.2016	Added Technology Details: SCIM	Andreas Abraham
0.21	16.08.2016	Added eGovernment Technology Details	Andrea Migliavacca Alberto Zanini
0.22	16.08.2016	Evaluated Identity Protocols	Felix Hörandner
0.23	21.09.2016	Added Details for ABCs and PIR	Jetzabel Serna
0.24	03.10.2016	Added Details for PKCS11	Andrea Migliavacca
0.25	13.10.2016	Evaluated Malleable Signatures	Christoph Striecks
0.26	14.10.2016	Evaluated Secret Sharing	Stephan Krenn
0.27	17.10.2016	Restructured Document	Felix Hörandner
0.28	18.10.2016	Evaluated SCIM	Andreas Abraham
0.29	25.10.2016	Evaluated PRE Types	Felix Hörandner
0.30	03.11.2016	Evaluated Authorization Protocols	Andreas Abraham
0.31	08.11.2016	Updated Introduction and Methodology	Felix Hörandner
0.32	29.11.2016	Evaluated PDP, POR, PIR, ORAM, and FHE	Stephan Krenn Thomas Lorünser
0.33	01.12.2016	Evaluated PRE vs ABE and redactable signatures vs anonymous credentials	Christoph Striecks Stephan Krenn
0.34	20.12.2016	Evaluated searchable encryption and added details for remote attestation	Christoph Striecks Stephan Krenn
0.35	20.12.2016	Evaluated KMIP and WebCrypto API	Christof Rabensteiner
0.36	28.12.2016	Added Technology Details: TPM and TEE	Franco Nieddu

0.37	13.01.2017	Evaluated TPM and TEE	Franco Nieddu
0.38	13.01.2017	Improved structure of document	Felix Hörandner
0.39	16.01.2017	Evaluated Authentication Technologies	Jetzabel Serna
0.40	16.01.2017	Evaluated Verifiable Computing	Felix Hörandner
0.41	16.01.2017	Evaluated Policy Languages	Florian Thiemer
0.42	16.01.2017	Added abstract and acronyms	Stephan Krenn
0.43	23.01.2017	Added introductions and floating text to core crypto technologies	Stephan Krenn
0.44	24.01.2017	Added methodology section	Stephan Krenn
0.45	24.01.2017	Added Introduction and introductions to eGovernment and eBusiness sections	Andrea Migliavacca
0.46	26.01.2017	Finalized additional crypto section and Appendix B	Christoph Striecks
0.47	27.01.2017	Finalized protocols section	Christof Rabensteiner
0.48	27.01.2017	Added Executive Summary	Felix Hörandner
0.50	14.01.2017	Reviewing and fixing	Felix Hörandner Stephan Krenn
0.51	14.02.2017	Added Conclusion	Elias Klughammer
0.52	24.02.2017	Polished eGovernment and eBusiness sections	Silvana Mura
0.53	03.03.2017	Reviewed Document	Florian Thiemer
0.54	07.03.2017	Addressed Review Comments	Felix Hörandner Stephan Krenn
0.56	08.03.2017	Re-Wrote Conclusion	Felix Hörandner
0.57	27.03.2017	Polishing	Felix Hörandner
1.0	31.03.2017	Final copy-editing	Stephan Krenn

Table of Contents

List of Figures

List of Tables

List of Acronyms

ABC	Attribute Based Credential
ABE	Attribute Based Encryption
AgID	Agenzia per l'Italia Digitale
API	Application Programming Interface
ASN	Abstract Syntex Notation
CAS	Central Authentication Service
CBE	Certificate-based Encryption
CLS	Certificate-less Encryption
CNS	Carta Nazionale dei Servizi
CRL	Certificate Revocation List
CRUD	Create, Read, Update, Delete
CSP	Cryptographic Service Provider
DLL	Dynamic Link Library
DPR	Decreto del Presidente della Repubblica
FHE	Fully Homomorphic Encryption
HPC	Healthcare Professional Card
HSM	Hardware Secure Module

HTTP	Hypertext Transfer Protocol
IBE	Identity-Based Encryption
IAM	Identity and Access Management
IdP	Identity Provider
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
JWT	JSON Web Token
KGC	Key Generation Center
KMIP	Key Management Interoperability Protocol
OCSP	Online certificate status Protocol
ORAM	Oblivious RAM
OTP	One-Time Password
PDP	Proof of Data Possession
PEC	Posta Elettronica Certificata
PEKS	Public-key Encryption with Keyword Search
PIN	Personal Identification Number
PIR	Private Information Retrieval
PKCS	Public Key Cryptography Standard
PKG	Private Key Generator
PKI	Public Key Infrastructure
POR	Proof of Retrievability
PRE	Proxy Re-Encryption
RAM	Random Access Memory
REST	Representational State Transfer
RS	Redactable Signature
SDK	Software Development Kit
SE	Secure Element; Searchable Encryption
S/MIME	Secure/Multipurpose Internet Mail Extension
SAML	Security Assertion Markup Language
SCIM	System for Cross-domain Identity Management
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SP	Service Provider
SPID	Sistema Pubblico Identita Digitale
SLO	Single Log Out
SOAP	Simple Object Access Protocol
SSO	Single Sign-On
TEE	Trusted Execution Environment
TLS	Transport Layer Security
TPASS	Threshold Password-Authenticated Secret Sharing
TPM	Trusted Platform Module
TRL	Technology Readiness Level
UMA	User-Managed Access
UNI	Ente Nazionale Italiano di Unificazione
URL	Uniform Resource Locator

W3C	World Wide Web Consortium
WS-*	Web Service
WSDL	Web Services Description Language
XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language

1 Introduction

A core activity within *CREDENTIAL* is the identification and evaluation of relevant base technologies especially in the fields of cryptography, identity and access management as well as authentication mechanisms, to develop a cloud-based data sharing platform. In the assessment, whose results are gathered in this report, we evaluate the relevant technologies based on their relevance for the *CREDENTIAL* project's vision according to a set of high-level criteria. Also, this report provides an overview of technologies relevant to three pilots in the domains of eGovernment, eHealth, and eBusiness but does not further evaluate them. In-depth descriptions of relevant technologies are provided to allow for further analysis or application. This assessment represents an input for the design and implementation of the *CREDENTIAL* components and raises directions for further research to enhance technologies or make them suitable for *CREDENTIAL*.

1.1 The *CREDENTIAL* Project

The goal of *CREDENTIAL* is to develop, test and showcase innovative cloud-based services for storing, managing, and sharing digital identity information and other critical personal data. This approach keeps the user in control while enjoying the benefits of a cloud solution. The use of sophisticated cryptographic mechanisms, such as proxy re-encryption [25] and redactable signatures [103], will enable a secure and privacy preserving information sharing network for cloud-based identity information in which even the identity provider cannot access the data in plain-text and hence protect access to identity data. The project goal is to extend the application of the *CREDENTIAL* approach to a comprehensive cloud system and to existing solutions by using and exploiting recognized standards and protocols.

CREDENTIAL's **basic architecture** is based on the integration of cryptographic mechanisms involving three key components, namely a user, the *CREDENTIAL* wallet, and a data receiver, as shown in Figure 1. The **user** owns data that might be securely stored or shared with other members of the *CREDENTIAL* wallet. A client application in the user's domain handles cryptographic operations involving the user's private key, such as signing or generating a re-encryption key. The ***CREDENTIAL* wallet** is a cloud-based data storage and sharing service offering benefits such as constant availability on the Internet, scalability, and cost effectiveness. The Identity and Access Management system implements a multi-factor authentication and authorizes access to the stored data. This leads to two main advantages: A proxy re-encryption system does not expose plain text data, therefore confidentiality of data shared and stored by *CREDENTIAL* wallet in the cloud is ensured; Once a re-encryption key is available for some specific set of data as defined by the user, these data can be shared with specified receivers even when the user or her client application are not available. The **data receiver** can be either another *CREDENTIAL* user or a service provider. This receiver decrypts and verifies the shared data in order to reach authorization decisions and further process the data.

The **data sharing process** involves the actors of *CREDENTIAL*'s basic architecture in the following steps:

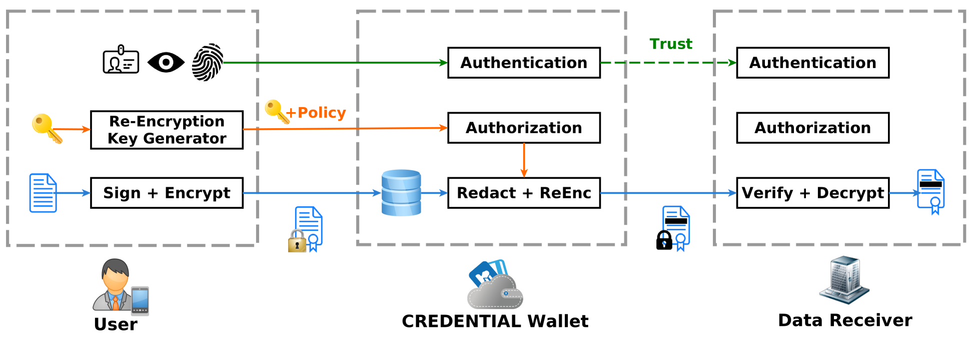


Figure 1: *CREDENTIAL*'s Basic Architecture

1. The user authenticates at the wallet to get read and write permission to her wallet account, which are used to upload signed and encrypted data.
2. To later share this data, the user generates a re-encryption key towards a selected data receiver in her trusted domain. Along with this key, the user defines a policy defining which data may be disclosed to which entity and installs it at the wallet.
3. When an authorized receiver tries to access the user's data, not required parts are redacted and the remaining parts are transformed into ciphertext for the data receiver by using the re-encryption key.
4. Finally, the data receiver is able to decrypt the data and verify the signature on the disclosed parts.

To showcase the functionality of the *CREDENTIAL* wallet and to demonstrate how a higher security and privacy can be achieved by the means of the *CREDENTIAL* wallet, **three different pilots** in the domains eGovernment, eHealth and eBusiness are developed within the project. A more detailed description of the scenarios can be found in Hörandner et al. [90].

- **eGovernment:** The pilot focuses on identity management to authenticate citizens and assess their eligibility for a service, based on sensitive identity attributes. Standardized identity protocols such as SAML or OpenID Connect are used in *CREDENTIAL*'s identity management data sharing process. Within these protocols, the service provider (i.e., the data receiver) triggers the process by requesting authentication and identity attributes from the identity provider (i.e., the *CREDENTIAL* wallet). The pilot will not only enable authentication via national eID solutions but also cross-border authentication according to the eIDAS regulation [54]. The wallet requests a consent to the user and generates a re-encryption key, whilst the service provider receives re-encrypted attributes disclosed in a selective way.
- **eHealth:** The pilot focuses on secure data sharing between patients, doctors, and further parties, in the field of type 2 diabetes. In particular, the process applies to patients, which use mobile devices to record their health data. The data is collected by a *CREDENTIAL*

eHealth mobile app which remotely stores them in the *CREDENTIAL* wallet. The user can decide who is allowed to access medical data and which specific parts are exposed, thus allowing authorized people to prepare and send medical advices back to the user. This data sharing approach offers several advantages to the patient, such as enhanced privacy by minimizing disclosed data, having a continuous control of health data, as well as saving time and money for personal visits.

- **eBusiness:** The pilot focuses on the integration of modular libraries implementing *CREDENTIAL*'s technologies into existing solutions to provide additional value. In particular, it tackles the issue of forwarding encrypted mails, which are nowadays increasingly used by companies to protect data and products, when employees are not at work. In fact, according to the current legislation, an employee has to provide her private keys to access the company e-mail system to give the possibility to other colleagues to still read and eventually take over incoming mail. Through proxy re-encryption, an employee can generate a re-encryption key towards an authorized colleague and hand this key to the mail server before leaving. The mail server is then able to re-encrypt incoming mail during the worker's absence and forward it to the authorized colleague.

1.2 Scope of this Deliverable

The objective of this document is to thoroughly investigate eligible technologies with the aim to select the most suitable technologies for *CREDENTIAL*. In this document, the technologies are described briefly, their relevance to *CREDENTIAL* is stated concisely, and they are evaluated according to high-level criteria, resulting in technology recommendations. In detail, this deliverable evaluates the following:

- **Core Cryptographic Schemes**, with a focus on technologies to securely share data, such as proxy re-encryption, and mechanisms to selectively disclose authentic subsets of shared data, in particular redactable signatures, are of crucial importance to reach our vision for *CREDENTIAL*.
- **Additional Cryptographic Schemes** might offer further security, usability or privacy benefits to the *CREDENTIAL* system. This includes for example secret sharing mechanisms (also applied to authentication processes) or privacy-enhancing technologies.
- **Authentication Technologies** represent another technological aspect for the cloud-based *CREDENTIAL* system. This deliverable not only evaluates state-of-the-art (cloud) authentication mechanisms, but also investigates technologies to bind (cryptographic) operations to the executing hardware, such as Trusted Platform Modules (TPM) or Trusted Execution Environments (TEE).
- **Identity and Access Management Technologies** are mostly based on standardized protocols and specifications (for example OpenID Connect, OAuth, XACML) in modern IAM systems. As such an IAM system also has to be designed as an integral part of *CREDENTIAL*, we assess state of the art technologies according to *CREDENTIAL*'s requirements and extensions possibilities with selected cryptographic tools.

Furthermore, this deliverable gives an overview of pilot-specific technologies required in different domains but does not make an evaluation of them.

1.3 Relation to Other Deliverables

The following list briefly describes other related deliverables and their connection to this deliverable:

D2.2 System security requirements, risk and threat analysis (1^{st} iteration): D2.2 collects security requirements relevant for *CREDENTIAL*. These requirements serve as input for the technology assessment in D4.1.

D2.3 Cloud identity wallet requirements: D2.3 focuses on functional, technical, organizational and legal requirements related to *CREDENTIAL*-related. In combination with D2.2's security requirements, these requirements are used to support the assessment of technologies.

D3.3 Recommendations on privacy-enhancing mechanisms: D3.3 analyzes threats to the user's privacy within the *CREDENTIAL* system and evaluates mitigation strategies. While D4.1 assesses and recommends the technological state of the art to be considered in the design of *CREDENTIAL*, D3.3 performs it's analysis influenced by the privacy issues and enhancements of these technologies.

D4.2 Security enhancements for basic cryptographic technologies: D4.2 will perform further research on core and additional cryptographic mechanisms suggested by this deliverable with the goal to improve or combine those technologies.

D4.3 Recommendations for improving identity protocols: D4.3 will investigate how the suggested and selected identity protocols can be enhanced to support our recommended cryptographic technologies.

D4.4 Guidelines for secure authentication to the cloud: D4.4 will investigate how the suggested technologies can be used to implement strong user authentication. If weaknesses are identified within the authentication technologies, D4.4 will furthermore try to enhance them.

D5.1 Functional Design: D5.1 will especially consider the technologies suggested by this deliverable in the design of the generic *CREDENTIAL* system.

1.4 Overview of Assessment Results

In Table 1, we present a concise summary of our assessment results. This table should introduce the reader to the technology clusters, the evaluated technologies and our findings. Further information on the evaluations is presented in Sections 3 to 6, while details on the technologies are given in Appendices A to G.

Fully Suitable **Recommended**	**Limited Suitability** **Requires Further Research**	**Currently Not of Interest** **Not Recommended**
	Core Cryptographic Technologies	
	Secure Data Sharing	
Classical Proxy Re-Encryption Conditional Proxy Re-Encryption	Proxy Re-Encryption with Keyword Search Certificate-Based PRE Certificate-Less PRE	Attribute-Based Encryption Identity-Based PRE Attribute-Based PRE Fully Homomorphic Encryption
	Authentic Data Disclosure	
For data sharing: Redactable Signatures	For identity data: Anonymous Credentials	
	Additional Cryptographic Technologies	
	Authentication	
	TPASS Distributed Password Verification Password-based Cryptography	
	Access to Encrypted Data	
	Searchable Encryption Proofs of Retrievability Provable Data Possession	Private Information Retrieval Oblivious RAM
	Other Technologies	
	Unlinkable Pseudonyms Secret Sharing Verifiable Computing	
	Authentication to the Cloud	
	Authentication Technologies	
FIDO UAF FIDO U2F OATH	SQRL	Mobile Connect
	Underlying Technologies	
TPM	TEE	
	Identity and Access Management	
	Identity Protocols	
OpenID Connect SAML		OpenID OAuth WS-Federation CAS Mozilla Persona WebID
	Authorization Protocols	
OAuth UMA		WS-Trust Kerberos
	Policies	
XACML		WS-Policy
	Cryptographic Protocols and APIs	
		W3C WebCrypto API KMIP
	Other Technologies	
	SCIM	

Table 1: Summary of Technology Assessment

1.5 Outline

The remaining sections are organized as follows: Section 2 gives a detailed description of the applied assessment methodology and the high-level evaluation criteria. The next four sections assess different technological aspects by introducing and evaluating relevant technologies. Section 3 focuses on core cryptographic technologies to securely share data and disclose authentic subsets. Section 4 provides an overview and evaluation of further cryptographic technologies that are not at the center of *CREDENTIAL* but which might provide further benefits. In Section 5, different authentication technologies are presented. Section 6 assesses protocols and technologies with a focus on identity and access management. Additionally, Section 7 provides an overview of technologies specific to the three pilot domains, namely eGovernment, eHealth and eBusiness. Section 8 summarizes the results of our assessment and provides concrete recommendations. In the appendices, we provide further details on relevant technologies, for core cryptographic technologies (Appendix A), additional cryptographic technologies (Appendix B), authentication technologies (Appendix C), and identity and access management protocols (Appendix D). Further appendices give details on pilot-specific technologies, namely eGovernment (Appendix E), eHealth (Appendix F), and eBusiness (Appendix G).

2 Assessment Methodology

In the following, we explain the methodology used in this document for evaluating the different technologies. Basically, we follow the following steps:

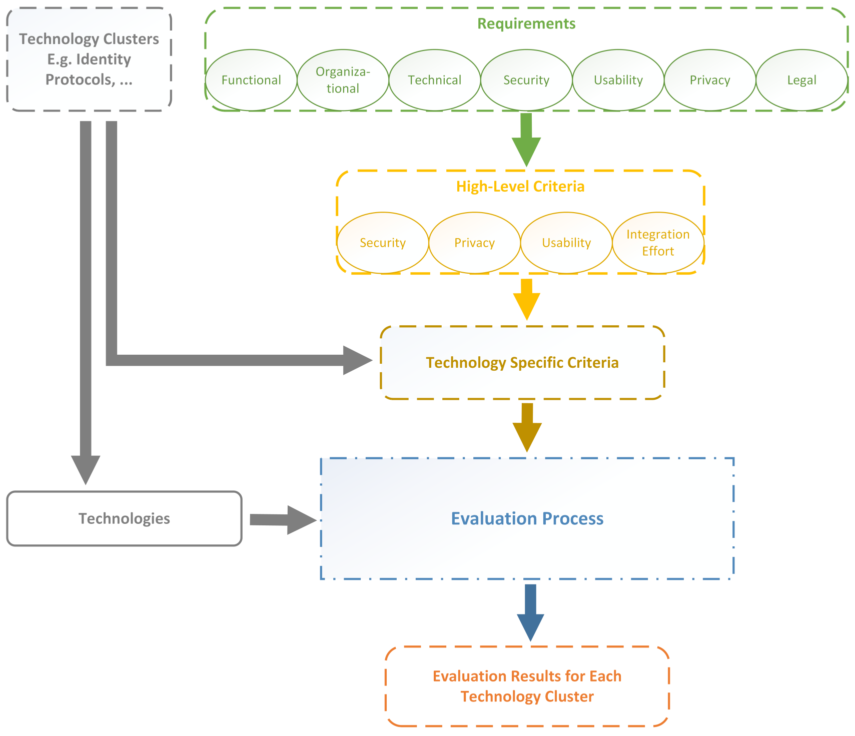


Figure 2: Selection Methodology

Clustering: In a first step, we define clusters of potentially relevant technologies, e.g., for selective disclosure of authentic data to other parties, or for authenticating users towards a cloud service. For each of those clusters, we collect promising instantiations from the literature. Furthermore, we collect all technologies that are pre-defined by the chosen pilot scenarios.

Fact sheets: Next, we fill in fact sheets for all potential technologies. The goal of those fact sheets is to get a concise overview of the offered features, maturity levels, or IPR issues. The structure and content of those fact sheets is described in more details in Section 2.1.

Pre-sorting: Based on the findings of the fact sheets, a first pre-sorting is performed to filter out technologies that—e.g., because of efficiency drawbacks or because clearly more suitable solutions exist—will not be further used within *CREDENTIAL*.

Assessment process: The remaining technologies are then evaluated in detail to obtain clear recommendations. We therefore define high-level evaluation criteria (cf. Section 2.2 for details). Those criteria cover both generic and technology specific aspects.

We then evaluate each technology (cluster) for those criteria, having in mind the different requirements for the *CREDENTIAL* wallet (e.g., as defined in D2.2 and D2.3). In the evaluation we are not following a consistent ordering of the evaluation criteria, but rather try to streamline the process. That is, we first concentrate on the most important criteria for each technology to be able to sort out unsuitable technologies as early as possible to avoid unnecessary effort.

An overview of this selection process is depicted in Figure 2.

Detailed descriptions: As a final step, certain technologies are specified in full details and can be found in the appendix. Besides the recommended technologies we also describe technologies that could be interesting for potential successors of the *CREDENTIAL* wallet if certain limitations (e.g., efficiency or usability wise) can be overcome; those technologies are typically also recommended for future research in the evaluation conclusions.

2.1 Fact Sheet Description

As mentioned earlier, we use fact sheets for briefly introducing all analyzed technologies. The purpose of those fact sheets is to give a compact overview not only about the potential and relevance of the different technologies, but also specify relevant further issues like the technology readiness level (TRL) and the technology status or known intellectual property aspects—all aspects that can potentially influence the decision whether or not to use a specific technology.

When specifying the TRL, we follow the European Commission's recommendation [53], i.e., we define the technology readiness levels as follows:

TRL 1: basic principles observed

TRL 2: technology concept formulated

TRL 3: experimental proof of concept

TRL 4: technology validated in lab

TRL 5: technology validated in relevant environment (industrially relevant environment in the case of key enabling technologies)

TRL 6: technology demonstrated in relevant environment (industrially relevant environment in the case of key enabling technologies)

TRL 7: system prototype demonstration in operational environment

TRL 8: system complete and qualified

TRL 9: actual system proven in operational environment (competitive manufacturing in the case of key enabling technologies; or in space)

In the following we now present a template of our fact sheets and explain in detail the possible semantics of each of the fields.

Name of Technology [Literature Reference]	
Type of Technology	Specifies the type of technology; for instance signature schemes, proxy re-encryption, or anonymous credential systems
Status	Gives information on the status of the technology. The following different status types are distinguished: • Concept paper • Technology report • Specification • Standard • Framework • Prototype • Project Additional information on the status such as current version of the technology and its publication year are provided.
TRL	To facilitate the assignment and reading, we cluster TRLs as follows: • Low (L): ideas and concepts, corresponding to TRL 1 or 2. • Medium (M): prototypes and demonstrators, corresponding to TRLs 3 to 7. • High (H): complete products or solutions, corresponding to TRLs 8 or 9.
Implementation	If available, a reference to an open source implementation is given here.
IPR (License Model)	Provides information on the intellectual property rights (IPR) of this technology, e.g., the licensing model
Brief Description: Briefly describes the technology	
Relevance to CREDENTIAL: Provides quantitative as well as – if important – qualitative information how relevant this technology is for *CREDENTIAL*.	

We may omit fields in our fact sheets if no information on, e.g., licensing models, were available.

2.2 High-Level Criteria

In the following, we provide more details on, and the rationale underlying, our high-level criteria for evaluating technologies, cf. also Figure 2. The main goal is to recommend technologies that are provably secure and privacy-preserving, offer high convenience to the users of the *CREDENTIAL* wallet, and are compatible with existing parts on the pilot partners' side. Specifically, we distinguish the following generic clusters of evaluation criteria:

Security: Security is one of the main categories for most technologies evaluated in this report, in particular including all cryptographic schemes and authentication and identity protocols. For instance, this category subsumes aspects such as:

- confidentiality of (identity) data;
- authenticity of the information received by a service provider;
- required level of trust into the wallet provider; or
- means for revoking expired or compromised identity information.

In particular for cryptographic schemes, security also covers whether a scheme is provably secure, and which computational hardness assumptions have to be made.

Privacy: While in the literature privacy is sometimes considered as one aspect of security, we consider it as a separate evaluation criterion because of the privacy-by-design and privacy-by-default aspects of *CREDENTIAL*. This category in particular evaluates whether:

- different actions by the same user can or cannot be linked by third parties;
- it can be checked that a specific action has been performed by a user; or whether
- a user can determine on a fine-granular level which information he wants to reveal to another party.

Similarly to security, also required complexity assumptions will be considered if appropriate.

Usability: Usability is of key importance when developing a system that shall be adopted by a broad range of users. Therefore, this cluster poses another important criterion when evaluating existing technologies. For instance, usability considers:

- the efficiency and performance including scalability aspects;
- convenience aspects for the user such as the support for Single Sign-On or Sign-Off; or
- whether and how much secret data a user needs to memorize or store locally on his device.

Integration Effort: While the rationale of the previous criteria were mainly user-centric, the integration effort cluster of evaluation criteria is about the suitability of a technology for the *CREDENTIAL* system. The goal is to recommend technologies that can practically be integrated into our solution. For instance, it considers aspects like:

- whether or not a technology is standardized;
- whether the technology is compatible with other (pilot-specific and thus pre-defined) technologies; or
- whether open-source implementations under appropriate licences exist to avoid re-implementations.

3 Core Cryptographic Technologies

In this section, we introduce and evaluate the cryptographic technologies which are at the very core of the *CREDENTIAL* tool chain. That is, we focus on cryptographic technologies that allow one to share data across different stakeholders—e.g., users, service providers—in an authentic yet privacy-preserving manner. Consequently, we focus on two main technology areas: techniques for efficient and secure sharing of data without requiring an unrealistic level of trust in the cloud storage provider, and techniques that allow one to blank out parts of authentic documents to give the data owner full control over which information he or she wants to share with a data receiver.

In Section 3.1 we compare different solutions for secure data sharing, in particular attribute-based encryption (ABE) and proxy re-encryption (PRE), and argue why the latter primitive is better suited for the *CREDENTIAL* wallet. Furthermore, we explain why fully homomorphic encryption (FHE) is currently not a viable alternative either.

Similarly, in Section 3.2 we introduce provably secure constructions for authentic data sharing with the option for only partial disclosure of information. In particular, we introduce the concepts of redactable signatures (RS) and their generalization, attribute-based credentials (ABC). We justify our choice for the former.

Now and for the rest of this document, we will first present fact sheets for all technologies to be evaluated. As already explained earlier, those fact sheets will contain a short description of the technology as well as its maturity level. This summary will then be followed by a detailed evaluation, focusing on the evaluation criteria introduced in Section 2.2. Detailed descriptions of the different technologies can be found in Appendix A.

3.1 Secure Data Sharing

In this section, we first justify a fundamental design decision of *CREDENTIAL*. Namely, we explain why we give proxy re-encryption priority over the almost equivalent alternative attribute-based encryption, cf. Section 3.1.1. Then, in Section 3.1.2 we do an in-detail analysis of the many different flavors of proxy re-encryption and chose those most suitable for usage within *CREDENTIAL*. Finally, in Section 3.1.3, we briefly discuss fully homomorphic encryption as a theoretically interesting though practically not yet usable alternative.

3.1.1 Attribute-Based Encryption vs. Proxy Re-Encryption

There are two practically efficient and probably secure cryptographic primitives that could be used to pursue the goals of *CREDENTIAL*, attribute-based encryption (ABE) and proxy re-encryption (PRE).

The former allows for creating fine-grained attribute-specific ciphertexts such that any user can decrypt the ciphertext if and only if the user has the corresponding attribute-specific secret key. For example, a ciphertext could address doctors with an attribute `doctor`. Now, any user with a doctor-specific secret key is capable to decrypt that ciphertext but nobody else is able to do so (e.g., no user with attribute patient). To delegate decryption rights, a delegator can use his secret-key material to derive attribute-specific secret keys and may send those keys (on a secure channel) to a delegatee. It follows that any delegatee is able to decrypt the ABE-ciphertexts of the delegator if and only if the delegatee is in possession of the proper attribute-specific secret key(s).

Alternatively, in PRE, each participant in the system has its own public and secret key pair. If, for instance, a patient wants to share data with a doctor, she encrypts her data under her own public key, and stores it in the cloud. Furthermore, the patient uses her own secret key and the public key of the doctor to compute a so-called *re-encryption key*. When the doctor now wants to access the data, the cloud would re-encrypt the encrypted data, such that it can now be de-ciphered using the doctor's secret key. As long as the cloud provider and the data receiver do not collude, the patient's data is secure and hidden from the cloud provider.

We next compare some important features of those two primitives.

Evaluation Criteria

We use the following two security and usability criteria to compare PRE and ABEs:

Access-Rights Adaptability (Usability): This criterion specifies whether a delegatee's access rights can adaptively be set by the delegator (i.e., access granted or revoked).

Trusted Proxy (Security): A proxy can be trusted, semi-trusted, or untrusted. For a trusted proxy, one must assume that the proxy does not perform any malicious actions and cannot be compromised, while an untrusted proxy could behave arbitrarily malicious at any point in time. A semi-trusted proxy is assumed to behave partially honest, i.e., it may for instance be allowed to leak arbitrary ciphertexts or use bias random coins, but still follows the protocol specification when talking to the user. The precise assumptions for semi-trusted proxies need to be clearly stated for each specific scenario.

Evaluation

Inherently, ciphertexts in ABE systems are more fine-grained (i.e., they are decryptable under many secret keys) in comparison to PRE where ciphertext are all-or-nothing (i.e., they are decryptable exactly under one specific secret key). That means that if a delegator wants to revoke access rights of a delegatee in an ABE system, the delegator has to trigger the proxy not to give any future ciphertext (that are associated with the delegatee's attributes) to the delegatee. In a PRE system on the other hand, the delegator has to trigger the proxy to delete the re-encryption key (associated to that delegatee) properly. Hence, in an ABE system, the proxy needs some kind of access control while in the PRE system, the proxy only needs to delete the appropriate re-encryption keys securely. (One can think of a hard disk that only contains ciphertexts. A dispose of that hard disk in the ABE case can be difficult while in the PRE case one is safe to dispose the disk when the re-encryption keys are stored in a different place.) Furthermore, to give more access rights to a delegatee, a delegator has to issue new attribute-specific secret-key material to the delegatee in an ABE system each time while in a PRE system, the delegator has to do nothing.

Further, both systems have to trust the proxy on a semi-level. That is that one needs access control which is done at the proxy's end in the ABE case while in the PRE case, one has to be sure that the proxy deletes the re-encryption keys securely.

Evaluation Conclusion

Although both techniques are almost equal on the evaluation level, we strive for PRE. That is, if one is going to use a constrained and not-fixed device (e.g., a mobile phone as planned in *CREDENTIAL*) and wants to give access to specific delegatees, one does not need the secret-key material each time when using PRE; which, however, is different to using ABE where one needs the delegator's secret-key material and (potentially heavy) secret-key computation on used device. Further, disposing ciphertexts (e.g., on a hard disk) is a problem in ABE systems since their ciphertext structure is more fine-grained in comparison to PRE. (Of course, one has to make sure that the re-encryption keys are not stored on the same hard disk in the PRE case.)

3.1.2 Proxy Re-Encryption

There are multiple different types of Proxy Re-Encryption that are implemented by multiple constructions exhibiting different properties.

Classical Proxy Re-Encryption [25]	
Type of Technology	Proxy Re-Encryption Type
Status	Concept paper (1998)
TRL	M - Multiple schemes were implemented and their performance was compared, for example in [129].
Implementation	`https://www.nics.uma.es/dnunez/nics-crypto` (LGPL license)
Brief Description: Proxy re-encryption is a public key encryption paradigm where a semi-trusted proxy, given a transformation key, can transform a message encrypted under the key of party A into another ciphertext to the same message such that another party B can decrypt it with its private key. Although the proxy can perform this re-encryption operation, it does not learn anything about the encrypted message.	
Relevance to CREDENTIAL: This algorithm could be used to provide proxy re-encryption operations.	

Identity-Based Proxy Re-Encryption [75]	
Type of Technology	Proxy Re-Encryption Type
Status	Concept paper (2007)
TRL	M - A proof of concept implementation is available.

Implementation	`https://github.com/nikosft/IB-PRE`
Brief Description: Identity-based encryption (IBE), introduced by Shamir [160], is a sub-category of public-key encryption in which the public key of an entity can be derived from identity information about that entity, such as an email address. An identity-based system usually relies on a trusted third party, the private key generator (PKG), which issues private keys to authorized entities. These private keys are derived from a master key and the verified identity information. The main advantage of IBE over PKE is that PKI is not required. In IBE, publicly known identity information, instead of public keys, is used for encryption. Therefore, there is no need to distribute public keys or to ensure their authenticity with certificates. Hence, PKI issuing these certificates is not required. However, IBE requires significant trust assumptions, which prevent wide-spread usage. IBE faces the key escrow problem, since all private keys are issued by the PKG. This PKG, therefore, has the ability to decrypt any ciphertext of its users. This problem is the motivation for certificate-less and certificate-based encryption.	
Relevance to CREDENTIAL: This algorithm could be used to provide proxy re-encryption operations.	

Certificateless Proxy Re-Encryption [166]	
Type of Technology	Proxy Re-Encryption Type
Status	Concept paper (2010)
TRL	L - The technology concept was formulated and applications were defined, but there is no claim of an implementation.
Brief Description: Certificate-less encryption (CLE), proposed by Al-Riyami and Paterson [10], neither uses certificates with all the associated public key infrastructure nor suffers from the key escrow problem. CLE is based on IBE, however, the third party does not generate the user's secret key on its own. Instead, this third party, now called key generation center (KGC), issues a partial private key (PPK) from its master key for the user's identity. This PPK and a secret value (SV) chosen by the user represent the actual decryption key. For encryption, the sender requires the publicly known identity string as well as key material derived from the user's SV. Certificate-less proxy re-encryption (CL-PRE), introduced by Sur et al. [166], applies this concept of CLE to PRE. The benefits of using CLE are that PKI is not required, while it does not suffer the key escrow problem. Since CLE is based on IBE, it enjoys the same advantage of not needing certificates to ensure the authenticity of encryption key material, as the recipient is determined through her identity information. Therefore, without certificates, PKI is superfluous. In addition, the distributed generation of the decryption key material solves the key escrow problem. As both, the KGC as well as the user, contribute input for the actual decryption key material, the KGC does not have enough information to decrypt on its own.	
Relevance to CREDENTIAL: This algorithm could be used to provide proxy re-encryption operations.	

Certificate-Based Proxy Re-Encryption [167]	
Type of Technology	Proxy Re-Encryption Type
Status	Concept paper (2013)
TRL	L - The technology concept was formulated and applications were defined, but there is no claim of an implementation.
Brief Description: Certificate-based encryption (CBE), introduced by Gentry [66], is similar to CLE, as it combines classical and identity-based encryption, while preserving their attractive features. In contrast to CLE, CBE uses certificates with a simplified PKI, which does not require the inconvenient checks for revocation status. A certification authority (CA) issues these certificates for the public key of a user-generated key pair, thereby binding the user's identity to the key material. Data is encrypted with the public key for the recipient's identity and the current time period. Decryption requires not only the private key but also a valid certificate the time period. Certificate-based proxy re-encryption (CB-PRE), proposed by Sur et al. [167], applies the concept of CBE to PRE. By introducing identity-based concepts, CBE enables implicit certification, which requires a recipient to be certified in order to decrypt. During encryption, the sender specifies the identity of the receiver and uses the presumably associated public key. In order to decrypt such a ciphertext, the receiver not only requires her private key but also a valid certificate linking the used identity to the used public key. Consequently, the sender does not have to check the authenticity of the used public key, as the receiver is implicitly required to be in possession of an appropriate certificate. For revocation, CBE only requires a simplified PKI without queries to a third party for the revocation status. Implicit certification requires recipients to possess currently valid certificates. As these certificates expire after a short time, the CA frequently has to re-certify the association between the user's identity and key material. To revoke this association, the CA is instructed to stop re-certifying the user's key material. However, a previously issued certificate stays valid for the remainder of its short lifespan.	
Relevance to CREDENTIAL: This algorithm could be used to provide proxy re-encryption operations.	

Conditional Proxy Re-Encryption [172]	
Type of Technology	Proxy Re-Encryption Type
Status	Concept paper (2009)
TRL	L - The technology concept was formulated and applications were defined, but there is no claim of an implementation.

Brief Description: Conditional proxy re-encryption [172] (C-PRE) and type-based proxy re-encryption [168] (TB-PRE) were independently introduced with the same motivation: to provide users with fine-grained control over the delegation of decryption rights. The previously discussed variants of proxy re-encryption enable the proxy to transform all of party A's ciphertexts once she provides a re-encryption key. In contrast, C-PRE (as we will also call TB-PRE) allows the user to specify which of her ciphertexts can be transformed by a re-encryption key. This is realized by tagging a ciphertext with a condition during encryption and only allowing to translate this ciphertext with a re-encryption key that satisfies the condition. For example, selective re-encryption of urgent mail highlights new possible applications of C-PRE. For the duration of her vacation, party A wants party B to be able to read and answer only her urgent mails. With C-PRE, a sender encrypts the mails for party A and tags them with additional information, for example urgent. Before leaving, party A generates a re-encryption key to party B for the condition urgent. Given this key, the mail gateway is only able to re-encrypt those mails tagged as urgent for party B. Party A's other mails remain confidential.
Relevance to CREDENTIAL: This algorithm could be used to provide proxy re-encryption operations.

Attribute Based Re-Encryption [118]	
Type of Technology	Proxy Re-Encryption Type
Status	Concept paper (2013)
TRL	L - The technology concept was formulated and applications were defined, but there is no claim of an implementation.
Brief Description: With Attribute Based Encryption (ABE) data is encrypted with regards to some attributes, which define an access policy. If a recipient wants to decrypt such a ciphertext, her private key has to reflect the required attributes, thereby fulfilling the access policy. Attribute Based Re-Encryption allows to re-encrypt ciphertexts that were encrypted for one policy to ciphertexts for another policy. The ABE concept relies on a trusted third party, which issues private keys after having checked the entity's attributes.	
Relevance to CREDENTIAL: ABE could be used to limit the re-encryption power of the proxy.	

Proxy Re-Encryption with Keyword Search [161]	
Type of Technology	Proxy Re-Encryption Type
Status	Concept paper (2010)
TRL	L - The technology concept was formulated and applications were defined, but there is no claim of an implementation.

Brief Description: Public-key encryption with keyword search (PEKS), introduced by Boneh et al. [27], enables users to search for a keyword in encrypted data stored by another party. To realize this, the sender tags data with a keyword during encryption, before handing the ciphertext to the recipient's storage provider. In order to search, the recipient generates a trapdoor for a keyword, hands this trapdoor to her storage provider, which then tests for matching ciphertexts. Proxy re-encryption with keyword search (PRES), presented by Shao et al. [161], allows to delegate both decryption and search rights to another party. As in PEKS, data is encrypted for a recipient, and tagged with a keyword. In addition, a proxy can re-encrypt the recipients's ciphertext for another user. Then, this user is able to generate trapdoors that can be used to search on the re-encrypted data as well as to decrypt these ciphertexts.
Relevance to CREDENTIAL: ABE could be used to limit the re-encryption power of the proxy.

Proxy re-encryption plays an integral role in the design of *CREDENTIAL* as it not only provides confidentiality for the user's sensitive data, but also enables secure end-to-end data sharing. However, there are multiple different types of proxy re-encryption, which provide further functionality or improvements.

In the following, we will first describe evaluation criteria and explain their motivation. Then, we use these criteria to evaluate the individual types of proxy re-encryption. Finally, we discuss our results and draw a conclusion.

Evaluation Criteria

This section lists different evaluation criteria and describes their impact.

Trust Requirements (Security) (Usability): Widely different trust assumptions are required depending on the used type of proxy re-encryption. Some types introduce a third party, which is involved in the generation of the participant's key material. If such a party inherently has full knowledge of the user's decryption key, which is also known as the key escrow problem, the user has to completely trust this third party (see Hoyle and Mitchell [92] for an in-depth discussion of the problem). Such a high trust requirement is hard to fulfill, especially in a system where a huge number of diverse users are present.

Revocation (Security): This criterion describes when revocation information is propagated to participants who want to use the revoked key material. Revocation might happen almost immediately, or periodic after a defined short timespan.

Scalability (Usability): When dealing with a huge number of users, for example in a cloud-based solution, the scalability of the chosen approach is important. Especially, supporting revocation turns out to cause scalability issues. That is, participants have to check the status of other participants' key material, in order to ensure the keys are still valid to use in an encryption process. For example, this status information can be provided through certificate revocation lists (CRLs) or the online certificate status protocol (OCSP), which cause substantial load for the public key infrastructure (PKI).

No Key Distribution Problem (Security) (Usability): The key distribution problem occurs when key material has to be handed to an entity, as this entity has to be sure of the keys' authenticity and integrity. In this criterion, we do not consider the distribution of key material from the key's owner to a participant who wants to use it, as this is a fundamental challenge which is solved in all types of proxy re-encryption. Instead, this criterion focuses on the generation of key material. If a third party is involved in the key generation, the resulting keys have to be sent to the actual owners. This transmission has to be sent over a secure channel or secured via additional means. Therefore, this might be hard to achieve, which is at least inconvenient for the involved participants.

Support for Policies (Security): This criterion examines the possibility to limit the power of misbehaving proxies and ciphertext receivers. By applying a policy, only certain ciphertexts might be re-encrypted or decrypted. Consequently, this limits which ciphertext can be decrypted.

Support for Search (Usability): This criterion evaluates if the proxy re-encryption type supports search on the encrypted data and if the search rights can also be delegated through the re-encryption operation.

Transformation (Usability): This criterion examines the different kinds of transformations, which are performed by the re-encryption operation of the individual proxy re-encryption types. Typically, ciphertext encrypted for one entity is transformed to ciphertext encrypted for another entity. However, other types of proxy re-encryption do not encrypt for entities but rather for identities referring to entities, or attribute sets which might be fulfilled by a number of entities. Therefore, the transformations are also from and to identities or attribute sets.

Evaluation

In this section, we evaluate the individual types of proxy re-encryption based on the previously identified criteria.

Trust Requirements: In *classical*, *cerificate-based*, and *conditional proxy re-encryption* as well as *proxy re-encryption with keyword search*, the key material is generated locally by the participant. Therefore, no trust in an external party is required.

With *certificate-less proxy re-encryption*, the keys are partially generated locally but also partly by a third party. Consequently, some limited trust has to be placed in this third party.

High trust into a third party is required when using *identity-based* or *attribute-based proxy re-encryption.* In those cryptographic mechanisms, the keys for an identity or the keys representing attributes are generated by a third party, which therefore also commands full decryption power.

Revocation: In *certificate-less* and *certificate-based proxy re-encryption*, the key material is only valid for a short period of time. Here, revocation is considered not renewing the key material or certification.

Similarly, in *identity-based* and *attribute-based proxy re-encryption*, we can add a validity period to an identity or as an attribute. Thereby, we achieve revocation based on a limited lifetime.

For revocation, *classical* and *conditional proxy re-encryption*, as well as, *proxy re-encryption with keyword search* build on public key infrastructure, which can be implemented in two flavors. Firstly, by using long-lived certificates, a sender can query the PKI through OCSP to check the status of the public key before using it, which allows to immediately propagate revocation information. Secondly, short-lived certificates are only valid for a limited timespan and revocation is achieved by not re-issuing a certificate after the period.

Scalability: Types of proxy re-encryption that employ a limited lifetime of key material have high scalability, as the associated infrastructure does not have to be queried for the current revocation status. However, keys or certificates have to be re-issued periodically. Nevertheless, re-issuing once per period can be significantly more efficient than checking the revocation status for each usage of the key, which might happen many times per time period. High scalability applies to *certificate-less*, *certificate-based*, *identity-based*, and *attributes-based proxy re-encryption*, as well as, proxy re-encryption types using PKI with short lived certificates, such as *classical*, *conditional*, and *proxy re-encryption with keyword search*.

In contrast, scalability is an issue for types of proxy re-encryption that employ long-lived certificates, which have to be checked for revocation before each use. This can apply to *classical* and *conditional proxy re-encryption*, as well as, *proxy re-encryption with keyword search*.

No Key Distribution Problem: In *classical*, *cerificate-based*, and *conditional proxy re-encryption* as well as *proxy re-encryption with keyword search*, the key material is generated locally by the participant. Therefore, this key material does not have to be distributed.

With *certificate-less proxy re-encryption*, the keys are partially generated locally but also partly by a third party. Consequently, the remote part has to be transmitted to the participant.

In *identity-based* and *attribute-based proxy re-encryption*, the keys for an identity or the keys representing attributes are generated by a third party, which therefore have to be distributed to the participants.

Support for Policies: *Conditional proxy re-encryption* provides support for policies. In this type, policies or attributes are attached to re-encryption keys and ciphertexts. The re-encryption operation is only successful if the attributes fulfill the policy. Thereby, proxies are limited in which ciphertexts a given re-encryption key can encrypt. This limits the potential for abuse.

Also, *attribute-based proxy re-encryption* can provide support for policies. This type of proxy re-encryption requires that the attributes of a re-encryption key match the policy attached to a ciphertext. By appending further attributes and policy conditions, the use or abuse of the re-encryption key can be limited.

In contrast, policies are not supported by the other types of proxy re-encryption, namely *classical*, *identity-based*, *certificate-less*, and *cerificate-based proxy re-encryption*, as well as, *proxy re-encryption with keyword search*,

Support for Search: *Proxy re-encryption with keyword search* is the only type that enables to search on encrypted data and makes it possible to delegate the search rights as well.

Transformation: Most types of proxy re-encryption transform ciphertext that was encrypted for one entity into ciphertext for another entity. This applies to *classical*, *certificate-less*, *cerificate-based*, and *conditional proxy re-encryption*, as well as, *proxy re-encryption with keyword search.*

In *identity-based proxy re-encryption*, data is not encrypted for an entity but rather for an identity, which refers to a currently existing or future entity. Therefore, in this type of proxy re-encryption, data encrypted for one identity is transformed into ciphertext for another entity.

In contrast, when using *attribute-based proxy re-encryption*, data is encrypted for attribute sets which fulfill a policy. Consequently, the re-encryption operation transforms data encrypted for an attribute set into data encrypted for another attribute set.

Evaluation Conclusion

	Classical PRE‡	**Identity-Based PRE**	**Certificate-Less PRE**	**Certificate-Based PRE**	**Conditional PRE‡**	**Attribute-Based PRE**	**PRE with Keyword Search‡**
Trust Requirements	L	H	M	L	L	H	L
Revocation	Immediate* / Periodic†	Periodic	Periodic	Periodic	Immediate* / Periodic†	Periodic	Immediate* / Periodic†
Scalability	L* / H†	H	H	H	L* / H†	H	L* / H†
No Key Distribution Problem	✓	✗	✗	✓	✓	✗	✓
Support for Policies	✗	✗	✗	✗	✓	✓	✗
Support for Search	✗	✗	✗	✗	✗	✗	✓
Transformation	$E \rightarrow E$	Id → Id	$E \rightarrow E$	$E \rightarrow E$	$E \rightarrow E$	Attr→Attr	$E \rightarrow E$

* denotes short-lived certificates; † denotes long-lived certificates; ‡ denotes the usage of PKI

Table 2: Comparison of Proxy Re-Encryption Types

The results of evaluating different proxy re-encryption types are summarized in Table 2. In conclusion, we recommend classical proxy re-encryption and as an improvement conditional proxy re-encryption. Further details on these proxy re-encryption types can be found in Appendix A.1.

We recommend to employ classical proxy re-encryption. In contrast to identity-based and certificate-less proxy re-encryption, this type requires no trust into a third parts, can achieve high scalability, and does not suffer from the key distribution problem. Certificate-based proxy re-encryption has the same benefits. However, classical proxy re-encryption has been established for a longer period of time and, therefore, also offers a greater variety of schemes implementing this type.

Furthermore, as an improvement over classical proxy re-encryption, we recommend conditional proxy re-encryption, which also provides support for policies. To limit the proxy's power to abuse given re-encryption key, we mainly consider conditional and attribute-based proxy re-encryption. However, in contrast to conditional proxy re-encryption, the attribute-based counterpart requires high trust assumptions and suffers from the key distribution problem. Also, in *CREDENTIAL* we want to share data from entity to entity and not focus on applying globally valid attributes to the users. Hence, conditional proxy re-encryption is more suitable.

Even though search functionality is key to provide a user-friendly experience, we do not fully recommend proxy re-encryption with keyword search. Only this type makes it possible to delegate search rights and, as it extends classical proxy re-encryption, it has similar properties, such as low trust requirements, high scalability, and no key distribution problem. However, we cannot fully recommend proxy re-encryption with keyword search. Sharing key material to perform search on encrypted data outside the proxy re-encryption ecosystem might be a better fit for *CREDENTIAL*'s use case, as this approach can be integrated into any proxy re-encryption type and scheme we choose to implement.

3.1.3 Fully Homomorphic Encryption

Fully Homomorphic Encryption [67]	
Type of Technology	Fully Homomorphic Encryption
Status	Concept paper (2009)
TRL	M - There are multiple implementations for fully homomorphic encryption and their performance has been compared.
Implementation	`https://github.com/shaih/HElib` (GPL)
Brief Description: Fully Homomorphic Encryption (FHE) allows to compute arbitrary functions on plaintexts only given the corresponding ciphertexts, without knowing any private key material or learning the plain values. In general, a scheme is homomorphic if algebraic operations propagate through the encryption. Fully homomorphic encryption supports two operations, namely addition and multiplication. With those operations it is possible to construct logic circuits and, therefore, evaluate arbitrary functions. However, fully homomorphic encryption schemes are not yet practical, as they are not efficient enough for most use cases.	

Relevance to CREDENTIAL: As fully homomorphic encryption schemes can evaluate arbitrary functions on the input's ciphertext, a multitude of usage scenarios is possible. For example, FHE could be used to perform access control, as a function could re-encrypt ciphertexts only if they satisfy some arbitrary policy. Furthermore, as FHE is very powerful, it could be used to derive new attributes from existing, encrypted attributes.

Fully homomorphic encryption schemes might enable powerful extensions to the *CREDENTIAL* data sharing platform, in particular when combined with proxy re-encryption schemes. For instance, one could then decide that a receiver is not allowed to access single data sets, but only arbitrary functions such as certain statistics over large amounts of data.

Despite its potential benefits, we omit a detailed evaluation of fully homomorphic encryption schemes here. This is because at the time of writing, such schemes are not practically efficient. For instance, despite recent advances (e.g., [51, 44]), the bootstrapping step (which is inherently needed in all existing schemes to enable the *fully* homomorphic feature) requires key sizes ranging between many megabytes up to gigabytes even when encrypting only single bits.

However, future advances in this area might lead to a revision of this evaluation. Furthermore, we believe that in particular the combination with proxy re-encryption might be a valuable area of research for privacy-preserving data sharing applications.

3.2 Authentic Data Disclosure

In traditional signature schemes, every single bit flip in the signed document will immediately invalidate the signature. However, one goal of *CREDENTIAL* is to give the data owner full control over which parts of potentially authentic (i.e., signed) data she wants to share with other parties. For instance, consider the following scenario: a user wants to authenticate herself to a service provider using her electronic identity card. The service provider requires a proof that the user is at least 65 years old, e.g., for granting a price reduction. Now the user should be able to proof that she is old enough to receive the discount, but the service provider should not learn all the other information contained in the eID.

We therefore analyzed various technologies that allow controlled modifications of a document after receiving a signature on it.

3.2.1 Malleable Signature Schemes

In the following, we first assess the different types of malleable signature schemes, and then compare different instantiations of the chosen primitive, redactable signatures.

Redactable Signatures [103]	
Type of Technology	Digital Signature Scheme
Status	Concept paper (2002)

TRL	L - The technology concept was formulated and applications were defined, but there is no claim of an implementation.
Brief Description: In documents that were signed using a redactable signature scheme, it is possible to redact arbitrary parts, while still being able to verify the signature. This redaction can be performed by any entity without requiring interaction with the original signer. A redactable signature scheme has to provide a way to ensure that only semantically sound redactions are possible.	
Relevance to CREDENTIAL: Redactable Signatures could be used to only disclose a minimized set of the user's attributes.	

Sanitizable Signatures [16]	
Type of Technology	Digital Signature Scheme
Status	Concept paper (2005)
TRL	M - The authors of this paper implemented their signature scheme and presented a performance analysis.
Brief Description: Documents that were signed using a sanitizable signature scheme, can be modified by an explicit redactor in a controlled and limited fashion, while still staying valid. A redactor is usually able to modify or remove designated areas of the document.	
Relevance to CREDENTIAL: Sanitizable Signatures could be used to only disclose a minimized set of the user's attributes by removing undesired parts. However, the ability to modify parts of the user's data has to be carefully investigated.	

Blank Digital Signature [81]	
Type of Technology	Digital Signature Scheme
Status	Concept paper (2013)
TRL	L - The technology concept was formulated and applications were defined, but there is no claim of an implementation.
Brief Description: In the blank digital signature scheme, the signer first defines a template, which consists of fixed parts, as well as variable parts with a set of values. This template is signed by the signer. A designated redactor is then able to select one of the predefined values for each variable. The signature still holds for such an instance of the template.	
Relevance to CREDENTIAL: The use of blank digital signatures could help to fulfill the requirement to only disclose a minimal set of information about the user.	

Functionality-wise, we strive for malleable signature schemes that allow for redacting signed documents at the wallet's end without any need of additional (secret) key material. We argue that sanitizable and blank digital signatures (as special cases of malleable signatures) are not suitable in the *CREDENTIAL* setting. That is that within the sanitizable and blank digital signatures context, the sanitizer and the proxy are in need of secret-key material, respectively, and hence it would require that the wallet must manage keys with each issuer. Redactable signature (RS) schemes in turn allow for redaction of signed document *without* the need of additional key material at the wallet. Hence, redactable signature schemes fit our needs functionality-wise.

In the following we therefore do not further evaluate sanitizable or blank signatures any further, but solely concentrate on redactable signature schemes. Due to the large body of work in this field and the many different flavors of redactable signature schemes, we perform an in-depth evaluation of representative schemes in order to identify the most suitable types and instantiations of RS for *CREDENTIAL*.

Evaluation Criteria

The following criteria have been chosen as a basis to evaluate different redactable signature schemes:

Unforgeability (Security): There exist numerous notions of unforgeability for signature schemes. We consider EUF-CMA (existential unforgeability under chosen message attacks, (one of) the standard unforgeability definitions for digital signature schemes) as the minimum security that needs to be guaranteed.

Privacy (Privacy): For redactable signature schemes privacy means that redacted data is hidden and cannot be reconstructed by a receiver.

Transparency (Privacy): This property guarantees that a third party cannot tell whether or not a redaction took place.

Unlinkability (Privacy): This property guarantees that an adversary cannot tell whether two redacted versions were generated from the same or different documents, if this cannot be decided based on the revealed data.

Model (Security): We analyze whether a scheme is secure in the standard model or whether it requires idealized assumptions such as the availability of random oracles.

Assumption (Security): We analze under which complexity assumptions a scheme can be proved secure. For schemes that can generically be built from a variety of concrete instantiations, this criterion cannot be evaluated but depends on the used building block.

Efficiency Criteria (Usability): We compare different aspects of efficiency, such as key or signature sizes, or signing, redaction, or verification costs.

Evaluation

In the following Table 3, we evaluated the schemes under the mentioned criteria.

Evaluation Conclusion

It is not possible to give a concrete recommendation which scheme to use in privacy-preserving applications or even within the *CREDENTIAL* project. This is because different application scenarios require different privacy features (e.g., unlinkability is important for identity provisioning but might be less important for sharing diabetes diaries). Also, the chosen schemes need

	Brzuska et al. [31]	**Pöhls et al. [147]**	**Ahn et al.[9]**	**Samelin et al. [157]**	**Derler et al. [50]**
			Security and Privacy Criteria		
EUF-CMA	✓	✓	✓(selectively)	✓	✓
Privacy	✓	✓	✓(strong)	✓	✓
Transparency	✓	✓	✓	✓	✓
Unlinkability	✗	?	✓	?	?
Model	Standard	Standard	Random Oracle	Standard	Standard
Assumption	Generic	Generic	CDH	Generic	Generic
			Efficiency Criteria		
Signature size	$O(n^2) \cdot \|\sigma\|$	$O(n) \cdot \|\sigma\|$	$O(m \log m)$	$O(n) \cdot \|\sigma\|$	$O(m) + \|\sigma\|$
Signing key size	$\|sk\|$	$\|sk\|$	$O(\log m)$	$\|sk\|$	$\|sk\|$
Verification key size	$\|vk\|$	$\|vk\|$	$O(\log m)$	$\|vk\|$	vk
Key generation costs	T_K	T_K	$O(\log m)$	T_K	T_K
Signing costs	$O(n) \cdot T_S$	$O(n) \cdot T_S$	$O(\log m)$	$O(n) \cdot T_S$	$T_S \cdot m$
Redaction costs	$O(r)$	$O(r)$	$O(r \log r)$	$O(r)$	$O(r)$
Verification costs	$O(n) \cdot T_V$	$O(n) \cdot T_V$	$O(\log m)$	$O(n) \cdot T_V$	$T_V \cdot m$

Table 3: Some constructions are generic in a way that they use an underlying schemes (e.g., signature or accumulator schemes); hence, we denote with vk, sk the verification key and the secret key of the scheme; we write σ to denote the signature; T_K, T_S, T_V denote the computational costs of key generation, signing, and verification, respectively. Let m and r be the maximum length of the message and the length of its redactions, respectively. Brzuska et al.'s, Pöhl et al.'s, Derler et al.'s schemes utilize a tree-based approach with n nodes. Ahn et al. consider quotable signature schemes.

to be compatible with all the other employed cryptographic primitives, in particular the chosen proxy re-encryption scheme. Depending on the concrete scenario any of the evaluated schemes could be used within our project.

3.2.2 Anonymous Credentials vs Redactable Signatures

In the following, we give a short overview of some of the most prevalent anonymous credential systems, which in some sense can be seen as an extension of redactable signature schemes. However, later in this section, we will not evaluate the different schemes against each other, but rather compare anonymous credential systems against redactable signature schemes on a primitive level.

Idemix Anonymous Credentials [34]	
Type of Technology	Anonymous Credentials

Status	Concept paper (2002)
TRL	M – Idemix has been trialled in operational environments, i.e., with real applications and real users.
Implementation	`https://abc4trust.eu/idemix/` (Apache 2 license)
Brief Description: In a system using Anonymous Credential, an organisation first checks a list of attributes regarding user, which are then issued as an anonymous credential. Those credentials can then be used to convince a verifier, for example a wine-merchant, that the user fulfills some requirements, such as being of age. The advantage in comparison to normal signature schemes is, that requirements can be proven while only revealing a minimal set of information about the user. Anonymous credentials allow to reveal single attributes, and to perform arithmetic, comparison as well as logic operations on the attributes, without actually revealing the data itself. Some features include 1. Minimal information disclosure 2. Pseudonymity 3. Conditional anonymity 4. Multi-show unlinkability The signature scheme underlying the idemix system are CL signatures.	
Relevance to CREDENTIAL: The use of anonymous credentials could help to fulfill the requirement to only disclose a minimal set of information about the user.	

U-Prove Anonymous Credentials [143]	
Type of Technology	Anonymous Credentials
Status	Technology report (2013), Version 1.1
TRL	M – U-Prove has been trialled in operational environments, i.e., with real applications and real users.
Open Source implementation	`http://research.microsoft.com/en-us/projects/u-prove/`
Brief Description: From a functionality point of view, the main difference to idemix is that UProve only supports single-show credentials, i.e., re-using a credential more than once makes the user's actions linkable in contrast to idemix, where an arbitrary number of presentations of the same credential can be kept unlinkable. The signature scheme underlying the UProve system are Brands signatures.	
Relevance to CREDENTIAL: The use of anonymous credentials could help to fulfill the requirement to only disclose a minimal set of information about the user.	

Persiano Anonymous Credentials [145]	
Type of Technology	Anonymous Credentials
Status	Concept paper (2004)
TRL	M – Proof of concept validation (lab environment) in highly dynamic scenarios such as vehicular networks.
Brief Description: From a functionality point of view, the Persiano system is comparable to idemix, cf. above.	
Relevance to CREDENTIAL: The use of anonymous credentials could help to fulfill the requirement to only disclose a minimal set of information about the user.	

Evaluation Criteria

When comparing anonymous credential systems against redactable signature schemes, we will focus on the following criteria. Note that as we are comparing two clusters of technologies also the criteria are rather clusters of properties rather than specific properties.

Supported functionality (Usability). This criterion analysis which functionalities going beyond the plain feature of selective disclosure are typically supported by anonymous credential systems and redactable signature schemes, respectively.

Properties (Privacy) (Security). We analyze here all security and privacy features of the two primitives with regard to the use cases identified within the *CREDENTIAL* project.

Efficiency and Scalability (Usability). This criterion evaluates the computational costs of both primitives, in particular when applying them to larger documents with many redactable blocks.

Flexibility (Integration effort). We here estimate how easy it will be to integrate the schemes with the remaining *CREDENTIAL* framework, including the combination with other primitives like proxy re-encryption. To guarantee an in-time delivery of the *CREDENTIAL* results, this is a key criterion.

Cloudifiability (Integration effort) (Usability). As we are aiming for a maximum degree of cloudification for our systems, we also analyze to which degree the two primitives can be cloudified with reasonable effort within the project.

Evaluation

Supported functionality. Anonymous credential systems, which are also described in detail in Appendix A.4, can be seen as an extension of unlinkable redactable signatures.[1] That is, they not only allow one to selectively disclose or hide specific blocks of a document towards the intended receiver, but typically also support a large variety of additional functionality, such as revocation, (scope-exclusive) pseudonyms, or predicates over attributes to allow a user, e.g., to prove that she is at least 18 years old without revealing the actual birth date. All those features are, e.g., covered by the most prominent anonymous credential schemes, Microsoft's UProve [143] and IBM's identity mixer (aka idemix) [34], see [35] for a formal treatment.

In terms of functionality, attributed-based credential systems are thus superior to redactable signature schemes.

Properties. In contrast to redactable signature schemes, existing anonymous credential systems typically do not support transparency, as they usually assume a known structure of a document (e.g., an electronic identity card) and thus it is always known whether or not

[1]Actually, it is possibly to build unlinkable redactable signatures from anonymous credential systems and vice versa, using additional cryptographic primitives such as commitments and generic zero-knowledge proofs of knowledge.

data fields have been removed. However, in the context of data sharing, this feature may be required depending on the concrete scenario and type of shared data. For instance, the number of redacted entries in a treatment history report might by itself already be privacy-sensitive information.

On the other hand, multi-show credentials (see above) canonically support unlinkability as an inherent feature of such schemes. Yet, the evaluation of redactable signatures in Section 3.2.1 shows that this feature can also be achieved by specific redactable signature schemes.

Efficiency and Scalability. Credential systems are usually optimized for relatively small numbers of attributes, but can be extended to an arbitary number of blocks, which however often causes high computational costs and larger keys; for instance, in the case of [34], each redactable message block causes one exponentiation in a group of large order.

On the other hand, there exist redactable signature schemes where each additional message block only requires one additional evaluation of a cryptographic hash, which is computationally less expensive than in the anonymous credential case.

Flexibility. Because of the larger variety of redactable signature schemes, as well as their lower algebraic and algorithmic complexity, we expect that an integration of redactable signatures with proxy re-encryption is easier than in the anonymous credential case. Also, as their structure is closer to standard signatures, we expect that they are easier to integrate with other used technologies like authentication protocols.

Cloudifiability. The high-level setup of anonymous credential systems does not involve a central cloud platform which computes presentation tokens to authenticate a user towards a service provider. Rather, this computation is assumed to be executed locally on the user side. However, this is not a suitable setting in the case of data sharing, where the data owner might not be online when the data is accessed by the receiver. Also, in the case of shared devices, having the credential locally on the device might not be desirable from a privacy and security point of view.

Because of the lower required research efforts for combining redactable signatures with proxy re-encryption schemes, we thus believe that also the privacy-preserving cloudification of redactable signatures will be easier to achieve in the required timeframe than for anonymous credential schemes.

Evaluation Conclusion

At this point, we opt for redactable signature schemes for being used in *CREDENTIAL*. This is due to the fact that transparency might be important feature in the data sharing scenarios, the higher efficiency of redactable signatures, and that the combination of redactable signatures with other required technologies seems more feasible than for full-fledged anonymous credential systems.

However, we recommend that to keep in mind the extended functionalities and features offered by anonymous credential systems, and integrate them whenever possible.

3.3 Section Conclusion

We assessed and evaluated the main cryptographic primitives which seemed to be of potential relevance for *CREDENTIAL*. We argued why we recommend proxy re-encryption over attribute-based encryption and redactable signatures over attribute-based credentials, even though also their counterparts would be viable alternatives with their own specific advantages. Within the concrete recommended technologies, we provided an overview of the available variants and gave concrete recommendations for usage within *CREDENTIAL*.

Full details on the discussed technologies can either be found in the original literature or in Appendix A.

4 Additional Cryptographic Technologies

In this section, we describe and evaluate additional cryptographic technologies which are relevant for *CREDENTIAL* in the context of authentication, access to encrypted data, and related technologies. Concretely, in Section 4.1, we list password-based authentication technologies in fact sheets. Password-based systems are for example relevant to store cryptographic key material. In Section 4.2, access-to-encrypted-data technologies are described and evaluated. That is since we inherently deal with encrypted data in the *CREDENTIAL*-wallet, cryptographic ways to access, search, audit, and retrieve encrypted data must be considered. Further related technologies are given and evaluated in Section 4.3; in particular, secret sharing (share a sensitive message securely among many servers), unlinkable pseudonyms, and verifiable computing (convince a client of a correct server computation) are considered.

4.1 Authentication

In this subsection, we cluster the password-based cryptographic technologies TPASS, Distributed Password Verification, and Asymmetric Password-based Cryptography in form of fact sheets. In Appendix B, we provide more details on TPASS.

TPASS (Threshold Password-Authenticated Secret Sharing) [33, 37]	
Type of Technology	Password Management and Secret Sharing
Status	Concept Paper (2015)
TRL	L - The construction has been presented in the paper, but there is no implementation.
Brief Description: In a TPASS system, the user's data, which includes her password, is distributed across multiple servers. Those servers engage in a protocol to authenticate the user and to reconstruct a strong secret, which can be used for further cryptographic operations. Even if a number of servers below the threshold are compromised, an attacker is not able to determine the user's data in a brute-force attack. The second paper describes an improved TPASS protocol that is more secure and provides better usability, than the one discussed above (Camenisch et al.). Firstly, the user only has to know her password to authenticate and to re-construct her secret key. No data from a prior setup has to be remembered by the user, which improves the mobility of this solution. Secondly, unlike in the previous approach, even if the user performs the protocol exclusively with malicious servers, they are not able to learn the user's password. Therefore, the described solution is also more secure.	
Relevance to CREDENTIAL: Such a TPASS system could be used to implement password based authentication, and furthermore, provide the user with means to store and access key material.	

Distributed Password Verification [38]	
Type of Technology	Password Management
Status	Concept Paper (2015)

TRL	M - An implementation and performance measurements were presented in the paper.
Brief Description: This paper describes an approach to distribute password verification across multiple servers. An adversary can only perform an offline brute-force attack on stored user data, if all servers are compromised. In contrast to other concepts, such as the above mentioned TPASS, recovery from corruptions is computationally inexpensive.	
Relevance to CREDENTIAL: This concept could be used to implement a password authentication mechanism, that is hard to attack offline, and which has an efficient process in place to recover compromised servers.	

Asymmetric Password-based Cryptography [20]	
Type of Technology	Password Management and Secret Sharing
Status	Concept Paper (2013)
TRL	L - The construction has been presented in the paper, but there is no implementation.
Brief Description: In the usual (symmetric) password based encryption (PBE) a secret key is derived by salting and hashing the password. This key is then used to encrypt communication between user and server. However, if the server is compromised and an attacker gains access to the hashed and salted passwords, which also act as keys, all prior communication can be decrypted. This paper examines approaches to implement asymmetric PBE. In an asymmetric PBE scheme a public and a private key is derived from the user's password. The server only uses such a public key for encryption, so even if compromised, the adversary is not able to decrypt anything. The paper describes an invasive variant of the PBE, which introduces a new key-derivation function, as well as a non-invasive variant, which can be used with already existing and deployed key-derivation functions.	
Relevance to CREDENTIAL: This concept could be used to derive asymmetric key material from user's passwords. Since those keys are asymmetric, an attacker would only be able to perform limited operations. For example, an attacker would only be able to encrypt but not decrypt messages.	

Especially technologies for distributed password verification could potentially be of interest for a *CREDENTIAL*-like system. For instance, a TPASS scheme could be used to securely store backup or recovery keys for users. However, for the time being we consider such mechanisms out of scope of the *CREDENTIAL* wallet, as they can be seen as an independent add-on to the developed applications and tools.

4.2 Access to Encrypted Data

In this subsection, we cluster and evaluate Searchable Encryption, Private Information Retrieval, Oblivious RAM, and Proofs of Retrievability and Provable Data Possession. More details on the given technologies can be found in Appendix B.

4.2.1 Searchable Encryption

Search on Encrypted Data [19]	
Type of Technology	Search
Status	Concept Paper (2007)
TRL	H - Multiple implementations allow search on encrypted data. For example: SAP SEEED, Google's Encrypted BigQuery, and Microsoft's Always Encrypted SQL Server
Implementation	`https://github.com/sashank/jsse` (LGPL)
Brief Description: Search on encrypted data usually deals with finding all documents that are associated with a keyword. The user has to perform a two-step setup process. Firstly, the user encrypts her documents and, secondly, creates an encrypted database of metadata about those documents. The encrypted documents and encrypted database are then handed to a server. To search for documents, the user would send a cryptographically encapsulated keyword to the server. The server would use it with the metadata database to figure out which documents should be sent back. There are multiple approaches to realize search on encrypted data. Those approaches differ in their security, that is how much information they leak, and their performance characteristics.	
Relevance to CREDENTIAL: Since credential might store a huge amount of the user's data, search functionality on the ciphertexts would be beneficial.	

A very promising approach in the private-key setting to searchable encryption (SE) is symmetric searchable encryption (SSE). This can be realized by Fully Homomorphic Encryption (FHE) [67], Oblivious RAM (ORAM) [72], and Private Information Retrieval (PIR) [46]. These building blocks offer well-researched security guarantees (without leaking significant information as, e.g., access and search pattern or index information). At the same time, unfortunately, all of the mentioned building blocks are very expensive in complexity and considered to be practically inefficient. (However, current research argues that ORAM is getting more practical [5].)

When one is willing to allow significant leakage (i.e., take reduced security guarantees into account), one can derive practical SSE schemes using Deterministic Encryption (DE) [19] and Order-Preserving Encryption (OPE) [26]. Schemes that offer such guarantees are used in, e.g., CryptDB [148]. However, it was shown that CryptDB (and their derivates) leak many crucial information regarding the search on the encrypted data [125].

In the public-key setting (i.e., where any external user can write the encrypted and searchable data using the public key), one can go for Public-Key Encryption with Keyword Search (PEKS) [27].

We point to [28] for an overall survey of the topic SE.

Evaluation Criteria

In the following, we briefly describe the evaluation criteria used for searchable encryption.

Security (Security). The security of SE scheme is evaluated in terms of the security model, leaking access and search pattern, as well as index information.

Efficiency (Usability). Efficiency is measured in terms of communication and computation overhead.

Single-user/multi-user setting (Usability). A SE scheme can be in the single-user and multi-user setting (i.e., how many users can write/access data).

Query expressiveness (Usability). The query can have different levels of expressiveness (e.g., equality of keyword or inner product).

Evaluation

Table 4 gives a detailed overview over current SE schemes with respect to the criteria mentioned above. (We emphasize that SE is vibrant area of research and we only cover well-established SE schemes here. However, we want to point out that more recent (theoretical) SE results are known and under investigation, e.g., [42].)

	Song et al. [163]	**Curtmola et al. [47]**	**Boneh et al. [27]**	**Shen et al. [162]**
Security	L	M	M	H
Efficiency	L	H	M	L
Single-user/multi-user	Single	Multi	Multi	Single
Query expressiveness	Equality	Equality	Equality	Inner product

Table 4: Comparison of searchable encryption schemes in terms of the mentioned evaluation criteria. H, M, and L means high, medium, and low, respectively.

Evaluation Conclusion

A concrete suggestion on which SE scheme to use within *CREDENTIAL* is highly non-trivial. It very much depends on the setting and how much leakage is acceptable in order to efficiently perform searchable encryption. Although Table 4 gives an overview of a selection of well-established SE scheme, one has to be careful with the details (in terms of security and efficiency) when one is willing to implement such a scheme in a concrete setting.

4.2.2 Private Information Retrieval

In general, private information retrieval (PIR) could offer a valuable feature to realize *CREDENTIAL*'s vision of a privacy-preserving identity management and data sharing platform. This is because such schemes (partially) hide the access patterns to different resources. Therefore, in particular when combined, e.g., with hidden or encrypted access control lists, the *CREDENTIAL* wallet would no longer learn who is accessing which data when.

Private Information Retrieval [70]	
Type of Technology	Private Information Retrieval
Status	Concept Paper (2007)
TRL	M - Multiple open source implementations are available.
Implementation	`http://percy.sourceforge.net/`, `https://crypto.stanford.edu/pir-library/`
Brief Description: Even if the data is encrypted, a cloud storage provider might use the access pattern of one or more users to learn something about the data or users. For example, the access pattern could expose the user's habits and privileges, or if multiple users access the same file reveal a collaborative relationship. Private Information Retrieval (PIR) is a method to access data items without revealing this access pattern. In PIR schemes with multiple servers, the data can be protected if up to a threshold of servers collude, and queries can be correctly answered if only a subset of servers responds, while some of these servers are byzantine (i.e. send malformed/incorrect answers). However, the performance characteristics are of vital importance of the practicability of such a PIR scheme. As a result, cloud providers cannot build behavior profiles and learn sensitive information through inference using access patterns.	
Relevance to CREDENTIAL: Private Information Retrieval could be used in *CREDENTIAL*, so that the cloud wallet does not learn which data the user tries to access. However, if a data receiver wants to privately access data, the re-encryption and signature redaction would have to be integrated, which might further limit the efficiency and privacy benefits, which therefore requires further research.	

However, there are fundamental limitations to PIR schemes which render them impracticable in large scale deployments. Namely, it is easy to see that for a storage server to not learn any information about which data has been accessed, it is inherently necessary that from the server's perspective every stored object and bit has to be treated in an identical way. This means that the storage server has to touch every stored bit for every single access to the data and perform a (typically highly efficient) computation on it. The communicational complexity varies depending on the security requirements and the number of independent storage servers being used, cf. Table 5.

Evaluation Conclusion

Given the high computational overhead per server, we do not consider PIRs practical for large scale deployment within *CREDENTIAL*, as the negative performance or cost impact seems to

	1-server information theoretic PIR	k-server information theoretic PIR	1-server computational PIR	k-server computational PIR
Computation per server	n	n	n	n
Overall communication	n	$n^{O(\frac{\log\log k}{k \log k})}$	$O(1)$ (amortized)	$O(1)$ (amortized)

Table 5: Computational and communication efficiency of existing PIR schemes for databases of size n.

strong. This is particularly true as in our concrete setting, the *CREDENTIAL* wallet would additionally have to re-encrypt all potentially accessed data for every user as it would not know which data is actually accessed, thereby causing an even higher computational overhead.

4.2.3 Oblivious RAM

Oblivious RAM (ORAM) can be seen as an extension of private information retrieval, where not only the read access patterns but also the write operations are disguised. As such, ORAM has inherently at least the same costs as PIR.

Oblivious RAMs [72]	
Type of Technology	Oblivious RAMs
Status	Concept Paper (1996)
TRL	M - Multiple research implementations, such as ObliviStore or PrivateFS, were used in relevant environments and compared in performance.
Brief Description: In contrast to Private Information Retrieval, Oblivious RAMs (ORAMs) not only allow to read data from but also to write data to a storage service without revealing any access pattern. ORAM schemes usually involve performing a search for the desired item as well as inserting a new item, in order to disguise the kind of operation, read or write. After some number of operations, the data has to be reorganized. Improvements on the efficiency of this reorganization is the focus of some research. In recent years, the emergence of cloud computing promoted effort to integrate ORAM into practical implementations, such as ObliviStore or PrivateFS.	
Relevance to CREDENTIAL: ORAMs could be used in *CREDENTIAL* in two ways. Firstly, the cloud wallet could use ORAMs to obliviously access data stored on an external attribute provider. Secondly, a data receiver, such as the user or a service provider, could use the ORAM method to access re-encrypted and redacted attributes from the cloud wallet.	

Despite of its potential positive privacy impact, ORAM is thus currently not recommended to be used within the *CREDENTIAL* project. This can also be underpinned by actual implementations such as PrivateFS [173] or ObliviStore [164]. The former, mounted as a local file system, achieves throughputs below 100kB/s, and a single-digit number of database queries when used remotely. The latter outperforms this by a factor of almost 20, but only supports a single client.

Evaluation Conclusion

In either case, the throughputs are currently insufficient for large-scale deployments; yet, a re-evaluation might become necessary if breakthrough results are found.

4.2.4 Proofs of Retrievability and Provable Data Possession

Provable Data Possession (PDP), introduced in [14], and Proofs of Retrievability (PoR), introduced in [111], are two independently introduced concepts for checking the integrity and availability of data stored on a remote server. Originally, there were some fundamental differences between PORs and PDPs. For instance, the former were only intended to allow for a limited number of checks, while the latter supported an unlimited number of integrity checks. In PORs, files were first encoded and then stored remotely while PDPs did not require any encoding, which in turn allowed PORs, in contrast to PDPs, to also correct minor corruptions. Also, there were subtle differences in the proposed security guarantees. However, in recent works the two primitives seem to converge to a unified approach, which is also why they are going to be evaluated as a single tool here.

For details on the technologies we refer to Appendix B.2.3 and Appendix B.2.4.

Proofs of Retrievability (PoR) [111]	
Type of Technology	Remote Data Checking
Status	Concept Paper (2007)
TRL	M - There are multiple open source implementations and the performance of a PoR construction was analyzed for example in [29].
Implementation	`https://code.google.com/archive/p/proof-of-retrievability/` (BSD license), `https://code.google.com/archive/p/compact-proofs-of-retrievability/` (BSD license)
Brief Description: Proofs of Retrievability is a method to check if a remotely-stored file is still available and intact. The objective of this method is to perform the check, while requiring the user to only store a minimal set of verification data and the storage provider to perform as few computations as possible. In particular, before uploading small amounts of data for spot-checking (referred to as “sentinel values”) are appended, the blocks are shuffled and then encrypted. In order to check the file, the user requests encrypted blocks that contain sentinel values, decrypts them and verifies the values.	
Relevance to CREDENTIAL: PoR could be used in two ways in *CREDENTIAL*. Firstly, if the cloud wallet outsources the data storage to an external attribute provider, the wallet could check if the data is still available and intact at that external entity. Secondly, the user could check if her data is available via the *CREDENTIAL* system.	

Provable Data Possession (PDP) [15]	
Type of Technology	Remote Data Checking
Status	Concept Paper (2007)

TRL	M - The performance of the initial PDP construction was analyzed in the referenced paper and multiple open source implementations are available.
Implementation	`https://code.google.com/archive/p/provable-data-possession/` (BSD license), `https://code.google.com/archive/p/scalable-and-efficient-provable-data-possession/` (BSD license)
Brief Description: Provable Data Possession (PDP) is very similar to Proofs of Retrievability (PoR), as it too is a method to check if a remotely-stored file is available and intact, while trying to keep the computational and storage requirements low for both the client and the server. In contrast to PoR, PDP were initially designed to support an unlimited number of challenges and to detect large corruptions of the file. However, in recent approaches both PoR and PDP converge towards a model that supports an unlimited number of challenges and even detects small corruptions.	
Relevance to CREDENTIAL: PDP, like PoR, could also be used to check if data is available and intact at the cloud wallet or at an external attribute provider.	

Evaluation Criteria

In the following we briefly describe the evaluation criteria used for PORs and PDPs.

Unlimited Verifications (Usability). We will only take into consideration schemes that support an unbounded number of integrity checks, as we believe that bounding the number of such checks a-priori is often not possible, in particular in the case of long-term storage.

Verifiability (Security, Usability). Privately verifiable PORs and PDPs only allow the legitimate owner of the data to verify whether they are still available, while in publicly verifiable schemes this task can be outsourced to a (semi-trusted) third party without violating the user's privacy. While publicly verifiable schemes can in particular be also used by the user herself, the appropriate choice here very much depends on the concrete use case and type of data being stored.

Computational Costs (Usability). The computational efficiency of a scheme can be measured regarding the tagging (or pre-processing) costs, as well as the costs for the server and the client when executing the protocol.

Communication Overhead (Usability). The communication costs determine how much data needs to be transferred (in addition to the data itself) when storing it on the server.

Storage Overhead (Usability). The storage overhead measures how many bits need to be stored in addition to the data itself.

Evaluation

Table 6 gives a detailed overview over chosen recent POR and PDP schemes with respect to the criteria mentioned above. Concerning the computational efficiency, we solely focus on the number e of full-length exponentiations of 2048 bit integers, as well as the number P of required pairings. This is because those terms usually determine the overall costs of the computations compared to simple field operations like multiplication or inversion, or point addition on elliptic curves. If none such operation is required, the computational costs are thus labeled as “small”.

Apart from this, we use κ to denote the security parameter, n to denote the overall number of files to be stored, and t to denote the number of file elements to be stored within a single tag. Finally, $\ell = n/t$ is the number file blocks and c is the number of file blocks being challenged in an execution of the protocol.

	S-PDP [13]	**SPOR [158]**	**EPOR [176]**	**SRPDP [82]**	**P-PDP [13]**	**PPOR [158]**	**SRPDP [82]**
Unbounded Verifications	✓	✓	✓	✓	✓	✓	✓
Verifiability	Private	Private	Private	Private	Public	Public	Public
Keysize	2048	2048	224	224	2048	224	224
Computation Tagging	$2\ell\kappa\mathsf{e}$	small	small	small	$2n\kappa\mathsf{e}$	small	small
Computation Costs Server	$(2ct + c)\mathsf{e}$	small	small	small	$c\mathsf{e}$	small	small
Computation Costs Client	$(c + 2)\mathsf{e}$	small	small	small	$(c + 2)\mathsf{e}$	$2\mathsf{P}$	$3\mathsf{P}$
Communication Overhead	$(c+1)\kappa+h$	$(2t + c + 1)\kappa + h$	$(c + 3)\kappa$	$(t + 1)\kappa$	$(c + 2)\kappa$	$(2t+c+2)\kappa$	$(t + 1)\kappa$
Storage Overhead	$\ell\kappa$	$\ell\kappa + t\kappa$	$\ell\kappa$	$\ell\kappa$	$\geq n\kappa$	$\ell\kappa+(t+1)\kappa$	$\ell\kappa$

Table 6: Evaluation details for POR and PDP schemes.

Evaluation Conclusion

Based on our analysis, the schemes of Xu and Chang [176] is recommended for privately verifiable scenarios, and the schemes of Hanser and Slamanig [82] are recommended for scenarios requiring private or public verifiability, respectively. Concerning the concrete setting of *CREDENTIAL*, we consider PDPs and PORs as interesting extensions of the core functionality. However, given the inherently necessary trust assumptions coming from our PRE based approach, we believe that such schemes are not stringently required for the wallet.

4.3 Other Technologies

In this subsection, we cluster (Un-)linkable Pseudonyms, Secret Sharing, and Verifiable Computing and provide evaluations for Secret Sharing and Verifiable Computing. In Appendix B, we provide more details on the described technologies.

4.3.1 Unlinkable Pseudonyms

In the following we briefly summarize the concept of unlinkable pseudonyms; for a detailed description, we refer to Appendix B.3.2.

Unlinkable Pseudonyms [36]	
Type of Technology	Unlinkable Pseudonyms
Status	Concept Paper (2015)
TRL	L - The construction has been presented in the paper, but there is no implementation.
Brief Description: If data from one server should be exchanged with another server, these items have to be linkable. However, to preserve privacy, especially sensitive data should be unlinkable.This paper describes an approach, where a central converter is used to establish individual user-pseudonyms for each server. These pseudonyms are derived from unique main identifier that each user has. The converter is the only entity that can link pseudonyms together, but it still does not learn pseudonyms or user identifiers involved. The converter only learns which servers want to exchange data.	
Relevance to CREDENTIAL: By making use of this concept, the individual components of the *CREDENTIAL* architecture as well as external services would not be able to link the user's data without express consent of a central conversion service. This could contribute towards the goal of improve the user's privacy.	

Unlinkable pseudonyms, in all their different flavors—i.e., also without the aforementioned conversion service as described, e.g., by Camenisch et al. [35]—, are important features for privacy-enhancing systems and applications. However, we omit a detailed evaluation here, as they are not inherently needed for realizing the use cases and pilots identified in *CREDENTIAL*. This is, because all the services considered in the project are stateful or need to unambiguously identify the user, and therefore the use of pseudonyms is not of direct benefit for the user, as a returning user needs to be recognized in either scenario. Furthermore, there is currently no need for users to, e.g., link different accounts with different service providers together. Finally, given the low maturity level of the technology, the implementation and adaption overhead might be too high for the scope of *CREDENTIAL*. However, for other use cases and future developments, they might offer a valuable extension to the features of the *CREDENTIAL*wallet.

4.3.2 Secret Sharing

Secret Sharing [159]	
Type of Technology	Secret Sharing
Status	Concept paper (1979)
TRL	H - Secret sharing is widely used and integrated into commercial products.
Implementation	`https://launchpad.net/libgfshare` (MIT license), `http://point-at-infinity.org/ssss/`, `https://github.com/rxl/secret-sharing` (MIT license)
Brief Description: A secret, such as a key, is split into shares, which are distributed amongst a group of participants. The shares of a sufficient number of participants have to be combined, in order to reconstruct the secret.	
Relevance to CREDENTIAL: In case the user's key was lost, secret sharing could be used to recover the key, if shares were distributed to multiple semi-trusted entities, like family members or secure servers.	

As described in detail in Appendix B.3.3, secret sharing schemes allow one to decompose data into multiple shares such that only predefined qualified subsets of shares are able to reconstruct the original data, while any non-qualified subset of shares does not learn any information about the shared information. In theory, arbitrary "access structures", i.e., sets of qualified subsets of shares can be realized, as long as every superset of a qualified set is again qualified. However, when using secret sharing to store data across multiple storage nodes, we believe that threshold schemes are the most reasonable choice. Such schemes realize access structures where, for some predefined threshold value t, every set of at least $t + 1$ shares can be used for reconstruction, while every set of at most t shares cannot. In the following evaluation we will thus focus on such schemes, as other access structures would lead to the situation that (a priori) more trusted nodes would receive more information, which would make them more interesting for attackers, which in turn would again decrease the potential trust.

Evaluation Criteria

Many extensions to the basic functionality of secret sharing schemes can be found in the literature. In the following, we briefly discuss those which would be relevant when using such schemes within the *CREDENTIAL* wallet.

Verifiability (Usability). A secret sharing scheme is verifiable if the storage nodes can—**interactively** or **non-interactively**, i.e., with or without communication among each other—verify whether the distributed shares are consistent with each other. In particular, when using secret sharing for distributed storage scenarios, this allows to detect malicious dealers (the parties decomposing the original data), and prevent them from storing inconsistent "garbage" on the storage servers, thereby preventing other parties from accessing the data.

Auditability (Usability). Auditability is closely related to proofs of retrievability and proofs of data possession, cf. Appendix B.2.2 and Appendix B.2.4. It allows the legitimate owner of some data to efficiently and remotely verify whether his data is still stored and available on all storage nodes. Depending on whether such checks can be outsourced to a third party without compromising data privacy, a scheme is **publicly** or **privately** auditable. Furthermore, in **batch** auditable schemes the availability of large amounts of stored data can be verified in a single protocol run at very low (amortized) costs for the servers and the verifiers.

Byzantine Fault Tolerance (Usability). In particular, when multiple users might have write-access to the same shared file at the same moment, inconsistencies in the stored data might occur. Byzantine fault tolerant (BFT) systems are able to solve this issue. While efficient solutions for replica-based storage solutions have been existing for quite some time, research for erasure-coding or secret sharing based systems has only started more recently.

For evaluating the efficiency of secret sharing schemes, the following three properties need to be considered.

Computational Efficiency (Usability). In particular, the computational costs per data block are a crucial evaluation criterion, when storing and retrieving large amounts of data.

Storage Efficiency (Usability). Again, in particular when storing large amounts of data, the required expansion factor per server may significantly determine the costs of the overall system. That is, this criterion considers the ratio between the size of the share stored on a single server and the size of the original data. For instance, this ration would be 1 in a fully replicated system.

Communication Efficiency (Usability). Besides the data that actually needs to be stored on each server, certain secret sharing schemes (or extensions thereof) require additional data to be transferred, e.g., for verifying the consistency of the distributed shares. As bandwidth is often a limiting factor on the end user's side, the overall required bandwidth should be as small as possible per share data block.

Finally, the security of secret sharing schemes is analysed based on the following criteria:

Computational vs. Information Theoretical Privacy (Security). Computational privacy means that an adversary having access to only a non-qualified subset of shares is not able to break the confidentiality of the stored data as long as some certain problem is computationally hard. On the other hand, information theoretical privacy means that even an attacker having access to a non-qualified subset of shares cannot infer any information about the stored data even if it has unlimited computational power. In particular, when storing long-term sensitive or highly classified information, information theoretical privacy is to be preferred.

Proactivity (Security). Secret shared data remains secret as long as an attacker is not able to compromise more than t of the storage nodes. As this might be a very strong assumption for long-term storage, proactive steps to re-share the data (in a way that guarantees that data leaked before the re-sharing are of no use to the adversary any more) are important. We therefore check whether proactive security steps are available for given secret sharing schemes.

Relation of t and n (Security). Secret sharing schemes have two main parameters: the overall number of storage nodes, n, and the threshold value of nodes required to reconstruct, t. In the literature, it is often recommended to set $t \approx \frac{n}{2}$. This is because an adversary might then corrupt have of the nodes while still not learning any information, while on the other hand also a denial of service attack on half of the nodes would not prevent the legitimate owner of the data to retrieve the stored files. We therefore also consider the imposed restrictions on the relation of n and t in our evaluation, as a higher "imbalance" might lead to less security or lower efficiency.

Evaluation

Because of the large amount of secret sharing schemes presented in the literature, we omit a full evaluation of the existing schemes here. This is also due to the fact that schemes are often not proposed including all their features, but various extensions (e.g., for proactivity or auditability) are often proposed independently in the literature. We therefore propose one potential set of building blocks that are compatible with each other and perform well in (most of) the above evaluation criteria.

- The secret sharing scheme proposed by Shamir [159] is **information theoretically private**. Furthermore, is is **perfect**, meaning that the shares that need to be sent and stored on each server have exactly the same size as the original data message; it can be shown that this cannot be improved for information theoretically private schemes. By itself, the scheme does not make any assumptions on the relation of n and t.
- An extension proposed by Krenn et al. [115] allows for **interactive verifiability** at negligible amortized (computational and communicational) costs for large data sets. The extension required $n \geq 3t + 1$.
- Demirel et al. [49] proposed an efficient extension for **public auditability**. For block-sizes larger than 64 bits, the scheme has practically constant communication efficiency and also high efficiency on the server sides. The scheme requires $n \geq 2t + 1$.
- The extension of Gupta and Gopinath [80] allows to proactively re-share secrets for Shamir's secret sharing scheme for $n \geq 2t+1$. However, similar to all other proactive steps found in the literature, the scheme comes at very high computational and communications costs between the servers.

4.3.3 Verifiable Computing

Verifiable Computing [65]

Type of Technology	Verifiable Computing
Status	Concept paper (2010)
TRL	M - There are multiple implementations with performance measurements.
Implementation	`http://www.pepper-project.org/#source` (BSD license)
Brief Description: In many scenarios, a weak client wants to outsource a computation to a more powerful worker. An example would be a mobile phone that accesses computing resources provided by a cloud service. However, the client has to be sure the requested function was performed correctly. With the concept of verifiable computing, the client is given the means to efficiently verify, whether the requested function was executed correctly, or not. In addition, in the approach presented in the referenced paper, the worker does not learn anything about the input or the function it is executing.	
Relevance to CREDENTIAL: Verifiable computing could be used to securely outsource computation tasks.	

Let us consider a party (called the client) which specifies some function f and has some input x (which will be outsourced to some third party called the server). At some point the client requests the server holding the data to evaluate the function f on the input x and the server returns y = f(x) to the client. Now the client wants to ensure that the server indeed computed y = f(x) correctly without having to re-perform the evaluation of the function on x locally. A client should only accept y in case of correct computation, but reject any incorrect y' with overwhelming probability. Verifiable computing (cf. [170, 43] for an overview) subsumes a number of different techniques which allow a client to be convinced of the correct computation of the server in the aforementioned scenario. Thereby, it is important that the client does not need access to the data locally and one requires that all means required by the client to check or ensure the correctness is more efficient than a local re-evaluation of the function f (at least in an amortized setting).

Evaluation Criteria

Public vs private verifiability (Usability). Either anyone can publicly verify the correctness of some computation performed by a server or the client requires some dedicated secret to check a computation for correctness.

Interactive vs. non-interactive (Usability). A client may either probe the server several times on the computation performed by the server (using several rounds of communication) until the probability that the server can convince a client of the correctness of an incorrect computation is reduced to some acceptable small probability. In the non-interactive setting, the server simply returns a certificate of correct computation (a proof) which can locally be checked by the client and the entire protocol thus only requires a single round of communication.

Expressiveness (Usability). Depending on the used approach the class of supported functions f can differ significantly. In the best case one a technique can deal with arbitrary arithmetic circuits (functions) or even arbitrary C code. Other techniques restrict the class of functions to polynomial functions of some fixed small degree (e.g., to linear or quadratic functions).

Communication (Usability). The number of rounds of communication between the client and the server and whether the communication cost depends on the concrete function f or not.

Server computation (Usability). The computational overhead required by the server to perform the entire protocol. Server overhead can usually assumed to be much higher than the cost of evaluating the function as the server is assumed to run in the cloud and can be efficiently scaled to the required needs.

Client computation (Usability). Considers the effort of the client compared to locally performing the computation. The efficiency is often considered in an amortized sense, i.e., the client has some setup costs which are performed once and then by verifying computations on different inputs the costs of this setup are amortized over time.

Authenticity (Security). One requires that a malicious server will not be able to convince a client of the correctness of a computation when the result is not correct. Formally, one distinguishes different types of an adversary. While a weak adversary is only allowed to submit one try, an adaptive adversary is given multiple tries where it learns for each try whether the client did accept or not.

Privacy (Privacy). Privacy comes in different flavors. Firstly, privacy can be related to the input of the computation from a server's point of view (so called input privacy). It requires that the server performing the computation does not learn the data on which the server is computing. Secondly, privacy can be related to the client (so called output privacy). It requires that client checking the correctness of the computation does not learn anything about the input to the computation.

Evaluation

There is a large body of literature on different techniques to realize verifiable computing and since verifiable computing does not constitute a core cryptographic technique within *CREDENTIAL*, we omit a full evaluation of all existing techniques. We just provide a brief overview of the different classes of techniques and we refer the reader to [170, 43, 32] for a more comprehensive overview:

- **Approaches based on interactive proofs or arguments:** These approaches typically require many rounds of interaction which limits the fields of practical application quite significantly (cf. [170]).
- **Approaches based on non-interactive proofs or arguments:** This approach relies on SNARKS and efficient ways of encoding the computations and is used by the main verifiable computing approaches available today like Gepetto or Pinoccio (cf. [170]).

- **Approaches based on fully-homomorphic encryption:** This approach can be generically constructed from garbled circuits and any FHE scheme [65]. Since FHE, however, still lacks practical efficiency, this approach cannot be considered efficient as of today.
- **Approaches based on homomorphic authenticators:** This approach is based on homomorphic MAC or signature schemes [43]. There are different constructions under various assumptions available that support different classes of functions (from linear polynomials to polynomials of arbitrary degree). For restricted classes of functions this approach is the most practical, although the client computation can only be made good in an amortized setting. Nevertheless, this approach is often viable when output privacy is of interest.

Evaluation Conclusion

In conclusion, verifiable computing could be used to securely outsource computations to a party that is not fully trusted, such as the *CREDENTIAL* cloud service in our case. Even though verifiable computing not a core technology for *CREDENTIAL*, it might be used to construct cryptographic primitives that are relevant for this project, and, therefore, verifiable computing is of interest for further research.

4.4 Section Conclusion

In this Section, we described and evaluated additional cryptographic technologies that are relevant to *CREDENTIAL*. For the evaluated technologies, we presented recommendations within the subsections. Full details for many specific technologies which are relevant for *CREDENTIAL* are given in Appendix B.

5 Authentication to the Cloud

One of the core elements of *CREDENTIAL* is to provide strong authentication to the cloud while high levels of privacy and usability are achieved. To this end, in this chapter we first introduce the basics of authentication, towards analyzing and evaluating relevant technological frameworks that enable strong authentication to the *CREDENTIAL* wallet. The selected frameworks allow the implementation of different authentication factors, which in turn provide different levels of usability and security; selected frameworks are introduced and evaluated in Section 5.2. Furthermore, despite the strength of the analyzed solutions, security at the client side, and particularly, when considering mobile devices could be drastically enhanced by taking advantage of secure HW-supported cloud authentication. Thus, in Section 5.3 we also analyze state of the art underlying authentication technologies that can support the secure execution of cryptographic algorithms while providing secure storage of the cryptographic material and associated credentials. For each cluster of technologies we first summarize in a fact sheet the main features offered by each technological framework. Secondly, we introduce a detailed evaluation criteria derived from the high level criteria introduced in Section 2. Afterwards, we perform the evaluation of each cluster of technologies, and finally, we conclude the chapter by providing a recommendation on technologies that could be implemented within the *CREDENTIAL* architecture.

5.1 Authentication Factors

The process allowing one entity to establish the identity of another entity is known as "Authentication" and it represents a major barrier for an attacker to circumvent a system, i.e. the more robust and sophisticated an authentication method is, the harder to be deceived. The implementation of strong authentication is an essential part of any security system [54] implying the combination of two or more elements that could be based on what you know (e.g., PIN, passwords, pass-phrases, etc.), what you have (e.g., keys, tokens, smart cards, etc.), and what you are - based on biometric methods that evaluate physiological attributes or behavioral singularities (e.g. fingerprints, iris, voice, etc).

In what follows we further describe the main authentication factors that could be implemented within the selected authentication frameworks which are later evaluated in Section 5.2.

- **Knowledge Factors:** Something the user knows, such as passwords, PINs, or secret questions.
- **Possession Factors:** Something the user possesses.
  - One-Time Passwords: One-Time Passwords (OTP) can be generated time-based or event-based. OTP are generated on an external token with a display, or on a smart phone. The particular generation process is bound to a user and paired with a server. Therefore, the possession of the device can be verified by comparing the device's OTP and the server's OTP. See OATH for further details.

  - SMS-TAN: A server creates an OTP, usually denoted as TAN, and sends it to the user's phone. The user then returns the TAN through another communication channel to prove its reception. The user's phone, or more precisely its SIM, cover the possession factor.
  - Challenge-Response: The remote entity creates a so-called challenge, which is transmitted to the user. The user then applies some kind of cryptographic method to the challenge and returns the outcome as response. Since the cryptographic operation usually involves a secret key that is bound to the user, the server is able to verify possession of that key material by examining the response. Hardware security tokens or software applications can be used to perform the involved cryptographic functions. Some of the technologies described below can be used to implement this approach: FIDO describes a possible implementation; Trusted Platform Modules and Trusted Execution Environments securely store and operate on key-material; PKCS #11 and ISO 24727 can be used to communicate with external tokens that provide cryptographic material and operations.

- **Inherence Factors:** Something the user is. This is usually described by biometric characteristics, which can either be physiological or behavioural. Physiological biometrics, consists of measurements taken from data obtained as part of the human body; leading techniques in this category are fingerprint, hand geometry, facial recognition, iris and retina recognition. Behavioral biometrics consist of measurements taken from the user's actions, many of them are indirectly measured from the human body. The leading behavioral techniques are voice recognition, handwritten signature, and keystroke dynamics.

5.2 Authentication Technologies

In this subsection we briefly introduced the selected authentication frameworks, namely, FIDO UAF, FIDO2F, Mobile Connect, OATH and SQRL. These frameworks aim to replace user names and passwords by implementing (passwordless) strong authentication mechanisms based in possession and inherent factors; these frameworks operate using a challenge-response authentication model. We evaluate the aforementioned technologies based on the high level evaluation framework defined in Section 2. The aim of this assessment is to contribute to a better understanding of which authentication framework is most suitable for providing secure access to the *CREDENTIAL* wallet considering its application scenarios.

FIDO UAF (Universal Authentication Framework) [57]	
Type of Technology	Authentication Framework
Status	Framework, Specification (2014), Version 1.0
TRL	H - FIDO UAF has been deployed in operational environments, for example by Google, and open source implementation are available.
Implementation	`https://github.com/eBay/UAF` (Apache 2 license), `https://fidoalliance.org/certification/fido-ready/`, `https://fidoalliance.org/certification/fido-certified/`
Brief Description: The FIDO UAF enables multi-factor security, where online services are able to specify which types of factors are sufficient.	

The Authentication Framework defines software layers for client- and server-side, as well as a protocol for their interaction. During registration of a service, the user generates a key pair on her device and sends the public key to the service. To authenticate a user, the server sends a challenge which has to be signed with the corresponding private key. Access to this private key is only granted after the user performed local authentication, such as fingerprint verification, with the required factors. This process requires client software and certified hardware, which would be an obstacle for user adoption. However, major corporations are members of the FIDO alliance, so hardware and software availability might not be a problem.
Relevance to CREDENTIAL: FIDO UAF could be combined with identity protocols that rely on out-of-band authentication. Since UAF client and server components are provided as separate software, they would just have to be integrated.

FIDO U2F (Universal Second Factor) [58]	
Type of Technology	Authentication Framework
Status	Framework, Specification (2015), Version 1.1
TRL	H - FIDO UAF has been deployed in operational environments, for example by Google, and open source implementations are available.
Implementation	`https://github.com/google/u2f-ref-code`, `https://fidoalliance.org/certification/fido-ready/`, `https://fidoalliance.org/certification/fido-certified/`
Brief Description: FIDO U2F defines strong authentication via a second factor that augments traditional authentication methods. U2F specifies a narrow set of functionality that is mainly concerned with the generation of key pairs and signatures on external devices. Interaction with those cryptographic operations is possible through a Javascript API in the browser. The authentication process is very similar to the one described in FIDO UAF.	
Relevance to CREDENTIAL: FIDO U2F could be combined with identity protocols that rely on out-of-band authentication.	

Mobile Connect [79]	
Type of Technology	Authentication Framework
Status	Version 2.0
TRL	H - Since the solution was introduced at Mobile World Congress 2014, 34 mobile network operators (MNOs) have launched the service in 21 countries, with plans for additional launches and trials to follow in 2016 and beyond.
Open Source Implementations	`https://developer.mobileconnect.io/content/sdks-and-test-applications`

Brief Description: Mobile Connect is a user authentication and identity service based on the OpenID Connect/OAuth2 standards. Mobile Connect is provided by individual mobile network operators and delivered through a standardised technical interface. Mobile Connect basic usage provides a one factor authentication based on the possession of a mobile number. It supports three different authenticators to do so: An SMS+URL authenticator is an SMS sent to the user's device with a unique one-time only URL that the user selects to prove that they are in possession of the device and have access to the SMS. USSD (Unstructured Supplementary Service Data) is a protocol used by mobile networks to communicate with a terminal (mobile device). The operator will push a message to the terminal and can require a response. The user's device will display a message such as "Press 1 for OK. Press 2 for Not OK". If a user has access to the mobile device and is able to respond (correctly) to it, they will be authenticated. SIM applets are applications that are stored within the SIM card and run on the mobile phone. When an authentication request occurs, a binary SMS will be sent to trigger the SIM applet. Once triggered, it will prompt the user for an input such as "Press OK to continue" or "Please enter your PIN". The SIM Applet will send a response back to the operator to validate the user. In all cases the user must needs to be connected to mobile network. The Mobile Connect framework also considers the usage of biometric factors, but these are dependent on mobile network operators' local authenticator implementations.
Relevance to CREDENTIAL: Mobile Connect could be considered as one of the alternatives to the authentication protocol as it provides a passwordless experience, with certain level of identity assurance.

OATH [97]	
Type of Technology	Authentication Reference Architecture
Status	Framework (2007), Version 2.0
TRL	H - Multiple implementations are used in operational environments, for example by the Google Authenticator and integrated into Forgerock's OpenAM.
Implementation	`https://github.com/google/google-authenticator`, `https://github.com/google/google-authenticator-android` (Apache License), `https://github.com/bdauvergne/python-oath` (BSD 3-clause), `https://github.com/archiecobbs/oathtoken`
Brief Description: OATH describes a reference architecture where open standards are used to adopt strong authentication. This reference architecture includes a client framework, which defines how client applications perform authentication methods by communicating with authentication tokens. Furthermore, it describes a validation framework, a risk evaluation framework, a provisioning and management framework, as well as a common data model. The innovative focus lies on research and integration of a one-time password algorithm that has event-based, time-based and challenge/response variants.	

Relevance to CREDENTIAL: The OATH Reference Architecture provides a broad overview and might serve as an introduction into relevant topics and technologies.

Secure Quick Reliable Login (SQRL) [69]	
Type of Technology	Authentication Specification
Status	Standard (Draft)
TRL	M - This concept has been realized in multiple Implementation for different platforms.
Implementation	`https://www.grc.com/sqrl/implementations.htm`, `https://github.com/geir54/android-sqrl` (GPL 3), `https://github.com/geir54/php-sqrl` (GPL 3)
Brief Description: A highly secure, comprehensive, easy-to-use replacement for usernames, passwords, reminders, one-time-code authenticators... and everything else. A high level vision of the protocol: "The website's login presents a QR code containing the URL of its authentication service, plus a nonce. The user's smartphone signs the login URL using a private key derived from its master secret and the URL's domain name. The Smartphone sends the matching public key to identify the user, and the signature to authenticate it."	
Relevance to CREDENTIAL: A very convenient implementation of secure authentication; Identifies crypto mechanisms for the convenient sharing of private keys between several devices. This specification helps to prevent service providers from tracking customers.	

In the following we briefly introduce the criteria used in our evaluation framework, afterwards we use these criteria to concretely evaluate the FIDO UAF, FIDO U2F, Mobile Connect, OATH and SQRL technologies. Finally, we discuss the most suitable authentication approaches for *CREDENTIAL* and its corresponding implications.

5.2.1 Evaluation Criteria

Privacy Criteria: Although many definitions of privacy exist, we consider the privacy criteria as the extent to which the user is given the control of his/her data; that is, that user is informed and gives his/her consent about what data is being shared, for which purposes and to which services.

Unlinkability (Privacy): Users being able to perform two or more transactions without those transactions being associated to the same user.

Minimal information disclosure (Privacy): As indicated in the eIDAS regulation, authentication for an online service should only consider processing those data that are relevant and not excessive to grant access to such as service.

User consent (Privacy): This criterion provides information whether a user is informed and consequently asked for her consent to share her data during the authentication process.

Security: It is strength of an authentication system in terms of covered risk and its efficiency to resist potential attacks, considering the risk they represent and its sophistication. In the authentication framework is relevant by means of providing the appropriate mechanisms to assure that authentication through fake or invalid credentials cannot be successful and at the same time providing secure communications protocols that will safeguard the information being transmitted from any malicious attacker. For the purpose of this evaluation we consider the following properties:

Integrity (Security): Ability of protecting exchanged authentication information from being altered by (malicious) unauthorized parties.

Confidentiality and data leakage (Security): The communication between the authenticating client and the authentication server should be done via the establishment of a secure communication channel. Furthermore, attacks to centralized databases storing sensitive data, such as, biometric data, result in both: high associated risks and great responsibility.

Unforgeability (Security): Capability of a solution to avoid impersonation, use of fake or revoke credentials, this includes the capability of revoking credentials.

Mutual authentication (Security): By means that the user authenticating can be assured that there is a legitimate entity behind the authentication server.

Trust assumptions and assurance levels (Security): This criterion will provide information of the trust assumptions needed in the authentication framework (trusted third parties) and insight of the strength of the authentication technologies suitable in each framework

Usability Criteria: Authentication system's quality of being user-friendly and closer to user needs and requirements, including acceptability, performance and ease of use. For instance, users might be reluctant to adopt novel solutions, if they are not able to authenticate them at the first try and in a reasonable time.

Performance efficiency: In terms of amount of resources needed by the authentication solution such as storage, CPU power, bandwidth, latency, etc.

Convenience for user (Usability): The quality of being able to meet the needs and expectations of a particular user segment for instance general public.

Integration Effort Criteria: Suitability to the *CREDENTIAL* architecture and its application scenarios in terms of how much effort is needed to integrate the authentication technology.

Implementation difficulty (Integration effort): In terms of technical requirements to be fully deployed and functional, and cost impact of the authentication system implementation effort.

Extensibility (Integration effort): This criterion describes the feasibility in extending the authentication framework which may be required by specific application domains. In this regard, the framework could potential extensions be integrated.

Interoperability (Integration effort): Convenience for the developer regarding existing infrastructure/services, etc. Costs and dependencies: economic impact of the technology in the overall authentication system (e.g. implementation costs, maintenance, etc.) and dependencies with regard to third party services or intermediaries fees.

Technology maturity and adoption levels (Integration effort): It provides an insight of the readiness level of the technology and its level of adoption and its practicability in terms of available libraries and licensing models.

5.2.2 Evaluation

FIDO Universal Authentication Framework (UAF) is an authentication framework that allows the implementation of secure authentication while providing enhanced privacy and high level of usability. In particular, this framework aims for a passwordless authentication, thus, enhancing security with multifactor authentication based on inherent and possession factors. FIDO UAF provides better end-user experience by preventing the burdensome of password management not only for users but for identity providers as well. Furthermore, FIDO provides the functionality of transaction confirmation; and decouples user verification from the authentication protocol.

Privacy: FIDO UAF aims at providing privacy in terms of *minimal information disclosure*, the FIDO UAF does not require a unique ID per device, instead it associates the authenticators to accounts in the relying party; and in this regard FIDO provides pseudonymity but *not unlinkability*. One of the mains strengths of FIDO UAF is that it avoids the collection and storage of biometric and other personal identifiable information. Furthermore, *user consent* is required in all actions.

Security: FIDO UAF provides end-to-end security as the framework relies on the use of asymmetric key cryptography, which enables secure communication providing integrity, confidentiality and unforgeability, as well as protecting the system from phishing and Man-in-the-middle (MITM) attacks. FIDO's client generates a key pair and the public key is sent to the authenticating server. Within the authentication process, the FIDO server sends a challenge which is signed by the client demonstrating the possession/ownership of the private key. Despite its security strengths, the FIDO authenticators may present different assurance levels. In particular, FIDO authenticators used for user verification rely on the existence of a trusted platform/module or secure element to store and protect key material (i.e. private keys). Furthermore, FIDO allows integration with different authentication technologies such as biometrics which are used to unlock the private key and which technically imply different security and maturity levels that can be indicated in a policy to describe the required or accepted authenticators. Taking the biometrics example, the biometric data is stored and verified locally and never transmitted the server, which minimizes the risks of potential massive biometric data leakages but on the other hand the security and protection of biometric data relies totally on the client side which is beyond FIDO's specification.

Integration effort: There exist mature and open source implementations of FIDO UAF which makes FIDO a suitable and relatively not expensive solution. FIDO supports integration with existing IAMs such as OpenSSO, however the implementation of FIDO servers and FIDO authenticators is needed. Its readiness for adoption has been proved by its widely deployment in solutions such as Google and Visa. The implementation effort, involves different dependencies w.r.t. OS versions, browsers and FIDO equipped devices (i.e. certified hardware). Finally, FIDO simplifies integration and extensibility, protocols and APIs support the inclusion of additional authentication methods, protocols and authenticators.

Usability: A high level of usability for end users and services provides is provided by means of passwordless authentication; user verifications are done through different kinds of authenticators installed which, are based on the device capabilities, for instance voice and fingerprint authentication.

FIDO Universal Two Factor Authentication (U2F) is a FIDO Alliance open protocol which provides strong authentication using asymmetric key in a challenge-response model. Unlike UAF, U2F does not work with different authenticator factors but it is limited to possession factors, concretely token-based authentication.

Privacy: User consent is required to log in to a particular website, but there exists no evidence that the user is informed about which data is being shared by the U2F such as app_id or registration of the device which in turns enables linkability. Open implementations such the one provided by google may possess additional privacy risks.

Security: U2F uses asymmetric key mechanisms to provide secure authentication. It implements a challenge-response model which relies on the use of a physical token to authenticate the user. The token generates the public/private key pairs and signs the challenge.

Usability: U2F provides usability in terms of being a passwordless solution, however, considering that users need to carry on an additional device (i.e. token), the levels of usability provided by tokens are undesirable for *CREDENTIAL*.

Integration effort: There exist mature implementations of the U2F; Nevertheless the client side need first to have a U2F token with a client side application that would act as a bridge between the token and the web API. Server side open implementations are available which can considerably reduce the effort; however, implementations provided by google may have additional dependencies such as the users having a google account.

Mobile Connect is an authentication framework based in OpenID Connect standards, which provides SIM-based authentication (i.e. the possession of a mobile number). It allows the implementation of secure authentication and high level of usability by means of passwordless authentication. Authenticators are either based on SMS, USSD or SIM toolkit. Additionally, local user verification through the use of biometrics is available by some MNOs.

Privacy: Mobile connect provides privacy by means of minimal information disclosure and user consent. The MNO confirms user credentials and user is asked to give consent for the sharing of his/her information. Mobile connect does not provide unlinkability, even though it claims to provide certain level of anonymous authentication is done through mobile phone authentication which is associated to user credentials and the intermediary (i.e. MNO) is able to learn about user's authentication to services provides.

Security: Mobile connect aims at reducing frauds by assuring that a real person is behind the account, which possess a device and a mobile number (SIM card). It offers three different levels of security assurance supporting three kind of authentication methods, namely SMS, USSD or SIM.

Integration effort: There exist mature and open source implementations of Mobile Connect which makes it a suitable solution, however costly due to the need of MNOs involvement as intermediaries. Its readiness for adoption has been proved by its widely deployment by 34 network operators in 21 countries. The implementation effort, involves dependencies on MNOs and users having an enabled SIM card. The high level of usability for end users and services provides is provided by means of passwordless authentication.

SQRL is an authentication framework which operates using a challenge-response model with asymmetric encryption. SQRL aims to replace usernames and passwords to minimize the disclosure of personal information. To achieve this, SQRL stores only a set of public keys which are used to verify the signature of an authenticator. It is assumed that if public keys are lost, this will only posses a minimum threat to the privacy and identity theft of users. Furthermore, SQRL aims to improve usability by implementing QR codes that need to be scanned for a user to be logged in.

Privacy: SQRL provides privacy in terms of minimal information disclosure, the identity required by the SQRL authentication server is derived from the master key stored in user's SQRL application. SQRL can provide pseudonymity but no unlinkability, user's transactions are associated to user's account which in turn is associated to the public key stored at the server side. User consent during authentication is requested.

Security: SQRL provides secure authentication with the use of asymmetric key cryptography. The client application derives site specific private/public key pairs from a master key and the domain, which decreases the probability of a MITM attack when modifying the website domain; nevertheless the real time MITM attack cannot be prevented. SQRL implements a challenge-response mechanism which generates a long random nonce encoded into a QR code which prevents replay attacks. The SQRL application reads the challenge and generates a signed response that can be verified with the public key stored at the authentication side. Similarly to FIDO, the master key needs to be protected. SQRL client uses SCrypt (a password-based key derivation function [144]) to encrypt the master key with a password.

Usability: SQRL does not works in offline mode but the implementation of QR-codes provide users with high usability levels. Users only need to manage a single password for the master key which is stored locally in the SQRL client device. On the downside, a single user cannot have more than one account in the same website using the same master key, as the key pairs are derived from the master key and the website domain.

Integration effort: Very few implementations of SQRL exist which do not cover the major platforms, for is potential adoption the SQRL client application needs to be available for the major mobile platforms.

OATH is an open authentication framework, which operates using a challenge-response model, and supports the HOTP, OCRA and TOTP algorithms. OATH supports hardware-based token as well as the so called soft tokens. Credentials can therefore be embedded into SIM cards and mobile phones TPM. OATH provides authentication of users, devices and transactions.

Privacy: OATH provides little privacy in the sense that high levels of linkability are present by storing users profile information as well as the storage of unique identifiers, where no minimal information disclosure is guaranteed. Authentication information is stored in databases which could also be vulnerable to data breaches. Additionally, the three validation models (i.e. distributed, centralized and wallet) allow the validation side to learn about users' activities.

Security: OATH's specification allows strong authentication through the integration of different authentication methods based on symmetric key challenge/response, PKI, OTP, biometrics, and cookies; authentication tokens (hardware and software based) which could be embedded into hardware tokens, SIM cards, mobile devices TPM; and a set of authentication protocols to provide secure communication such as SSL/TLS, HTTP Auth, Kerberos and WS-Security. High levels of assurance could be achieved by monitoring transactions and user behavior which may result in additional privacy concerns regarding users IP addresses and behavioural patterns.

Usability: The convenience and user experience is enhanced by providing a wide range of authenticators the go from hardware based such as hardware tokens to the user or soft tokens embedded in mobile devices. It provides online and offline authentication and high level of interoperability, practically suitable for all devices.

Integration effort: OATH claims to address secure authentication challenges with standard and open technology which enables solutions in all kind of devices. The framework is designed with flexible modules aimed to be interoperable with current standards and identity management systems.

5.2.3 Evaluation Conclusion

FIDO UAF is the most complete approach as it provides high security, usability and acceptable levels of privacy; high assurance levels could be achieved by implementing different types of authenticators. FIDO UAF allows smooth integration with existing IAM systems such as OpenSSO which is considered in *CREDENTIAL*'s core architecture for Identity and Access Management. FIDO UAF provides extensibility in terms of authentication methods, protocols and authenticator types. The technology has proven high levels of maturity and wide adoption, with open source implementations available. Contrary to FIDO UAF, FIDO U2F provides limited possibilities for authenticators, which could be translated as usability and costs issues. Similarly to FIDO UAF, OATH provides extensive flexibility, usability and strong security, however, little to no privacy. Mobile connect, although promising, is limited to authenticators that require the involvement of MNO's as intermediaries, which turns out to be more complex in terms of reachability, costs, and privacy. Similarly to FIDO UAF, SQRL seems a very interesting approach. However, very few implementations are available and its maturity and adoption are still in its infancy. Furthermore, the fact the users cannot have multiple accounts in a single domain increases the privacy risks and limits the potential of *CREDENTIAL* when considering multiple accounts to access the wallet.

5.3 Underlying Technologies for Authentication

Most authentication mechanisms rely on cryptography to convince a communication partner of the user's identity. In order to execute cryptographic algorithms securely and to provide a protected storage location for keys on the other hand, a hardware based root of trust is necessary. Therefore, this section investigates crypto processors, called Trusted Platform Modules (TPM). As TPMs only provide a limited set of operations and more flexibility might be required, another technology allowing to execute arbitrary algorithms, called Trusted Execution environment (TEE), is analyzed.

5.3.1 Trusted Platform Module

Trusted Platform Module (TPM) [77]	
Type of Technology	Cryptographic Chip
Status	Specification Version 2.0, Standard (ISO/IEC 11889:2015)
TRL	H - TPMs have been integrated into many modern devices, such as phones and computers.
Brief Description: A TPM is a separate hardware module in the user's device that stores keys and provides cryptographic operations. The operating system executes these operations, but it cannot extract the involved private keys. An example for the use of a TPM is Android's Keystore and Cryptography API. Strong authentication could be implemented by performing local authentication, for example with a fingerprint, which unlocks access to key material. The server can then verify that the local authentication succeeded, if the user is able to create a signature with a protected private key.	
Relevance to CREDENTIAL: A Trusted Platform Module would facilitate the implementation of authentication methods.	

Evaluation

A Trusted Platform Module is a crypto processor created by the Trusted Computing Group. The core functionality of a TPM is to provide a tamper-resistant, hardware-backed root-of-trust. Furthermore, TPMs implement specific cryptographic primitives and algorithms next to providing a secure storage for keys. The two main specifications for the TPM are version 1.2 and version 2.0. A TPM complying to version 1.2 only supports SHA-1 and RSA. Although version 2.0 supports more cryptographic algorithms (including ECC, AES), the TPM is not capable of creating the paring-keys used in proxy re-encryption. Therefore, TPMs cannot directly be used for the key material of the algorithms used in *CREDENTIAL*. It may be possible to create a key material within the TPM and use it to encrypt the private key used for proxy re-encryption. Every time the key is used, it gets deciphered in the TPM and the plain key would reside in memory during cryptographic operations.

TPM version 2.0 introduced different forms of the TPM, that is, among others, the discrete and the firmware TPM as described in Appendix C.2.1. Firmware TPMs are not based on an own crypto processor but on other technologies enabling hardware-based security. This mechanism might be flexible enough to introduce our own cryptographic schemes, but as it is based on Trusted Execution Environments, we refer to the subsequent Section 5.3.2.

Evaluation Conclusion

The evaluation of this technology showed that it is not possible to solely use TPMs in the *CREDENTIAL* project for key management (lack of required functionality). TPMs are not capable to perform the required operations needed by *CREDENTIAL*. Anyhow, the TPM could be used to encrypt the private key for proxy re-encryption. The proxy re-encryption key could be written in encrypted form to the disk and when the key is used it gets decrypted by the TPM. In this way, it would also be possible to export the private key of a user to any other device. The downside of this approach is that the (plain) private key would reside in RAM during creation, pairing, exportation and other operations. If used, this approach should be further evaluated (in terms of security constraints) during a later stage of the project.

5.3.2 Trusted Execution Environment

Trusted Execution Environment (TEE) [146]	
Type of Technology	Cryptographic Chip
Status	Specification (2011), Version 1.0
TRL	H - TEEs have been realized by multiple vendors, such as ARM with its TrustZone and Intel with the Trusted Execution Technology.
Brief Description: The TEE is a secure and isolated execution environment, which protects the integrity and confidentiality of data and code being processed inside it. The device's operating system can only communicate with the TEE through well-defined channels. In comparison to a TPM, a TEE provides more flexible functionality, since arbitrary algorithms can be executed. A TEE can be used to implement strong authentication as described for the TPM use case. However, the local authentication, for example the extraction and comparison of fingerprint data, could also be implemented in this secure environment.	
Relevance to CREDENTIAL: A Trusted Execution Environment would further secure local authentication methods.	

Evaluation

A TEE is an execution environment running parallel to a rich operating system. Applications executed within the trusted environment (called Trusted Applications) have a higher security level than applications residing in the rich environment. Specified by the GlobalPlatform, TEEs were originally created for the increasing need of security in the mobile market. Key creation and storage within the TEE is possible although it does not provide the same security level as for example a Secure Element. The TEE itself is a processor feature enabled in most Android devices. To be more precise, the ARM TrustZone, one implementation of the TEE, is used in numerous Android devices. For instance, devices running with the popular Qualcomm SnapDragon support ARM's TrustZone. Not all Android devices actually have implemented a TEE or any other mean for hardware based security at all. Some older and/or cheaper devices may only have software solutions. Nevertheless, a TEE provides great flexibility. A Trusted Application could be used for the creation of proxy re-encryption keys, the pairing process and would also support exportability. Currently, the creation of Trusted Applications for the Android TEE represents a challenge for third-party developers.

Evaluation Conclusion

Within this section the Trusted Execution Environment was evaluated in respect to the requirements for *CREDENTIAL*. The evaluation led to the conclusion that a TEE would meet the security and usability requirements of *CREDENTIAL*. However, writing third-party Trusted Applications to implement our own cryptographic schemes on TEEs requires considerable effort.

5.4 Section Conclusion

This section presented the evaluation of those authentication technologies relevant to *CREDENTIAL*'s wallet, as well as the underlying authentication technologies which could support the execution of cryptographic operations and the secure storage of the cryptographic material. We conclude that FIDO UAF is the most suitable approach for providing authentication to the *CREDENTIAL*'s wallet, as it meets all requirements of *CREDENTIAL*, mainly because of the features provided, but also because of its suitability of integration. Furthermore, as introduced before, FIDO's relies on the security of the device for the local management and storage of the cryptographic material; therefore, as a second step, underlying authentication technologies were evaluated, concluding that TPM and TEE would meet the *CREDENTIAL* and FIDO's requirements for authentication. Nevertheless, further investigation on both technologies should be provided in order to assess the main improvements and customizations needed by the *CREDENTIAL* architecture. Finally, the detailed description of the discussed technologies could be found in Appendix C.

6 Identity and Access Management Protocols

Identity and access management (IAM) is a concept that combines processes, policies and technologies for managing if, how, and when entities access resources. This area of technology includes a variety of protocols that manage identification, authentication, and authorization of users towards service- and identity providers. *CREDENTIAL* leverages IAM technologies because it needs to authenticate users and manage how users and service providers access shared resources. By building on established standards and technologies, *CREDENTIAL* avoids *reinventing the wheel* and eases integration into current IAM systems.

In this section we present a variety of IAM technologies and evaluate their usefulness with regard to *CREDENTIAL*. We grouped these technologies into the following categories:

- **Identity protocols** specify the communication of identity information between entities.
- **Authorization Protocols** define how a service provider gains and proves authorization.
- **Policies** express access rules on services, functions, and resources.
- **Cryptographic APIs** enable to remotely perform operations on cryptographic material.
- **Other Technologies** do not fall in any of the mentioned categories.

Each category follows this structure: We summarize the technologies of the category in **fact sheets**. We present a list of evaluation **criteria** and explain why those criteria are important for *CREDENTIAL*. We **evaluate** the technologies and conclude by explaining if and why we **recommend** a technology for *CREDENTIAL*.

6.1 Identity Protocols

Identity protocols specify the communication of identity information between entities. They initiate an authentication process and specify how to handle the authorization process. The actual authentication process is usually done out of band. An identity protocol may also include further functionality, such as the discovery of the user's identity provider or how to handle the user's attributes. *CREDENTIAL* relies on these protocols for exchanging attributes between users, IdPs and SPs.

This section is structured as follows: Section 6.1.1 presents the fact sheets of technologies that can serve as identity protocols. Section 6.1.2 presents the evaluation framework, which describes all evaluation criteria and their relevance for *CREDENTIAL*. Section 6.1.3 evaluates the technologies and Section 6.1.4 discusses the results.

6.1.1 Fact Sheets

This section introduces technologies that can serve as identity protocols. A detailed description of SAML and OpenID Connect can be found in Appendix D.1.

Security Assertion Markup Language (SAML) [83]	
Type of Technology	Identity Protocol
Status	Standard (2005), Version 2.0
TRL	H - This protocol has been implemented in multiple commercial products and open source implementation are available.
Implementation	`saml.xml.org/wiki/saml-open-source-implementations`
IPR (License Model)	`http://www.oasis-open.org/committees/security/ipr.php`
Brief Description: SAML is a XML-based protocol that can be used to exchange authentication, authorization and attribute data. In this protocol, an Identity Provider authenticates the user and in case of success issues an assertion. Such assertions are used by Service Providers as basis for authorization decisions. Since the actual authentication is out of band, a number of methods may be applied. SAML provides confidentiality, integrity and authenticity through XML Encryption and XML Signature. It also supports single sign on.	
Relevance to CREDENTIAL: SAML could be used as underlying identity protocol, which would still provide us with the freedom to implement strong authentication methods. SAML is also a base protocol used in eIDAS [54].	

OpenID [140]	
Type of Technology	Identity Protocol
Status	Specification (2007), Version 2.0; deprecated
TRL	H - This protocol has been implemented in multiple commercial products and opens source implementations are available.
Implementation	`http://openid.net/developers/libraries/obsolete/`
IPR (License Model)	`http://openid.net/intellectual-property/`
Brief Description: OpenID is an identity protocol, which can be used to authenticate users to service providers in the web environment. Since the actual authentication method is not predefined, it is possible to integrate strong authentication. OpenID also defines a process to discovery the user's identity provider given her identity-string. Furthermore, there is an attribute exchange extension, which defines how a service provider could access data about one of its users.	
Relevance to CREDENTIAL: OpenID could be used as identity protocol. However, since it is deprecated, its successor OpenID Connect should be considered instead.	

OpenID Connect [154]	
Type of Technology	Identity Protocol
Status	Specification (2014), Version 1.0
TRL	H - This protocol has been implemented in multiple commercial products and open source implementations.
Implementation	`http://openid.net/developers/libraries/`
IPR (License Model)	`http://openid.net/intellectual-property/`

Brief Description: OpenID Connect is an identity layer on top of OAuth 2.0. It allows services to verify the user's identity, to obtain the user's attributes and to gain delegated access to the user's additional server resources. The confidentiality, integrity and authenticity of identity data and attributes is ensured by exchanging them as JSON Web Tokens (JWT). Those tokens support standard encryption and signature mechanisms through the JSON Web Encryption (JWE) and JSON Web Signature (JWS) specifications. The OpenID Connect protocol suite also includes extensions that specify: the automatic discovery of a user's OpenID Provider, the dynamic registration of new service providers, and session management.
Relevance to CREDENTIAL: OpenID Connect could be used as Identity Protocol in *CREDENTIAL*. Its core specification and extensions tackle many issues *CREDENTIAL* faces.

WS-Federation [73]	
Type of Technology	SOAP extension
Status	Standard (2009), Version 1.2
TRL	H - This standard has been implemented in multiple commercial products and open source implementations are available.
Implementation	`http://www.zxid.org/` (Apache2 license)
IPR (License Model)	`https://www.oasis-open.org/policies-guidelines/ipr`
Brief Description: WS-Federation, which extends the functionality of WS-Trust, allows the federation of different security realms, for example different companies. In order to do so, WS-Federation provides mechanisms to broker identity, attribute, authentication and authorization assertions, as well as, trust relationships between realms.	
Relevance to CREDENTIAL: WS-Federation could be used to federate different security realms.	

Central Authentication Service (CAS) [121]	
Type of Technology	Identity Protocol
Status	Specification (2015), Version 3.0.2
TRL	H - Multiple implementations are used in operational environments and open source implementations are available.
Implementation	`https://github.com/apereo/cas`
IPR (License Model)	`https://www.apereo.org/licensing`
Brief Description: CAS defines an identity protocol that provides single sign-on in the web environment. The CAS identity provider issues security tickets, which have to be validated by contacting the identity provider again. The process flow of CAS is similar to OpenID Connect and SAML. However, OpenID Connect for example provides additional features. These features include discovery, dynamic client registration, and claims in an externally verifiable token.	

Relevance to CREDENTIAL: CAS could be used as identity protocol. However, from an initial analysis, it provides less features than similar protocols, and those protocols should therefore be preferred.

BrowserID / Mozilla Persona [124]	
Type of Technology	Identity Protocol
Status	Specification (2011); deprecated
TRL	H - An operational system is offered by Mozilla and open source implementations are available.
Implementation	`https://github.com/mozilla/persona` (MPL 2.0)
IPR (License Model)	`https://developer.mozilla.org/en-US/docs/MDN/About#Copyrights_and_licenses`
Brief Description: The BrowserID protocol was defined for Mozilla's Persona service. After successful authentication, the user generates a key pair for which an Identity Authority issues a certificate. The key pair and certificate are stored in the browser and used to sign assertions about the user's identity and attributes. These assertions can be verified by a service with the certificate.	
Relevance to CREDENTIAL: BrowserID could be used as identity protocol.	

WebID [156]	
Type of Technology	Identity Protocol
Status	Specification (Draft 23, 2017)
TRL	H - Multiple implementations are used in operational environments and open source implementations are available.
Implementation	`https://www.w3.org/wiki/WebID_Protocol_Implementations`
IPR (License Model)	`https://www.w3.org/Consortium/Legal/2002/ipr-notice-20021231`
Brief Description: In the WebID Protocol, the user generates a key pair and issues a self-signed certificate, which is included in requests as TLS client certificate. In addition, the user publishes her public key and identity data on a HTTPS-secured web server. A service that receives the user's request, can authenticate her by comparing the certificate's public key with the public key served by the secure web server.	
Relevance to CREDENTIAL: The WebID Protocol could be used as authentication protocol. Since authentication within this protocol is based on the possession of key material, strong authentication would have to be implemented at the user's device to grant access to this key material.	

6.1.2 Evaluation Criteria

This section presents the evaluation criteria for identity protocols. Firstly, we list **security** criteria:

Security (Security): Security considerations have to be made as sensitive identity and authentication data are handled by identity protocols. The transfer of data between IdP and SP is a major concern. This can be achieved on the transport level, for example by using SSL/TLS, or on the content level, with encryption, digital signatures, etc. However, there might be further implementation-specific measures, which should be described in the protocol specification.

Authentication of service providers (Security): This criterion specifies how service providers are authenticated against the identity provider. Such an authentication is a major mechanism to prevent phishing attacks, where a user exposes data to an adversary.

Furthermore, we also consider **usability** criteria:

Single sign-on (SSO) (Usability): SSO simplifies the sign-on process, as users only have to sign-in once. The identity provider remembers that a user authenticated by binding the authentication data to her sessions. As a result, unnecessary interaction steps can be skipped when a service provider requires authentication for an already authenticated user.

Single logout (SLO) (Usability): SLO allows to log the user out of the identity provider as well as multiple services at once. This simplifies the log-out process, as the users do not have to interact with each service provider at which they are signed in.

User consent (Usability): This criterion evaluates whether the user is asked for consent to share an explicit set of her data with a service provider. From a privacy point of view, informing a user about the implications of a pending data sharing operation and gaining her consent is a very desirable property, especially when designing a user-centric system.

SP-initiated/IdP-initiated (Usability): This criterion evaluates if the authentication process can be initiated by the service provider, the identity provider, or both.

Identity provider (IdP) discovery (Usability): This criterion evaluates if there is a way to automate the discovery of a user's preferred identity provider. Such a discovery mechanism would also facilitate the dynamic association between identity and service provider.

Token size (Usability): This criterion compares the size of the exchanged authentication and attribute data. Authentication data allows to verify the user's authenticity, while attribute data describes characteristics about the user. Depending on the protocol, these two types of data might not be returned in the same response. In order to compare the sizes, we measure the response containing the authentication data and possibly attributes, as well as additional responses necessary to collect a total of five attributes. The token size mainly depends on the data format and as well as the use of signatures or encryption.

Finally, the required **integration effort** is another important aspect:

Extensibility (Integration Effort): This criterion describes how easy it is to extend the protocol. An extension might be necessary to realize application-specific requirements. Therefore, the protocol should define extension points that specify how enhancements may be integrated.

Format of the identifier (Integration Effort): This criterion evaluates whether the user's identifier is fully specified by the protocol or if custom identifiers can be used.

Format or names of additional user attributes (Integration Effort): Apart from the user's identifier other user-specific attributes might be transferred. This criterion evaluates whether the format or the names of these additional attributes are completely specified, which facilitates interoperability. However, some use cases might require the integration of custom attributes and formats.

Level of adoption (Integration Effort): This criterion evaluates how widely the protocol is deployed, which might be an indicator for the protocol's practicability.

Open source libraries (Integration Effort): This criterion evaluates if implementations are freely available. A healthy mix of multiple implementations in diverse programming languages facilitates the adoption of a protocol, as technical requirements regarding the operational environment can be satisfied.

Interoperability (Integration Effort): This criterion evaluates the interoperability between actors in the protocol using different implementations. If the specification leaves of room for interpretation, this might result in diverging implementation that have interoperability issues. Also, if the protocol defines optional functionality or requires multiple endpoints, the configuration of an actor should be accessible within the protocol.

Metadata (Integration Effort): This criterion evaluates whether the protocol specifies metadata, which describes the configuration of the service and identity provider. Such metadata therefore allows for an automated association between service and identity provider and facilitates interoperability.

Registration of service providers (Integration Effort): This criterion evaluates how service providers register at the identity provider in the individual protocols. A fully specified process could facilitate the automatic association of service and identity provider and thereby simplify their integration.

Data exchange format (Integration Effort): This criterion describes the format in which identity and authentication data can be exchanged. The format has an influence on the size of the transmitted data. Also, some formats may be easier to handle on certain platforms. Additionally, the format also specifies which other data standards can be embedded. For example, JWT can be integrated with JSON. In general, data is mainly exchanged through HTTP parameters, JSON, and XML.

Transfer protocol (Integration Effort): This criterion describes over which protocol data is transferred between the identity and service provider.

Bindings (Integration Effort): This criterion evaluates via which methods data can be transferred between the identity and service provider. In contrast to the transfer protocols, the bindings might further specify how such a protocol is used to exchange data. Generally, there are two types of communication channels. Firstly, the front channel hands data between identity and service provider through the user agent. For example, the parameters in an HTTP redirect can be used to transmit data between identity and service provider. Also, the same can be achieved with automatically submitting forms. Additionally, data can be passed indirectly, by providing a one-time URL for subsequent communication. Secondly, in the back channel, identity and service provider communicate directly.

6.1.3 Evaluation

In this section, the described identity protocols are compared and evaluated according to the selected criteria (security, usability, and implementation effort criteria). Note that some properties or features can still be fulfilled or improved if not further specified recommendations of the specification are implemented or if the specifications are extended and amended according to specific needs. Even though OAuth is an authorization protocol (see Section 6.2.1), it is commonly also used as identity protocol. Therefore, we also include OAuth in this evaluation. The following technologies are not part of the evaluation because they are not considered relevant for *CREDENTIAL*:

Mozilla Persona: Mozilla stopped developing and deploying Persona, therefore, it is deprecated. On November 30th, 2016, Mozilla will shut down the persona services[2].

WebID: Within *CREDENTIAL*, we want to use identity protocols which are accepted in the community and often utilized to ensure interoperability and extensibility. Since WebID is not as popular and often used as other protocols, it is therefore not relevant for *CREDENTIAL*.

Based on the **security criteria**, the individual protocols are evaluated and discussed.

Security: *SAML* 2.0 makes it possible to use XML signatures [18] as well as XML encryption [95] to protect the exchanged messages and assertions. Furthermore, the use of security protocols such as SSL/TLS or IPSec on the transport layer is recommended. A detailed security analysis is given by [86].

OpenID does not require the use of TLS/SSL but recommends it. Optionally, the exchanged messages can be signed with a shared secret that was established through a Diffie-Hellman key exchange. Security considerations are listed by [140].

OAuth employs SSL/TLS to protect its communication and to authenticate the server. However, no further signatures or encryption of the payload is supported by the core specification. Proposed standards define how these additional security features can be

[2]`https://developer.mozilla.org/en-US/Persona`

achieved by transmitting SAML assertions [40] or JWT [107]. The security of OAuth is discussed in detail by [120].

OpenID Connect protects its communication with SSL/TLS. Furthermore, JWT can be used to encrypt and sign identity data. The core specification discusses security aspects in detail.

In *WS-Federation*, the transmitted data is protected by SSL/TLS. Furthermore, XML signatures and encryption can be applied to the individual messages. A rigorous security proof of the WS-Federation protocol was presented by [76].

The *CAS* specification recommends but does not require the use of SSL/TLS. Signatures or encryption on the token or identity data are not specified.

Authentication of service providers: *SAML*, *OAuth*, *OpenID Connect*, and *WS-Federation* fully specify how identity providers authenticate service providers. In *SAML* and *WS-Federation*, a service provider signs its requests for user authentication, which are sent to the identity provider. This identity provider then verifies the signature using the certificate, which was included in the service provider's metadata. The *OpenID Connect* core specification defines multiple methods to authenticate clients.

For *OAuth* the authentication of service providers is partially specified. One approach is to authenticate the client based on a previously issued id and secret. However, the *OAuth* specification says to conduct any authentication method that fulfills the requirements of the authorization server.

In *OpenID* and *CAS*, the authentication of service providers is not specified. The authentication of services is not foreseen in the *OpenID* specification, as service providers dynamically associate with the user's preferred identity provider. *CAS* recommends but does not define the authentication of service providers.

The following evaluates and discusses the six identity protocols based on the identified **usability** criteria.

Single sign-on (SSO): All six identity protocols support single sign-on. While *OAuth* does not explicitly define SSO functionality, single sign-on can be achieved by using the same authorization server for multiple clients.

Single logout (SLO): Single logout is supported by *SAML*, *OpenID Connect*, *WS-Federation*, and *CAS*. *SAML* supports single logout through the Single Logout Profile [93] While single logout is not part of the core specification of *OpenID Connect*, the draft for session management [48] introduces this functionality. For *WS-Federation* and *CAS*, single logout at the identity provider as well as at active service providers is defined in the core specifications.

In contrast, neither *OpenID* nor *OAuth* support single logout.

User consent: In both the *SAML* and *WS-Federation* protocols, service providers register their required attributes in a metadata document that is exchanged during the association with an identity provider. Based on this information, the user is able to make an informed authorization decision at the identity provider.

For *OpenID*, *OAuth*, and *OpenID Connect*, the service provider specifies the required attributes in each authentication request.

In the *CAS* protocol, the service provider does not specify which user attributes are required. Therefore, identity provider and user have no information regarding the necessary consent.

SP-initiated/IdP-initiated: In all six protocols, the authentication process can be initiated by the service provider. Additionally, *SAML* and *OpenID Connect* allow the identity provider to initiate this process. For example, in *OpenID Connect*, such an IdP-initiated authentication process is realized by sending the user to a specific URL at the service provider.

Identity provider discovery: Identity provider discovery is supported by *SAML*, *OpenID*, and *OpenID Connect*. The discovery of the user's preferred identity provider is a core feature of *OpenID*. For *SAML*, this process is specified in the Identity Provider Discovery Profile [93]. Also, an extension [155] makes this possible for *OpenID Connect*.

In contrast, *OAuth*, *WS-Federation*, and *CAS* do not provide support for the discovery of identity providers. The general *OAuth* framework leaves discovery entirely unspecified. Although a very early draft [109] defines this functionality, we cannot yet consider the discover process as being supported by the *OAuth* ecosystem. The *WS-Federation* specification informally lists approaches to discover the user's identity provider but does not specify a process.

Token size: Due to the complex *SAML* specification and the use of XML, the messages and assertions are rather large. A signed assertion containing five attributes has approximately 5.5 kByte.

Similarly, *WS-Federation* encodes the token in XML, which is accordingly rather large. If a signed SAML assertion is returned as a token, the size is also approximately 5.5 kBytes.

In *OpenID*, the transmitted messages and assertions are small compared to complex, XML-based approaches. A signed response containing five attributes has approximately 1 kByte.

In *OAuth*, access tokens are typically just a short string and can be used to verify the user's authenticity. Further attributes have to be loaded through an unspecified API endpoint. This endpoint may offer data in any format, which also has an impact on the overall size. To compare the data size to other implementations we make the reasonable assumption that an endpoint provides the same five attributes as a signed JWT. As a result, the combined token size is approximately 700 bytes.

OpenID Connect identity providers issue id tokens as JWT. Those tokens are smaller than XML representations, but require more space than simple encodings of the identity data due to the added security mechanisms. A typical response containing an access token as well as an id token with five attributes has about 600 bytes.

In *CAS*, authentication data are represented by tickets, which can be validated at the identity provider. This validation response might also contain additional attribute data encoded as XML or JSON. Therefore, the token size is calculated using a ticket and the validation response. A typical size for ticket and response containing five attributes encoded as JSON is approximately 600 bytes.

In the following, the individual identity protocols are evaluated and discussed according to the **integration effort** criteria.

Extensibility: *SAML*'s architecture, the protocol's messages, as well as the assertion format were designed for extensibility.

OpenID is easily extensible, as the core specification defines a mechanism for extending the protocol. This mechanism makes it possible to add further parameters in an extension-specific namespace to the exchanged messages.

OAuth serves as a framework, where some components are only partially defined. The specification explicitly states that those components should be described by extensions, and therefore offers extension mechanisms.

OpenID Connect, like OAuth, was designed with the goal of being easily extensible.

The *WS-Federation* specification was designed to be extensible. The XML schema makes the extension at message-level possible.

In the design of *CAS*, the extensibility considerations are mainly limited to the possibility of defining custom attributes. Compared to other standards, CAS does not provide as many explicit extension points for the protocol itself.

Format of the identifier: *SAML*, *OpenID*, and *OpenID Connect*, partially define the formats of identifiers. *SAML* specifies multiple formats for identifiers but also allows custom definitions. *OpenID* specifies two formats for the user's identifier, namely XRI and URL. These identifiers serve as a basis for discovering the user's preferred identity provider. In *OpenID Connect*, the discovery extension [155] defines formats for the user's identifiers, as these identifiers have to be converted to a hostname used in the IdP discovery process.

In contrast, *OAuth*, *WS-Federation*, and *CAS* do not specify the format of identifiers.

Format or names of additional user attributes: An extension for *OpenID* [139] and the core specification of *OpenID Connect* define a standard set of attributes as well as their format. Custom attributes can be used in these protocols as well.

For *SAML*, *OAuth*, *WS-Federation*, and *CAS*, the format and names of additional attributes are not defined.

Level of adoption: *SAML*, *OAuth*, and *OpenID Connect* are widely deployed. SAML is especially adopted in eBusiness, eHealth, and eGovernment [179]. [1] provides a list of services where OAuth is deployed. Services employing OpenID Connect include Google, Amazon, and Microsoft.

OpenID and *CAS* only have a medium level of adoption. While OpenID was deployed on more than 9 million websites in 2009 according to [30], its use has significantly decreased in favor of its successor OpenID Connect. CAS is mainly only deployed in the higher education sector.

WS-Federation only has a low level of adoption, as this standard was not embraced by the identity management community.

Open source libraries: For *SAML* [2, 96], *OpenID* [138], *OAuth* [1], and *OpenID Connect* [60], multiple open source implementations are available in different programming languages.

In contrast, only a limited number of implementations is available for *WS-Federation* and *CAS*. For example, *WS-Federation* has been implemented by Microsoft, Oracle, or Ping Identity.

Interoperability: *SAML*, *OpenID*, *OAuth*, and *OpenID Connect* allow for interoperability between service and identity provider of the individual protocols. The Kantara Initiative [96] highlighted the interoperability of *SAML* by testing an extensive list of implementations. There are no official conformance tests for *OpenID*, however, [141] presents a matrix of performed interoperability tests. For *OpenID Connect*, extensions for IdP discovery [155] and dynamic registration of clients [153] particularly facilitate interoperability between different actors. The OpenID foundation also certifies the interoperability of various implementations [61]. *WS-Federation* specifies minimum requirements in order to facilitate interoperability. Also, based on [74], interoperability tests were performed.

In contrast *OAuth* and *CAS* do not provides such a high support for interoperability. Since *OAuth* is just a framework, multiple required components are only partially defined, which severely limits interoperability. Multiple extensions [151, 109] try to close this gap, but they were not yet finalized at the time of writing. *CAS* is a comparatively simple protocol which limits the extent of interoperability issues. However, since CAS does not support metadata, the configuration and capabilities of the actors is not documented, which impedes interoperability. To the best of our knowledge, no interoperability tests have been performed.

Metadata: *SAML*, *OpenID Connect*, and *WS-Federation* support metadata. In *SAML* and *WS-Federation*, the configuration of service and identity provider can be specified through XML metadata documents. These documents describe the providers' configuration, required or offered attributes, as well as key material. For *OpenID Connect*, the configurations of both the identity and the service provider can be described through the extensions for IdP discovery [155] and dynamic registration of clients [153], respectively.

In contrast, *OpenID*, *OAuth*, and *CAS* do not use metadata. *OpenID* does not use metadata, as service providers dynamically associate with the user's preferred identity provider, which was discovered through the user's identifier. The *OAuth* specification does not define metatdata. However, a proposed standard for client registration [151] as well as a very early draft for authorization server discovery [109] describe how to exchange configuration data. As these documents are not yet finalized, we cannot consider metadata as being supported by the OAuth ecosystem. *CAS* does not support the description or exchange of metadata.

Registration of service providers: *SAML*, *OpenID Connect*, and *WS-Federation* fully specify the registration process of service providers at the identity provider. In *SAML*, metadata is used to register a service provider at an identity provider. The identity provider verifies the structure as well as the signature of the provided metadata. Similarly, *WS-Federation* exchanges metadata documents for this registration process. For *OpenID Connect*, an extension makes it possible to dynamically register service providers at the identity provider [153].

The *OAuth* specification describes that certain types of clients should be registered but does not define how. A extension [151] was proposed, which specifies this process.

OpenID and *CAS* do not specify the registration of service providers. *OpenID* service providers do not have to register at the identity provider. Instead, an association is established dynamically when the user enters an identity pointing to the identity provider. In the *CAS* specification, the registration of service providers is recommended but not specified.

Data exchange format: All six protocols use HTTP GET or POST parameters to transfer data. While *OpenID* exclusively uses those parameters, they are only used to a negligible extend in *SAML* and *WS-Federation* to transport XML-based messages. In comparison, *OAuth* and *OpenID Connect* employ both HTTP parameters and JSON messages. Finally, in *CAS*, data can be exchanged optionally as XML or JSON through request parameters.

Transfer protocol: All six protocols exchange data via HTTP. In addition, *SAML* and *WS-Federation* also support SOAP.

Bindings: In all six protocols, the parameters of HTTP redirects can be used to exchange data between service and identity provider via the user agent. For *OpenID* and *CAS*, such redirects are the only data exchange mechanism. Besides HTTP redirects, *SAML*, *OAuth*, and *OpenID Connect* [106] also support automatic form submissions for the data transfer. The artifact binding, which allows to fetch the data from a sent one-time URL, is supported by *SAML* and *WS-Federation*.

6.1.4 Evaluation Conclusion

The results of the evaluation are presented by Table 7.

SAML and OpenID Connect performed best in our protocol evaluation. The former, SAML was designed for the secure exchange of authentication and authorization data given by a subject. The latter, OpenID Connect builds on OAuth and is in particular an identity layer on top of OAuth 2.0. It was developed to simplify the exchange of authentication data and information about the end-users. We recommend these two protocols because both have in common, the widespread usage especially in eBusiness, eGovernment and eHealth with good ecosystems and implementations. Moreover, both support the usage of metadata, the option to register service providers and offer high interoperability. SAML and OpenID Connect support the SP and IdP initiated authentication, which gives additional flexibility. OpenID Connect is one of the recent published protocols utilizing JSON based messages, whereas SAML uses XML-based messages.

	SAML 2.0	**OpenID 2.0**	**OAuth 2.0**	**OpenID Connect 1.0**	**WS-Federation 1.2**	**CAS 3.0**
			Security Criteria			
Security	H	M	M	H	H	L
Authentication of service providers	H	L	M	H	H	L
			Usability Criteria			
Single sign-on (SSO)	✓	✓	✓	✓	✓	✓
Single logout (SLO)	✓			✓	✓	✓
User consent	H	H	H	H	H	M
Identity provider (IdP) discovery	✓	✓		✓		
SP-initiated / IdP-initiated	SP, IdP	SP	SP	SP, IdP	SP	SP
Token size	$>$ 5.5 KB	$>$ 1 KB	$>$ 700 Bytes	$>$ 600 Bytes	$>$ 5.5 KB	$>$ 600 Bytes
			Integration Effort Criteria			
Extensibility	H	H	H	H	H	M
Format of the identifier	M	M	L	M	L	L
Format or names of additional user attributes	L	M	L	M	L	L
Level of adoption	H	M	H	H	L	M
Open source libraries	H	H	H	H	M	M
Interoperability	✓	✓		✓	✓	
Metadata	✓			✓	✓	
Registration of service providers	H	L	M	H	H	L
Data exchange format	Params, XML	Params	Params, JSON	Params, JSON	Params, XML	Params, JSON, XML
Transfer protocol	HTTP, SOAP	HTTP	HTTP	HTTP	HTTP, SOAP	HTTP
Bindings	Redirect, Form, Artifact	Redirect	Redirect, Form	Redirect, Form	Redirect, Artifact	Redirect

Table 7: Comparison of Identity Protocols

The bigger message size of XML-based message leads to a transmission overhead. SAML is has powerful features but can be complex when implementing, to the contrary, OpenID Connect offers an easy way of implementing functionality.

OAuth is widely used for authentication and inter-operates very nicely with OpenID Connect. It was not strictly developed for identity management. Instead, it was developed with a strong focus on authorization rather than authentication. OpenID is an earlier published protocol which is deprecated because there are already better successors available. The WS-Federation protocol can be used to easily extend an already existing WS-* system. The existing WS-* system is necessary otherwise it is not possible to extend the system. WS-Federation is not as widespread used as other protocols, because of the lower acceptance in the IdM community. The main advantage of CAS is the simple specification and the easy usage. Besides this fact, CAS is the weakest protocol in our evaluation, especially because security features are only optional.

6.2 Authorization Protocols

Authorization protocols manage access control on resources between entities. They define how a service provider would gain and prove authorization. *CREDENTIAL* can leverage authorization protocols for granting and revoking access on resources like shared documents.

This section is structured as follows: Section 6.2.1 presents the fact sheets of three authorization protocols. Section 6.2.2 presents the evaluation criteria and their relevance for *CREDENTIAL*. Section 6.2.3 evaluates the protocols and Section 6.2.4 discusses the results.

6.2.1 Fact Sheets

This section introduces the protocols OAuth, UMA, and WS-Trust. The detailed description of these protocols can be found in Appendix D.2.

OAuth [84]	
Type of Technology	Authorization Protocol
Status	Standard (2012), Version 2.0
TRL	H - This protocol has been implemented in multiple commercial products and open source implementations are available
Implementation	`http://oauth.net/code/`
IPR (License Model)	`https://www.ietf.org/ipr/`
Brief Description: OAuth is an authorization protocol, where the user is able to delegate access rights to her server resources to third parties. This delegation does not require users to share their credentials, but relies on the issuance of access tokens. OAuth can also be used as an identity protocol under the assumption that a user can only grant access to an API, if the user first successfully completed an authentication process. As a consequence, the user's authenticity can be verified by checking the validity of an access token.	
Relevance to CREDENTIAL: Only a limited set of required functionality is defined in OAuth, but it leaves the possibility to integrate single sign on, strong authentication and advanced cryptography. Therefore, OAuth could be used as underlying identity protocol.	

User-Managed Access (UMA) [114]	
Type of Technology	Authorization Protocol
Status	Specification (2015), Version 1.0.1
TRL	H - Multiple implementations are used in operational environments and open source implementations are available.
Implementation	`https://kantarainitiative.org/confluence/display/uma/UMA+Implementations`
IPR (License Model)	`https://kantarainitiative.org/confluence/pages/viewpage.action?pageId=41025689`
Brief Description: User-Managed Access (UMA), which is based on OAuth 2.0, allows users to manage access to individual resources residing on any number of resource servers through a single authorization server. UMA improves upon OAuth by introducing a protection API, which specifies interactions between resource servers and authorization servers. This allows to protect the APIs of multiple resource servers through a single authorization server. Furthermore, UMA formalizes a requesting party that may not be the resource owner. Thereby, this requesting party may initiate access requests, which have to satisfy a user-defined policy to be successful. The concept of such a distinct requesting party also makes it possible to include legal aspects into granting access to APIs.	
Relevance to CREDENTIAL: In credential, UMA could be used as a powerful standardized authorization framework. It allows to externalize APIs, such as attribute providers, while still managing authorization policies at a central point. Furthermore, the concept of a requesting party makes it easier to realize use cases, where multiple parties want to access the user's data, such as the eHealth pilot, where the patient as well as the physician require access to the patient's data.	

WS-Trust [117]	
Type of Technology	Authorization Protocol
Status	Standard (2012), Version 1.4
TRL	H - This protocol has been implemented in multiple commercial products and open source implementations are available.
Implementation	`https://cxf.apache.org/` (Apache2 license)
IPR (License Model)	`https://www.oasis-open.org/policies-guidelines/ipr`
Brief Description: WS-Trust is an extension of WS-Security that specifies methods for issuing, renewing, and validating security tokens. These tokens represent the basis for authorization decisions at the services.	
Relevance to CREDENTIAL: WS-Trust could be used in web service environments as an authorization protocol.	

Kerberos [126]	
Type of Technology	Authorization Protocol
Status	Standard (2005), Version 5

TRL	H - Multiple implementations are used in operational environments and and open source implementations are available.
Implementation	`http://web.mit.edu/kerberos/`, `https://www.h5l.org/`, `http://www.gnu.org/software/shishi/`
IPR (License Model)	`https://www.ietf.org/ipr/`
Brief Description: The Kerberos Protocol is based on symmetric key cryptography and therefore does not require public key infrastructure. An Authentication Server (AS) requires proof of possession of a shared symmetric key to authenticate user. The AS then issues a ticket, which confirms a successful authentication. With this ticket a Ticket Granting Server performs an authorization decision and, in case of success, issues another ticket, which attests permission to access a service.	
Relevance to CREDENTIAL: The Kerberos Protocol provides basic functionality for the authentication and authorization process. Since authentication within this protocol is based on the possession of key material, strong authentication would have to be implemented at the user's device to grant access to this key material.	

6.2.2 Evaluation Criteria

This section describes and motivates the evaluation criteria for the presented authorization protocols. The individual protocols have common features which allows us to compare them with each other. The evaluation and the resulting protocol recommendation is based on the differences between those protocols.

Extensibility (Integration Effort): This criterion describes the potential of an authorization protocol to be extended by further functionality. It is important that the chosen protocol can be extended so that we are able to introduce customizations which make it suitable for other use cases. This can significantly influence the authorization protocol decision.

Interoperability (Integration Effort): The possibility of being interoperable with other systems is described with this criterion. Interoperability requires a common understanding of the protocol's aspects by all communication partners. For instance, fully and strictly specified interfaces can be an indicator of high interoperability because other systems can easier communicate with this interfaces. Also, when the specification is more flexible and allows for optional fields or extensions, a configuration documenting these customizations can facilitate the system's interoperability. This criterion is important because *CREDENTIAL*'s interfaces should be used across different applications.

Adoption (Integration Effort): With the criterion adoption, the acceptance within the community and the adoption in well known systems is described. This criterion indicates if the protocol is practical in use and easy to integrate. A high adoption implies that more developers are improving and extending the implementations.

Flexibility (Integration Effort): This criterion describes how flexible the protocol can be adopted for different use cases. For example, if a protocol specification defines a system's architecture in high detail the possible use cases can be limited. Instead, when the specification leaves free space for customizations, then the protocol can be seen as flexible and more use cases are covered out of the box.

Separation of Resource and Authorization Server (Usability): This criterion describes if the protocol assumes that resource and authorization server are in the same organizational context and use out of band communication. This communication between the entities has to be considered, as a common understanding of the to be protected resources and authorization decisions is required. By separating the resource and authorization server through well defined interfaces and protocols, they can be deployed on different providers or trust domains.

External Requester (Integration Effort): This criterion indicates whether the protocol considers the requester to be an external entity or the owner of the resource. In a data sharing use case, such as in *CREDENTIAL*, data might be shared by the resource owner with another party. Therefore, a protocol inherently supporting external requesters would simplify the process.

Process Flow (Integration Effort): This criterion describes on a high level how the authorization process is envisioned in the individual protocols. This process flow also has an impact on the knowledge which is required to successfully authorize a user. There are two approaches: Firstly, the client application has to invoke a token service before accessing the resource server. This received token can then be used to access a resource. In this flow, the client has to have the knowledge that it needs a token and how this token can be acquired. Secondly, the client may try to access the resource server. If unauthorized, the resource server supplies the information where and how authorization can be obtained.

6.2.3 Evaluation

This section evaluates the authorization protocols OAuth2, UMA, and WS-Trust according to the listed criteria. In contrast to the other authorization protocols, Kerberos does not focus the web-based environment. Therefore, Kerberos is not relevant for the *CREDENTIAL* project, as its general use case differs significantly from *CREDENTIAL*'s requirements with respect to an authorization protocol.

Extensibility: *OAuth2* was designed high extensibility in mind, which is shown for instance in the amount of other protocols build upon OAuth2, such as OpenID Connect and UMA. OAuth2 is a powerful basis which leaves space for customization to use it in different use cases.

UMA is built upon OAuth2. Basically, it is an extension of OAuth2. This protocol's extensibility is medium because the definition is more specific than in OAuth2.

The extensibility of *WS-Trust* is medium because the specification does not let as much space for customizations as OAuth2. Since WS-Trust is an extension of WS-Security it can only be used within WS-* systems. Nevertheless, it defines some extension mechanisms which make it extensible.

Interoperability: *UMA and WS-Trust* have a similar high level of interoperability. Both have well defined specifications so that client applications can easily communicate with the interfaces when following the specification.

In contrast, *OAuth2* is a framework rather than a strict specification, which can be utilized in many different fields. Therefore, some aspects are left undefined which represents an obstacle for integrating client applications. Basically, the specification leaves plenty of room for interpretations, therefore, clients can struggle when trying to communicate with the interfaces. In our opinion has OAuth2 a low interoperability.

Adoption: The adoption of *OAuth2* is high because the protocol is used for many different web services and as a basic framework for other protocols such as UMA and OpenID Connect. For example, big players such as Google [3] or Dropbox [4] are using OAuth2 as protocol to provide access to their APIs.

The adoption level of *UMA* is low. This is because the protocol is still a newer standard and not as well known as the others. Even though UMA is built on OAuth2, the applications are limited in comparison to OAuth2, since UMA focuses on a more specific use case.

WS-Trust has a medium level of adoption. It is more used within enterprises rather than for online web services.

Flexibility: *OAuth2* has a high level of flexibility, as it can be utilized for many different use cases. The space for own customizations gives the protocol a high flexibility. For instance, OAuth2 was developed as authorization protocol but it is often used as identity protocol as well.

The *UMA* protocol specification is specific about the architecture and its process flow, limiting its uses. However, it still tries to be as generic as possible. Therefore, in comparison, this standard is rated as medium flexible.

WS-Trust has a high level of flexibility in our comparison. The WS-* specifications can be applied to a variety of use cases.

Separation of Resource and Authorization Server: The *OAuth2* specification assumes that both resource and authorization server are operated in the same organizational context sharing knowledge of the issued authorization tokens. Therefore, the communication between these two entities is internally realized.

Whereas, in *UMA* and *WS-Trust*, these both entities are clearly separated. The required communication between them as well as the processing of tokens is defined in the specifications.

External Requester: *OAuth2* assumes that the entity requesting access is the resource owner. Basically, a requester wants to access her resources when using a web service. The flexibility of OAuth2 allows to also use the protocol in the way where the requester is not the

[3]`https://developers.google.com/identity/protocols/OAuth2`
[4]`https://www.dropbox.com/developers/reference/oauth-guide`

resource owner. However, this customization requires additional conceptual and implementation effort.

In contrast, *UMA* and *WS-Trust* are designed that external requester can request access to a service.

Process Flow: In *OAuth2*, the process flow consists of two main steps. Firstly, the authorization token has to be acquired at the authorization server. Secondly, the token is being used at the API to get access.

UMA and *WS-Trust* differ from *OAuth2*. They consist of three basic steps. Firstly, a user tries to access the resource server's API, which provides information about where and how authorization can be obtained. Next, the requester collects a token as specified. Finally, the requester utilizes the token to get access at the resource server's API.

6.2.4 Evaluation Conclusion

	OAuth2	**UMA**	**WS-Trust**
Extensibility	H	M	M
Interoperability	L	H	H
Adoption	H	L	M
Flexibility	H	M	H
Separation of Resource and Authorization Server	✗	✓	✓
External Requester	✗*	✓	✓
Process Flow	Auth'z → API	API → Auth'z	API → Auth'z

* denotes that this property can be achieved with extra implementation effort, but not out of the box.

Table 8: Authorization Protocols Comparison

The evaluation results are summarized and discussed in this subsection. Table 8 summarizes the results in a compact and structured way.

We recommend to use the UMA extension of OAuth to implement the data sharing aspects of *CREDENTIAL*. As the UMA protocol is based on OAuth2, it can be seen as improved specification specialized on authorizing access to whole documents. The well specified protocol has high interoperability and medium flexibility. Furthermore, it is positive that this protocol formalizes an external requester, which reflects *CREDENTIAL*'s use case.

We also recommend to use OAuth within *CREDENTIAL* for access control to other services than the data sharing mechanism. In our evaluation, OAuth2 performed well because the specification is flexible and leaves space for extensions. Hence, it can be used to cover more use cases than UMA and WS-Trust. Nevertheless, this flexibility comes together with additional implementation effort, because the protocol needs to be customized to the specific needs. The high adoption was also positive for this protocol.

We do not recommend to use WS-Trust within *CREDENTIAL*. One disadvantage is that WS-Trust hast to be used within a WS-* system, which takes away the freedom to combine other technologies. Nevertheless, the protocol has good and powerful approaches but other protocols are a better fit for *CREDENTIAL*'s needs.

6.3 Policies

This section describes the evaluation of the policy technologies *XACML*, *WS-Policy* and *WS-SecurityPolicy*. These technologies are used to express access rules on services, functions and resources. They are designed in a way to enable machine to machine communication and allow the systems to automatically consider security constraints on their interactions. While all of these technologies have the goal to express certain rules in a formal notation as a policy, they have different application domains. *XACML* is specifically used as an attribute based access control policy language. Thus, it addresses protection of resources and services against unauthorized access. *WS-Policy* is the core policy framework for *WS-**. It introduces basic expressions for policy constraints on web services. *WS-SecurityPolicy* is an extension for *WS-Policy*. It profiles specific security constraints for web services and thus enables authenticated and confidential access to these services.

The evaluation will be performed in a way to assess the technologies individually and not competitively because they try to solve different things. This section is structured as follows: Section 6.3.1 presents the policy fact sheets. Section 6.3.2 defines and motivates the evaluation criteria. Section 6.3.3 performs the evaluation and Section 6.3.4 discusses the results.

6.3.1 Fact Sheets

WS-Policy [177]	
Type of Technology	Policy Definition
Status	Specification (Recommendation, 2007), Version 1.5
TRL	H - This standard has been implemented in multiple commercial products and open source implementations are available.
IPR (License Model)	`https://www.oasis-open.org/policies-guidelines/ipr`
Brief Description: WS-Policy is an XML specification that allows to define policies for entities of a web service environment. Those entities include web services, which want to advertise their policies, as well as, web service consumers that specify their policy requirements. When multiple communication partners define policies, those policies can be used to negotiate a mode of operation that fulfills the requirements of all parties. As WS-Policy can be used to describe general purpose policies, some policies have an impact on messages exchanged, for example the used transport protocol, while others have no wire manifestations, such as privacy policies.	

Relevance to CREDENTIAL: WS-Policy could be used to specify policies that govern the access to credential APIs, which include required permissions for requesters as well as security methods.

WS-SecurityPolicy [116]	
Type of Technology	Policy Definition
Status	Standard (2007), Version 1.2
TRL	H - This standard has been implemented in multiple commercial products and open source implementations are available.
IPR (License Model)	`https://www.oasis-open.org/policies-guidelines/ipr`
Brief Description: WS-SecurityPolicy extends WS-Policy by refining the expressiveness of policies concerning security aspects. Such a security policy includes for example requirements regarding signatures and encryption of specific message elements, transport level security, timestamps, or supported security tokens. Therefore, with security policies web service and consumer are able to express their requirements, which can be further used to negotiate mechanisms between these parties in order to satisfy the security aspects.	
Relevance to CREDENTIAL: WS-SecurityPolicy could be used to specify policies that define security methods.	

eXtensible Access Control Markup Language (XACML) [134]	
Type of Technology	Policy Definition
Status	Standard (2013), Version 3.0
TRL	H - Multiple implementations are used in operational environments and open source implementations are available.
Implementation	`https://github.com/att/XACML` (MIT license), `https://github.com/wso2/balana` (Apache 2 license)
IPR (License Model)	`https://www.oasis-open.org/policies-guidelines/ipr`
Brief Description: XACML is an xml-based standard for attribute based access control. Authorization policies that are built into client applications are difficult to update. The XACML model therefore encourages the separation of access decisions and the point where those decisions are used. This decoupling leads to three main components. Firstly, a policy information point (PIP) provides attribute data. Secondly, a policy decision point (PDP) performs an authorization decision based on those attributes. Thirdly, a policy enforcement point (PEP) acts based on a decision, that is, it allows or denies access to some resource. As a result, policies can be changed on the fly and are immediately effective.	
Relevance to CREDENTIAL: Credential could make use of XACML's sophisticated access control system to delegate authorization decisions to one central policy decision point.	

6.3.2 Evaluation Criteria

The criteria used in the policy technology evaluation are described and motivated in this subsection.

Extensibility (Integration Effort): This criterion describes the extensibility of each policy technology. This may be necessary for *CREDENTIAL* applicability of a chosen technology since new cryptographic algorithms are implemented and have to be reflected within the policies.

Interoperability (Integration Effort): Policy description languages usually are not working on their own. They are enabler for other technologies and mostly used as a configuration for these tools. Therefore, this interoperability criterion describes how well a policy technology can operate within an ecosystem of different tools and technologies that rely on this specific policy framework. It evaluates how easily the policy technology can be integrated in other technologies.

Adoption (Integration Effort): This criterion describes how much the policy technology is accepted, used and relevant in various tools and products.

Open Source Libraries (Integration Effort): This criterion considers how much open source libraries for each technology are available, how much of the functionality of the policy technology is implemented, how well accepted the open source libraries are and if they target the Java platform.

Technology Independence (Integration Effort): This criterion describes the level of independence the technology has against other technologies. If a policy framework is designed in a way to work only with specific technologies, the policy framework is considered to have a low technology independence. If the policy framework can be used in a wide variety of technologies and does not restrict the technologies to be used with, it is considered to have a high technology independence.

6.3.3 Evaluation

This subsection details the evaluation results according to the above defined evaluation criteria for the policy technologies.

Extensibility: The *XACML* specification defines extension points for some of their XML attributes. Custom functions can be added to the XACML policies through the *FunctionId* attribute. Matching of attributes with certain values can be extended with the *MatchId* attribute. Deriving a policy decision while having multiple policies involved are resolved by using a *RuleCombiningAlgId* or *PolicyCombiningAlgId*. Both elements can be customized and new combining algorithms that may be necessary for *CREDENTIAL* can be included.

The *WS-Policy* specification defines extension points within the policy expressions and enable custom element or attribute information attached to the policies. The specification defines a very generic handling for policy processors, how they should treat the extensions. However, it is up to extension developer to provide a policy processor implementation together with the extension.

The *WS-SecurityPolicy* specification defines guidelines for writing new security policy similar to the predefined policies in the specification.

Interoperability: *XACML* is developed in a way that it can be integrated in many process flows whenever an access control decision has to be made. It defines an own process flow which can be executed independently from the surrounding system. The integration point of the *XACML* model is the policy enforcement point where a given request is intercepted and the policy decision is performed. The single access point and the lightweight integration (for example through an interceptor pattern) gives the *XACML* specification a high interoperability evaluation.

WS-Policy is developed as the policy framework for web services. It configures the underlying web service implementation according to the given policies. It is independent from any web service implementation and thus enables communication between client und service implementations with different technologies.

WS-SecurityPolicy uses the *WS-Policy* framework and provides a set of security policies defined within *WS-Policy*. Therefore, they share the same interoperability level.

Adoption: There exists multiple *XACML* implementations ranging from open source software to commercial products.

WS-Policy and *WS-SecurityPolicy* are part of multiple web service framework implementations and are widely used in any product that features SOAP and WSDL.

Open Source Libraries: A couple of open source libraries are available for *XACML*. Most of them support the latest standard version 3.0 from *XACML*. The vast majority of these products are implemented in Java.

Technology Independence: *XACML* has no requirements or implications of the surrounding framework or software where it will be integrated. Thus, it can be used in any application where access control has to be enforced. On the contrary, *WS-Policy* and *WS-SecurityPolicy* are developed for web services which express their interfaces through WSDL.

6.3.4 Evaluation Conclusion

The evaluation results are summarized and discussed in this subsection. Table 9 summarizes the results in a compact and structured way.

Every of the assessed policy technology has a high evaluation towards the interoperability evaluation criteria. Except for the technology independency where the *WS-Policy* and *WS-SecurityPolicy* technologies received a low evaluation. This is a result of the fact, that these technologies are designed for web services using WSDL and SOAP to describe their interfaces. Furthermore, they build the basis to configure protocols for the *WS-** stack. Choosing *WS-Policy* and *WS-SecurityPolicy* as the policy language has therefore a strong impact on remaining technologies to use. Therefore, we do not recommend to use *WS-Policy* or *WS-SecurityPolicy*. *XACML* received a high evaluation in each of the categories. We recommend it as the access control standard for the access control management system.

	XACML	**WS-Policy**	**WS-SecurityPolicy**
Extensibility	H	H	H
Interoperability	H	H	H
Adoption	H	H	H
Open Source Libraries	H	H	H
Technology Independence	H	L	L

Table 9: Policy Standard Comparison

6.4 Cryptographic Protocols and APIs

Cryptographic protocols and cryptographic APIs specify how to perform cryptographic operations on key material residing in another trust domain. They enable standardized and vendor agnostic interfaces by consolidating the requirements of different scenarios. *CREDENTIAL* can leverage these specifications by mapping its needs onto established standards. In turn, we expect to improve *CREDENTIAL*'s interoperability and ease its market introduction. This section is structured as follows: Section 6.4.1 introduces three cryptographic standards. Section 6.4.2 describes the evaluation criteria and their relevance for *CREDENTIAL*. Section 6.4.3 evaluates the standards and Section 6.4.4 discusses the results.

6.4.1 Fact Sheets

This section introduces the W3C Web Crypto API and KMIP. The detailed description of these standards can be found in Appendix D.4.

Web Cryptography API [171]	
Type of Technology	Cryptography API
Status	Specification (W3C Recommendation, 2017)
TRL	H - This standardized API has been implemented in modern browsers.
IPR (License Model)	`https://www.w3.org/Consortium/Legaln/2002/ipr-notice-20021231`

Brief Description: The Web Cryptography API specification defines a JavaScript API for browsers, which provides access to cryptographic functions and key material. This API separates the web page's sandbox from the actual cryptography, which provides two benefits. Firstly, as the cryptographic functions are natively implemented in the browser, they are more efficient than JavaScript implementations. Secondly, the browser manages the keys and is therefore able to enforce access restrictions on them. For example, access control decides for which cryptographic functions a key may be used, and whether raw key material may be extracted.
Relevance to CREDENTIAL: The W3C Crypto API could be used to perform cryptographic operations in the browser.

Key Management Interoperability Protocol (KMIP) [137]	
Type of Technology	Cryptography Interoperability
Status	Standard (2016), Version 1.3
TRL	H - The KMIP standard has been implemented by many different vendors.
Implementation	`https://wiki.oasis-open.org/kmip/KnownKMIPImplementations`
IPR (License Model)	`https://www.oasis-open.org/policies-guidelines/ipr`
Brief Description: KMIP is a protocol for interoperable communication between key management server and cryptographic clients. This protocol enables the key management on the server which consists of operations like create key pair, check, revoke, validate, etc. Furthermore, the protocol supports also cryptographic operations like encrypt, decrypt and more. Basically, the protocol supports base objects and managed objects. Base objects are uses within the protocol messages and can be parts of managed objects. Managed objects are for example cryptographic keys, digital certificates and templates that reside on the server and can be referenced with identifiers. Managed objects also have attributes associated, which can be added, modified or deleted.	
Relevance to CREDENTIAL: This protocol describes a standard key management protocol which is highly relevant for the *CREDENTIAL* project. In *CREDENTIAL*, the re-encryption keys have to be generated in a user-controlled environment. KMIP, could be used to enable interoperable communication between the *CREDENTIAL* cloud service and the user's environment. The high level of TRL together with the offered functionality show the relevance related to *CREDENTIAL*.	

6.4.2 Evaluation Criteria

In the evaluation we focus on different aspects of integration effort and security. Therefore, we introduce the following evaluation criteria and motivate their relevance:

Extensibility (Integration Effort): This criterion describes how difficult it is to extend the specification and how the specification deals with extensions. An easily extendible specification eases integration because it facilitates adding custom functionality.

Interoperability (Integration Effort): This criterion analyses aspects of the specification that influence interoperability of *CREDENTIAL*. Theses aspects include how detailed an interface is described, if there is room for interpretation and how flexibility is dealt with. Interoperability eases integration of *CREDENTIAL* with existing solutions, which can save resources in the development process.

Adoption (Integration Effort): This criterion reviews the acceptance of the specification across the community. It examines both the prevalence and usage, which gives feedback on the practicability and maturity of the specification. These factors ease the integration process because they decrease implementation and maintenance costs.

Component Reuse (Integration Effort): This criterion describes how many parts of the specification can be used without extending the specification. Using many components without adaptions decreases the need to extend the specification and thus simplifies its integration.

Access Restriction (Security): This criterion describes if the specification is able to enforce access restrictions on cryptographic material and services and if the specification allows us to model trust boundaries between acting parties. Access restriction policies on cryptographic material are necessary f.i. to keep the users private key safe.

Semantic Closeness (Integration Effort): This criterion describes how well *CREDENTIAL*'s problem domain maps onto the problem domain of the specification. A close relation to the original domain facilitates integration because it guarantees that present and future design decisions of the specification address the needs of *CREDENTIAL*.

Freedom of Implementation (Integration Effort): This criterion describes if deployment of the specification leaves a certain degree of freedom or if it adds unnecessary constraints. It examines different aspects such as the runtime environment, the transport layer and the feature set. A greater latitude of implementation eases integration because it increases the greatest common denominator with other components.

6.4.3 Evaluation

This section evaluates two cryptographic interoperability specifications, namely *KMIP* and *W3C Web Cryptography API*. We consider these specifications for two reasons: Firstly, we need to manage key material such as the users private key and re-encryption keys. Secondly, we need to exchange keys across different trust domains and regulate the exchange with access policies.

Extensibility: *KMIP* is easily extensible. The specification reserves values for objects, operations and attributes for extensions. Furthermore, KMIP offers profiles [136], which consist of a subset of operations and objects. These profiles can be tailored to fit a special scenario, such as the one of *CREDENTIAL*.

Web Crypto is less extensible. Although the API design is flexible and anticipates changes in cryptography, W3C explicitly discourages vendors to implement proprietary extensions. Vendor neutral extensions are only available via updating the W3C's specification or adding an extension specification. The road from filing an extension to the extension being accepted by the W3C and implemented by browser vendors is long. Alternatively, one could add extended functionality with a polyfill implementation.

Interoperability: *KMIP* focuses, as the name says, on interoperability. This becomes apparent in the level of detail put into specifying operations, error handling and message format. Moreover, KMIP offers an operation named `query` [137, Section 4.25 Query], which allows a KMIP client to *interrogate a KMIP server to determine its capabilities and protocol mechanisms.*

WebCrypto is less interoperable: The specification is cross-browser compliant and discourages vendor specific extensions, which reduces fragmentation. However, the specification does not require conforming agents to offer any algorithm and it's up to developers to check if an algorithm is supported and recommended.

Adoption: *KMIP* SDKs are available for several programming languages and the standard has a long list of supporting vendors[5], including but not limited to HP, Dell, IBM. At the time of writing we could not find any studies on the market share of KMIP.

WebCrypto hits an adoption rate of 80% worldwide and 90% in Europe[6]. These figures indicate an acceptable support across browser vendors on both desktop and mobile platforms. Netflix leads the development of the WebCrypto standard and uses it in its media streaming service.

Component Reuse: *KMIP* offers many components that can be used right away. In this respect *CREDENTIAL* can benefit from KMIP.

CREDENTIAL needs its own types and schemes and *WebCrypto* does not offer such components. Even a polyfill implementation for these types and schemes cannot leverage WebCrypto because the proxy reencryption schemes and the WebCrypto schemes are not related enough.

Access Restriction: A *KMIP* Server can restrict access to cryptographic material. In KMIP the trust boundary can be simply modelled between devices. Furthermore, KMIP offers to implement custom and fine-grained access policies.

WebCrypto also focuses on providing cryptographic services without exposing cryptographic material to the client. However, access restrictions are imprecise: Its only possible to specify *key usage* and *extractability.* Furthermore, those policies are globally defined, so different trust domains always share the same access rights.

[5]`https://wiki.oasis-open.org/kmip/KnownKMIPImplementations`
[6]`http://caniuse.com/#feat=cryptography`

Semantic Closeness: *KMIP* use case is semantically distant from *CREDENTIAL*'s use case, because it is intended for key exchange and key management in enterprise networks. KMIP mentions cloud related use cases [135], but those still differ in scale and semantics from *CREDENTIAL*. An obstacle lies in the role reversal that KMIP entails: In KMIP, the device that manages the keys acts as the server; In *CREDENTIAL*, keys are managed by the user. Therefore, the users device needs to act as a server.

WebCrypto on the other hand is semantically close to the *CREDENTIAL* scenario. The specification covers use cases [171, Section 2 Use Cases] like encrypted document sharing and encrypted cloud storage.

Freedom of Implementation: *KMIP* is a communication protocol and it can be implemented in any environment or language. However, KMIP requires the use of TLS and mutual authentication [136, Section 3.1.3 Client Authenticity], which might not be how authentication is handled in *CREDENTIAL*. Also, a *minimal* KMIP server has to offer a variety of operations and objects [136, Section 5.1 Baseline Server] which are not directly needed in *CREDENTIAL*.

WebCrypto is a Javascript API, which implies that an application that relies on this API is likely to run in browser. The standard does not enforce restriction on the transport layer, so this decision is up to the vendor (Chromium requires *secure origins* [7], Firefox does not). As already mentioned, WebCrypto does not strictly require conforming agents to offer any specific algorithm.

6.4.4 Evaluation Conclusion

This section summarizes the results and decides on whether or not we recommend a specification. Table 10 gives an overview of the evaluation results.

KMIPs strengths lie in its interoperability, extensibility and the freedom to implement custom access restrictions. The problem with KMIP is that it does not address the *CREDENTIAL*'s scenario: A user's device that runs server style applications a mutual TLS authentication are requirements that do not fit *CREDENTIAL*. Because of these incompatibilities we do not recommend to use KMIP for *CREDENTIAL*.

Although *WebCrypto* fits *CREDENTIAL*'s problem domain well, there are too many factors that make a stand against its integration: It's lacking important schemes and types and it does not offer fine-grained access policies. Furthermore, we cannot close these gaps because of WebCrypto's missing extensibility. Therefore, we do not recommend to integrate WebCrypto into *CREDENTIAL*.

Instead, we recommend to embed the cryptographic operations into other required process flows to satisfy *CREDENTIAL*'s requirements with minimal implementation and communication overhead.

[7] `https://www.chromium.org/Home/chromium-security/prefer-secure-origins-for-powerful-new-features`

	KMIP	WebCrypto
Extensibility	H	L
Interoperability	H	M
Adoption	M	H
Component Reuse	M	L
Access Restriction	H	L
Semantic Closeness	L	H
Freedom of Implementation		
Runtime Environment	✓	✗
Transport Layer	✗	~
Feature Set	~	✓

Table 10: Evaluation Results of KMIP and WebCrypto

6.5 Other Technologies

This section presents and evaluates the SCIM protocol. It describes the technology in a fact sheet, weighs the benefits and drawbacks for *CREDENTIAL* and concludes with a summary.

6.5.1 Fact Sheet

System for Cross-domain Identity Management (SCIM) [98]	
Type of Technology	User Management Protocol
Status	Specification (2015), Version 2.0
TRL	H - SCIM is used in operational environments, for example in Microsoft's Azure, and open source implementations are available.
Implementation	`http://www.simplecloud.info/`
IPR (License Model)	`http://trustee.ietf.org/trust-legal-provisions.html`
Brief Description: System for Cross-domain Identity Management (SCIM) specifies how resources and in particular user's identities can be managed in the cloud. It facilitates interoperability, as users can be moved between different cloud services. Additionally, SCIM allows SSO across multiple services. Also, cloud services can discover the configuration of other services automatically, which allows them to associate without human interaction. SCIM consists of two specifications: Firstly, it defines an extensible schema that describes the format of resources. Secondly, SCIM specifies a HTTP-based protocol.	
Relevance to CREDENTIAL: In Credential, SCIM could be used to import/export users between Credential and an external cloud service. Furthermore, the possibility of also moving the user's encrypted data around the cloud has to be investigated.	

Appendix D.5.1 provides a detailed description of SCIM.

6.5.2 Evaluation

Basically, the SCIM protocol was developed for user provisioning, which is realized in the protocol using REST and provides CRUD methods to manage digital identities. The focus is clearly on managing identities whereas authentication and authorization is out of band.

The SCIM core schema is based on an object model which is used to describe users, data, and other objects. It also allows custom objects which improves the extensibility of the protocol. Such a data description facilitates interoperability and enables easy migration between different providers. Furthermore, SCIM supports configurations which document the deployment of a SCIM system and the applied customizations. Such configurations again improve the protocol's interoperability.

However, some aspects also highlight why SCIM is not perfectly suitable for *CREDENTIAL*. SCIM has a low adoption, which is a disadvantage as few implementations are available and for the migration of user data other providers would also have to support it. Furthermore, additional functionalities are required to implement the *CREDENTIAL* system which are conceptually not considered in SCIM. For example, the data storage and access mainly deals with encrypted data, which is not foreseen in SCIM. Also, SCIM seems to be inconvenient to use, as can be see by the way multi-valued attributes are being handled.

6.5.3 Evaluation Conclusion

Resulting on our research and evaluation, we are not recommending the usage of SCIM for *CREDENTIAL*. The reason for this decision is, that beside many positive features, such as extensibility, interoperability, and support for migration, SCIM does not consider the usage or encrypted or signed data, and much less the employment of advanced cryptography. Using SCIM in *CREDENTIAL* would cost too much effort because it has to be extended considerably.

However, SCIM introduces interesting ideas, such as a schema to describe the data which facilitates migration to other providers. Therefore, SCIM would be a candidate for further research, to enhance it with our advanced cryptography and integrate the protocol or core concepts at a later project stage.

6.6 Section Conclusion

This section introduced and evaluated a variety of IAM technologies because these technologies aid to solve many of *CREDENTIAL*'s challenges. We recommended OpenID Connect an SAML as identity protocols because they are widely spread and flexible. For authorization we recommended UMA to perform access control over the flexible data sharing process in *CREDENTIAL* as well as OAuth 2 for additional authroization needs. We recommended XACML as policy because it can be combined with other technologies, whereas the technologies from the WS-* stack are bound to the WS ecosystem. We did not recommend W3C Web Crypto and SCIM because of lacking cryptographic primitives. We also did not recommend KMIP because of its semantic distance to *CREDENTIAL*'s scenario. Appendix D provides further details to the examined technologies.

7 Pilot-Specific Technologies

In this section, an overview of the pilot-specific technologies is presented. *CREDENTIAL*'s three pilots are eGovernment, eHealth and eBusiness. In all three use cases data sharing and security is of utmost importance, as users provide sensitive personal data which have to be protected. All three use cases are introduced by a general description to provide an overview:

eGovernment: An overview of the technologies used in the eGovernment pilot will be provided. Those technologies include Italian and general digital authentication mechanisms as well as international identity protocols.

eHealth: In eHealth, it is especially important to adopt existing standards. Therefore, this is a specific need of healthcare use case. Well known standardization methods like HL7, ISO and IHE will be used to show the possibility of implementation of health care devices into the *CREDENTIAL* wallet.

eBusiness: Three standard technologies are relevant for the eBusiness use case. To achieve secure mail forwarding, especially PEC (Posta Elettronica Certificata) and S/MIME are of interest. Also, information on the Italian identity protocol SPID is given.

7.1 Overview of eGovernment Technologies

In this paragraph, we provide an overview of relevant technologies used in the eGovernment pilot. The technologies mainly refer to hardware and software security aspects: CNS, PKCS#11 and IS0 7816 are focused on the physical secure token used in the pilot, while CSP and STORK/eIDAS adapters are about software providers and identity protocol adapters.

CNS (Carta Nazionale dei Servizi) [6]	
Type of Technology	Smartcard file system standard
Status	Specification. Widely adopted since 2004 with about 50 million cards currently owned by Italian citizens.
TRL	H - CNS standard is well known and widely adopted in Italy. CNS Standard has demonstrated to be still valid and no major changes have been made from its first release to date.
IPR (License Model)	The CNS file system specification is public, but its implementation is reserved to the Italian Public Administration.

Brief Description: CNS is a national smartcard standard that is widely used within Italy. CNS is a ISO 7816 smartcard with a client authentication X509 certificate on-board to allow digital authentication over the Internet. CNS does not contain the user's photo or other biometric data, but stores several personal data such as name, surname, fiscal code, etc. These data are stored in different files and format (mainly ASN.1 and tag-length-value coding) in order to assure broad compatibility with existing software. It also contains a Netlink structure, which is dedicated to store some medical data known when the card is issued and that can be modify by other "authorized" healthcare professional cards (HPC). Netlink derived keys are stored on CNS in order to allow mutual authentication with HPC. It is possible to create and store digital signature keys, using a file system section reserved to authorized Certification Authorities.
Relevance to CREDENTIAL: This smartcard should be used to perform secure user authentication on the net.

CSP (Cryptographic Service Provider) [123]	
Type of Technology	Windows cryptographic libraries
Status	Standard. Widely adopted in Italy in order to allow client authentication between a user browser with CNS smartcard and a secure online service on the Internet. This technology has been recently enriched to create a new technology called "smart card mini driver".
TRL	H - CSP technology has been first developed by Microsoft in late '90 and no major changes have been made from its first release to date.
IPR (License Model)	CSP license typically depends on vendor who implements libraries.
Brief Description: A cryptographic service provider (CSP) contains implementations of cryptographic standards and algorithms. At a minimum, a CSP consists of a dynamic-link library (DLL) that implements the functions in CryptoSPI (a system program interface). Most CSPs contain the implementation of all of their own functions. Some CSPs, however, implement their functions mainly in a Windows-based service program managed by the Windows service control manager. Others implement functions in hardware, such as a smart card or secure coprocessor. If a CSP does not implement its own functions, the DLL acts as a pass-through layer, facilitating the communication between the operating system and the actual CSP implementation. The usage of CSP allows, on a Windows platform, to use Italy's CNS smartcard in a client (mutual) authentication with an identity provider. The user is asked to enter the smartcard's PIN in order to sign the challenge received by the server. The signing process is done using the smartcard private key.	
Relevance to CREDENTIAL: A CSP should be used to enable client authentication via CNS on PCs.	

PKCS #11 [132]

Type of Technology	API Specification for Cryptographic Tokens
Status	Standard (2015), Version 2.40
TRL	H - The PKCS #11 standard has been implemented by many vendors and is in use in operational environments.
Implementation	`https://github.com/OpenSC/OpenSC/wiki`
IPR (License Model)	Open OASIS standard
Brief Description: PKCS #11 defines a standardized API to access functionality of cryptographic tokens. Those cryptographic tokens are devices, such as smart cards and hardware security modules (HSM), which hold key material and perform cryptographic operations.	
Relevance to CREDENTIAL: PKCS #11 would facilitate the implementation of authentication methods. For example, the authentication factor possession could be implemented by connecting a cryptographic token and using it to generate a signature. Such a signature could then be use demonstrate the possession of private key material to a server.	

ISO 7816 [100]	
Type of Technology	International smartcard standard
Status	Standard
TRL	H - Technology widely adopted in smart card worldwide market.
IPR (License Model)	ISO 7816 is an international specification; every ISO specification is available in PDF format and can be bought on ISO or UNI web sites.
Brief Description: ISO 7816 is a smart card international standard. It refers mainly to contact smart card, and does not cover contact-less smart card. It contains 15 “chapters” which covers a wide variety of smart card technological aspects, such as physical characteristics, commands allowed for security operations, and so on.	
Relevance to CREDENTIAL: ISO 7816 could be relevant in the project, as the Lombardy Region smart card (CNS) is a ISO 7816 compliant card.	

STORK/STORK 2.0 Framework [3]	
Type of Technology	Identity Protocol
Status	Framework, Specification Version 2.0
TRL	H - Pilots available in domains eHealth, eLearning, eBanking, and Public Services for Businesses
IPR (License Model)	STORK components can be downloaded and integrated by contacting the STORK consortium.
Brief Description: The aim of the STORK and the STORK 2.0 project was the achievement of cross-border eID interoperability across Europe. The main goal of the project is to contribute to a realization of a single European electronic identification and authentication area, establishing interoperability of different approaches at national and EU level. In the project, people can use their national eID to establish new e-relations with foreign electronic services (public or private service providers). SAML is the base protocol used in STORK/STORK 2.0.	

Relevance to CREDENTIAL: In the *CREDENTIAL* project, a national STORK IdP could be used in order to access to foreign SP. Further, an IdP could be enhanced with *CREDENTIAL* cryptographic libraries to perform proxy re-encryption/decryption and malleable signatures.

eIDAS Interoperability Framework [54]	
Type of Technology	Identity Protocol
Status	Specification Version 1.0, valid since 17th September 2014, EUs countries have to be compliant from 1st July 2016
TRL	M - Open source implementations are available
Implementation	`https://joinup.ec.europa.eu/software/cefeid/news/eidas-sample-implementation-v10`
IPR (License Model)	A sample implementation of eIDAS interoperability framework is public and can be downloaded from a European Commission web site.
Brief Description: The eIDAS regulation ensures that people can use their national electronic identification schemes to access public services in other EUs countries where eIDs are available. The eIDAS regulation will also create a European internal market for electronic signatures, electronic seals, time stamp, etc. The eIDAS specifications are in line with Regulation No 910/2014 of the European Parliament and of the council on electronic identification and trust services for electronic transactions in the internal market, which regulates the acceptance of national eID across borders in the EU. The specifications build the technical basis for meeting the requirements of the eIDAS regulation and the corresponding implementation acts.	
Relevance to CREDENTIAL: The eIDAS regulation will be relevant at least for cross-border eID authentication and electronic signatures interoperability. Technical improvements being addressed for operational security towards eIDAS include crypto-suites for secure channels (TLS) and SAML signature/encryption, encryption of assertions to avoid attacks in the browser, trusted distribution of gateway meta-data (signature and encryption certificates, node addresses, and so on) by extended TSL or SAML meta-data.	

7.2 Overview of eHealth Technologies

Healthcare organizations are rather productive in defining healthcare specific standards for healthcare specific issues such as medical content structuring and encoding. Nevertheless, when it comes to more generic functionalities such as data sharing and security, there is a strong tendency to adopt existing standards to the specific needs of healthcare use case. This adaption is usually referred to as “profiling” with the resulting “profiles” defining constraints on existing standards in order to foster interoperability when using these standards in healthcare.

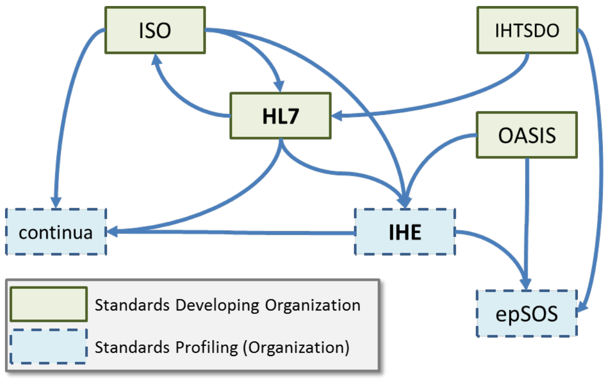

ISO
IHTSDO
HL7
OASIS
continua
IHE
epSOS
Standards Developing Organization
Standards Profiling (Organization)

Figure 3: Most Prominent eHealth Standardization Bodies

Figure 3 sketches some of the most prominent standardization bodies and profiling initiatives which are relevant for healthcare-IT:

- Especially in the domain of mobile health and medical devices healthcare-IT adapts existing ISO standards (i.e. ISO/IEEE 11073).
- HL7 is most prominent SDO that solely focusses on healthcare. HL7 standards range from content representation formats to permission catalogues. Through a co-operation with ANSI some HL7 standards are as well ISO norms.
- IHTSDO is a non-for-profit organization that releases the most comprehensive terminology in healthcare which is an about to substitute many of the existing specialized terminologies.
- IHE (Integrating the Healthcare Enterprise) defines profiles on existing standards which allow implementing eHealth use cases using existing COTS. Typical origins of IHE profiles are HL7 (e.g. document type specifications), OASIS (e.g. ebXML-based sharing of health data) and DICOM (radiology).
- Continua is another profiling organization that defines interoperability profiles for medical devices on top of ISO/IEEE 11073. Continua as well provides a reference architecture for connecting personal health devices to health records by using IHE profiles.
- epSOS is an example for an initiative (European FP7 LSP project) that further profiled existing IHE profiles. By this existing international profiles are further constrained to common European requirements (e.g. as derived from the data protection directive) and to cross-border sharing of health data.

In the following sections some established healthcare-IT standards and profiles are sketched which either may be relevant for *CREDENTIAL* or even shall be considered for the *CREDENTIAL* eHealth use case due to their wide acceptance with vendors and clinics.

Cross-Enterprise Document Sharing (XDS) [94]	
Type of Technology	API and metadata specification for sharing medical documents among healthcare professionals
Status	Specification - 23.07.2010
TRL	H - de facto healthcare-IT standard which is implemented by many IT-vendors.

Brief Description: IHE XDS defines a health record infrastructure within a single domain. It specifies transactions (protocols and interfaces) for four interoperable actors:

- Document Registry: management of document metadata and provisioning of means for document query and registration within a domain
- Document Repository: distributed stores for medical documents
- Document Source: provisioning of medical data to be used by other care providers
- Document Consumer: client application that allows doctors to search and retrieve medical documents that had been provided by other actors

IHE has defined several profiles for adding further functionality to XDS based infrastructures. Among these are:

- Cross-Community Access (XCA): gateways for mediating data query and retrieval across XDS domains
- Cross-Community Fetch (XCF): variant of XCA for implementing stateless cross-domain-gateways (driven by epSOS use cases)
- Cross-Enterprise Document Workflow (XDW): managing workflow data within an XDS registry/repository
- XDS-i: adaptation of XDS paradigms and services for sharing image data through DICOM messages
- Mobile Access to Health Documents (MHD): capsule to XDS infrastructures based on JSON and FHIR for connecting mobile devices as data sources

Relevance to CREDENTIAL: IHE XDS is the de facto healthcare-IT standard for record based infrastructures. The *CREDENTIAL* eHealth pilot shall use an IHE XDS compliant backend for managing medical data. It shall use IHE XDS transactions for making medical data available to health professionals.

Patient Identifier Cross-Referencing (PIX) and Patient Demographics Query (PDQ)[94]	
Type of Technology	Patient identity data types and API for managing patient indexes within healthcare domains

Status	Specification - 23.07.2010
TRL	H - de facto standard integrated and implemented by most of the healthcare-IT vendors
Brief Description: A common problem in healthcare IT is that the same patient is registered with different identifiers within different applications. This not only holds for cross-enterprise use cases (e.g. a hospital and a resident physician referencing the same patient using different IDs) but even within enterprises (e.g. radiology systems introducing their own patient IDs). IHE PIX/PDQ provides means for managing patient IDs within and across enterprises. Among the service provided are querying for patient IDs by demographics (PDQ) and registration and mapping of identifiers issued by different applications (PIX). PIX/PDQ are defined as profiles on top of three different families of standards: • PIXv2/PDQv2: based on HL7v2 messages (EDI alike) • PIXv3/PDQv3: based on HL7v3 messages (XML) • PIXm/PDQm: optimized for mobile devices (JSON and FHIR)	
Relevance to CREDENTIAL: IHE PIX/PDQ is the de facto standard for managing patient identifiers in cross-enterprise use cases. It interplays well with IHE XDS and is supported by most of the existing health record solutions (commercial and open source). Therefore *CREDENTIAL* identity management components shall be able to cope with IHE PIX/PDQ actors and transactions for the eHealth use case. In case that *CREDENTIAL* eHealth use case introduces additional IDs for patients, IHE PIX shall be used to match these with existing patient identifiers.	

Audit Trail and Node Authentication (ATNA) [94]	
Type of Technology	Paradigms and technical means (mutual node authentication and audit trail writing) for securing network nodes and applications
Status	Specification - 23.07.2010
TRL	H - IHE ATNA is the most elaborated profile on top of RFC3881
Brief Description: With its ATNA profile, IHE introduces the notion of a secure node. A secure node only allows (mutually) authenticated actors to utilize services on that node and ensures that an audit trail entry is written for each access to protected data that is stored on that node. By this a foundation level of security (with respect to authenticity and authorization of users and non-repudiation of activities) is provided to all applications which are to be deployed on a secure node. Technically IHE ATNA specifies constraint profiles on IETF TLS, IETF Syslog and other standards to ensure a defined level of security. IHE ATNA only defined means for writing to an audit trail repository. The recently published profile “Add RESTful Query to ATNA” adds a further actor and transactions for securely reading data from an audit trail repository through a defined REST interface.	

Relevance to CREDENTIAL: Non-Repudiation is a rather rigid requirement for most eHealth use cases (including the ones recently discussed for *CREDENTIAL*). Therefore at least the ATNA audit trail repository shall be considered as a required component for the implementation of the *CREDENTIAL* eHealth use case.

Cross-Enterprise Security and Privacy Authorization (XSPA) [130]	
Type of Technology	Standard for cross-enterprise security and privacy profile
Status	Specification - 12.06.2010
TRL	L - Working Draft and no adoption from any big vendor
IPR (License Model)	OASIS IPR Policy [131]
Brief Description: XSPA is a standard which describes the exchange mechanism of privacy policies, consent directives and authorizations for electronic health record systems in an interoperable manner. XSPA profiles have been developed for WS-Trust, XACML and SAML.	
Relevance to CREDENTIAL: XSPA describes profiles used in the healthcare field, which is covered in a pilot project of *CREDENTIAL*. Because the current status is still working draft, the relevance has to be evaluated in more detail.	

Cross-Enterprise User Assertion (XUA) and Cross-Enterprise User Assertion - Attribute Extension (XUA++)[94]	
Type of Technology	Identity protocol
Status	Specification - 15.07.2015
TRL	M - Various implementation exists.
Brief Description: IHE XUA defines constraints on SAMLv2.0 for reflecting specific requirements of eHealth applications. Additionally it specifies which and how certain SAML token profiles may be used within healthcare scenarios. In its XUA++ profile IHE defines, how typical identity attributes of healthcare professionals (e.g. organizational affiliation, professional role) are to be mapped onto SAML attribute statements. IHE XUA++ builds upon the OASIS XSPA standard by limiting and constraining the set of attributes and code systems to be used. In 2015 IHE Germany published a complete mapping table for IHE XUA, OASIS XACML and IHE XDS. By this role/attribute based access control on medical data can easily be enforced for health professionals who had been authenticated using the IHE XUA profile.	
Relevance to CREDENTIAL: For its eHealth use case *CREDENTIAL* shall consider the constraints and attribute definitions from IHE XUA(++) in order to ease the enforcement of permissions on medical resources.	

Document Digital Signature (DSG)[94]	
Type of Technology	Document Signature
Status	Specification - 12.03.2012
TRL	M - Implementation for IHE DSG exists as a trial implementation

Brief Description: IHE DSG defines how to apply detached and enveloping W3C XML signatures for signing (sets of) medical documents and how XAdES extended attributes shall be used for specific eHealth scenarios. Additionally DSG gives normative guidelines on how signed documents are to be stored with electronic health records and how signature validation shall be implemented.
Relevance to CREDENTIAL: As digitally signed documents are the rare exception in regular care, *CREDENTIAL* will with this respect start on an almost blank sheet. Nevertheless the IHE DSG profile shall be considered in a respect that it reflects typical healthcare business requirements and is designed to be easily integrated into existing care workflows.

Document Encryption (DEN)[94]	
Type of Technology	Document encryption
Status	Specification - 19.08.2011 for Trial Implementation
TRL	M - Implementation for IHE-DEN exists
Brief Description: IHE DEN defines how health data shall be encrypted for transmission and storage. By this DEN supports the notion of end-to-end encryption where medical data is only disclosed to the creator and consumer of medical content while all intermediary actors may only process encrypted content.	
Relevance to CREDENTIAL: IHE DEN shall be used as guidance about how data encryption may be integrated with typical health data sharing scenarios. Especially definitions about how encrypted data is to be handled in conjunction with other healthcare-IT standards should be considered.	

Advanced Patient Privacy Consent (APPC)[94]	
Type of Technology	Content model and encoding for patient privacy consents
Status	Specification - 09.09.2016
TRL	M - Trial Implementation
Brief Description: The permissions given to health professionals for accessing identifiable patient data are controlled by the patient's consent. A respective consent lists the organizations involved in a care setting, states the purposes for which data is to be processed and defines further constraints on data processing as agreed between the patient and her carers. IHE APPC specifies a content model and encoding for digital consent documents which may be imported into an access control system for setting respective permissions on the patient's medical data. IHE APPC is designed to be interoperable with common IAM standards such as SAML and XACML while specifically considering existing resource management standards such as IHE XDS.	
Relevance to CREDENTIAL: Participants in the *CREDENTIAL* eHealth use case are requested to give consent to the processing of their personal data. IHE APPC shall be considered as a means for encoding and processing such consents in a manner that allows for a seamless integration into existing healthcare-IT standards.	

Clinical Document Architecture (CDA)[87]	
Type of Technology	Document markup standard
Status	Approved
TRL	H - Implemented and supported by most of the big healthcare-IT vendors.
Brief Description: With its reference information model (RIM) HL7 defines a model and "grammar" for defining structured messages that are shared between healthcare-IT systems. The RIM itself only defines data types and rules for combining and expanding these types. For each type a normative XML binding is defined which allows for a direct derivation from model to implementation. The HL7 Clinical Document Architecture defines a model for structured documents on top of HL7 RIM data types and rules. Concrete document schemes (e.g. a discharge letter or a lab report) can be specified by constraining and instantiating the CDA model. By this such document schemas derive the properties of CDA: 1) Persistence, 2) Stewardship, 3) Potential for authentication, 4) Context, 5) Wholeness and 6) Human readability. Each CDA document is made up from a header and a body which itself is split into sections. Each section represents an area of interest (e.g. medication) and may be further structured into coded entries which represent machine readable clinical statements (e.g. defining the concrete doses and intake instructions for a specific medication). By ignoring the abstract model layer and focusing on the normative CDA XML binding CDA can as well be used as a markup language for clinical documents.	
Relevance to CREDENTIAL: During the last years international SDO/SPO (e.g. HL7 and IHE), European projects (e.g. epSOS) and national initiatives (e.g. ELGA in Austria) defined many CDA document, section and entry templates that can be re-used by *CREDENTIAL*.	

Fast Healthcare Interoperability Resources (FHIR)[88]	
Type of Technology	Class definitions for healthcare concepts
Status	Specification - DSTU2 (final standard) will be approved in 2016
TRL	H - Implemented and supported by most of the healthcare-IT vendors
Brief Description: Due its properties (wholeness, stewardship, etc.) CDA is rather heavy-weight and complex. For better reflecting recent tendencies in healthcare such as mobile health devices, REST service interfaces and the use of web standards (e.g. JSON, OAuth) HL7 developed the FHIR standard, which builds upon modular resource definitions (e.g. "patient", "care plan", "medication") that can be combined to implement arbitrary complex and interoperable information objects. A strong focus of FHIR is on implementability and reduction of complexity. Each resource definition is specified based on abstract data types and comes with normative bindings to XML and JSON. While CDA documents are always self-contained, FHIR follows the approach to consider a clinical information as a network of connected resources which may be managed independently and follow their own lifecycle.	

Relevance to CREDENTIAL: Especially when data is provided through personal health devices, *CREDENTIAL* eHealth use case should consider HL7 FHIR as the preferred standard for representing such data. The same holds for any information exchange which is rather message than document oriented.

7.3 Overview of eBusiness Technologies

In this paragraph a brief overview of technologies used in eBusiness pilot is given. The three technologies listed are well-known or de-facto standards in electronic mail delivery and Italian identity provider scenarios.

PEC (Posta Elettronica Certificata) [8]	
Type of Technology	Email with the same legal value as registered mail (regulated by italian laws)
Status	Specification. Used in Italy since 2005.
TRL	H – the technology is widely used. There are now more than 8 million PEC mailbox and more than 200 million PEC messages every 2 months.
Implementation	`http://www.openpec.org/`
IPR (License Model)	Depends on specific implementation
Brief Description: PEC, compared to traditional e-mail, ensures: authenticity of the sender (that is to say the mail account); integrity of sent message; no delivery refusal; matching between the delivery receipt and the message sent by the user. Providers are required to have a logging system, which tracks and stores all system events for 30 months, except for the mails written by the sender. The PEC Manager's engine implements the receipt management, enveloped messages metadata and the messages signature.	
Relevance to CREDENTIAL: *CREDENTIAL* features can be applied to implement a mail forwarding for encrypted mail solution. The PEC Engine has to be able to store re-encryption keys.	

S/MIME [149]	
Type of Technology	Standard defined for signing and encrypting of mails.
Status	Standard, version 3 proposed in 1999, actual version 3.2
TRL	H – the technology is widely used in many email clients
IPR (License Model)	S/MIME is currently defined in an Internet Standards Track document (RFC 5751). Code Components extracted from the document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
Brief Description: S/MIME is a widely accepted protocol for sending digitally signed and encrypted messages. It is mainly used for email messages and it provides security services such as message integrity, non repudiation of origin and data security with encryption.	

Relevance to CREDENTIAL: The technology is not relevant for identity and access management but has to be considered when forwarding encrypted mails.

SPID [7]	
Type of Technology	Identity Protocol
Status	Specification. SPID specifications are available from early 2015 and the first identity providers delivering SPID authentications are online since March 2016. This protocol is currently used by many Italian public administrations to securely authenticate users on the Internet. Over 1 million SPID credentials have been issued so far (update March 2017). Also some private companies are currently working to integrate the protocol.
TRL	H – the technology is used by production services
IPR (License Model)	Depending on specific implementation
Brief Description: The protocol used in SPID is mainly SAML 2.0, compatible with eIDAS specifications, with some restrictions (for example, just two types of bindings are allowed).	
Relevance to CREDENTIAL: *CREDENTIAL* could integrate this protocol or enhance it with advanced cryptography to selectively disclose the user identity's data.	

8 Conclusion

This deliverable assessed technologies that are relevant for the implementation of *CREDENTIAL*'s vision: 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. In this assessment, we presented an overview of relevant technologies, grouped technologies into clusters, defined technology-specific criteria based on high-level criteria, and evaluated the technologies according to these criteria for the application within *CREDENTIAL*. Our assessment results range from recommendations if the technology fits our use case well, to technologies with limited applicability which might require further research, and finally to technologies that are not recommended due to better alternatives or clashes with goals of *CREDENTIAL*. We summarize the assessment results for our four main categories of technologies in the following lines and visualize them in Table 1:

Core Cryptographic Technologies are fundamental cryptographic mechanisms that allow to securely share data from end-to-end across the cloud and ensure the data's authenticity even if only a subset is disclosed. For *secure data sharing*, we recommend to use classical proxy re-encryption or conditional proxy re-encryption to cryptographically enforce access policies. Proxy re-encryption with keyword search could provide search functionality, and certificate-less or certificate based proxy re-encryption could improve scalability, but further research is required to apply a suitable mechanism. Identity-based and attribute-based proxy re-encryption are not recommended due to their high trust requirements. Attribute-based encryption offers a viable alternative; our main reason for striving for proxy re-encryption is that it seems to be more flexible from a user point of view. Finally, fully homomorphic encryption is currently not sufficiently efficient. For the *disclosure of authentic data*, we recommend the usage of redactable signatures in our cloud-based setting. While anonymous credentials offer additional privacy benefits, their application in the cloud requires further research.

Additional Cryptographic Technologies are cryptographic means that are not at the core of *CREDENTIAL* but might provide further benefits. Therefore, these technologies should be considered for further research in the course of this project. TPASS, password-based cryptography and especially distributed password verification could improve the security of password-based authentication, which is however not in the core interest of *CREDENTIAL*. Searchable encryption minimizes information leakage when performing searches on the user's data and also proofs of retrievability and provable data possession offer benefits by ensuring the integrity and availability of data stored at the *CREDENTIAL* cloud service. While private information retrieval and oblivious RAM would represent privacy improvements, we currently do not recommend them for the use with large datasets due to insufficient efficiency. Additionally, using unlinkable pseudonyms could be a further privacy enhancement, secret sharing could facilitate backups of key material, and verifiable computing techniques could ensure that data were correctly transformed while moving through the cloud.

Authentication to the Cloud is a fundamental task within *CREDENTIAL* where the user is directly involved and therefore not only privacy and security aspects but also usability is of critical importance. As *authentication technologies*, we recommend FIDO UAF and U2F as well as OATH, since these technologies are widely adopted, user-friendly, and compatible with different authentication methods, such as biometric authentication techniques which are especially considered within the project. SQRL seems an interesting approach and its concept would be of interest for further research. Mobile connect is not recommended to be used within *CREDENTIAL* as it requires a mobile network operator as intermediary. Also, we evaluated *underlying technologies* to improve the security of keys used within the authentication and data sharing process by binding them to the hardware. We recommend to employ trusted platform modules. As these trusted hardware modules only support a limited set of cryptographic mechanisms, trusted execution environments should be further researched to also integrate more advanced cryptography.

Identity and Access Management focuses on protocols to be used within *CREDENTIAL* to exchange identity assertions between entities, define and demonstrate permissions, access cryptography and manage the user's data. As *identity protocol*, we recommend OpenID Connect and SAML since their flexibility, extensibility and interoperability features and broad adoption fit *CREDENTIAL*'s vision to implement a practical identity provider. For *authorization*, we recommend to use OAuth and its User-Managed Access (UMA) profile to demonstrate permission to access *CREDENTIAL*'s server-side functionality through a suitable and standardized protocol. *Policies* can be expressed, evaluated and enforced through the powerful XACML specification and its implementations. We also evaluated means to remotely *access cryptography*, but neither the Web Cryptography API nor KMIP were closely aligned with *CREDENTIAL*'s vision, so a custom integration appears more suitable. Also, SCIM introduces interesting concepts and should be considered for further research.

List of References

[1] OAuth 2.0 Web Page. `http://oauth.net/2/`. Accessed: 14/07/2016.

[2] SAML Open Source Implementations. `http://saml.xml.org/wiki/saml-open-source-implementations`. Accessed: 07.07.2016.

[3] STORK - Secure idenTity acrOss boRders linKed 2.0. `https://www.eid-stork2.eu/`, Accessed: 13.03.2017.

[4] *11th International Conference on Availability, Reliability and Security, ARES 2016, Salzburg, Austria, August 31 - September 2, 2016*. IEEE Computer Society, 2016.

[5] Ittai Abraham, Christopher W. Fletcher, Kartik Nayak, Benny Pinkas, and Ling Ren. Asymptotically Tight Bounds for Composing ORAM with PIR. In Serge Fehr, editor, *Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part I*, volume 10174 of *Lecture Notes in Computer Science*, pages 91–120. Springer, 2017.

[6] AgID (Agenzia per l'Italia Digitale). Carta nazionale dei servizi (CNS). `http://www.agid.gov.it/agenda-digitale/infrastrutture-architetture/carta-nazionale-servizi`, Accessed: 13.03.2017.

[7] AgID (Agenzia per l'Italia Digitale). Sistema Pubblico per la gestione dell'Identità Digitale - SPID. `http://www.agid.gov.it/agenda-digitale/infrastrutture-architetture/spid`, Accessed: 13.03.2017.

[8] AgID (Agenzia per l'Italia Digitale) and former CNIPA (Centro Nazionale per Informatica per la Pubblica Amministrazione). Posta Elettronica Certificata (PEC). `http://www.agid.gov.it/agenda-digitale/infrastrutture-architetture/posta-elettronica-certificata`, Accessed: 13.03.2017.

[9] Jae Hyun Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, Abhi Shelat, and Brent Waters. Computing on Authenticated Data. *J. Cryptology*, 28(2):351–395, 2015.

[10] Sattam S. Al-Riyami and Kenneth G. Paterson. Certificateless Public Key Cryptography. In Chi-Sung Laih, editor, *Advances in Cryptology - ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003, Proceedings*, volume 2894 of *Lecture Notes in Computer Science*, pages 452–473. Springer, 2003.

[11] Dmitri Asonov. Private information retrieval - an overview and current trends. In *GI Jahrestagung (2)*, pages 889–894, 2001.

[12] Giuseppe Ateniese, Karyn Benson, and Susan Hohenberger. Key-Private Proxy Re-encryption. In Marc Fischlin, editor, *Topics in Cryptology - CT-RSA 2009, The Cryptographers' Track at the RSA Conference 2009, San Francisco, CA, USA, April 20-24, 2009.*

Proceedings, volume 5473 of *Lecture Notes in Computer Science*, pages 279–294. Springer, 2009.

[13] Giuseppe Ateniese, Randal C. Burns, Reza Curtmola, Joseph Herring, Osama Khan, Lea Kissner, Zachary N. J. Peterson, and Dawn Song. Remote Data Checking Using Provable Data Possession. *ACM Trans. Inf. Syst. Secur.*, 14(1):12:1–12:34, 2011.

[14] Giuseppe Ateniese, Randal C. Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary N. J. Peterson, and Dawn Xiaodong Song. Provable Data Possession at Untrusted Stores. In Ning et al. [127], pages 598–609.

[15] Giuseppe Ateniese, Randal C. Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary N. J. Peterson, and Dawn Xiaodong Song. Provable Data Possession at Untrusted Stores. In *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 598–609, 2007.

[16] Giuseppe Ateniese, Daniel H. Chou, Breno de Medeiros, and Gene Tsudik. Sanitizable Signatures. In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, *Computer Security - ESORICS 2005, 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005, Proceedings*, volume 3679 of *Lecture Notes in Computer Science*, pages 159–177. Springer, 2005.

[17] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30, 2006.

[18] Mark Bartel, John Boyer, Barb Fox, Brian LaMacchia, and Ed Simon. XML Signature Syntax and Processing (Second Edition). `https://www.w3.org/TR/xmldsig-core/`, 2008.

[19] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and Efficiently Searchable Encryption. In *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, pages 535–552, 2007.

[20] Mihir Bellare and Viet Tung Hoang. Adaptive Witness Encryption and Asymmetric Password-Based Cryptography. In *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, pages 308–331, 2015.

[21] Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993.*, pages 62–73. ACM, 1993.

[22] Patrik Bichsel. *Cryptographic Protocols and System Aspects for Practical Data-minimizing Authentication.* PhD thesis, K.U.Leuven, Belgium, March 2012.

[23] G. R. Blakley and David Chaum, editors. *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*. Springer, 1985.

[24] George R. Blakley. Safeguarding Cryptographic Keys. *Proceedings of the National Computer Conference*, 48:313—317, 1979.

[25] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible Protocols and Atomic Proxy Cryptography. In Kaisa Nyberg, editor, *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, volume 1403 of *Lecture Notes in Computer Science*, pages 127–144. Springer, 1998.

[26] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. Order-Preserving Symmetric Encryption. In *Proceedings of the 28th Annual International Conference on Advances in Cryptology: The Theory and Applications of Cryptographic Techniques*, EUROCRYPT '09, pages 224–241, Berlin, Heidelberg, 2009. Springer-Verlag.

[27] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public Key Encryption with Keyword Search. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer, 2004.

[28] Christoph Bösch, Pieter Hartel, Willem Jonker, and Andreas Peter. A Survey of Provably Secure Searchable Encryption. *ACM Comput. Surv.*, 47(2):18:1–18:51, August 2014.

[29] Kevin D. Bowers, Ari Juels, and Alina Oprea. Proofs of Retrievability: Theory and Implementation. In Radu Sion and Dawn Song, editors, *Proceedings of the first ACM Cloud Computing Security Workshop, CCSW 2009, Chicago, IL, USA, November 13, 2009*, pages 43–54. ACM, 2009.

[30] Brian Kissel. OpenID 2009 Year in Review. `http://openid.net/2009/12/16/openid-2009-year-in-review/`, 2009.

[31] Christina Brzuska, Heike Busch, Özgür Dagdelen, Marc Fischlin, Martin Franz, Stefan Katzenbeisser, Mark Manulis, Cristina Onete, Andreas Peter, Bertram Poettering, and Dominique Schröder. Redactable Signatures for Tree-Structured Data: Definitions and Constructions. In *Applied Cryptography and Network Security, 8th International Conference, ACNS 2010, Beijing, China, June 22-25, 2010. Proceedings*, pages 87–104, 2010.

[32] Johannes Buchmann, Denise Demirel, David Derler, Lucas Schabhüser, and Daniel Slamanig. *PRISMACLOUD D5.8 Overview of Verifiable Computing Techniques Providing Private and Public Verification.* 2 2016.

[33] Jan Camenisch, Robert R. Enderlein, and Gregory Neven. Two-Server Password-Authenticated Secret Sharing UC-Secure Against Transient Corruptions. In *Public-Key*

Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings, pages 283–307, 2015.

[34] Jan Camenisch and Els Van Herreweghen. Design and Implementation of the *idemix* Anonymous Credential System. In Vijayalakshmi Atluri, editor, *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washington, DC, USA, November 18-22, 2002*, pages 21–30. ACM, 2002.

[35] Jan Camenisch, Stephan Krenn, Anja Lehmann, Gert Læssøe Mikkelsen, Gregory Neven, and Michael Østergaard Pedersen. Formal Treatment of Privacy-Enhancing Credential Systems. In Orr Dunkelman and Liam Keliher, editors, *Selected Areas in Cryptography - SAC 2015 - 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers*, volume 9566 of *Lecture Notes in Computer Science*, pages 3–24. Springer, 2015.

[36] Jan Camenisch and Anja Lehmann. (un)linkable pseudonyms for governmental databases. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 1467–1479. ACM, 2015.

[37] Jan Camenisch, Anja Lehmann, Anna Lysyanskaya, and Gregory Neven. Memento: How to Reconstruct Your Secrets from a Single Password in a Hostile Environment. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 256–275, 2014.

[38] Jan Camenisch, Anja Lehmann, and Gregory Neven. Optimal Distributed Password Verification. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 182–194, 2015.

[39] Jan Camenisch and Anna Lysyanskaya. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 61–76. Springer, 2002.

[40] B. Campbell, C. Mortimore, and M. Jones. Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants. RFC 7522 (Proposed Standard), May 2015.

[41] Ran Canetti and Susan Hohenberger. Chosen-Ciphertext Secure Proxy Re-Encryption. In Ning et al. [127], pages 185–194.

[42] David Cash, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Highly-Scalable Searchable Symmetric Encryption with Support for boolean Queries. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 353–373. Springer, 2013.

[43] Dario Catalano. Homomorphic Signatures and Message Authentication Codes. In Michel Abdalla and Roberto De Prisco, editors, *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, volume 8642 of *Lecture Notes in Computer Science*, pages 514–519. Springer, 2014.

[44] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 3–33, 2016.

[45] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, FOCS '95, pages 41–, Washington, DC, USA, 1995. IEEE Computer Society.

[46] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private Information Retrieval. *J. ACM*, 45(6):965–981, November 1998.

[47] Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pages 79–88. ACM, 2006.

[48] B. de Medeiros, N. Agarwal, N. Sakimura, J. Bradley, and M. Jones. OpenID Connect Session Management 1.0 - draft 26. Technical report, OpenID Foundation, February 2016. `http://openid.net/specs/openid-connect-session-1_0.html`.

[49] Denise Demirel, Stephan Krenn, Thomas Lorünser, and Giulia Traverso. Efficient and Privacy Preserving Third Party Auditing for a Distributed Storage System. In *11th International Conference on Availability, Reliability and Security, ARES 2016, Salzburg, Austria, August 31 - September 2, 2016* [4], pages 88–97.

[50] David Derler, Henrich C. Pöhls, Kai Samelin, and Daniel Slamanig. A General Framework for Redactable Signatures and New Constructions. In *Information Security and Cryptology - ICISC 2015 - 18th International Conference, Seoul, South Korea, November 25-27, 2015, Revised Selected Papers*, pages 3–19, 2015.

[51] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 617–640. Springer, 2015.

[52] ETSI. TS 102 778; Electronic Signatures and Infrastructures (ESI); PDF Advanced Electronic Signature Profiles, 2009.

[53] European Commission. Technology Readiness Level Definitions. http://www.nasa.gov/pdf/458490main_TRL_Definitions.pdf.

[54] European Commission. Regulation (EU) No 910/2014 of the European Parliament and the Council on Electronic Identification and Trust Services for Electronic Transactions in the Internal Market, 2014.

[55] Liming Fang, Willy Susilo, Chunpeng Ge, and Jiandong Wang. Interactive Conditional Proxy Re-Encryption with Fine Grain Policy. *Journal of Systems and Software*, 84(12):2293–2302, 2011.

[56] FIDO Alliance. FIDO UAF Architectural Overview, 2014. https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-overview-v1.0-ps-20141208.html#fido-uaf-authenticator accessed April 2016.

[57] FIDO Alliance. FIDO UAF Protocol Specification v1.0, 2014. https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-protocol-v1.0-ps-20141208.html.

[58] FIDO Alliance. FIDO UAF Architectural Overview, 2017. https://fidoalliance.org/specs/fido-u2f-v1.1-id-20160915/fido-u2f-overview-v1.1-id-20160915.pdf accessed March 2017.

[59] ForgeRock. OpenUMA, UMA protocol flow summarized diagram. https://forgerock.org/openuma. Accessed: June 2016.

[60] OpenID Foundation. Libraries, Products, and Tools. http://openid.net/developers/libraries/. Accessed: 08/09/2016.

[61] OpenID Foundation. OpenID Certification. http://openid.net/certification/. Accessed: 08/09/2016.

[62] Aurélien Francillon, Quan Nguyen, Kasper Bonne Rasmussen, and Gene Tsudik. Systematic Treatment of Remote Attestation. *IACR Cryptology ePrint Archive*, 2012:713, 2012.

[63] Aurélien Francillon, Quan Nguyen, Kasper Bonne Rasmussen, and Gene Tsudik. A Minimalist Approach to Remote Attestation. In Gerhard Fettweis and Wolfgang Nebel, editors, *Design, Automation & Test in Europe Conference & Exhibition, DATE 2014, Dresden, Germany, March 24-28, 2014*, pages 1–6. European Design and Automation Association, 2014.

[64] Taher El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In Blakley and Chaum [23], pages 10–18.

[65] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, pages 465–482, 2010.

[66] Craig Gentry. Certificate-Based Encryption and the Certificate Revocation Problem. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 272–293. Springer, 2003.

[67] Craig Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford, CA, USA, 2009. AAI3382729.

[68] Gibson Research Corporation. SQRL Client-Side Key Management. `https://www.grc.com/sqrl/key-flow.htm`.

[69] Gibson Research Corporation. SQRL Secure Quick Reliable Login, 2014. `https://www.grc.com/sqrl/sqrl.htm`.

[70] Ian Goldberg. Improving the Robustness of Private Information Retrieval. In *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*, pages 131–148, 2007.

[71] Ian Goldberg. Improving the Robustness of Private Information Retrieval. In *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*, pages 131–148. IEEE Computer Society, 2007.

[72] Oded Goldreich and Rafail Ostrovsky. Software Protection and Simulation on Oblivious RAMs. *J. ACM*, 43(3):431–473, 1996.

[73] Marc Goodner and Anthony Nadalin. Web Services Federation Language (WS-Federation) Version 1.2. Technical report, OASIS, 2009.

[74] Marc Goodner, Sidd Shenoy, and Lloyd Burch. WSFED TC Interop Scenarios. `https://www.oasis-open.org/committees/download.php/25931/WSFED-TC-InteropScenarios-ED01.doc`, 2007.

[75] Matthew Green and Giuseppe Ateniese. Identity-Based Proxy Re-encryption. In Jonathan Katz and Moti Yung, editors, *Applied Cryptography and Network Security, 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007, Proceedings*, volume 4521 of *Lecture Notes in Computer Science*, pages 288–306. Springer, 2007.

[76] Thomas Groß, Birgit Pfitzmann, and Ahmad-Reza Sadeghi. Proving a WS-Federation Passive Requestor Profile with a Browser Model. In Ernesto Damiani and Hiroshi Maruyama, editors, *Proceedings of the 2nd ACM Workshop On Secure Web Services, SWS 2005, Fairfax, VA, USA, November 11, 2005*, pages 54–64. ACM, 2005.

[77] Trusted Computing Group. Trusted Platform Module - 2.0: a brief introduction. `http://www.trustedcomputinggroup.org/wp-content/uploads/TPM-2.0-A-Brief-Introduction.pdf`.

[78] Trusted Computing Group. Trusted Platform Module (TPM) Summary. `http://www.trustedcomputinggroup.org/trusted-platform-module-tpm-summary/`.

[79] GSMA. Mobile Connect. http://www.gsma.com/personaldata/mobile-connect, Accessed: 20.03.2017.

[80] V. H. Gupta and K. Gopinath. G_{its}^2 VSR: An Information Theoretical Secure Verifiable Secret Redistribution Protocol for Long-term Archival Storage. In *Fourth International IEEE Security in Storage Workshop, SISW 2007, San Diego, California, USA, September 27, 2007*, pages 22–33. IEEE Computer Society, 2007.

[81] Christian Hanser and Daniel Slamanig. Blank Digital Signatures. In Kefei Chen, Qi Xie, Weidong Qiu, Ninghui Li, and Wen-Guey Tzeng, editors, *8th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '13, Hangzhou, China - May 08 - 10, 2013*, pages 95–106. ACM, 2013.

[82] Christian Hanser and Daniel Slamanig. Efficient Simultaneous Privately and Publicly Verifiable Robust Provable Data Possession from Elliptic Curves. In Pierangela Samarati, editor, *SECRYPT 2013 - Proceedings of the 10th International Conference on Security and Cryptography, Reykjavík, Iceland, 29-31 July, 2013*, pages 15–26. SciTePress, 2013.

[83] Thomas Hardjono, Nate Klingenstein, and Scott Cantor. SAML Version 2.0 Errata 05. Technical Report May, OASIS, 2012.

[84] D. Hardt. The OAuth 2.0 Authorization Framework. Internet Engineering Task Force (IETF), October 2012. http://tools.ietf.org/html/rfc6749.

[85] Ian Hickson. HTML5 Web Messaging. W3C Recommendation, May 2015. https://www.w3.org/TR/webmessaging.

[86] Frederick Hirsch, Rob Philpott, and Eve Maler. Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0. Technical report, OASIS, 2005.

[87] HL7. CDA Release 2. http://www.hl7.org/implement/standards/product_brief.cfm?product_id=7, Accessed: 20.03.2017.

[88] HL7. FHIR Documentation. https://www.hl7.org/fhir/, Accessed: 20.03.2017.

[89] HL7. Reference Information Model. http://www.hl7.org/implement/standards/rim.cfm, Accessed: 20.03.2017.

[90] Felix Hörandner, Stephan Krenn, Andrea Migliavacca, Florian Thiemer, and Bernd Zwattendorfer. CREDENTIAL: A framework for privacy-preserving cloud-based data sharing. In *11th International Conference on Availability, Reliability and Security, ARES 2016, Salzburg, Austria, August 31 - September 2, 2016* [4], pages 742–749.

[91] R. Housley. Cryptographic Message Syntax (CMS). STD 70, Internet Engineering Task Force (IETF), September 2009. http://www.rfc-editor.org/rfc/rfc5652.txt.

[92] Mark P. Hoyle and Chris J. Mitchell. On Solutions to the Key Escrow Problem. In Bart Preneel and Vincent Rijmen, editors, *State of the Art in Applied Cryptography, Course*

on Computer Security and Industrial Cryptography, Leuven, Belgium, June 3-6, 1997. Revised Lectures, volume 1528 of *Lecture Notes in Computer Science*, pages 277–306. Springer, 1997.

[93] John Hughes, Scott Cantor, Jeff Hodges, Frederick Hirsch, Prateek Mishra, Rob Philpott, and Eve Maler. Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0 - Errata Composite. Technical report, OASIS, 2009.

[94] IHE. IT Infrastructure Technical Frameworks. `http://ihe.net/Technical_Frameworks/`, Accessed: 20.03.2017.

[95] Takeshi Imamura, Blair Dillaway, and Ed Simon. XML Encryption Syntax and Processing (W3C Recommendation). `https://www.w3.org/TR/xmlenc-core/`, 2002.

[96] Kantara Initiative. SAML Interoperable Implementations, Tools, Libraries and Services. `http://kantarainitiative.org/programs/iop-saml/`. Accessed: 07.07.2016.

[97] Initiative for Open AuTHentication. OATH - The Initiative for Open AuTHentication , 2016. `http://www.openauthentication.org/specification`.

[98] Internet Engineering Task Force (IETF). Definitions, Overview, Concepts, and Requirements, 2015. `https://tools.ietf.org/html/rfc7642` accessed in July 2016.

[99] Internet Engineering Task Force (IETF). System for Cross-Domain Identity Management: Core Schema, 2015. `https://tools.ietf.org/html/rfc7643` accessed in July 2016.

[100] ISO). 7816-1:2011 Identification cards – Integrated circuit cards. `https://www.iso.org/standard/54089.html`, Accessed: 13.03.2017.

[101] Anca-Andreea Ivan and Yevgeniy Dodis. Proxy Cryptography Revisited. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2003, San Diego, California, USA*. The Internet Society, 2003.

[102] Johannes Buchmann and Denise Demirel and Andreas Happe and Stephan Krenn and Guilia Traverso and Thomas Lorünser. Secret Sharing Protocols for Various Adversary Models. EU H2020 PRISMACLOUD Project Deliverable, 2016.

[103] Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner. Homomorphic Signature Schemes. In Bart Preneel, editor, *Topics in Cryptology - CT-RSA 2002, The Cryptographer's Track at the RSA Conference, 2002, San Jose, CA, USA, February 18-22, 2002, Proceedings*, volume 2271 of *Lecture Notes in Computer Science*, pages 244–262. Springer, 2002.

[104] M. Jones. OpenID Connect Front-Channel Logout 1.0 - draft 00. OpenID Foundation, February 2016. `http://openid.net/specs/openid-connect-frontchannel-1_0.html`.

[105] M. Jones and J. Bradley. OpenID Connect Back-Channel Logout 1.0 - draft 02. OpenID Foundation, February 2016. `http://openid.net/specs/openid-connect-backchannel-1_0.html`.

[106] M. Jones and B. Campbell. OAuth 2.0 Form Post Response Mode. Technical report, OpenID Foundation, April 2015. http://openid.net/specs/oauth-v2-form-post-response-mode-1_0.html.

[107] M. Jones, B. Campbell, and C. Mortimore. JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants. RFC 7523 (Proposed Standard), May 2015.

[108] M. Jones and D. Hardt. The OAuth 2.0 Authorization Framework: Bearer Token Usage. Internet Engineering Task Force (IETF), October 2012. http://tools.ietf.org/html/rfc6750.

[109] M. Jones, N. Sakimura, and J. Bradley. OAuth 2.0 Authorization Server Discovery Metadata. (Draft 03), July 2016.

[110] P. Jones, G. Salgueiro, M. Jones, and J. Smarr. WebFinger. Internet Engineering Task Force (IETF), September 2013. http://tools.ietf.org/html/rfc7033.

[111] Ari Juels and Burton S. Kaliski Jr. PORS: Proofs of Retrievability for Large Files. In Ning et al. [127], pages 584–597.

[112] Kantara Initiative. The Three Phases of the UMA Profile of OAuth. https://docs.kantarainitiative.org/uma/draft-uma-core.html#UMA-phases. Accessed: June 2016.

[113] Kantara Initiative. Webinar Slides from 16 May 2015. http://kantarainitiative.org/confluence/download/attachments/17760302/UMA%20webinar%202015-05-16.pdf?api=v2. Accessed: June 2016.

[114] Kantara Initiative. User-Managed Access (UMA) – Profile of OAuth 2.0, 2015. https://docs.kantarainitiative.org/uma/draft-uma-core.html. Accessed: June 2016.

[115] Stephan Krenn, Thomas Lorünser, and Christoph Striecks. Batch-Verifiable Secret Sharing With Unconditional Privacy. In *ICISSP*, 2016. (in press).

[116] Kelvin Lawrence, Chris Kaler, Anthony Nadalin, Marc Goodner, Martin Gudgin, David Turner, Abbie Barbir, and Hans Granqvist. WS-SecurityPolicy 1.2 incorporating Approved Errata 01. Technical report, OASIS, 2012. http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/errata01/ws-securitypolicy-1.2-errata01-complete.html.

[117] Kelvin Lawrence, Chris Kaler, Anthony Nadalin, Marc Goodner, Martin Gudgin, David Turner, Abbie Barbir, and Hans Granqvist. WS-Trust 1.4 incorporating Approved Errata 01. Technical report, OASIS, 2012. http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/ws-trust.html.

[118] Kaitai Liang, Liming Fang, Willy Susilo, and Duncan S. Wong. A Ciphertext-Policy Attribute-Based Proxy Re-encryption with Chosen-Ciphertext Security. In *2013 5th International Conference on Intelligent Networking and Collaborative Systems, Xi'an city, Shaanxi province, China, September 9-11, 2013*, pages 552–559. IEEE, 2013.

[119] Benoît Libert and Damien Vergnaud. Unidirectional Chosen-Ciphertext Secure Proxy Re-encryption. In Ronald Cramer, editor, *Public Key Cryptography - PKC 2008, 11th International Workshop on Practice and Theory in Public-Key Cryptography, Barcelona, Spain, March 9-12, 2008. Proceedings*, volume 4939 of *Lecture Notes in Computer Science*, pages 360–379. Springer, 2008.

[120] T. Lodderstedt, M. McGloin, and P. Hunt. OAuth 2.0 Threat Model and Security Considerations. RFC 6819 (Informational), January 2013.

[121] Drew Mazure, Susan Bramhall, Howard Gilbert, Andy Newman, Andrew Petro, Robert Oschwald, and Misagh Moayyed. CAS Protocol 3.0 Specification. Technical report, APEREO, 2015.

[122] Carlos Aguilar Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. XPIR: Private Information Retrieval for Everyone. *PoPETs*, 2016(2):155–174, 2016.

[123] Microsoft. Cryptographic Service Providers. `https://msdn.microsoft.com/library/windows/desktop/aa380245%28v=vs.85%29.aspx`, Accessed: 21.03.2017.

[124] Mozilla. Persona. `https://developer.mozilla.org/en-US/Persona`. Accessed: 07/08/2016.

[125] Muhammad Naveed, Seny Kamara, and Charles V. Wright. Inference Attacks on Property-Preserving Encrypted Databases. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 644–655, New York, NY, USA, 2015. ACM.

[126] C. Neuman, T. Yu, S. Hartman, and K. Raeburn. The Kerberos Network Authentication Service (V5). RFC 4120, Internet Engineering Task Force (IETF), July 2005. `http://www.rfc-editor.org/rfc/rfc4120.txt`.

[127] Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors. *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*. ACM, 2007.

[128] David Nuñez, Isaac Agudo, and Javier Lopez. A Parametric Family of Attack Models for Proxy Re-encryption. In Cédric Fournet, Michael W. Hicks, and Luca Viganò, editors, *IEEE 28th Computer Security Foundations Symposium, CSF 2015, Verona, Italy, 13-17 July, 2015*, pages 290–301. IEEE Computer Society, 2015.

[129] David Nuñez, Isaac Agudo, and Javier Lopez. NTRUReEncrypt: An Efficient Proxy Re-Encryption Scheme Based on NTRU. In Feng Bao, Steven Miller, Jianying Zhou, and Gail-Joon Ahn, editors, *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '15, Singapore, April 14-17, 2015*, pages 179–189. ACM, 2015.

[130] OASIS. Cross-Enterprise Security and Privacy Authorization (XSPA) TC. `https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xspa`, Accessed: 20.03.2017.

[131] OASIS. Intellectual Property Rights (IPR) Policy. https://www.oasis-open.org/policies-guidelines/ipr, Accessed: 20.03.2017.

[132] OASIS). PKCS #11 Cryptographic Token Interface Base Specification Version 2.40 Errata 01. http://docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/errata01/os/pkcs11-base-v2.40-errata01-os.pdf, Accessed: 13.03.2017.

[133] OASIS. Security Assertion Markup Language (SAML) V2.0 Technical Overview. https://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf accessed in March 2016.

[134] OASIS. eXtensible Access Control Markup Language (XACML) Version 3.0 - Committee Specification 01, 2010. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.pdf. Accessed: June 2016.

[135] OASIS. Key Management Interoperability Protocol Use Cases Version 1.2. OASIS Working Draft, 2013. https://www.oasis-open.org/committees/download.php/49644/kmip-usecases-v1.2-wd10.doc accessed in July 2016.

[136] OASIS. Key Management Interoperability Protocol Profiles Version 1.3. OASIS Standard, 2016. http://docs.oasis-open.org/kmip/profiles/v1.3/kmip-profiles-v1.3.html accessed in Feb 2017.

[137] OASIS. Key Management Interoperability Protocol Specification Version 1.2, Edited by Kiran Thota and Kelley Burgin. OASIS Standard, 2016. http://docs.oasis-open.org/kmip/spec/v1.3/kmip-spec-v1.3.html accessed in Feb 2017.

[138] OpenID Foundation. Libraries for Obsolete Specifications. http://openid.net/developers/libraries/obsolete/.

[139] OpenID Foundation. OpenID Attribute Exchange 1.0 - Final. https://openid.net/specs/openid-attribute-exchange-1_0.html, 2007.

[140] OpenID Foundation. OpenID Authentication 2.0 - Final. http://openid.net/specs/openid-authentication-2_0.html, 2007.

[141] OSIS. http://osis.idcommons.net/wiki/I3:Cross_Solution_OpenID_Identity_Provider_x_Relying_Party_Results, 2008.

[142] Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.

[143] Christian Paquin and Gregory Zaverucha. U-Prove Cryptographic Specification v1.1 (Revision 2). Technical report, Microsoft Corporation, 2013.

[144] Colin Percival. Stronger Key Derivation via Sequential Memory-Hard Functions.

[145] Giuseppe Persiano and Ivan Visconti. An Efficient and Usable Multi-show Non-transferable Anonymous Credential System. In Ari Juels, editor, *Financial Cryptography, 8th International Conference, FC 2004, Key West, FL, USA, February 9-12, 2004. Revised Papers*, volume 3110 of *Lecture Notes in Computer Science*, pages 196–211. Springer, 2004.

[146] Global Platform. The Trusted Execution Environment: Delivering Enhanced Security at a Lower Cost to the Mobile Market, June 2015. `http://www.globalplatform.org/documents/whitepapers/GlobalPlatform_TEE_Whitepaper_2015.pdf`.

[147] Henrich Christopher Pöhls, Kai Samelin, Hermann de Meer, and Joachim Posegga. Flexible Redactable Signature Schemes for Trees - Extended Security Model and Construction. In *SECRYPT 2012 - Proceedings of the International Conference on Security and Cryptography, Rome, Italy, 24-27 July, 2012, SECRYPT is part of ICETE - The International Joint Conference on e-Business and Telecommunications*, pages 113–125, 2012.

[148] Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. CryptDB: Protecting Confidentiality with Encrypted Query Processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 85–100, New York, NY, USA, 2011. ACM.

[149] B. Ramsdell and S. Turner. Secure/multipurpose internet mail extensions (s/mime) version 3.2 message specification. RFC 5751, Internet Engineering Task Force (IETF), January 2010. `http://www.rfc-editor.org/rfc/rfc5751.txt`.

[150] Kai Rannenberg, Jan Camenisch, and Ahmad Sabouri, editors. *Attribute-based Credentials for Trust: Identity in the Information Society.* Springer, 2015.

[151] J. Richer, M. Jones, J. Bradley, M. Machulak, and P. Hunt. OAuth 2.0 Dynamic Client Registration Protocol. `http://www.ietf.org/rfc/rfc7591.txt`, July 2015.

[152] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On Data Banks and Privacy Homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

[153] N. Sakimura, J. Bradley, and M. Jones. OpenID Connect Dynamic Client Registration 1.0 incorporating Errata Set 1. Technical report, OpenID Foundation, November 2014. `http://openid.net/specs/openid-connect-registration-1_0.html`.

[154] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore. OpenID Connect Core 1.0. Technical report, OpenID Foundation, April 2014. `http://openid.net/specs/openid-connect-core-1_0.html`.

[155] N. Sakimura, J. Bradley, M. Jones, and E. Jay. OpenID Connect Discovery 1.0 incorporating Errata Set 1. Technical report, OpenID Foundation, November 2014. `http://openid.net/specs/openid-connect-discovery-1_0.html`.

[156] Andrei Sambra, Stephane Corlosquet, Andrei Sambra, Henry Story, and Tim Berners-Lee. Web identity and discovery (webid 1.0). W3C editor's draft 05, W3C, 2014. https://www.w3.org/2005/Incubator/webid/spec/identity/.

[157] Kai Samelin, Henrich Christopher Pöhls, Arne Bilzhause, Joachim Posegga, and Hermann de Meer. On Structural Signatures for Tree Data Structures. In *Applied Cryptography and Network Security - 10th International Conference, ACNS 2012, Singapore, June 26-29, 2012. Proceedings*, pages 171–187, 2012.

[158] Hovav Shacham and Brent Waters. Compact Proofs of Retrievability. In Josef Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008. Proceedings*, volume 5350 of *Lecture Notes in Computer Science*, pages 90–107. Springer, 2008.

[159] Adi Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, 1979.

[160] Adi Shamir. Identity-Based Cryptosystems and Signature Schemes. In Blakley and Chaum [23], pages 47–53.

[161] Jun Shao, Zhenfu Cao, Xiaohui Liang, and Huang Lin. Proxy Re-Encryption with Keyword Search. *Inf. Sci.*, 180(13):2576–2587, 2010.

[162] Emily Shen, Elaine Shi, and Brent Waters. Predicate Privacy in Encryption Systems. In *Proceedings of the 6th Theory of Cryptography Conference on Theory of Cryptography*, TCC '09, pages 457–473, Berlin, Heidelberg, 2009. Springer-Verlag.

[163] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical Techniques for Searches on Encrypted Data. In *2000 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 14-17, 2000*, pages 44–55. IEEE Computer Society, 2000.

[164] Emil Stefanov and Elaine Shi. ObliviStore: High Performance Oblivious Cloud Storage. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 253–267. IEEE Computer Society, 2013.

[165] Ron Steinfeld, Laurence Bull, and Yuliang Zheng. Content Extraction Signatures. In Kwangjo Kim, editor, *Information Security and Cryptology - ICISC 2001, 4th International Conference Seoul, Korea, December 6-7, 2001, Proceedings*, volume 2288 of *Lecture Notes in Computer Science*, pages 285–304. Springer, 2001.

[166] Chul Sur, Chae Duk Jung, Youngho Park, and Kyung Hyune Rhee. Chosen-Ciphertext Secure Certificateless Proxy Re-Encryption. In Bart De Decker and Ingrid Schaumüller-Bichl, editors, *Communications and Multimedia Security, 11th IFIP TC 6/TC 11 International Conference, CMS 2010, Linz, Austria, May 31 - June 2, 2010. Proceedings*, volume 6109 of *Lecture Notes in Computer Science*, pages 214–232. Springer, 2010.

[167] Chul Sur, Youngho Park, Sang-Uk Shin, Kyung Hyune Rhee, and Changho Seo. Certificate-Based Proxy Re-encryption for Public Cloud Storage. In Leonard Barolli, Ilsun You, Fatos Xhafa, Fang-Yie Leu, and Hsing-Chung Chen, editors, *Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2013, Taichung, Taiwan, July 3-5, 2013*, pages 159–166. IEEE Computer Society, 2013.

[168] Qiang Tang. Type-Based Proxy Re-encryption and Its Construction. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *Progress in Cryptology - INDOCRYPT 2008, 9th International Conference on Cryptology in India, Kharagpur, India, December 14-17, 2008. Proceedings*, volume 5365 of *Lecture Notes in Computer Science*, pages 130–144. Springer, 2008.

[169] W3C. XML Advanced Electronic Signatures (XAdES), 2003. `https://www.w3.org/TR/XAdES/`, Accessed: 06.03.2017.

[170] Michael Walfish and Andrew J. Blumberg. Verifying Computations Without Reexecuting Them. *Commun. ACM*, 58(2):74–84, January 2015.

[171] Mark Watson. Web Cryptography API. W3C Recommendation, January 2017. `http://www.w3.org/TR/WebCryptoAPI/`.

[172] Jian Weng, Robert H. Deng, Xuhua Ding, Cheng-Kang Chu, and Junzuo Lai. Conditional Proxy Re-Encryption Secure Against Chosen-Ciphertext Attack. In Wanqing Li, Willy Susilo, Udaya Kiran Tupakula, Reihaneh Safavi-Naini, and Vijay Varadharajan, editors, *Proceedings of the 2009 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2009, Sydney, Australia, March 10-12, 2009*, pages 322–332. ACM, 2009.

[173] Peter Williams, Radu Sion, and Alin Tomescu. PrivateFS: A Parallel Oblivious File System. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 977–988. ACM, 2012.

[174] WSO2, Amila Jayaskerar Research Assistant at Indiana University Bloomington. Understanding XACML Policy Language (Extended Assertion Markup Language) – Part 1, 2011. http://wso2.com/library/articles/2011/10/understanding-xacml-policy-language-xacml-extended-assertion-markup-langue-part-1/. Accessed: June 2016.

[175] Keita Xagawa and Keisuke Tanaka. Proxy re-encryption based on learning with errors. In *Proceedings of the Symposium on Cryptography and Information Security*, 2010.

[176] Jia Xu and Ee-Chien Chang. Towards Efficient Proofs of Retrievability. In Heung Youl Youm and Yoojae Won, editors, *7th ACM Symposium on Information, Compuer and Communications Security, ASIACCS '12, Seoul, Korea, May 2-4, 2012*, pages 79–80. ACM, 2012.

[177] Prasad Yendluri, Ümit Yalçinalp, Frederick Hirsch, Maryann Hondo, Asir Vedamuthu, David Orchard, and Toufic Boubez. Web services policy 1.5 - framework. W3C recommendation, W3C, September 2007. `http://www.w3.org/TR/2007/REC-ws-policy-20070904`.

[178] Jing Zhao, Dengguo Feng, and Zhenfeng Zhang. Attribute-Based Conditional Proxy Re-Encryption with Chosen-Ciphertext Security. In *Proceedings of the Global Communications Conference, 2010. GLOBECOM 2010, 6-10 December 2010, Miami, Florida, USA*, pages 1–6. IEEE, 2010.

[179] Bernd Zwattendorfer, Thomas Zefferer, and Arne Tauber. The prevalence of SAML within the european union - an empirical study. In Karl-Heinz Krempels and José Cordeiro, editors, *WEBIST 2012 - Proceedings of the 8th International Conference on Web Information Systems and Technologies, Porto, Portugal, 18 - 21 April, 2012*, pages 571–576. SciTePress, 2012.

A Details for Core Cryptographic Technologies

In the following sections, we provide details for the cryptographic primitives evaluated and assessed in Section 3. In particular, regarding data sharing, we describe PRE and its many different aspects in Appendix A.1, followed by a description of FHE in Appendix A.2. For selective disclosure of authentic data, we describe malleable signature schemes and attribute-based credential systems in Appendix A.3 and Appendix A.4, respectively.

Besides motivation and high-level descriptions, we provide formal specifications of all interfaces and semi-formal definitions of the security properties provided by each technology. Furthermore, we provide references to the original literature for further reading.

A.1 Proxy Re-Encryption

Proxy re-encryption, introduced by Blaze, Bleumer, and Strauss [25], enables a proxy to transform a ciphertext for one entity to a ciphertext for another entity without revealing the underlying message to this proxy.

A.1.1 Classical Proxy Re-Encryption

This section first describes the operations involved in a proxy re-encryption scheme. Subsequently, the trust relationship regarding the proxy is explained. Finally, this section provides an example application that highlights new possibilities compared to traditional encryption schemes.

In contrast to traditional asymmetric encryption, proxy re-encryption introduces two new operations, namely re-encryption (ReEnc) and the generation of a re-encryption key (ReKeyGen). Figure 4 illustrates the relationships between those operations. The two standard operations for asymmetric encryption as well as the additional operations of a proxy re-encryption scheme are described below. An argument for the global parameters was omitted to keep the descriptions concise. Please note that we use A and B as subscript to denote Alice and Bob, respectively.

$\texttt{KeyGen}() \rightarrow (sk_A, pk_A)$: This operation generates and returns a key pair for user Alice, containing a private key sk_A and a public key pk_A.

$\texttt{ReKeyGen}(sk_A, sk_B, pk_B) \rightarrow (rk_{A\rightarrow B})$: This operation takes the private key sk_A of Alice and key material of Bob to create a re-encryption key $rk_{A\rightarrow B}$ that can be used to transform data encrypted for Alice to ciphertexts for Bob. Depending on the particular scheme, Bob might only have to provide his private key sk_B or public key pk_B. The output is the re-encryption key $rk_{A\rightarrow B}$.

$\texttt{Enc}(M, pk_A) \rightarrow (C_A)$: Given a public key pk_A and a message M, this operation encrypts the message into a ciphertext C_A for the owner of the key pair to which pk_A belongs. The output is the ciphertext C_A.

ReEnc$(C_A, rk_{A\to B}) \to (C_B)$**:** The parameters of this operation are a re-encryption key $rk_{A\to B}$ and a ciphertext C_A for Alice, where Alice also provided her private key in the re-encryption key generation. This operation transforms the ciphertext C_A into a ciphertext C_B for Bob, who presented the second key material in the re-encryption key generation. The ciphertext C_B is returned.

Dec$(C_A, sk_A) \to (M)$**:** On input of a secret key sk_A and a ciphertext C_A, this operation decrypts the ciphertext C_A into its plain message M if C_A was encrypted for the key pair (sk_A, pk_A). The message M is returned.

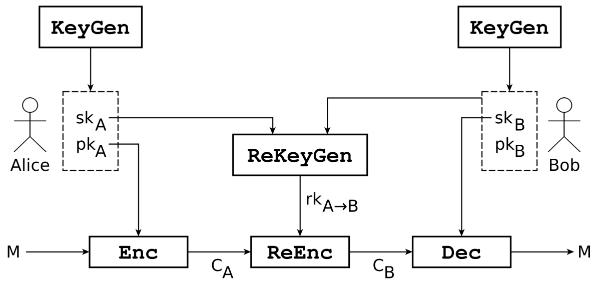


Figure 4: Proxy Re-Encryption

Proxy re-encryption leads to a system where the proxy does not have to be fully trusted and where the entity for which the message was originally encrypted stays in control of granting re-encryption rights. The proxy performs the re-encryption operation without seeing the underlying plaintext or having direct access to secret key material. Therefore, the proxy only requires limited trust. The entity for which the message was originally encrypted stays in control, as it has to provide key material to generate a re-encryption key.

The applications of proxy re-encryption are manifold. To outline the capabilities introduced by proxy re-encryption, we consider the email forwarding example [25]. In this example, Alice wants to forward her encrypted emails to Bob for the duration of her vacation. With traditional encryption, Alice would have to hand her private key to Bob in order for him to decrypt her mails. However, by making use of proxy re-encryption, Alice can generate a re-encryption key and hand this re-encryption key to the email server, which re-encrypts all of Alice's incoming email for Bob. As a result, Bob is able to decrypt Alice's emails with his own private key without requiring knowledge of Alice's key material.

A.1.2 Conditional Proxy Re-Encryption

Conditional proxy re-encryption [172] (C-PRE) and type-based proxy re-encryption [168] (TB-PRE) were independently introduced with the same motivation: to provide users with fine-grained control over the delegation of decryption rights. The previously discussed variants of proxy re-encryption enable the proxy to transform all of Alice's ciphertexts once she provides a re-encryption key. In contrast, C-PRE (as we will also call TB-PRE) allows the user to specify which of her ciphertexts can be transformed by a re-encryption key. This is realized by tagging a ciphertext with a condition during encryption and only allowing to translate this ciphertext with a re-encryption key that satisfies the condition.

The definition of a C-PRE scheme is presented below:

`KeyGen`$() \rightarrow (sk_A, pk_A)$: This operation generates and returns a key pair for user Alice, containing a private key sk_A and a public key pk_A.

`ReKeyGen`$(sk_A, w, pk_B) \rightarrow (rk_{A \xrightarrow{w} B})$: Compared to classical `ReKeyGen`, a condition keyword w has to be provided, which limits the re-encryption power of the resulting re-encryption key $rk_{A \xrightarrow{w} B}$. As we are mainly interested in non-interactive schemes, we do not consider Bob's private key sk_B in the algorithm definition.

`Enc`$(M, pk_A) \rightarrow (C_A)$: A ciphertext C_A that can be re-encrypted is conditioned with keyword w.

`ReEnc`$(C_A, rk_{A \xrightarrow{w} B}) \rightarrow (C_B)$: A ciphertext C_A conditioned with keyword w' can only be translated for another user to ciphertext C_B, if the re-encryption key $rk_{A \xrightarrow{w} B}$ was created for the same condition $w = w'$.

`Dec`$(C_B, sk_B) \rightarrow (M)$: On input of a secret key sk_A and a ciphertext C_A, this operation decrypts the ciphertext C_A into its underlying plain message M if C_A was encrypted for the user who owns the key pair that includes sk_A. The output is the plain message M.

Subsequent research by Zhao et al. [178] and Fang et al. [55] introduced C-PRE schemes with fine-grained access control. These schemes allow to define a policy over multiple conditions, which has to be satisfied to re-encrypt a ciphertext. Considering the urgent mail example, Alice could hand the mail gateway a re-encryption key, which allows to transform urgent or sales mails that were sent during her vacation in June.

In conclusion, C-PRE extends classical PRE by providing delegators with additional control over the delegation of decryption rights based on conditions. Classical PRE allows the proxy to re-encrypt any of the user's ciphertexts after a re-encryption key was issued. In contrast, with C-PRE, a ciphertext with a condition can only be transformed if the condition is satisfied by the re-encryption key. Also, trust assumptions and scalability of C-PRE are similar to classical PRE, as PKI is employed to solve the key distribution problem.

A.1.3 Certificate-Less Proxy Re-Encryption

Certificate-less proxy re-encryption (CL-PRE), introduced by Sur et al. [166], applies the concept of Certificate-less encryption (CLE) to PRE. Therefore, CLE and CL-PRE enjoy the same benefits in comparison to other types of encryption or proxy re-encryption.

The scheme definition by Sur et al. [166] was adapted to achieve a consistent notation. Note that the public parameters *params* generated in Setup were omitted in subsequent definitions. The definition of a CL-PRE scheme is presented below.

`Setup`$(k) \rightarrow (msk, params)$: The public parameters $params$ are established according to the security parameter k and a master secret key msk is generated

`Partial-Private-Key-Extract`$(msk, id_A) \rightarrow (d_A)$: The KGC generates a partial private key d_A for the user with identity id_A from the master secret key msk.

`Set-Secret-Value`$(id_A) \rightarrow (x_A)$: The user with identity id_A generates a secret value x_A for herself.

`KeyGen`$(d_A, x_A) \rightarrow (sk_A, pk_A)$: With her partial private key d_A and secret value x_A the user generates her key pair (sk_A, pk_A).

`ReKeyGen`$((id_A, pk_A, sk_A), (id_B, pk_B)) \rightarrow (rk_{A \rightarrow B})$: This operation takes the participants' identities and key material to output the re-encryption key $rk_{A \rightarrow B}$.

`Enc`$(M, id_A, pk_A) \rightarrow (C_A)$: Given a public key pk_A, an identity id_A and a message M, this operation encrypts the message into a ciphertext C_A for the owner of the key pair to which pk_A belongs. The output is the ciphertext C_A.

`ReEnc`$(C_A, rk_{A \rightarrow B}) \rightarrow (C_B)$: The parameters of this operation are a re-encryption key $rk_{A \rightarrow B}$ and a ciphertext C_A for Alice, where Alice also provided her private key in the re-encryption key generation. This operation transforms the ciphertext C_A into a ciphertext C_B for Bob, who presented the second key material in the re-encryption key generation. The ciphertext C_B is returned.

`Dec`$(C_B, sk_B) \rightarrow (M)$: On input of a secret key sk_A and a ciphertext C_A, this operation decrypts the ciphertext C_A into its underlying plain message M if C_A was encrypted for the user who owns the key pair that includes sk_A. The output is the plain message M.

Certificate-less encryption (CLE), proposed by Al-Riyami and Paterson [10], neither uses certificates with all the associated public key infrastructure nor suffers from the key escrow problem. CLE is based on IBE, however, the third party does not generate the user's secret key on its own. Instead, this third party, now called key generation center (KGC), issues a partial private key (PPK) from its master key for the user's identity. This PPK and a secret value (SV) chosen by the user represent the actual decryption key. For encryption, the sender requires the publicly known identity string and key material derived from the user's SV.

The benefits of using CLE are that PKI is not required, while it does not suffer the key escrow problem. Since CLE is based on IBE, it enjoys the same advantage of not needing certificates to ensure the authenticity of encryption key material, as the recipient is determined through her identity information. Therefore, without certificates, PKI is superfluous. In addition, the distributed generation of the decryption key material solves the key escrow problem. As both, the KGC as well as the user, contribute input for the actual decryption key material, the KGC does not have enough information to decrypt on its own. Hence, CLE does not require as much trust in a third party as IBE.

Support for revocation can also be achieved similar to IBE by issuing new key material for each time period. Again, the identity information is extended with a validity time period. This information is used by the KGC to issue new partial private keys. To revoke the association between a user and key material from an KGC, the KGC is advised to stop issuing new partial private keys.

In comparison to PKI with long-lived certificates, CLE schemes, like IBE schemes, offer favorable scalability, as only one re-issuing operation of key material is required per time period instead of one certificate status verification per encryption process.

However, CLE also suffers from the key distribution problem. Since the KGC generates the partial private keys, it also has to securely distribute that key material.

In conclusion, certificate-less schemes are very similar to identity-based schemes, except they have lower trust requirements. CLE schemes do not rely on PKI to ensure the correct recipient during encryption. Additionally, revocation can be supported by periodically re-issuing key material with a short lifespan. Therefore, CLE schemes, like IBE schemes, offer favorable scalability in comparison to classical schemes relying on PKI. Furthermore, CLE does not suffer from the key-escrow problem, as the key generation is distributed between KGC and user. Hence, CLE does not require as much trust in a third party as IBE. However, the key distribution problem still remains.

A.1.4 Certificate-Based Proxy Re-Encryption

Certificate-based proxy re-encryption (CB-PRE), proposed by Sur et al. [167], applies the concept of CBE to PRE. Therefore, CBE and CB-PRE enjoy the same benefits in comparison to other types of encryption or proxy re-encryption.

Certificate-based encryption (CBE), introduced by Gentry [66], is similar to CLE, as it combines classical and identity-based encryption, while preserving their attractive features. In contrast to CLE, CBE uses certificates with a simplified PKI, which does not require the inconvenient checks for revocation status. A certification authority (CA) issues these certificates for the public key of a user-generated key pair, thereby binding the user's identity to the key material. Data is encrypted with the public key for the recipient's identity and the current time period. Decryption requires not only the private key but also a valid certificate for the time period.

The scheme definition by Sur et al. [167] was adapted to achieve a consistent notation. Note that the public parameters *params* generated in Setup were omitted in subsequent definitions. The definition of a CB-PRE scheme is presented below:

`Setup`$(k) \rightarrow (msk, params)$: The public parameters *params* are established according to the security parameter k and a master secret key msk is generated.

`KeyGen`$() \rightarrow (sk_A, pk_A)$: This operation generates and returns a key pair (sk_A, pk_A) for user Alice, containing a private key sk_A and a public key pk_A.

`Certify`$(msk, \tau, id_A, pk_A) \rightarrow (cert'_{A,\tau})$: The CA uses the master secret key msk to certify that the public key pk_A belongs to a user with identity id_A for a limited time period τ, resulting in the certificate $cert'_{A,\tau}$.

`Consolidate`$(\tau, cert'_{A,\tau}, cert_{A,\tau-1}) \rightarrow (cert_{A,\tau})$: This function consolidates previous and the current certificate, returning the certificate $cert_{A,\tau}$.

`ReKeyGen`$((cert_{A,\tau}, sk_A), (id_B, pk_B)) \rightarrow (rk_{A\rightarrow B})$: Compared to classical `ReKeyGen`, the delegator's certificate $cert_{A,\tau}$ and the delegatee's identity id_B are also needed to create a re-encryption key $rk_{A\rightarrow B}$.

`Enc`$(M, pk_A, \tau, id_A) \rightarrow (C_A)$: In comparison to classical `Enc`, a sender also specifies the recipient's identity id_A and the current time period τ to encrypt a message M.

`ReEnc`$(C_A, rk_{A\rightarrow B}) \rightarrow (C_B)$: The parameters of this operation are a re-encryption key $rk_{A\rightarrow B}$ and a ciphertext C_A for Alice, where Alice also provided her private key in the re-encryption key generation. This operation transforms the ciphertext C_A into a ciphertext C_B for Bob, who presented the second key material in the re-encryption key generation. The ciphertext C_B is returned.

`Dec`$(C_B, sk_B, cert_{B,\tau}) \rightarrow (M)$: To decrypt a ciphertext C_B, not only the recipient's secret key sk_B, but also her valid certificate $cert_{B,\tau}$ is required.

By introducing identity-based concepts, CBE enables implicit certification, which requires a recipient to be certified in order to decrypt. During encryption, the sender specifies the identity of the receiver and uses the presumably associated public key. In order to decrypt such a ciphertext, the receiver not only requires her private key but also a valid certificate linking the used identity to the used public key. Consequently, the sender does not have to check the authenticity of the used public key, as the receiver is implicitly required to be in possession of an appropriate certificate.

For revocation, CBE only requires a simplified PKI without queries to a third party for the revocation status. Implicit certification requires recipients to possess currently valid certificates. As these certificates expire after a short time, the CA frequently has to re-certify the association between the user's identity and key material. To revoke this association, the CA is instructed to stop re-certifying the user's key material. However, a previously issued certificate stays valid for the remainder of its short lifespan.

In conclusion, certificate-based schemes provide greater scalability due to a simplified PKI in comparison to classical schemes, while requiring lower trust assumptions than identity-based schemes. CBE supports revocation based on certificates with a short lifespan, which are re-certified regularly, but do not have to be checked for validity. IBE and CLE follow a similar approach, where new key material is issued periodically. Compared to classical schemes with extensive PKI, these solutions provide greater scalability, as only one re-certify or issue operation has to be performed per time period, instead of one status check per encryption.

A.1.5 Proxy Re-Encryption with Keyword Search

Public-key encryption with keyword search (PEKS), introduced by Boneh et al. [27], enables users to search for a keyword in encrypted data stored by another party. To realize this, the sender tags data with a keyword during encryption, before handing the ciphertext to the recipient's storage provider. In order to search, the recipient generates a trapdoor for a keyword, hands this trapdoor to her storage provider, which then tests for matching ciphertexts.

Proxy re-encryption with keyword search (PRES), presented by Shao et al. [161], allows to delegate both decryption and search rights to another party. As in PEKS, data is encrypted for a recipient, Alice, and tagged with a keyword. In addition, a proxy can re-encrypt Alice's ciphertext for another user, called Bob. Then, Bob is able to generate trapdoors that can be used to search on the re-encrypted data as well as to decrypt these ciphertexts. The PRES definition of Shao et al. was adapted to achieve a consistent notation. Note that the public parameters params generated in Setup were omitted in subsequent definitions. The definition of a PRES scheme is presented below:

`KeyGen`$() \to (sk_A, pk_A)$: This operation generates and returns a key pair (sk_A, pk_A) for user Alice, containing a private key sk_A and a public key pk_A.

`ReKeyGen`$(sk_A, sk_B) \to (rk_{A\to B})$: This operation takes the private key sk_A of Alice and key material of Bob to create a re-encryption key $rk_{A\to B}$ that can be used to transform data encrypted for Alice to ciphertexts for Bob. The first schemes require both parties to provide private key material. The output is the re-encryption key $rk_{A\to B}$.

`Enc`$(M, pk_A, w) \to (C_{A,w})$: During encryption with public key pk_A, messages M are associated with a keyword w, resulting in ciphertext $C_{A,w}$.

`ReEnc`$(C_{A,w}, rk_{A\to B}) \to (C_{B,w})$: Like classical `ReEnc`, but search rights are also delegated.

`Dec`$(C_{B,w}, sk_B) \to (M)$: On input of a secret key sk_B and a ciphertext $C_{B,w}$, this operation decrypts the ciphertext C_B into its underlying plain message M if C_B was encrypted for the user who owns the key pair that includes sk_B. The output is the plain message M.

`Trapdoor`$(sk_B, w) \to (T_w)$: From a private key sk_B an encrypted query T_w for keyword w can be generated.

`Test`$(C_{B,w}, pk_B, T_w) \to (yes \text{ or } no)$: With a trapdoor T_w it is possible to determine if a ciphertext C_B has been associated with keyword w.

For example, PRES can be used to only retrieve urgent mail that was delegated by another user. As Alice goes on vacation, she wants Bob to be able to read her mails, which are encrypted and tagged with keywords, such as urgent. Therefore, Alice hands a re-encryption key to the mail gateway. From the huge bulk of Alice's mail, Bob only is interested in her urgent mail. Hence, Bob generates a trapdoor for the keyword urgent, which is then used by the gateway to test the individual mails. Bob only receives matching ciphertexts. In conclusion, PRES extends classical PRE with search functionality. Users can delegate both decryption and search rights to others. As PRES is based on classical PRE, it uses PKI to solve the key distribution problem and has similar trust requirements and scalability characteristics.

A.1.6 Properties of Proxy Re-Encryption Schemes

This section explains the most important properties of proxy re-encryption schemes. These properties will subsequently be the basis for the evaluation of different proxy re-encryption schemes.

Unidirectional, Bidirectional [101]: A proxy re-encryption scheme is *unidirectional* if a delegation of decryption rights from Alice to Bob does not also allow re-encryption from Bob to Alice. If such a delegation also enables re-encryption in the opposite direction, then the scheme is *bidirectional.*

Single-Hop, Multi-Hop [41]: A proxy re-encryption scheme is *single-hop* if an initial ciphertext can only be transformed into a re-encrypted ciphertext, but further re-encryption of that already re-encrypted ciphertext is not possible. In contrast, a proxy re-encryption algorithm is *multi-hop* if an already re-encrypted ciphertext can be further re-encrypted.

Collusion-Safeness (Master Key Security) [17]: A proxy re-encryption scheme is *collusion-safe* if the proxy and delegatee are not able to learn the delegator's private key. Even though they collude by sharing key material

Interactive, Non-Interactive [17]: A proxy re-encryption scheme is *non-interactive* if the delegator is able to generate a re-encryption key without having to interact with the delegatee or a trusted third party. In such a scheme, only the delegator's private key material can be involved in the generation of a re-encryption key. If key material of both participants is required, the scheme is *interactive*, as these participants have to interact to generate the re-encryption key, for example as shown by Canetti and Hohenberger [41].

Key-Private [12]: A proxy re-encryption scheme is *key-private* if an adversary is not able to identify any participant of a re-encryption key, even when given all public keys and extensive interaction abilities. These abilities include obtaining the re-encryption of any chosen ciphertext as well as getting any re-encryption key except for the re-encryption key being analyzed.

Lattice-Based [175]: Lattice-based algorithms rely on lattices as their underlying mathematical principle. Lattices are of particular interest, since no algorithm has yet been discovered that could harness the benefits of quantum computers in the cryptographic analysis of lattices. Therefore, lattice-based algorithms might still stay secure once quantum computers become available.

Security Model: Since proxy re-encryption is an extension of public key encryption, the same security notions serve as a basis to model a scheme's security. However, these models are extended by giving the adversary access to re-encryption keys and, in the case of CCA-security, to a re-encryption oracle. Of particular interest are adapted versions of indistinguishability under chosen-plaintext attacks (IND-CPA [17]), under (adaptive) chosen-ciphertext attacks (IND-CCA1/IND-CCA2 [41]), or with harmless mangling (IND-RCCA [119]). In their paper, Nunez et al. [128] presented a detailed description of the differences between the individual notions. Note, however, that these security notions are mainly applicable to classical proxy re-encryption. Other types of proxy re-encryption might require adapted security models to accommodate their added functionality.

Models of Computation [21]: The models of computation specify in which setting the security of schemes are proven. Two models are relevant to this survey. In the standard model, the adversary is only limited by the available amount of time and computational power. The random oracle model simplifies proofs by introducing genuinely random functions.

A.2 Fully Homomorphic Encryption

The question of computing on encrypted data was raised in a paper by Rivest, Adleman, and Dertouzos [152] in 1978 in which the encryption functions fulfilling this task were dubbed "privacy homomorphisms." A common problem motivation is the following. Consider a database of sensitive (e.g., salary) data. Due to the sensitivity of the data, one might choose an appropriate data encryption to ensure confidentiality. In this sequel, one is usually not able to perform certain arbitrary tasks or computations on the encrypted data (without the explicit access to the key material) that violates certain security properties. Of course, a user can trivially download his encrypted data, decrypt it, performing a certain task on the decrypted data, encrypt the altered data again, and store it back to the database. However, note that this is only possible if one has explicitly access to the (secret) key material under which the data was encrypted. To circumvent the explicit knowledge of the (secret) key material to performing a certain task, one might try to construct specific encryption algorithms which enable the execution of an at least limited (i.e., not arbitrary) task without having access to that private key material. One example is the widely known text-book RSA public-key functionality:

- Consider a public key (e, N) and a secret key (d, N) of the RSA system. The public key is used to encrypt a message, the secret key is used for decryption. Encryption works as follows: $c = m^e \bmod N$ while decryption using the secret key reverts the ciphertext c to m again as $m = c^d \bmod N = (m^e)^d \bmod N$. (This works since $1 = ed \bmod \varphi(N)$ holds where φ is the Euler totient function.)

It is easy to see that the RSA function allows for performing a certain task (namely multiplication of encrypted messages) under the public key (e, N) on the ciphertexts without having explicitly access to the secret key (d,N):

- Consider that the ciphertext of a message m_1 is constructed as $c_1 = {m_1}^e \bmod N$.
- A second ciphertext of a message m_2 yields $c_2 = {m_2}^e \bmod N$.
- Without knowing the secret (decryption) key, one can perform the multiplication of encrypted messages as follows:
  - $c_{12} = c_1 * c_2 \bmod N = {m_1}^e * {m_2}^e \bmod N = (m_1 * m_2)^e \bmod N$
- Decryption of c_{12} yields $m_1 * m_2 = ((m_1 * m_2)^e)^d \bmod N$ as desired.

Encryption schemes that allow for such limited operations on the ciphertexts are called (partially) homomorphic encryption scheme. In the cryptographic literature exist a way more examples of such schemes, e.g., the well-known schemes of ElGamal [64], Paillier [142], etc. Note that some of these scheme do not necessarily allow for the task of multiplication of the encrypted messages; indeed, some of them allow for addition. However, note that any of these schemes above allow only to perform one specific task, i.e., the multiplication or addition of encrypted messages.

A long open cryptographic question was:

How to perform arbitrary tasks (i.e., function evaluations) on encrypted data publicly?

In 2009, Craig Gentry gave the first "practical" solution of such a cryptographic scheme, dubbed fully homomorphic encryption (FHE) system [67]. The scheme is based in hard problems in ideal lattices. Since then, a lot of progress has been made. Unfortunately, many schemes are still far away from being deployed in realistic and plausible real-world scenarios. More formally, a FHE scheme for an arbitrary evaluation function f consists of four efficient probabilistic algorithms (Gen, Enc, Dec, $Eval$) as follows:

$\texttt{Gen}(1^k) \rightarrow (pk, sk)$: This operation generates and returns a key pair (pk, sk) for the security parameter k as input in unary representation.

$\texttt{Enc}(pk, M) \rightarrow C$: This operation encrypt the message M with a private key pk and outputs the ciphertext C.

$\texttt{Dec}(sk, C) \rightarrow M$: This operation decrypts the given input ciphertext with the secret key sk and outputs the message M.

$\texttt{Eval}(pk, C) \rightarrow C'$: This operation evaluates the given input public key pk and ciphertext C and outputs a ciphertext C'.

For correctness, for $C \leftarrow Enc(pk, M)$ as computed above, we in particular require that for any evaluation of a ciphertext $C' \leftarrow Eval(pk, C)$, it holds that $Dec(sk, C') = f(M)$ for an arbitrary computable function f. (Note that the function f is specified by the system.)

A.3 Malleable Signatures

Digital signatures are utilized in different fields such as the government or corporate environment. The basic idea of digital signatures is to maintain data integrity and authenticity as well as ensure non-repudiation. For example, after signing a document it is not possible to change its content and still have a valid signature.

If a message is digital signed, the signature can be utilized to check the message source (authenticity) and also their non-repudiation. Non-repudiation describes that the author of a message cannot dispute that he/she was not the originator of the signed message. The digital signature can moreover be used to proof the integrity of the message. Conventional digital signatures have been adopted for the usage with the most common data formats. Common signature formats are PDF Advanced Electronic Signature (PAdES) [52], XML Advanced Electronic Signature (XAdES) [169] and CMS Advanced Electronic Signature where CMS [91] stands for Cryptographic Message Syntax.

Besides the conventional digital signatures there are the so-called malleable signatures. The general idea of malleable signatures is to be able to change the data content of a signed document, while still maintaining the validity of the signature. This does not mean that it is possible to change random parts in a message. The changeable parts as well as legal transformations have to be predefined. It can be distinguished between three main categories of malleable signatures, namely redactable signatures, blank digital signatures and sanitizable signatures. These three categories are described in further detail in the following specification section.

Malleable signatures are following two main principles, beside the properties of conventional digital signatures, which are first unforgeability and second editability. Unforgeability describes a property which ensures that the signature to verify is not a fake digital signature. Also, it describes that it should not be possible to tamper the digital signature. The second principle is the principle which makes malleable signatures different from conventional digital signatures. Basically, it should be possible to edit a message in a limited and predefined manner while maintaining a valid signature.

A.3.1 Redactable Signatures

Redactable signatures were introduced in 2001 by Steinfeld et al. [165] utilizing redactable signatures which allows the originator to redact parts of a signed message while the signature stays valid on the remaining parts. After the message has been transferred to the receiver she is now able to read the message except the redacted parts. Moreover, the receiver is also able to check with utilizing the signature both the message's source as well as the integrity of the message. The advantage in using redactable signatures is that message parts, which the originator doesn't want to show, can be blacked out.

Figure 5 illustrates the basic idea of redactable signatures. This basic idea is based on splitting the message to sign in different parts the so-called message blocks. Each of this message blocks are going to be used for creating the signature. The hash value is calculated for each of these message blocks. The resulting hash values are utilized to generate the signature of the document. The sign operation requires a key pair related to the signer. The private key of the signer is utilized to create the signature of the message including all message blocks. If some message blocks should not be visible for the receiver, they can be redacted. This is the case when the originator doesn't want the receiver to get all information from the message. This occurs if a message contains necessary information but also secret information which the originator does not want to share. To redact a block of a message, the redact operation has to be performed. The redact operation takes as input parameters the message, the related signature, the public key of the signer and the so-called modification instruction. The modification instruction describes which message blocks have to be redacted. This could simply be a list of indexes describing which message blocks should be redacted. In the example the message block 3 is marked to be redacted. According to this, the hash value of message part 3 has to be used and the actual message block 3 has to be redacted (removed). The receiver receives depending on the used scheme either the modified message together with the hash of the modified part and the signature or a modified message and the updated signature.

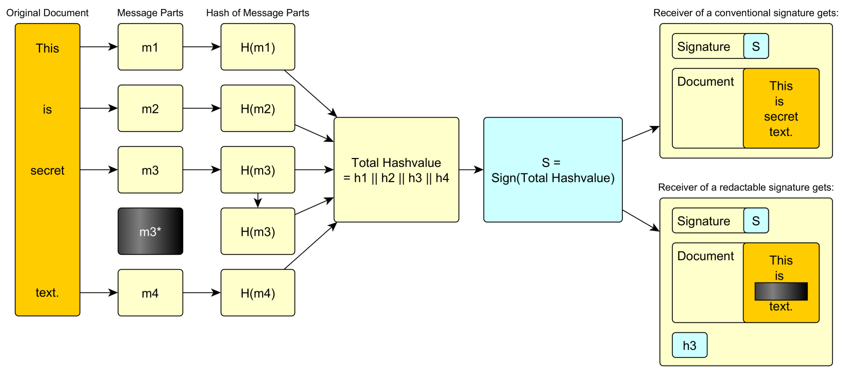


Figure 5: Redactable Signature Concept

Redactable signatures can have different properties depending on the scheme utilized to redact a message. Possible properties have been identified and listed with a detailed description below.

- **Unforgeability:** This property ensures that no one other than the signer can generate a valid signature. Therefore, a valid signature ensures the message's authenticity and if the message has been redacted that the redaction has been performed correctly.
- **Privacy:** The privacy property ensures that the message receiver is not able to recover any redacted information.
- **Transparency:** This property ensures that it is not possible for a third party to decide who the redactor of the message was.

The main redactable signature operations are described as follows:

- **Key Generation:** $KeyGen(\lambda) \rightarrow (sk, pk)$

  The $KeyGen$ operation is an algorithm to generate a key pair consisting of the private (secret) key sk and the corresponding public key pk with utilizing a security parameter λ as input. The key pair is generated for the signer.
- **sign:** $Sign(sk, msg) \rightarrow (\sigma)$

  The $Sign$ operation takes as input a message msg and the private key sk. Each message block of the message is used to create the signature σ, which is the operation's output.
- **Redact:** $Redact(msg, \sigma, pk, mod) \rightarrow (msg', \sigma')$

The redact operation *Redact* takes as input the message *msg*, the signature σ, the public key *pk* of the signer and an instruction *mod* for modifying the message. A modification instruction, basically, describes which message blocks from the message have to be redacted (removed). After the message has been modified according to the instruction the algorithm updates the signature accordingly. The modified message together with the modified signature is returned.

- **Verification:** $Verify(msg', \sigma', pk) \rightarrow val$

  The verification operation $Verify$ takes as input the message *msg*, the signature σ and the public key *pk* of the signer. Depending on the scheme it can also require the hash value of the redacted message block. The return value *val* of this algorithm shows if the signature for the given message is valid.

A.3.2 Blank Digital Signatures

Blank digital signatures were introduced by Hanser and Slamanig in 2013 [81]. This type of digital signatures is part of the editable signatures. The general idea of blank digital signatures is that the originator signs a document template. This template consists of parts for which a value is either fixed of can be chosen from a predefined set.

Figure 6 illustrates the process flow of blank digital signatures. First the originator creates the message template with its predefined choices. After successful creation the originator signs the template. Next, the originator hands the template and appropriate permissions to the instantiator. This instantiator creates an instance of the document. During the instantiation procedure, the instantiator selects her choices contained in the message template. The result of this instantiation procedure is a message instance. This message instance is signed by the instantiator and send to the recipient. The recipient is now able to verify that the received instance was derived from a valid template.

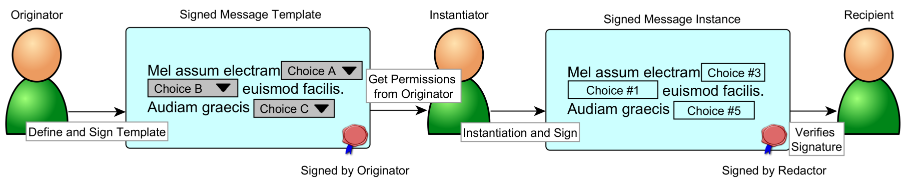


Figure 6: Blank Digital Signature Concept [81]

Some properties of blank digital signatures are listed as below [81]:

- **Unforgeability:** In the context of blank digital signatures, unforgeability describes that nobody should be able to forge the template signature or the message instance signature without having knowledge about any secret information. Secret information are describing the private keys of the originator and the instantiator as well as the secret signing key of the instantiator.

- **Immutability:** This property ensures that the instantiator (proxy) can only compute instantiations of message templates which are explicitly intended by the originator.
- **Privacy:** The privacy property ensures that no entity without knowing the private keys of the originator sk_{Orig}, the private key of the instantiator $skInst$ and the secret template signing key sk^{τ}_{Inst} should be able to determine elements of the template message, which haven't been revealed through instantiations yet.

Blank digital signatures are utilizing different operations. The operations are listed and described as follows.

- **Key Generation:** $KeyGen(\lambda, size_t) \rightarrow (sk, pk)$

  The $KeyGen$ operation generates a key pair consisting of the private (secret) key sk and the corresponding public key pk and requires two input parameter. The first parameter λ denotes a security parameter and the second parameter $size_t$ describes the template size.

- **Sign Template:** $Sign(\tau, sk_{Orig}, pk_{Inst}) \rightarrow (\sigma_\tau, sk^{\tau}_{Inst})$

  The $Sign$ operation takes as input the template message τ, the private key of the originator sk_{Orig} and the public key of the instantiator pk_{Inst}. The operation returns the signature of the template σ_τ as well as the secret signing key for the instantiator sk^{τ}_{Inst}.

- **Verify Template:** $VerifyTemp(\tau, \sigma_\tau, pk_{Orig}, pk_{Inst}, sk^{\tau}_{Inst}) \rightarrow val$

  The verify template operation $VerifyTemp$ takes as input the template message τ together with the related template signature σ_τ, the public keys from the originator pk_{Orig} and the instantiator pk_{Inst}, and the secret signing key of the instantiator sk^{τ}_{Inst} and verifies the signature of the template message. The return value val shows if either the signature is valid or not.

- **Instantiate:** $Instantiate(\tau, \sigma_\tau, M, sk^{\tau}_{Inst}, sk_{Inst}) \rightarrow \sigma_M$

  The instantiation operation $Instantiate$ takes as input the message template τ with the related template signature σ_τ, a message instance M, the secret template signing key of the instantiator sk^{τ}_{Inst} and also the private key of the instantiator sk_{Inst}. The operation outputs the signature σ_M for the corresponding message instance M.

- **Verify Instance:** $VerifyInst(M, \sigma_M, pk_{Orig}, pk_{Inst}) \rightarrow val$

  The verify instance operation $VerifyInst$ is validating the signature σ_M of an instantiated message. It takes as input the message instance M together with the message instance signature σ_M and the public keys pk_{Orig} and the instantiator pk_{Inst}. The return value val describes if the signature is valid for the given message instance.

A.4 Anonymous Credentials

Attribute-Based Credentials (ABCs) technologies have been designed to enhance users' privacy, and for several years have been investigated as part of anonymous credential systems and group signatures. ABCs are issued like ordinary cryptographic credentials (e.g., X.509 credentials) using a digital (secret) signature key; basically a PKI with privacy enhancing features. In ABCs, the main enhancing feature is that credential's attributes could be transformed into unlinkable and non-transferable presentation tokens[8] able to protect the holder's privacy, while offering the same level of security.

Overview

A typical ABC system includes the following entities and interactions:

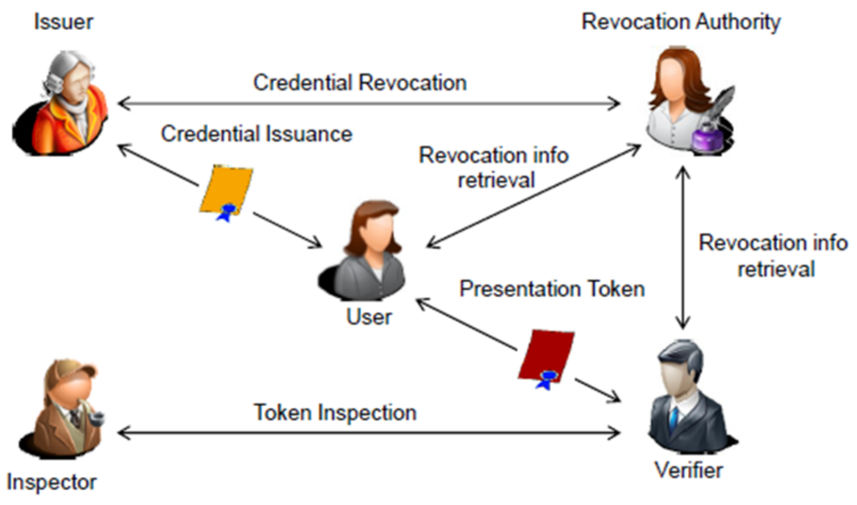


Figure 7: ABC System: Entities and Interactions [150]

- **Issuer:** it is an infrastructure-based (trusted) identity provider also known as an attribute authority; this entity or organization is responsible of issuing credentials. It is also responsible for vouching for the correctness of the information contained in the credentials.
- **Users:** entities to which the identity providers (issuers) will issue the (ABC) credentials. Such credentials are used to assert claims about user's identity to service providers.
- **Verifier:** any relying party willing to protect access to resources, information or services. Revocation Authority: this (not a mandatory) entity is responsible for revoking issued credentials and preventing their further usage.
- **Inspector:** (not mandatory) entity which consists of a trusted authority comprised of either a single entity or by a multi-party cooperation. The inspector's role is to de-anonymize the user under specific situations (e.g., misuse or liability). Ideally, the capability of inspection should be done in a distributed fashion, and it must be compliant

[8] A presentation token is a digitally signed container of attribute information [143]

with a presentation policy that specifies which information should be recoverable by an inspector and under which circumstances.

Operations

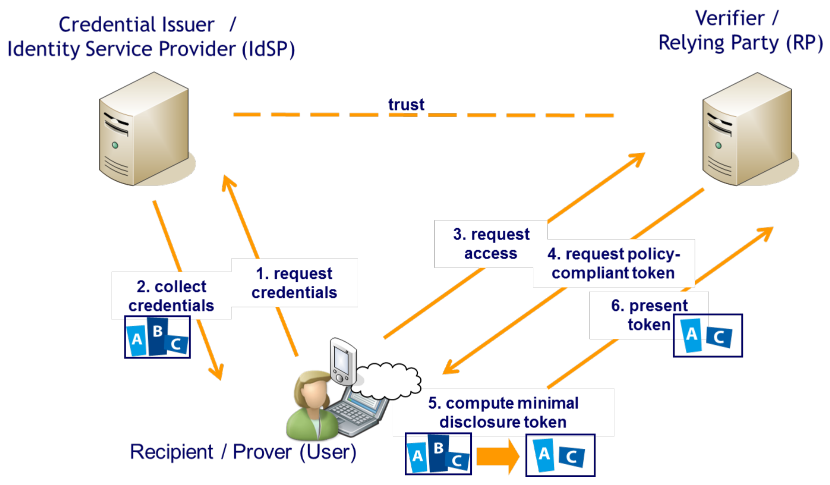


Figure 8: ABC System: Basic Operations [150]

Figure 8 illustrates the following operations:

- **Issuance:** the credential issuance is a multi-round interactive protocol between the issuer and the user. The Issuer defines the issuance requirements that must be met by the User in order to get a credential.
- **Presentation:** In a presentation protocol the Verifier sends the User a presentation policy, which defines, among other things, what credentials are accepted and what kind of proof is required from the User. Presentation uses commitment schemes to enable users to commit a certain message without showing it to the Verifier, and zero-knowledge proofs, to enable a User to prove to a Verifier that they know the secret behind the committed message (credential).
- **Revocation:** The revocation mechanism requires that both users and verifiers have the most recent revocation information from the corresponding revocation authority, which is the entity responsible for revoking credentials and for making available updated revocation information. Users are responsible for updating their non-revocation evidence.
- **Inspection:** presentation tokens are by default fully anonymous. Nevertheless, full anonymity could lead to misuse, abuse or even fraud. Thus, inspection allows the user to encrypt one or more attribute values under the public key of a trusted inspector under specific inspection grounds.

Properties

Attribute-based credentials are a combination of several cryptographic building blocks, which include signatures, pseudonyms, zero-knowledge proofs, encryption and revocation mechanisms. Together, those building blocks enable a number of security and privacy features that guarantee that 1) a user cannot create valid presentation tokens without having the proper underlying credentials and keys; and 2) presentations tokens do not reveal more information than what is intentionally disclosed by the user, both security and privacy features are achieved by the following properties.

- **Multi-show unlinkability:** a credential can be used multiple times without the resulting evidence becoming linkable.
- **Selective disclosure of attributes:** allows users to prove only a subset of attributes to a verifier.
- **Predicate proofs:** consist of statements that allow to prove a property of an attribute without disclosing its actual value, example of these statements are the logical operators.
- **Proof of holdership (ownership):** a cryptographic evidence for proving ownership or possession of a credential without disclosing the credential.
- **Non-transferability:** key binding can be used to bind one or more credentials of a user to the same secret and discourage users to perform credential pooling.
- **Unforgeability:** users cannot create new credentials or change attribute values in the credentials they obtained by from an issuer
- **Scope-exclusive pseudonyms:** a certified pseudonym unique for a specific scope and secret key, i.e. a single pseudonym can be created for each credential.
- **Carry-over attributes:** it relies on the assumption that the user already possesses a credential, from which a given attribute can be included into the new credential without disclosing the attribute value to the Issuer.
- **Cross-credential proofs:** allows users to prove relations between attributes from two or more credentials without revealing them to the verifier.
- **De-anonymization:** it is an optional feature that allows an authority to reveal the identity of a user in cases of accountability and non-repudiation.
- **Revocation:** in case of misuse, it allows the revocation of issued credentials to (misbehaving) users. Thus, revoked credentials cannot longer be used to generate presentation tokens.

A.4.1 Idemix

There exist a number of implementations of attribute based credentials, which are based on different cryptographic primitives. Among the most successful technologies the Identity Mixer (Idemix) - an anonymous credential system developed by IBM Research- enables strong authentication while at the same time provides relevant privacy properties. The basic Idemix ABC System comprises of protocols for a user to join a system, register with an issuing organization and obtain multi-show credentials to further present them to a verifying organization; thus the entities in the system are users who obtain and show credentials, issuing organizations and organization verifying credentials (cf. Figure 9). There exist an additional (optional) organization known as de-anonymization organization.

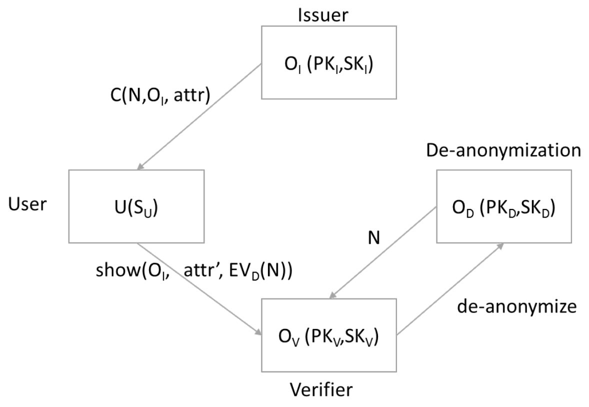


Figure 9: Idemix

A User U requests a credential C containing certain attributes ($attr$) to an issuing organization O_I; U establishes a pseudonym with O_I. O_I produces C by signing a statement containing $attr$ and N. Using a zero-knowledge proof, U presents (shows) the credential to a verifying organization O_V, Convincing O_V of possessing a signature generated by O_I on a statement containing $attr$ and N, and knowing the master secret key S_U associated to N. The de-anonymization mechanism requires the existence of a de-anonymization organization O_D. U encrypts N with O_D's PK_D. The encryption is verifiable $EV_D(N)$, i.e. O_V has a proof that O_D can decrypt and reveal N from O_V's show protocol transcript. [39]

Idemix provides a number of privacy properties described below [22]:

- **Proof of ownership:** idemix allows a user to proof possession of the credential without involving the IdP.
- **Selective disclosure of attributes:** Using verifiable encryption Users can choose which attributes to disclose or what to prove about them.
- **Unlinkable multi-use:** Using the zero-knowledge property of the proof provides unlinkability of different showings of a credential.

- **Non-transferability:** the user's master secret is linked to all user's credentials, sharing a credential implies sharing one's master secret.
- **Predicate proofs:** it supports proof of predicates over attributes, for instance proving equality among attributes.
- **Cross-credential proofs:** users can prove that attributes encoded in different credentials fulfill a specified predicate.

B Details for Additional Cryptographic Technologies

B.1 Authentication

In this subsection, we provide more details on the password-based cryptographic technology TPASS; you can find the fact sheets in Section 4.1.

B.1.1 TPASS

Access to potentially sensitive data is currently often protected by password based authentication schemes. However, such schemes are inherently vulnerable to, e.g., dictionary attacks. One mitigation option is to request users to use long and truly random passwords. Alternatively, if the password verification is done by an online server, one can let the server throttle verification attempts after a certain number of subsequent errors. However, this approach still suffers from the vulnerability that the password can be guessed offline if the server gets compromised.

A natural solution to this problem is provided by TPASS (threshold password-authenticated secret sharing). In such schemes, the verification process is spread across multiple servers. If the correct password is entered, the stored information (e.g., a strong cryptographic key for further applications) is revealed to the user. However, if a wrong password is entered, the information does not get disclosed. The password and the stored data now remain secure also against offline attacks as long as no more than a predefined threshold of servers gets compromised.

Despite this confidentiality guarantees, the first TPASS protocols were still not fully secure in the real world, as they did not provide any guarantees if a user accidentally tried to authenticate to a malicious set of servers, which is a common situation in phishing attacks, were a user is requested to enter his password, e.g., on a fake website. More recent TPASS solutions [33] do not suffer from this problem anymore. That is, they even if the user is persuaded to authenticated to fake servers, an attacker is not able to learn neither the password attempt nor the stored data. Note here that even if more than the threshold of servers gets compromised, they still have to mount an offline attack against the password, i.e., they do not learn the password in the plain. Furthermore, as long as no more than the threshold of original servers gets compromised, it is also infeasible for an attacker to change the data that was stored by the user. That is, as long as sufficiently many servers remain honest, the user is guaranteed that the downloaded data corresponds to the data that he originally stored spread across the servers – even if all the servers he tries to authenticate to are fake.

On a very high level, TPASS protocols have the following two interfaces:

- $Store(pw, d, (S_i)_{i=1}^n, t)$: On input a password pw, the data d to be stored on the Server S_i, and a threshold t, this algorithm outputs partial data $(\sigma_i)_{i=1}^n$ to be sent to the different servers.
- $U.Retrieve(pw', (S_i')_{i=1}^t) \leftrightarrow \{S.Retrieve(\sigma_i')\}_{i=1}^t$: This is an interactive protocol between the user and a set of t servers. The user takes as input a password attempt pw' and the

identities of the servers, while each server takes as input his partial data. At the end of the interaction, the user either obtains data d' or an error symbol $\perp$.

Besides `correctness`, requiring that an honest user interacting with honest servers is always able to obtain his original data, TPASS schemes need to guarantee the security features discussed above.

As a simplified variant of TPASS, `distributed password verification protocols` can be used to verify the correctness of a password attempt in a way that guarantees security against offline attacks as long as not too many servers get compromised. The difference to TPASS protocols is that in distributed password verification protocols, the user does neither store nor retrieve any associated data, but the servers merely learn whether the password attempt was correct, and can then decide, e.g., whether or not to grant the user access to some online resource.

B.2 Access to Encrypted Data

In this subsection, we provide more details on Search on Encrypted Data, Private Information Retrieval, Proofs of Retrievability, and Provable Data Possession; the fact sheets can be found in Section 4.2.

B.2.1 Search on Encrypted Data

Consider a database stored by a potentially untrusted third party. It is natural that a user—who wants to store data in a database—encrypts this data under his (public) key before uploading it. Naively, as with homomorphic encryption, computing certain functions or tasks on encrypted data is a non-trivial task without the explicit access to the corresponding key material. Here, we consider searching on encrypted data. One of the early works on that are given by Song, Wagner, and Perrig [163] in 2000. Of course, trivially, the user can download all of his encrypted data, decrypt it (using his key material) and search on the plaintext data for some (e.g., string) patterns. Unfortunately, the communication overhead of such approach is linear in the size of the database. Hence, one is interested in more efficient solutions. Searchable encryption tries to solve this problem, i.e., provide a secure cryptographic solution such that the search communication complexity is lower than downloading the entire encrypted database and performing the search on the decrypted data.

Searchable encryption comes in at least two different forms, i.e., the private-key and public-key flavor. (The private-key case is also often dubbed symmetric searchable encryption.) The different forms encompass for example oblivious RAM [72], fully homomorphic encryption [67], or public-key encryption with keyword search [27]. In this overview, we start with describing the private-key (or, symmetric) case, dubbed searchable symmetric encryption (SSE), and follow the work of Curtmola et al. [47] for the definitions. In this scenario, a user encrypts the data to be stored on the server with his (private) key and can deploy access patterns to the data before the encryption. This is done by creating a trapdoor for the key on keywords which are associated to the encrypted data. Essentially, this allows for an efficient search. Hence, only the user (with the corresponding key) is able to search on the encrypted data while keeping the communication complexity low in comparison to the trivial solution above. We note that this approach usually needs pre-processing of the data to be stored and, thus, the complexity of pre-processing is as least as large as the unencrypted data set. However, once the encrypted data is stored on the server, the (communication) complexity is low.

More formally, a SSE scheme in the the sense of [47] consists of five efficient algorithms (Gen, Enc, Trap, Search, Dec) as follows:

- **Key generation.** $Gen(1^k)$, on input the security parameter 1^k in unary, outputs a (private) key K.
- **Encryption.** $Enc(K, D)$, on input a key K and a data set $D = (D_1, ..., D_n)$, outputs a ciphertext set $C = (C_1, ..., C_n)$ and a secure index I.
- **Trapdoor generation.** $Trap(K, w)$, on input a key K and a keyword w, outputs a trapdoor T.
- **Searching.** $Search(I, T)$, on input the secure index I and the trapdoor T, outputs a set X of (lexicographically-ordered) document IDs.
- **Decryption.** $Dec(K, C)$, on input a (private) key K and a ciphertext C, outputs a message M or an error.

For correctness, we have that for all possible data sets D, for all ciphertext-secure-index pairs $(C, I) \leftarrow Enc(K, D)$, for all keywords w, we have that the search on I and the trapdoor (generated from $Trap(K, w)$) yields the D-corresponding IDs in X as well as $Dec(K, C) = D$.

For the public-key searchable encryption scenario, we follow the work of Boneh, Di Crescenzo, Ostrovsky, and Persiano [27] which define public-key encryption with keyword search (PEKS). Consider an email scenario where Alice sends Bob an encrypted message under Bob's public key. With a PEKS system, Alice can determine keywords that are attached to the encrypted mail in a specific (i.e., encrypted) way. Bob can now give a trapdoor for a keyword to the email server such that this server is able to search for that keyword attached to the emails (e.g., if the emails are labeled with the keyword "urgent").

A PEKS systems consists of four efficient algorithms (Gen, PEKS, Trpd, Test) as follows:

- **Key generation.** $Gen(1^k)$, on input the security parameter 1^k in unary, outputs a public and private key pair (pk, sk).
- **Searchable encryption generation.** $PEKS(pk, w)$, on the public key pk and a keyword w, outputs a searchable encryption S.
- **Trapdoor generation.** $Trpd(sk, w)$, on input the secret key and a keyword w, outputs a trapdoor t.

- **Testing.** $Test(pk, S, t)$, on input a public key pk, a searchable encryption S, and a trapdoor t, outputs "valid" or "not valid". In particular, for $S = PEKS(pk, w)$ and $t = Trpd(sk, w')$, Test outputs "valid" if and only if "w=w'".

B.2.2 Private Information Retrieval

Nowadays, knowledge about online users' preferences are well known as an important asset for service providers, and for many years those preferences were treated a secret for all entities except for the server itself, always under the assumption that the server was a trusted entity and would never employ such information against users [45, 11]. However, for some years, this situation has raised many security and privacy concerns, and it has been demonstrated that there are no reasons for such assumptions. On the one side, users' privacy might be compromised due to potential security (database) leaks, and on the other side, service providers could just sell or exchange this information without users being aware of it. In this regard, private information retrieval solutions are aimed at enabling users to keep their preferences private even from the server (service provider). Thus, PIR is the task that allows the user to retrieve a record from a (database) server, while hiding the identity of the record from the (database) server [45, 11, 71]. PIR protocols could be applied in a number of application scenarios including access to the cloud. There exist a number of approaches that go beyond applying anonymization techniques or downloading an entire database, these approaches are commonly classified either as theoretical private information retrieval or computational private information retrieval. 1) Theoretical Private Information Retrieval is the approach where the privacy of users is protected independently from the computational power of the attacker, i.e. the server is not able to determine information of user's query even with unlimited computational power [11, 71]. 2) Computational Private Information Retrieval is the approach where in contrast it is assumed that the privacy of the users' query is protected by considering an attacker with limited computational power (polynomial-time computation) [71].

XPIR: Private Information Retrieval for Everyone

The XPIR consists of a PIR protocol that enables users to retrieve information from a database without revealing what has been retrieved based on a 'computationally-Private Information Retrieval (cPIR)' scheme that do not need the database to be replicated (single server PIR) [122]. Generally speaking, there are two types of PIR protocols: single server PIR, and multi-server PIR. In a single server PIR protocol, only one server hosts and serves the database. Users' queries will be answered by one server. By contrast, in a multi-server PIR protocol, the database is replicated to multiple servers and the queries will be answered jointly by the servers.

The main idea is based on lattice-based cryptography and Ring-LWE which make cPIR practical and feasible even for users that do not have access to a high-end server. A lattice is defined as a set of points in n-dimensional space (Rn) with a periodic structure. In addition, the authors used the time as an important metric which is needed for a client to privately retrieve an element.

Generally, the proposed protocol is based on simple cPIR which is formally described as follows:

- Setup: set up an instance of the crypto-system with k security bits
- Query generation (encryption): This process is done to retrieve an element (e.g. i_0). To this end, for the i-th query element q_i, if $i \neq i_0$, then a random encryption of zeros is generated. Otherwise (if $i = i_0$), a random encryption of ones is generated. Ultimately, the ordered set $q_1, ..., q_n$ will be sent to the database.
- Reply generation: In this phase, the bits that can be absorbed in a ciphertext are considered $\mathcal{L}_0$. Then, all the m_i chunks of $\mathcal{L}_0$ are split, and they are called $m_{i,j}$. As a result, R_j will be calculated as a summation of absorbed chunks.
- Reply extraction (decryption): This phase decrypts the coordinates of the reply vector R and recovers all the chunks of the cipher text.

cPIR: Improving the Robustness of Private Information Retrieval

The cPIR protocol tries to improve the robustness of a basic PIR [71]. The main steps of the proposed protocol are as follows <:

- Allowing more responding servers to collude without compromising privacy, and be Byzantine, respectively. The authors also present a t-private v-Byzantine-robust k-out-of-$\mathcal{L}$ PIR protocol for any $0 < t < k$ and $v < k - \lfloor\sqrt{kt}\rfloor$.
- Improving the scalability of the proposed method in the previous step by proposing a PIR system which has hybrid privacy protection. In fact, the authors aim to amend the proposed protocol in terms of engaging more servers in the collusion procedure, i.e. queries have information-theoretic protection if a limited number of servers collude (up to t servers). However, in case of happening collusion by all the servers (more than t servers), queries imply computational protection.
- Improving the functionality of the proposed protocol in the first step in terms of communication cost. The authors suggest to use information-theoretic protection to the contents of the database against collusions of limited numbers of the database servers. It is expected that this can be done at no additional communication cost or increase in the number of servers, i.e. this improvement can add τ-independence to the proposed protocol, for $0 \leq \tau < k - t - v(2 - v/k)$, with no increase in the number of servers or in communication cost.

Available implementation

Percy++ is a C++ implementation of the information-theoretic private information retrieval (PIR) protocols, which provide t-private v-Byzantine-robust τ-independent k-out-of-$\mathcal{L}$ private information retrieval.

- k-out-of-$\mathcal{L}$: there are $\mathcal{L}$ distributed database servers, and only replies from k of them are needed.

- t-private: no coalition of up to t servers receives any information at all about the block of interest
- v-Byzantine-robust: up to v of the servers that do reply might give incorrect answers;
- τ-independent: the database is split between the servers so that no coalition of up to τ of them can determine the contents of the database itself ($\tau = 0$ means all the servers just have a complete copy of the database)

Any choice of t, v, τ, k and $\mathcal{L}$ will work, so long as they satisfy the following conditions:

They are all non-negative integers.

$$0 < t \leq t + \tau < k \leq \mathcal{L},$$
$$0 \leq v < k - t - \tau - \mathcal{L}$$

B.2.3 Proofs of Retrievability

Proofs-of-retrievability (POR) systems are closely related to PDP systems. The idea is to give a short verifiable proof (done by a server) that a specific file on a server is in some sense authentic and unchanged without the necessity to deliver the entire file to a verifier (e.g., a client). Note that this is different in comparison to a PDP system since in a POR system the client is able to retrieve the file.

More concretely, a client encodes a file and delivers it to a server (e.g., Amazon S3). The client can now check specific spots of the remote file in a challenge-response manner without retrieving the entire file back from the server. Concretely, the client determines a subset of file blocks to be checked, sends a request, and the server response with computed result over those blocks that can be verified. The client now is able to check if the response and is able to verify that the file is unchanged in any meaningful manner. One property of a POR system is that the response is shorter in communication than retrieving the entire file. Such systems can be used in scenarios with backup system where a backup file is checked not on a regularly basis but only from time to time. For example, the framework of Juels and Kaliski (JK) [111] assures that a file which is stored on a server can be retrieved with high probability without sending the entire file to the verifier. Further works (e.g., the Shacham-Waters (SW) framework [158]) can be used to reduce the communication complexity of such proofs of retrievability further. The Bowers-Juels-Oprea (BJO) POR framework [111, 29] generalizes both earlier frameworks of JK and SW and results in variants and improvements of previous PORs. The BJO framework follows the JK framework and defines a POR scheme consisting of the algorithms Gen, Encode, Challenge, Respond, Verify, and Extract:

- **Key generation.** $Gen(k, pp)$ takes as input the security parameter k and the public parameters of the system, and outputs a secret key k for the symmetric case (or a public and private key pair (pk,sk) in the asymmetric case; however, we focus on the symmetric case here). This is run by the client.

- **Encoding a file.** $Encode(k, F, s)$ takes as input the secret key, a file F and a client state s, and return an encoded file F' and a file handle h. This is also run by the client.
- **Generating a challenge.** $Challenge(h, k, s)$ takes as input a file handle h, a secret key k, and a client state s, and outputs a challenge value c. This is run by the client.
- **Generating a response.** $Response(h, c)$ takes as input a file handle h and a challenge c, and outputs a response r. This is run by the server.
- **Verify a response.** $Verify(h, c, r, k, s)$ takes as input a file handle h, a challenge c, a response r, a secret key k, and a client state s, and outputs a verdict "true" or "False". This is run by the client.
- **Extract a challenge.** $Extract(h, k, s, pp)$ takes as input a file handle h, a secret key k, a client state s, and the public system parameters, and outputs a file F. This is run by the client.

To construct a POR system out of a POR scheme, we proceed in two phases as follows:

- **The challenge-response phase.** The client runs Gen to obtain k. We assume that the server is of possession of an (encoded) file F. (The encoding could be an error-correction code.) Further, the client computes a challenge c via Challenge and sends c to the server. The server calculates the response r which the client verifies via $Verify$. This can be as many times as the client likes to perform. However, if any of the responses do not verify, the client aborts else it accepts.
- **The extracting phase.** The client is able to retrieve F via Extract. Note that in many cases the file has to be decoded with an error-correction code.

B.2.4 Provable Data Possession

Storing data on untrusted servers is very common in our digital world. It ranges from e-mail systems (e.g., Google's Gmail) over online storage systems (e.g., Apple's iCloud) to content management systems (e.g., Microsoft's SharePoint). It must be the case that data stored in either of such system remain in full control of the client, i.e., no server should be allowed to alter or modify data stored in its databases without the permission of the client. To check whether the data at the server's side is still present and unaltered, a client might want to retrieve a proof of data possession. In particular, these proofs should be trustworthy in a sense that no server should be able to provide valid proofs for altered client data.

In the following, we will describe a Provably Data Possession (PDP) system that allows to give such proofs of data possession. Consider a client-server scenario, a PDP system allows a client to securely and efficiently check whether data is correctly and consistently stored at a server's side without retrieving all the data. In particular, consider a setting in which a server is untrusted; i.e., it might alter or delete client's data stored at the server's end without being delegated to do so by a client. Trivially, the client can retrieve all the data and check for consistency. Clearly, one is interested in more efficient solutions. That is the client should not have to retrieve all of the data, but at the same time is still satisfied by being able to verify that the server keeps the right data (i.e., exactly the data that was provided by the client before).

The idea of PDP provides the solution for securely verifying client's data stored on an untrusted server in a very efficient manner. Say that a client wants to store a data file on a server. The client and the server can use a PDP system to proof that the data stored on the server was not altered in any means. Briefly summarized, the client generates and stores a public and secret key pair which is the result of setting up the PDP system. Further, the client takes one or more data blocks and tags those data blocks using the secret and public key pair. The public key, the data blocks, and the associated tags are sent to the server and are no longer needed at the client's end, i.e., the client deletes the data blocks and the tags and only keeps the public and the private key pair. Further, the client generates a challenge that corresponds to some index of some data block stored at the server. This challenge is then sent to the server and the server provides a proof of possession using the challenge, the public key, the data blocks, and the associated tags. Having generated the proof of possession, the server sends this proof to the client which is now able to verify the proof (with the help of the secret key), i.e., the client checks if the data is still consistent with that what the client had expected.

More technically, a PDP system uses a PDP scheme where a PDP scheme consists of four efficient algorithms (Gen, Tag, GenProof, CheckProof) (where we follow the definition of [14]):

- **Key generation.** $Gen(k)$ takes as input the security parameter k and outputs a public and secret key pair (pk, sk). This is run by the client.
- **Tagging a data block.** $Tag(pk, sk, m)$ takes as input the public and secret key pair as well as a data block m and returns the verification tag t. This is also run by the client.
- **Generating a proof.** $GenProof(pk, B, c, S)$ takes as input the public key pk, an ordered collection of data blocks B, a challenge c, and an ordered collection S (which contains verification metadata), and outputs a proof P. This is run by the server.
- **Checking a proof.** $CheckProof(pk, sk, c, P)$ takes as input the public and the secret key pair, a challenge c, and a proof P, and outputs a verdict "true" or "false." (If and only if CheckProof returns "true" then the proof verifies.) This is run by the client.

To construct a PDP system out of a PDP scheme, we proceed in two phases as follows:

- **The setup phase.** The client runs Gen to obtain (pk, sk) and tags each data block $m_1, ..., m_l$ using the tagging algorithm Tag. The public key, the data blocks $m_1, ..., m_l$, and the associated tags $t_1, ..., t_l$ are sent to the server.
- **The challenge phase.** The client generates a challenge c such that c indicates on which data block m_i the client would like to receive a proof of possession. The challenge is sent to the server. Further, the server generates a proof P of possession using the challenge c, the public key pk, the data blocks $m_1, ..., m_l$, and the tags $t_1, ..., t_l$. The client receives the proof P from the server and verifies if $CheckProof$ outputs "true."

B.3 Other Technologies

In this subsection, we provide more details on Verifiable Computing, Unlinkable Pseudonyms, Secret Sharing, and Remote Attestation; the fact sheets on some of the technologies can be found in Section 4.3.

B.3.1 Verifiable Computing

Given the increasing popularity and decreasing prices of cloud computing, more and more tasks are currently being outsourced into the cloud. However, letting a server perform sensible computations requires substantial, potentially unjustified, trust into this server. Namely, one needs to trust the server that the requested computations were computed correctly on the correct inputs. It is therefore required to develop efficient means to verify the correctness of the results returned from a server.

A naive solution to this task would be to let the server perform the computations, and then download all the outsourced data, verify its integrity, execute all the computations locally, and compare the results. Obviously, this approach renders the entire idea of outsourcing computations to the cloud void. Thus, a refined formulation of the task considered in verifiable computing is how the correctness of the result provided by the server can be verified in a way that requires substantially less computational effort than performing the entire computation locally.

Slightly more formally, a verifiable computing scheme consists of three parties. The *client* provides some input x to the *server*, who is requested to evaluate a function f on x, i.e., the server is requested to compute $y = f(x)$. If this computation was done correctly, then the *verifier* should accept y, otherwise it should reject with high probability. In the literature one also finds security models involving, e.g., multiple servers; yet, such settings are beyond the scope of this overview.

We refer to Buchmann et al. [102] for an exhaustive overview of the state of the art.

A verifiable computing scheme consists of four probabilistic polynomial time algorithms *(KeyGen, ProbGen, Compute, Verify).*

- **KeyGen:** This algorithm takes as input the security parameter and the function f to be evaluated, and returns a secret key sk, a public verification key vk, and an evaluation key ek.
- **ProbGen:** This algorithm takes as input the secret key sk and the client's input x, and outputs a decoding value d_x and a public encoding e_x of the data x.
- **Compute:** This algorithm performs the actual computation, taking the evaluation key ek and the encoding value e_x as inputs. It outputs an encoding d_y of $y = f(x)$.

- **Verify:** The verification algorithm takes as inputs the verification key vk, the encoding e_y of y, and the decoding d_x value of x. It either outputs y if the computation was performed correctly, or an error symbol otherwise.

Let us next discuss some of the **key aspects** of verifiable computing:

Efficiency: In a verifiable computing scheme, the client typically has to perform some pre-processing on the input x. If this pre-processing and the work required from the verifier are less than the costs of directly performing the computation locally, the scheme is said to be *efficient*. Depending on the concrete scenario, also *amortized efficiency* might be sufficient. In this case, the client has to perform a computationally expensive setup phase once, while the verifier's costs are comparatively low. Thus, after a certain number of computations, the average costs become less than those of local computations.

Privacy: Privacy means that the server and/or the verifier do not learn anything about the client's input x apart from what can actually be inferred from y. For instance, when processing sensitive data such as electronic health records, it might be required that the cloud provider performing the computations never gets access to the plaintext data but only has access to, e.g., encryptions thereof. Similarly, in such a scenario it might also be required that the verifier (e.g., an insurance company) only learns statics on the client's blood pressure or blood sugar level, but not the concrete measurements.

Verifiability: Here one distinguishes between public and private verifiability. A scheme is said to be *publicly* verifiable if any third party learning y and potentially the client's public verification key vk can check the correctness and integrity of the result. On the other hand, a scheme is *privately verifiable* if the verification requires access to secret information only known by the client.

Functions: The class of functions supported by a verifiable computing scheme determines the flexibility of the scheme. For instance, certain schemes found in the literature can only be used to evaluate polynomials up to a certain degree (which is sufficient, e.g., when one only wants to compute standard statistics on the client's data), while other schemes support arbitrary arithmetic circuits or even arbitrary C code. As a rule of thumb, the efficiency of a scheme decreases with its generality.

B.3.2 Unlinkable Pseudonyms

Pseudonyms can be thought of as privacy-friendly counterparts to public keys by allowing users to control the linkability of their actions. If a user decides that two actions should be linkable, he can reveal the same pseudonym twice; if, however, two actions should be unlinkable, the user can use different pseudonyms such that it is infeasible to determine whether the two pseudonyms have been generated by the same or two different parties. On the other hand, the receiver of a pseudonym is ensured that the user knows a corresponding secret key, and therefore is guaranteed that only legitimate holder of that key can reproduce a pseudonym.

The controlled linkability is achieved by using *scopes*. A scope is a specification of the context, which is known to the user and the receiver of the pseudonym. For instance, for an email service, the scope could simply be the email address: the user would then be linkable over multiple logins to his email account (which is inherently necessary as the user could otherwise not access previous emails), but he would not be linkable over different email accounts. Similarly, a service provider could use the date and time to derive the scope. Then, for instance, multiple logins of the same user within a fixed time-period could be detected, while different access across those time periods could not be associated with each other. By using (application dependent) appropriate time periods, the service provider would not learn access patterns of the user, while it would still be guaranteed that the user did not share his login credentials with others, as this would result in a high number of collisions within each time period.

The interfaces of scope exclusive pseudonyms are as follows:

- $KeyGen(1^\lambda)$: On input the security parameter λ, this algorithm outputs a secret key sk for a user.
- $NymGen(scope, sk)$: Taking a *scope* and a user secret key sk, this algorithm outputs a pseudonym nym for the given user under the given scope, together with a proof π showing it's well-formedness and knowledge of sk. The generation of nym is deterministic.
- $NymVerify(scope, nym, \pi)$: This algorithm takes as input a scope, a pseudonym, and a proof, and returns a single bit indicating whether to accept or to reject the pseudonym and proof for the given scope.

The security requirements for scope exclusive pseudonyms are *completeness, unlinkability, unforgeability*. Completeness requires that an honest user can, for every scope, generate pseudonyms and proofs which are accepted by an honest receiver. Unlinkability says that no receiver can decide whether two pseudonyms for different scopes have been derived from the same sk or from different secret keys. Finally, unforgeability says that only the legitimate owner of sk can produce a valid pseudonyms and proof for this secret key.

For a formal treatment of scope exclusive pseudonyms, we refer to Camenisch et al. [35].

B.3.3 Secret Sharing

Consider a user who wants to store personal information on potentially untrusted servers in a way that guarantees availability and confidentiality. Assume furthermore that the user plans to access the data from numerous different devices. Then simply encrypting the data before sending it to the servers is not a user-friendly solution, as the user would have to always carry his secret decryption key with him. This becomes even worse if the user intends to share the data with other users in the system, which would result in complex key management issues. In addition, an encryption scheme can only give computational privacy guarantees to the user. That is, a computationally unbounded adversary could always recover the data contained in the ciphertexts, and thus violate the confidentiality of the data. As an attacker's capabilities in the far future cannot be reliable estimated at the time of storing the data (e.g., given recent progress in the area of quantum computers), this encrypt-then-store approach is questionable when long-term privacy is an issue.

An alternative and fully key-less approach to the initial problem is offered by *secret sharing schemes*, which were independently introduced by Shamir [159] and Blakely [24]. Those are schemes which split the given data m into multiple, say n, shares such that only predefined *qualified* subsets of the shares can be used to recover the initial data, while all other subsets of shares do not contain any information about m. Depending on the scheme, this can even be guaranteed in an information theoretic sense, i.e., even a computationally unbounded adversary would not be able to infer any information about the message from such an *unqualified* subset of shares.

Let us elaborate this a bit more formally for so-called *threshold secret sharing schemes*, which are most reasonable to be used within the context of *CREDENTIAL*. In such schemes the user first defines a number n of servers he wants to store his data on, as well as a threshold t such that the original message can be recovered from any $t + 1$ shares, while no subset of up to t shares contains any information about the data. The user would then apply the resulting scheme to the m, and send one share σ_i to each server S_i . Now, the confidentiality of the user's data would be guaranteed as long as no more than t servers become corrupt and collaborate.

The interfaces of such basic threshold secret sharing schemes are as follows:

- $\mathit{Share}(n, t, M, m)$: This algorithm takes as input the number n of servers, the threshold t, the space M, of potential message to be shared, and the secret message $m \in M$. The algorithm then outputs hares $\sigma_1, ..., \sigma_n$.
- $\mathit{Reconstruct}(n, t, M, \{\sigma_{i_k}\}_{k=1}^{l})$: Taking $l > t$ shares as inputs, this algorithm reconstructs the shares data and returns m.

The basic security properties of secret sharing are *privacy* and *completeness*. The former says that the reconstruction algorithm always returns the original data if the input shares were computed honestly. The latter guarantees that no *unqualified* set of shares contains any information about the data.

Extensions

The above security requirements can be extended in various ways, depending on the concrete application scenario of secret sharing schemes. In the following, we briefly discuss some of those extensions:

- **Robustness:** The basic security requirements only guarantee the privacy of the data, and that the original data can be reconstructed from an honest qualified set of shares. However, they do not give any security guarantees if one or more of the servers maliciously alter their shares before handing them back for download to the user. *Robustness* now guarantees that the user is always able to reconstruct the initial message from sufficiently many (or all) shares, even if some of those shares have been modified. Furthermore, there exist schemes that allow the user to determine which shares have been altered.

- **Auditability:** If one uses secret sharing schemes for long-term storage of rarely accessed data, it is desirable to have efficient means to remotely check whether all shared data is still available at the servers without having to download the entire data. A secret sharing scheme is *auditable* if such a procedure is offered. Furthermore, it is called *publicly auditable* if the remote check does not need to be done by the user (i.e., the data owner) himself, but can be outsourced to an external entity such that even an external auditor collaborating with a subset of corrupt servers still cannot learn any information about the shared data.

- **Verifiability:** In certain situations, it is necessary that the user can prove that he actually sent consistent shares to the different servers, i.e., that the shares stored in the different locations would actually reconstruct to a valid file. This might, e.g., be required if servers should take over liability in the case that they lose the user's data, as otherwise a malicious user could simply send inconsistent garbage to the servers and then sue them for having lost his data. Secret sharing schemes offering an efficient way for the user to prove the consistency of the distributed shares are called *verifiable*.

- **Proactivity:** In the original setting, one assumes that no more than t servers become corrupt and pool their shares. However, in the case of long-term storage, this might be a very strong assumption, as it would imply that no more than t servers become compromised over the entire life time of the system. In *proactive* secret sharing schemes, the servers can periodically re-randomize their shares without involving the users. That is, the servers interactively update all their local shares in a way that guarantees that all data leaked to an adversary before the update becomes void after the re-randomization. In this way, it is no longer sufficient if the adversary corrupts $t + 1$ servers during the entire life-time of the system, but these corruptions need to happen between two re-randomizations, which can be dynamically adjusted. Advanced schemes also allow sufficiently many servers to jointly re-construct the share of another server in a privacy friendly way, which is, e.g., useful in cases of data loss or if one server goes out of business and gets replaced by another one.

Efficiency

From a computational point of view, secret sharing schemes are very efficient in the sense that the sharing and reconstruction algorithms can be executed highly efficiently even for large amounts of data. However, in the information theoretic sense, the communicational costs of such a scheme are equivalent to storing n redundant copies of a file when sharing a file; when reconstructing a file, at least $t + 1$ times the size of m needs to be downloaded. This is due to the fundamental observation that every share needs to be at least as large as the initial data.

Alternatively, when information theoretic security is not required, one can use computationally private secret sharing schemes, in which case the multiplicative communicational overhead can be reduced to $\frac{n}{t+1}$ and 1 for sharing and reconstruction, respectively.

For a detailed overview of the state of the art we refer to Buchmann et al.[102].

B.3.4 Remote Attestation

Remote attestation, as a part of trusted computing, allows for verifying the correctness of the internal state of a remote device across a network, and ideally for detecting remote software attacks. For example, software companies can use remote attestation to identify unauthorized modifications to software, or more generally communicating parties can gain assurance about the other's integrity and trustworthiness.

Remote attestation differs from traditional software attestation in the sense that for the latter, the verifier directly communicates with the prover, with no other hops in between. Yet, in contrast to secure hardware solutions, no dedicated components are required for remote attestation, making it easier to integrate on low-cost devices. The following description in large parts follows the approach of Francillon et al. [62, 63].

A remote attestation protocol consists of the following algorithms:

- $\mathit{Setup}(1^\lambda)$: On input the security parameter λ, this algorithm outputs a secret long term key k.
- $\mathit{Attest}(k, s)$: Taking a key k and the state s of a device, this algorithm outputs an attestation token tok. The generation of tok is deterministic; however, to avoid trivial attacks when generating the token, the prover's internal state consists of its actual state and a nonce provided by the verifier.
- $Verify(k, s, \mathit{tok})$: This algorithm takes as input a key k, a device state s, and an attestation token tok, this algorithm outputs a single bit indicating whether the attestation token matches the specified state.

The basic security properties of a remote attestation protocol need to guarantee *attestation unforgeability* and *completeness*. The former requires that a malicious prover (not being able to compromise the used key) cannot generate an attestation token with is accepted in the verification algorithm for any state $s' \neq s$ of its own choice; the latter requires that honest prover are always able to succeed in the protocol.

The basic definitions of remote attestation do not prevent an adversary from using the correct state for the attestation procedure, and then replace it by an arbitrarily altered state. Therefore, the following (minimum) properties need to be satisfied.

Exclusive access. Only *Attest* must have access to the key k, as otherwise the adversary could simulate the attestation algorithm for the target state s, while actually using a different state s'.

No leaks. *Attest* does not leak anything about k except for attestation tokens tok. This is necessary as otherwise it would be possible to construct examples where over the time the adversary could infer the value of k after many invocations of the attestation algorithm. This also needs to be addressed in concrete implementation with regards to side-channel attacks.

Immutabililty. The code of *Attest* is immutable and directly executed from this immutable memory. Otherwise, the adversary might be able to replace the code of *Attest* by a malicious piece of code between the transfer of the *Attest* code and its execution.

Uninterruptibility. The execution of *Attest* cannot be interrupted, as otherwise the adversary could let a malicious piece of code be attested by replacing the bad parts by correct code just for the time of the execution of *Attest*.

Controlled invocation. In order to not skip important parts of *Attest*, the algorithm must only be invoked from its intended entry point.

As a conclusion, while remote attestation is intuitively easy to achieve from message authentication codes and related primitives, it has to be noted that their real-world implementation and realization poses massive challenges to protocol designers and software engineers.

C Details for Authentication to the Cloud

At the core of CREDENTIAL is the strong authentication component, which will enable secure authentication to the Cloud. In this respect, Appendix C.1 introduces the details of each authentication framework identified as useful for the CREDENTIAL wallet. Furthermore, in order to provide a higher level of security assurance, in particular when using soft-type of authenticators and mobile devices, a cluster of underlying technologies is also considered. Appendix C.2 introduces the details of the evaluated underlying technologies.

C.1 Authentication Technologies

In this section, we describe the main architecture, modules and features offered by the evaluated frameworks. In particular, more details about the authenticators, methods and protocols are given. Appendix C.1.1 introduces a high-level architecture and specification of both FIDO UAF and FIDO U2F frameworks. Appendix C.1.2 introduces the details of OATH followed by Mobile Connect, which is described in Appendix C.1.3. SQRL authentication framework and its main operations are described in Appendix C.1.4. At the end of this section more details of biometric authentication is describes. Biometric processes and techniques are highlighted in order to provide a quick overview on those technologies that could be potentially integrated as authenticators in most of the introduced frameworks.

C.1.1 FIDO

In July 2012 a non-profit organization called the FIDO (Fast Identity Online) Alliance was formed. The FIDO Alliance is an industry consortium consisting of more than 200 members. The FIDO Alliance addresses the user's problem to remember multiple usernames and passwords and also the lack of interoperability utilizing strong authentication devices. The main goal of the FIDO Alliance is to develop specifications, which should help the user to securely authenticate to online services, basically, to increase security using web applications. These specifications define a new standard utilized for security devices and browser plugins. FIDO specifications are designed to protect the user privacy.

The FIDO Alliance has developed two sets of specifications. These two specifications are separated into two different user experiences. The first specification describes the passwordless user experience. This passwordless user experience is based on the Universal Authentication Framework (UAF). The UAF enables the possibility to use alternative authentication methods such as fingerprint or other biometric authentication methods to allow passwordless authentication to a web application. Moreover, UAF enables multifactor authorization, which means that it is possible to combine authentication methods in order to increase security. These combined authentication methods can be for example fingerprint + PIN.

The second user experience is the second factor user experience, which is defined in the Universal 2nd Factor (U2F) specification. The main idea of U2F is to strengthen the security of online services by adding a strong second authentication factor to an already existing username and password infrastructure. These strong second authentication factors include secure devices such as a USB dongle. This USB dongle is registered to a specific user and the ownership of this dongle authenticates the user.

Architectural Overview

FIDO is based on a so-called client-server architecture. In the commercial way the usernames and passwords have to be stored on the client side as well as on the server. Storing usernames and passwords on the server brings up security issues such as a user's credentials getting stolen by a thief.

FIDO is increasing security with different mechanisms. One example would be that the user's passwords no longer have to be stored on the server. Furthermore, utilizing FIDO prevents against phishing or man-in-the-middle attacks. FIDO consists of an extended server-client architecture. The extended part is the so-called authenticator.

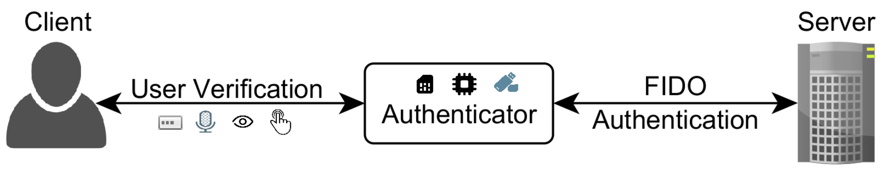


Figure 10: FIDO Basic Concept

Figure 10 shows the basic concept of FIDO. The first thing that happens if a user tries to log on a web service is that he or she has to verify his or her identity to the FIDO authenticator. This verification procedure is supported by various mechanisms such as fingerprint, speech recognition or by simply username and password. After successfully verifying to the FIDO authenticator the authentication procedure starts. Next, after a successful authentication procedure the user is logged onto the web service. To be able to prove the identity of a user and the authenticator to the server a preliminary step has to be performed. This preliminary step is a binding between the user and the authenticator and between the authenticator and the server. After this preliminary step is taken, which is called registration, the server can be sure that the user and authenticator's identities are proven.

Process

FIDO works with utilizing protocols with strong public key cryptography mechanisms to increase security for authentication. In order to achieve this level of security, a registration step has to be performed. A new key pair is generated by the client device in the registration step. The client device retains the private key and the public key is registered with the web service. By proving the ownership of the private key, the client device is authenticating to the web service. The user has to unlock the private keys stored on the client device before using it. FIDO supports secure and user-friendly mechanisms to unlock the private keys such as fingerprint scan, speech recognition or using a second factor device.

- **Registration:** The FIDO registration process is necessary if a user is using a web service for the first time. Figure 11 illustrates the registration procedure. This procedure starts with the user selecting one of the FIDO authenticators available on both the web service and the user device. Next, the user has to unlock the FIDO authenticator with one of the available unlock mechanisms such as the fingerprint reader. After successfully unlocking the authenticator, the user device generates a new public-private key pair for the user device, the web service and also for the user account. The private key is stored on the device and will never leave it. The public key is sent to the web service so that the user account is associated with this key. This registration procedure only has to be performed by using a new web service the first time. Following this step, the user is able to use the authenticator to authenticate at the service.

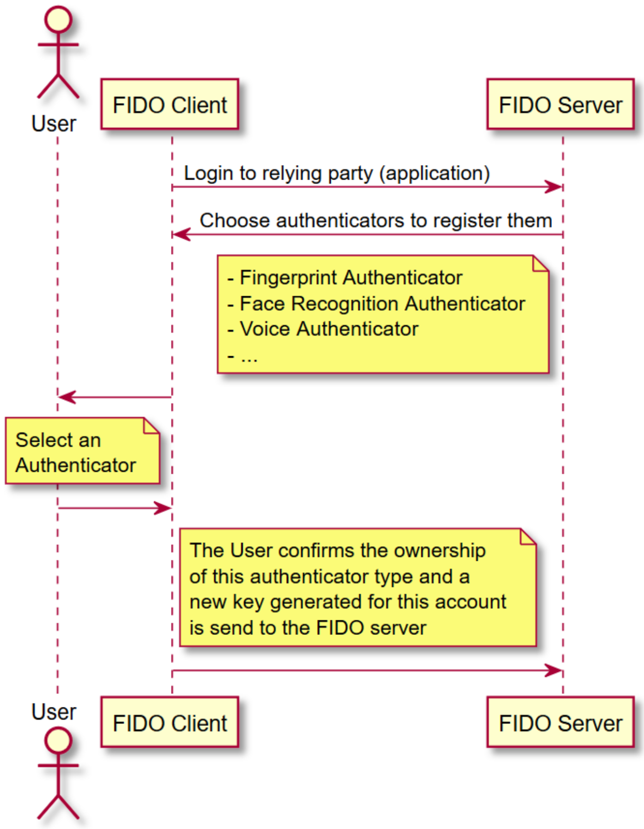

User
FIDO Client
FIDO Server
Login to relying party (application)
Choose authenticators to register them
- Fingerprint Authenticator
- Face Recognition Authenticator
- Voice Authenticator
- ...
Select an Authenticator
The User confirms the ownership of this authenticator type and a new key generated for this account is send to the FIDO server
User
FIDO Client
FIDO Server

Figure 11: FIDO Registration [57]

- **Login:** A user can log into a web service using a previously registered device. The login procedure is shown in Figure 12. It starts with unlocking the FIDO authenticator. The unlock mechanism has to be the same as at the registration procedure. The device selects the corresponding key related to the user and the web service. Next, the web service is asking the device to sign a challenge. The challenge is signed by the client device with the correct key. After signing, the challenge is sent back to the web service. The following step will be that the user is successfully logged onto the web service.

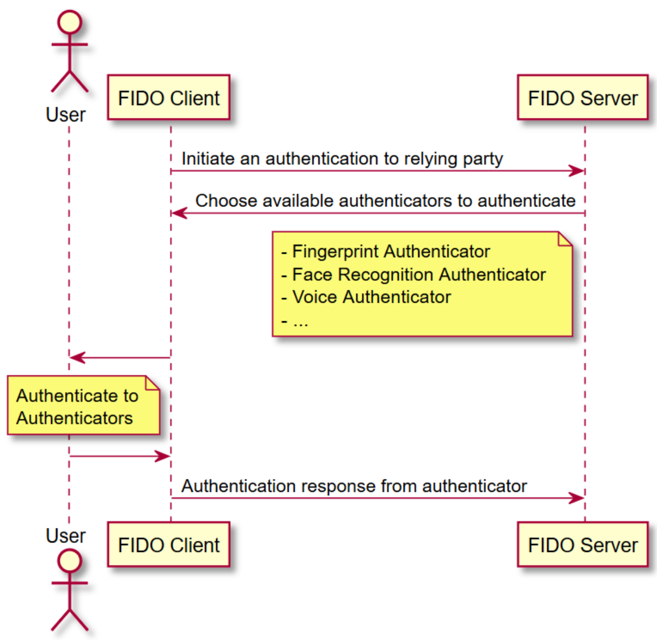

User
FIDO Client
FIDO Server
Initiate an authentication to relying party
Choose available authenticators to authenticate
- Fingerprint Authenticator
- Face Recognition Authenticator
- Voice Authenticator
- ...
Authenticate to Authenticators
Authentication response from authenticator
User
FIDO Client
FIDO Server

Figure 12: FIDO Login [57]

Universal Authentication Framework (UAF)

The UAF specification focuses on the passwordless user experience. Moreover, it has been developed to define open, scalable and interoperable mechanisms to securely authenticate user to web services. Besides the passwordless authentication, UAF also offers multifactor security such as fingerprint + PIN.

Figure 13 describes the FIDO UAF architecture from a high level view. Moreover, it also describes the interaction of and between the components. Each component is described in the subsequent sections.

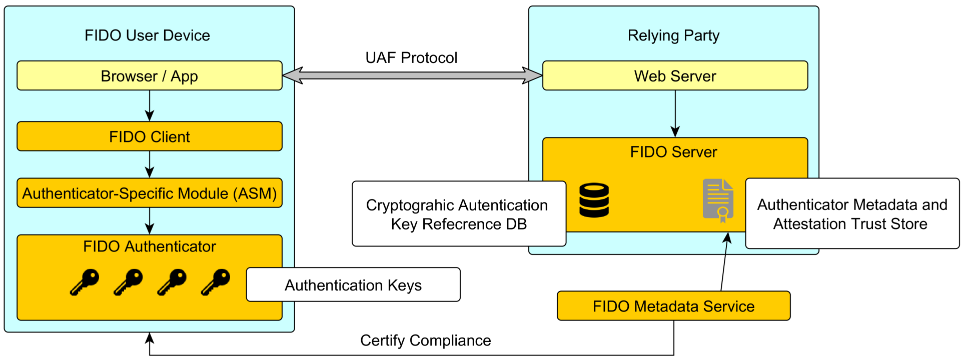


Figure 13: FIDO UAF High-Level Architecture View [56]

The **FIDO client** is a part of the user device and is responsible for the interaction with the FIDO authenticators utilizing the FIDO authentication abstraction layer via API. The client also interacts with the so-called agent which can for example be the app or the browser. It uses agent-specific interfaces to communicate with the FIDO server. The FIDO client is designed to be implemented on various operating systems and web browsers on various systems.

The **FIDO UAF server** is responsible for the interaction with the web sever of the relying party. The web server is utilized for the communication to the FIDO client via the device agent on the user's device. Another important responsibility of the FIDO server is to manage the association between registered authenticators and user accounts. Moreover, it is also responsible for validating the user's authentication and transaction confirmation responses.

The **FIDO protocols** are utilized to carry messages between the user's device and relying party. FIDO protocol messages are used for the authenticator registration, user authentication, secure transaction confirmation and authenticator deregistration.

The **FIDO Authenticator** is able to create key material related to the relying party. The authenticator is located on the user's device or it is connected to it. This FIDO authenticator is a secure entity that interacts with the **FIDO authenticator abstraction layer** which provides a uniform API. This API enables the use of authenticator based cryptographic mechanisms and operations.

Universal Second Factor (U2F)

FIDO U2F aims strong authentication on the web while preserving the user's privacy. U2F utilizes the so-called U2F devices as second factor which are carried by the user. The user has to register the U2F device so that the device is linked to his account and to create a new public-private key pair. The user can use this U2F device as second factor when authenticating to a web service. The basic idea from U2F is to make the authentication of already existing web services stronger. This idea should be realized by adding a strong second factor to the existing authentication infrastructure. In fact, the U2F device mentioned earlier constitutes as this strong second factor. A few examples of U2F devices are USB devices, standalone NFC devices or standalone Bluetooth devices. After successfully registering a U2F device, the user is able to use this device as a second authentication factor. Both the registration as well as the authentication are exposed through JavaScript APIs on the browser side and native APIs on the mobile operating system side.

C.1.2 Initiative for Open AuTHentication (OATH)

The OATH specification [97] intends to provide strong authentication by leveraging open standards. OATH framework supports several authentication methods in multiple devices such as, mobile phones. OATH has been designed to support high level of interoperability it facilitates the integration of open, scalable and interoperable mechanisms to securely authenticate user to web services. A high level architecture is depicted in Figure 14

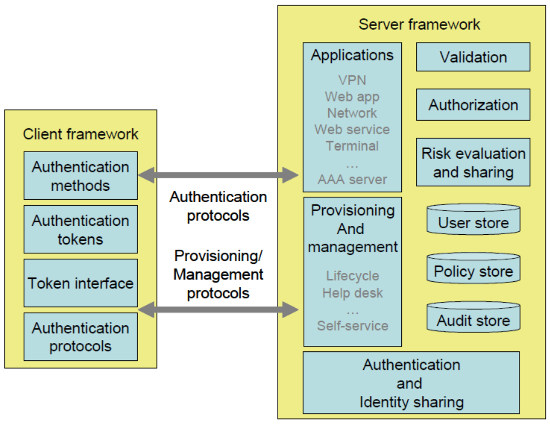


Figure 14: OATH high level architecture [97]

Client Framework

The flexible client framework enables standard-based integration of multiple forms of strong authentication to authenticate users and devices. It supports a wide range of authentication methods (Biometric, certificate, OTP, Transaction signing), standalone and embedded tokens (e.g. smart card, SIM card, TPM) and it communicates using standard authentication protocols (e.g. SSL/TLS, WS-Security, EAP-*).

Authenticators

Different levels of assurance can be provided during the authentication process. OATH supports hardware- and software- based authenticators to confirm that the user is who he/she claims to be. Soft-based authenticators can be implemented on multiple devices (e.g. smart card), which enable the combination of multiple authentication methods in a single authenticator or token.

Protocols

In order to exchange authentication data between the client and the server, over-the-wire authentication protocols are implemented. The OATH reference architecture considers existing standard protocols (e.g. EAP, SSL/TLS, Kerberos, WS-Security) and each of them support one or more authentication methods.

Server Framework

The server framework consists of the following modules

- Provisioning and management framework: responsible of accommodating secure and reliable delivery of software modules and security credentials to any client device.

- Validation framework: validates authentication credentials and communicates contextual information to the risk evaluation framework.
- Applications: The application receives user's credentials and communicates with the validation module.
- Authorization: Once user has been authenticated, the authorization modules is responsible of granting/denying access to a certain resource (based on applicable access policy).
- User store: Stores all user and profile information.
- Policy store: Stores all policies by component or in a common stored, depending on the model used.
- Audit store: a central repository for all audit and operational events.
- Authentication and Identity sharing: implements technology primitives to enable sharing of authentication of identity information.
- Risk evaluation and sharing: determines the risk associated to a particular transaction.

C.1.3 Mobile Connect

The Groupe Speciale Mobile (GSM) was created in 1982 by the Confederation of European Posts and Telecommunications (CEPT) to design a pan-European mobile technology. In 1995 on the foundation of the GSM the GSM Association was formed. This organization represents the interests of almost 800 mobile operators across the world and related companies. GSMA supports the standardization, implementation and promotion of Global System for Mobile Communication GSM.

Due to the fast growth of the digital economy and the increasingly development of digital identity services the GSMA have developed the Personal Data Programme based on the GSMA Mobile Identity Programme with the aim to help digital service providers and consumers assuring privacy and security.

The GSMA Personal Data has developed the Mobile Connect, a new standard in digital authentication. It is a universal log-in solution that uses the security of the mobile devices and requires user permission for sharing personal information, protecting user privacy. Mobile Connect associates the mobile phone with the user allowing authentication with different levels of security, in this way remembering usernames and passwords are not needed, they are replaced by a mobile number and access to the mobile device.

Mobile Connect is an identity service based on the OpendID Connect & OAuth2 standards. Mobile Connect is offered by mobile network operators and delivered thanks to a standardised technical interface.

The user authentication is provided by the mobile user operator. For this purpose, Mobile Connect is using two APIs:

- The API Exchange provides the Discovery API that allows identifying the end user's mobile operator and whether Mobile Connect is available for that network, also providing the URLs for the Mobile Connect service associated to the end user's network.

  Figure 15 displays how API Exchange is working and the steps are described next.

  1. The SP application calls Discovery API to look for the operator the user is associated. Operator details are returned.
  2. The SP application calls user's operator.

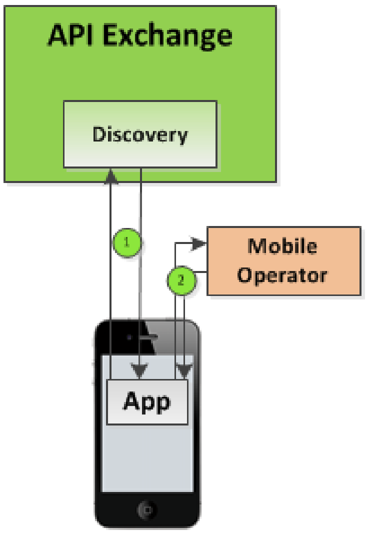


Figure 15: API Exchange Flow

- The Mobile Connect API is an OpenID Connect API that allows the use of Mobile Connect user account to authenticate a user.

Both APIs are RESTful API with JSON responses.

Due to the SPs could need several Levels of Assurance (LoA) during the authentication process; the Mobile Connect API provides different secure ways to confirm that the user is who they claim to be. The mobile operators use authenticators to make this. There are different types of authenticators which provide different LoA as follows:

1. Seamless Authentication: authentication is automatically performed by the operator when the user is connected through the operator network;
2. SMS+URL authenticator is an SMS sent to the user's device with a unique one-time only URL that the user selects to prove that they are in possession of the device and have access to the SMS;

3. USSD (Unstructured Supplementary Service Data) is a protocol used by mobile networks to communicate with a terminal (mobile device). The operator will push a message to the terminal and can require a response. The user's device will display a message such as "Press 1 for OK. Press 2 for Not OK". If a user has access to the mobile device and is able to respond (correctly) to it, they will be authenticated;
4. SIM Applet authenticator is an application that is stored within the SIM card and run on the mobile phone. When an authentication request occurs, a binary SMS will be sent to trigger the SIM applet. Once triggered, it will prompt the user for an input such as "Press OK to continue" or "Please enter your PIN". The SIM Applet will send a response back to the operator to validate the user;
5. Smartphone Application authenticator is a native application that allows the end user to manage their verification.

It is worth to mention that the LoA the SP requires and the user can use is dependent on the authenticator implementations the mobile operator supports, e.g. some mobile operators cannot support biometric factors such as fingerprint, as a second factor authentication. Once the authentication is performed the Mobile Connect uses a unique identifier called PCR (Pseudonymous Customer Reference) to reference a specific end user. The PCR ensures the user's privacy protection and guarantees this PCR represents an actual end user. The PCR will be same for each authentication request. The use of PCR as the key identifier easier the integration with existing accounts systems, mapping the PCR identifier to the existing user-id.

Mobile Connect Login

As Figure 16 shows, the user could use Mobile Connect in order to get access to protected resources, provided by a SP, using mobile device instead of user and password as credentials.

Once the user clicks on the Mobile Connect button asking the service (step 1), the SP will use the Discovery service for identify the end user's operator (step 2). Discovery service will try to retrieve Operator details such as Mobile Connect API address. If this is not possible the Discovery service will use an Operator User Interface where the user will be asked to enter their mobile phone number.

The information retrieved will be returned in the Discovery response (step 3), and will be used to create and send the Mobile Connect authentication request to the user's operator (step 4). Depending on the Level of Assurance required by the Service Provider an appropriate Authenticator will be used by the Authentication Service. Then the user will be asked through his/her mobile to act in order to perform the authentication (step 5).

After the user have completed the action proving he/she possess the mobile device (step 6), an authentication response is sent to the SP (step 7). This response contains the PCR unique for the end user. At this moment the user is granted to access the requested resource (step 8). In order to do so, the user must be connected to mobile network either on their operator's network or not.

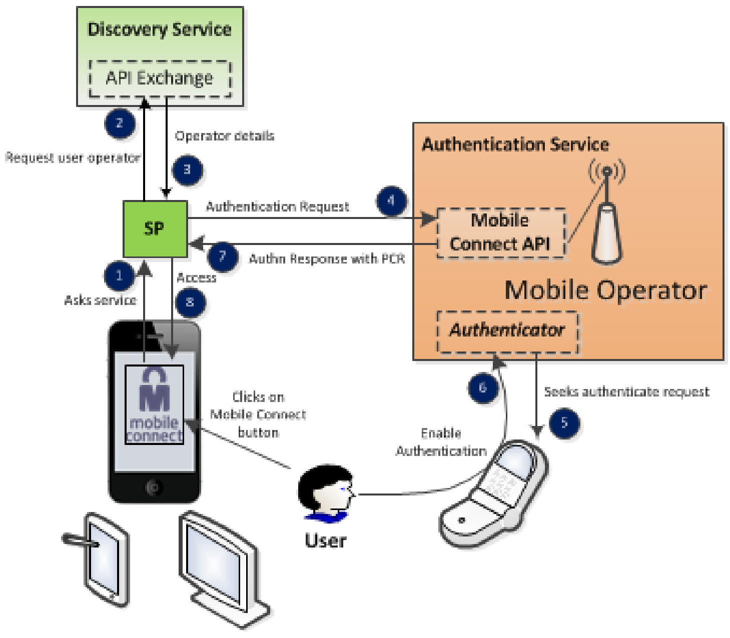


Figure 16: Use of Mobile Connect by the User

Basically Figure 17 depicts the login process where the Discovery service and the Authentication service from the mobile operator are involved.

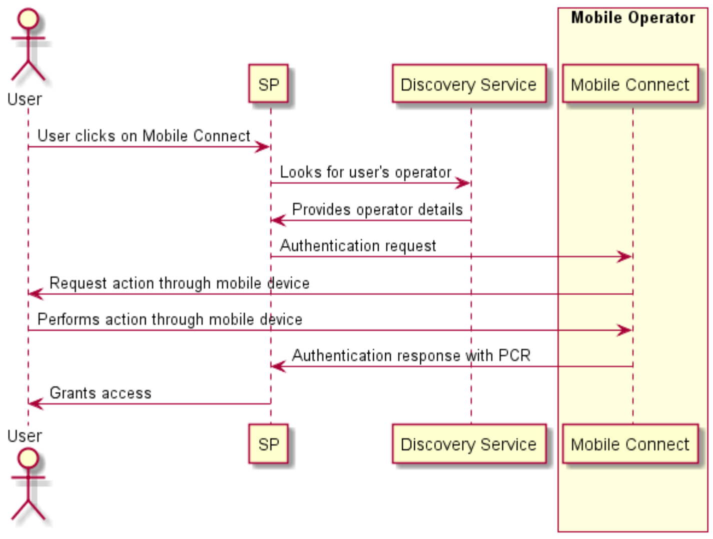

Mobile Operator
User
SP
Discovery Service
Mobile Connect
User clicks on Mobile Connect
Looks for user's operator
Provides operator details
Authentication request
Request action through mobile device
Performs action through mobile device
Authentication response with PCR
Grants access

Figure 17: Mobile Connect Login

Mobile Connect Privacy

Mobile Connect service, through the privacy principles agreed by the members of the GSMA, is committed to protect the user privacy, developing security practices and assuring the security of their data.

C.1.4 Secure Quick Reliable Login (SQRL)

SQRL [69] is a draft open standard for secure website login and authentication proposed by Steve Gibson of Gibson Research Corporation in October 2013 as a way to simplify the process of Authentication protocol, without revealing any information about the transaction to a third party. SQRL standard provides a passwordless anonymous identification and authentication framework based on cryptographic challenges, asymmetric key schemes, standard URL schemes and QR codes.

Specification

SQRL basically works by presenting, in a server login page a cryptographic URL challenge that must be processed by a client application (a mobile application or a browser extension). When processing a challenge URL, the client application must generate, from a master key, a site-specific public key pair. With the private key part, the application signs the URL (excluding the protocol). The final step involves making a POST request to the aforementioned URL with the public key and the generated signature. In the server side, the server, using the POST URL, the public key and the signature, can easily verify that everything matches. As long as the master key remains the same, the server can subsequently send challenges and receive responses that will always correspond to the same identity key. The main benefits of this approach are that the user has no longer to manage site-specific passwords and that the server side knows nothing about the identity of the person accessing the service (anonymity).

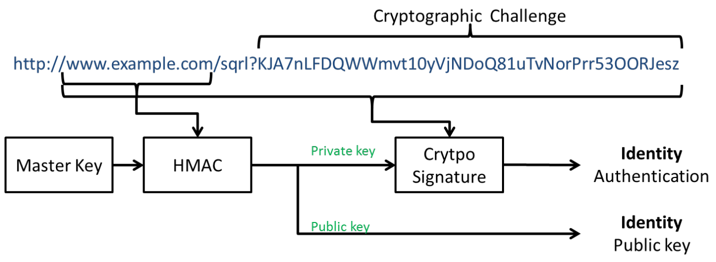


Figure 18: SQRL Working Principle

Registration and authentication: The SQLR registration process is necessary if a user is using a web service for the first time. Figure 19 illustrates the registration procedure. This procedure starts with the user trying to access a service without a previous session. The service will present a QR Image (containing the challenge URL) and an anchor element linking to the aforementioned challenge URL and using a special protocol (e.g. sqrl://www.example.com/sqrl?...).

The main objective of using a special protocol is that such URLs can be automatically captured by SQRL specific applications or extensions and automatically handle the authentication logic. The registration and authentication protocols are the same. The only difference is that at the server side, the SQRL server will check if the public key has already been used (authentication). If so, the server will recover information associated to such user, if not, it will register the user as new.

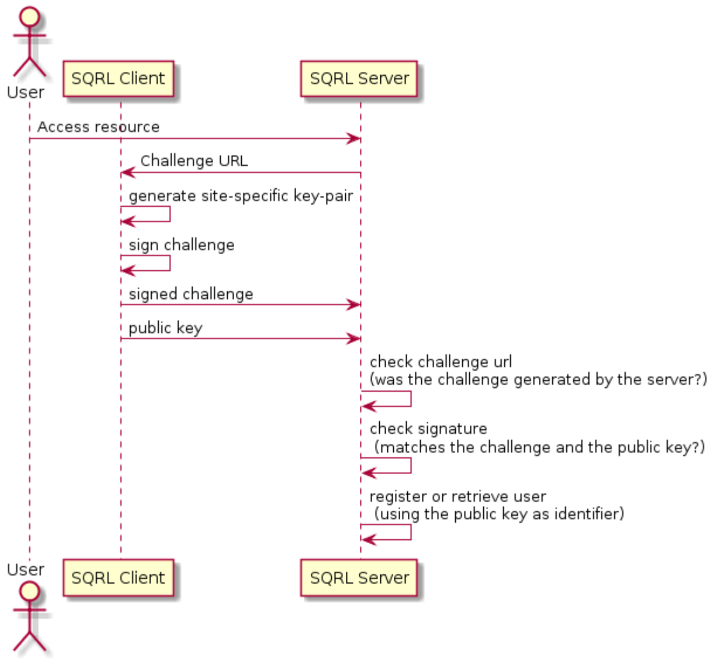

User
SQRL Client
SQRL Server
Access resource
Challenge URL
generate site-specific key-pair
sign challenge
signed challenge
public key
check challenge url
(was the challenge generated by the server?)
check signature
(matches the challenge and the public key?)
register or retrieve user
(using the public key as identifier)
User
SQRL Client
SQRL Server

Figure 19: SQRL Registration and Authentication

Key Management: In previous sections the algorithm has been simplified in order to avoid cryptographic underlying techniques to obscure the main concepts behind the framework. The main issue that can rise with the previous description is that "what happens if the master key is lost?" The answer to this question is dealt in the full specification with the "ID Lock mechanism": which is based in creating a second "master key" (Identity Unlock Key) different from SQRL's regular identity master key. This Identity Unlock Key must be never stored in a SQRL client (it can be printed onto paper as a graphic QR code or exported as a short text file and stored securely). The only way a website can accept a new identity for some user is by the presence of a "identity unlock key" which will allow users to remove their previous identity so, even with full access to the old master key, attackers will no longer be able to impersonate them There are other issues such as how the master keys are protected within the devices that are not covered within the specification as they have to be dealt at the implementation level given the high dependency with OS or HW features. Figure 20 depicts the flow and interconnectivity of cryptographic keying information within a SQRL client. There, EnHash is essentially sixteen (16) chained iterations of SHA-256[16] EnScrypt is essentially a number of chained iterations of Scrypt[17]. The exact number of iterations is determined by an execution time which is a parameter to the function.

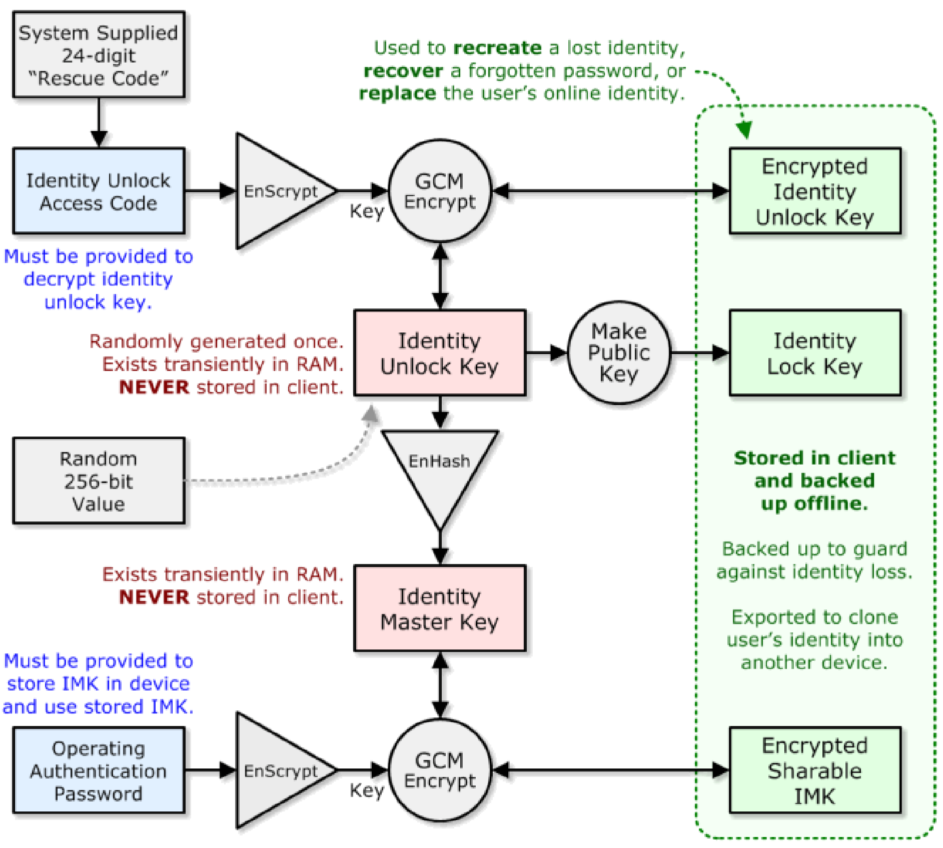

System Supplied 24-digit "Rescue Code"
Used to recreate a lost identity, recover a forgotten password, or replace the user's online identity.
Identity Unlock Access Code
EnScrypt
Key
GCM Encrypt
Encrypted Identity Unlock Key
Must be provided to decrypt identity unlock key.
Randomly generated once. Exists transiently in RAM. NEVER stored in client.
Identity Unlock Key
Make Public Key
Identity Lock Key
Random 256-bit Value
EnHash
Stored in client and backed up offline.
Backed up to guard against identity loss.
Exported to clone user's identity into another device.
Exists transiently in RAM. NEVER stored in client.
Identity Master Key
Must be provided to store IMK in device and use stored IMK.
Operating Authentication Password
EnScrypt
Key
GCM Encrypt
Encrypted Sharable IMK

Figure 20: SQRL Client-Side Key Management [68]

C.1.5 Biometrics

Common authentication methods either based on what you know, or what you have involve properties that can be forgotten, lost, stolen or eventually disclosed; and furthermore, do not authenticate the user as such. Contrary to this, biometric authentication, which is based on what you are, allows users to be authenticated by referring to those physiological or behavioral characteristics that are uniquely associated to them, and presumed to be neither replicable nor transferable. Therefore, biometric authentication promises to severely enhance the security of authentication systems. Biometric systems aimed at verifying the identity of individuals (authentication) basically carry out three processes: i) enrollment, ii) storage, and iii) verification.

1. The enrollment process is the initial step. It collects user's biometric sample (e.g., fingerprint), which requires user's interaction. Ideally it is performed only once; however, most systems require the collection of several samples. Once the collection of samples is done, a feature extractor algorithm analyzes the sample, extracts and measures specific biometric features (e.g., fingerprint minutiae).

2. The storage process is performed at the user registration phase. It consists in storing recently extracted features (i.e. biometric template) locally or remotely; and additional information, which allows the authentication system to associate a biometric template to an individual.

3. The verification process is performed on each authentication attempt. It takes a new sample of the biometric data, and provides a comparison between the new sample and the one stored during enrollment. Unlike conventional authentication methods, biometric systems provide a percentage of similarity between samples, i.e. an individual's identity is confirmed only if the resulting percentage is above a predefined threshold.

Techniques

Biometric authentication techniques are classified by the type of characteristics evaluated: physiological attributes or behavioral singularities. The leading biometric techniques used in inherence-based authentication are those briefly introduced next. Physiological biometrics

Physiological Biometrics: It consists of measurements taken from data obtained as part of the human body. This category includes:

- Fingerprints: it identifies the lines convergence points (minutiae matching).
- Facial recognition: it captures a sequence of images, and extracts features from the images ensemble
- Hand geometry: it extracts hand features, such as shape, appearance, length and perimeters of fingers.
- Iris recognition: it identifies the location, shape and size of random patterns in the external iris of the eye; it transforms the iris rim into a rectangular shape texture.
- Retinal identification: it maps the blood vessels in the back of the eye.

Behavioral Biometrics: It consists of measurements taken from user's actions, some of them indirectly measured from the human body, e.g., voice recognition. Techniques within this category include:

- Voice recognition: it analyses power and spectral samples of the speech, building a statistical pattern from them.
- Keystroke dynamics: it identifies user's typing pattern. It measures and compares the series of user specific timing events also known as "typing signature". Samples could be taken either from conventional keyboards or from touch screens (key tap dynamics).
- Handwritten signature: it uses a digital version of the signature. Modern sensors can also measure pen position, pressure and inclination in a three-dimensional way.

C.2 Underlying Technologies for Authentication

As algorithms based on software-only solutions are always vulnerable by malicious applications implant by potential attackers, further technologies are necessary in order to mitigate this attack vector. Those technologies make use of non-manipulable cryptographic hardware tokens representing a root of trust. In this context, two technologies are broadly used in the field nowadays: Trusted Platform Modules (TPMs) and Trusted Execution Environments (TEEs). While TPMs only support a very limited set of operations, TEEs are more flexible. The following sections investigate those technologies in more detail and explore the availability on android devices on the other hand.

C.2.1 TPM

TPMs are independent pieces of hardware which tackle security requirements by providing a hardware based root of trust. This includes key storage or key generation in a secure and trusted manner, but also user authentication or a secure random generator. Another important aspect of a TPM is the monitoring of the secure boot process. Usually, the TPM is implemented in a chip or microcontroller that is physically separated from the remaining hardware. Standardized by Trusted Computing Group (TCG), TPMs are used in a variety of domains. A lot of today's PCs and laptop are making use of TPMs to gain security. According to TCG over 100 million PCs and laptops were sold with incorporated TPMs in 2007.

A set of components of a TPM version 1.2 can be seen in Figure 21.

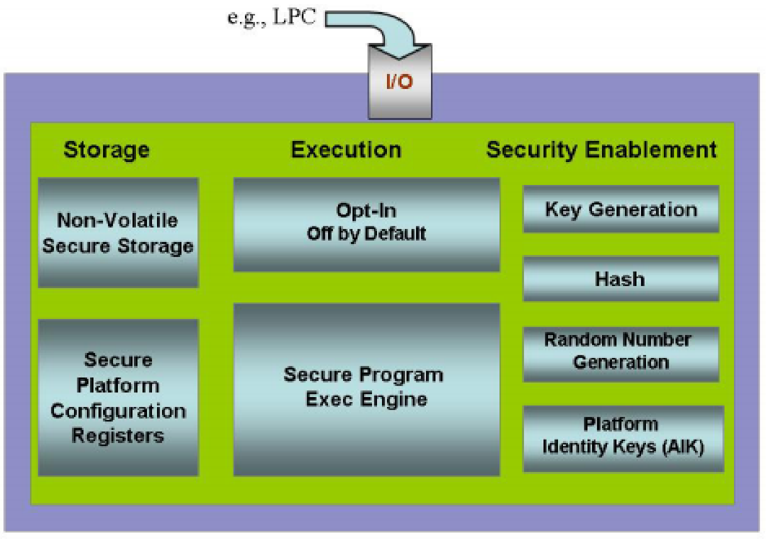

e.g., LPC
I/O
Storage
Execution
Security Enablement
Non-Volatile Secure Storage
Opt-In Off by Default
Key Generation
Hash
Random Number Generation
Secure Platform Configuration Registers
Secure Program Exec Engine
Platform Identity Keys (AIK)

Figure 21: Components of the TPM 1.2 [78]

With TPM version 2.0, TCG specified five different implementations for the TPM (all of the following specified in [77]). This was done because of the increasing need for security in embedded system. Version 2.0 is a more flexible approach, in order to provide security features also in systems were it may not be feasible (because of space or cost) or not necessary to provide maximum security with a separate microchip.

Discrete TPM: Provides the highest level of security but is also the most expensive implementation. A discrete chip is built in the device (hardware based).

Integrated TPM: Is still hardware based but the TPM is integrated into another chip which purpose is not solely security. This level is not tamper-resistance, making it less secure than the discrete TPM.

Firmware TPM: The TPM is executed in a trusted execution environment (TEE). It is not required to use an own, isolated chip for this level of the TPM. Resources for the TPM, like keys, remain in the TEE. The drawback of this method is that the TPM must rely on other implementations providing security.

Software TPM: Basically just a software emulator of the TPM. This TPM provides the lowest level of security. It is vulnerable to any software bug in the operating system and tampering. The main application of a software TPM is for testing and building a prototype.

Virtual TPM: Cloud environments (for example "Internet of Things") may also use this TPM. The virtual TPM provides the same security features as a physical TPM would do, but the TPM is shared between multiple virtual machines and relies on a hypervisor.

The main advantage of TPMs is that it protects the device from external software attacks. Without proper authorization it is nearly impossible to access protected data. TCG also created a specification for mobile architectures, in order to address the vulnerability of such mobile devices, namely the "Mobile Trusted Module" (TPM Mobile).

Android devices mostly utilize the TEE. Most chip manufacturers make use of the "ARM TrustZone". It may be possible for the TPM to run within a TEE as a "Trusted Application". This would result in a firmware TPM.

TPMs are heavily used in windows phones.

C.2.2 TEE

A "Trusted Execution Environment" is an execution environment which runs parallel to a rich operating system (OS), like Android. This is usually achieved through virtualization of the main processor thus creating two environments, a rich environment where the rich OS performs its tasks and another trusted, isolated environment, the TEE. The specification for the TEE is developed by GlobalPlatform. The most prominent implementation of the TEE is the "TrustZone" (ARM).

The TEE was designed for the growing market of connected devices including but not limited to mobile phones. Drawbacks of the increasing usage of smart devices is its expanding attack surface hence the need for security in this area. The main motivation for the creation of an open standard was to addresses this problem.

As mentioned earlier, the TEE is an execution environment which allows the execution of trusted code. During the secure boot process, the TEE is authenticated and isolated. This step builds the necessary root of trust. Trusted code which is executed within the isolated environment is called “Trusted Application” (TA). These applications have a higher level of security than applications which run in the rich environment and offer some functionality to the rich OS. The two levels of security are realized by the TEE by preventing access from the rich environment to the hard- and software resources of the TEE. Such resources could for example be a secure key storage implemented in hardware or a so-called “Secure Element” (SE). Furthermore, the TAs gain access to the internal APIs of the TEE [146].

It is only possible for the rich OS to communicate with the TEE via well-defined channels. TAs within the TEE are also isolated from each other and it is not possible for a TA to gain unauthorized access to any data used by another TA. The basic architecture of a TEE, including TAs and the rich environment can be seen in Figure 22.

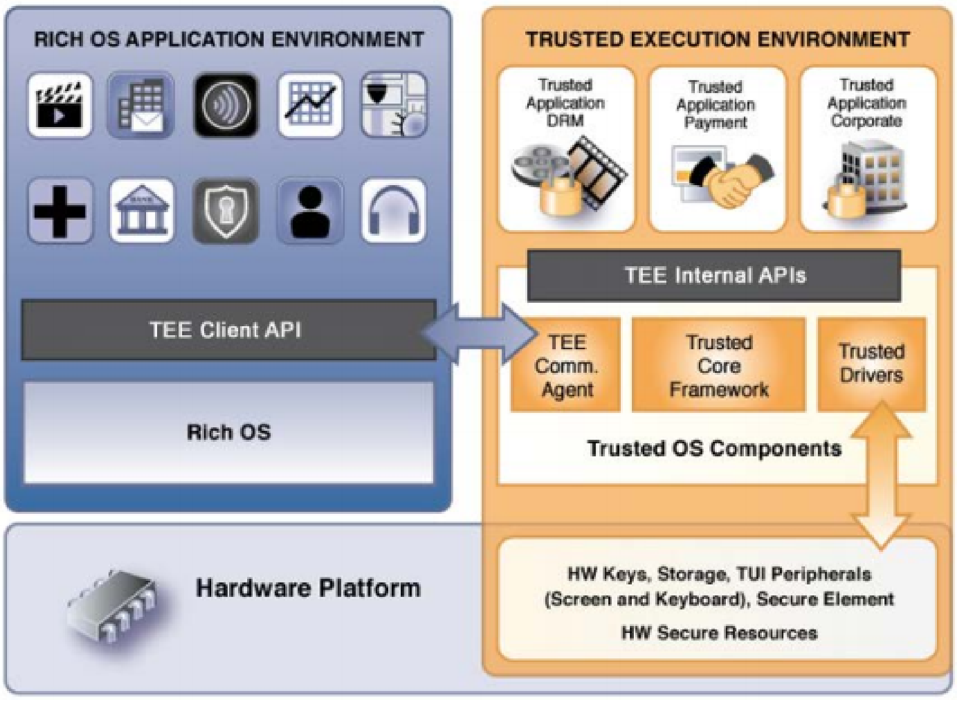

RICH OS APPLICATION ENVIRONMENT
TRUSTED EXECUTION ENVIRONMENT
Trusted Application DRM
Trusted Application Payment
Trusted Application Corporate
TEE Internal APIs
TEE Client API
TEE Comm. Agent
Trusted Core Framework
Trusted Drivers
Rich OS
Trusted OS Components
Hardware Platform
HW Keys, Storage, TUI Peripherals (Screen and Keyboard), Secure Element
HW Secure Resources

Figure 22: Architecture of the TEE [146]

With version 4.3 (API level 18) Android improved its security framework by introducing the possibility for hardware-backed keys. Android introduced a new Java Security Provider, namely the “AndroidKeyStore” provider, which offers an interface to generate and save keys in secure hardware. Once the key resides in secure hardware it remains non exportable. Also the Android OS has no read-access to the key hence the key is bound to the device. This also applies for rooted devices. On non-rooted devices a created key can only be used by the application which created the key (identified with the UID), but it may be possible for rooted devices to circumvent this security feature. The Android KeyStore framework was further enhanced with Android version 6.0 (API level 23). As things stand at the moment (December 2016), according to Android Studio, 76.9% of all Android Devices support API level 18 or higher. Anyhow, if the device doesn't support hardware-backed keys, for example if the device lacks the necessary hardware, the KeyStore API defaults to a software solution. Furthermore, the "AndroidKeyStore" provider only supports a predefined set of cryptographic algorithms. This set is not extensible.

The security of Android devices (KeyStore API) usually relies on the TEE. Most chip manufacturers for Android are integrating a TEE in modern Android devices, namely the "ARM TrustZone". This includes Qualcomm (market leader), Texas Instruments and Samsung. The TrustZone is a TEE which introduces two "worlds" that run in parallel, a "secure world" and a "normal" world. As the names imply, the normal world is where the rich OS, Android in this case, resides (thus non-secure) and in the secure world trusted code is executed.

Nevertheless, it is not guaranteed that the TrustZone or any other TEE is implemented in an arbitrary Android device. It is also possible that an Android phone is shipped with no form of TEE incorporated. Anyhow, most middle- to high-end Android phones are using the TrustZone.

D Details for Identity and Access Management Protocols

In the following sections, we provide details for the identity and access management technologies, which were evaluated and assessed in Section 6. First, Appendix D.1 provides details on the identity protocols OpenID Connect and SAML. Then, Appendix D.2 further describes the authorization protocols OAuth, UMA, and WS-Trust. Appendix D.3 explains policy languages, namely WS-Policy and XACML. Subsequently, Appendix D.4 presents the W3C Web Crypto API and KMIP. Finally, Appendix D.5 details SCIM.

D.1 Identity Protocols

This section completes the fact sheets of OpenID Connect and SAML from Section 6.1.1.

D.1.1 OpenID Connect

OpenID Connect [154] is an identity protocol on top of OAuth 2.0 [84], which allows service providers (SP) to delegate user authentication to an identity provider (IdP), to obtain data about the user, and to gain authorization to access additional resources. As a result of this process the identity provider issues JSON Web Tokens (JWT) to the service provider, containing identity and attribute information. By offering these data as JWT, confidentiality, integrity and authenticity can be ensured. By delegating the authentication to the user's preferred identity provider, usability improvements can be achieved. The user only has to memorize one set of credentials for her identity provider instead of sets for each service. Furthermore, with single sign-on (SSO), information about the user's recent authentication is stored and can be reused to complete a subsequent authentication request without requiring user interaction. OpenID Connect supports server-side and client-side web applications as well as mobile apps. Extensions for OpenID Connect also include the automatic discovery of a user's identity provider, dynamic registration of new service providers, and session management.

This section is organized as follows: To begin with, the general authentication process and the involved steps are described. The subsequent paragraphs explain the optional extensions for discovery, dynamic registration, and, finally, session management.

Authentication Process

The authentication process of OpenID Connect involves interaction between user, service and identity provider. In order to access a protected resource at the service provider, the user first has to authenticate via the identity provider. Then, the proof of authentication, the user's attributes, as well as authorization to access further resources have to be transmitted back to the service provider. The service provider can use this information to request further data about the user and, ultimately, reach an access control decision regarding the user's original request. Figure 23 illustrates the steps of the OpenID Connect process for the so called authorization code flow. These following paragraphs elaborate on these steps.

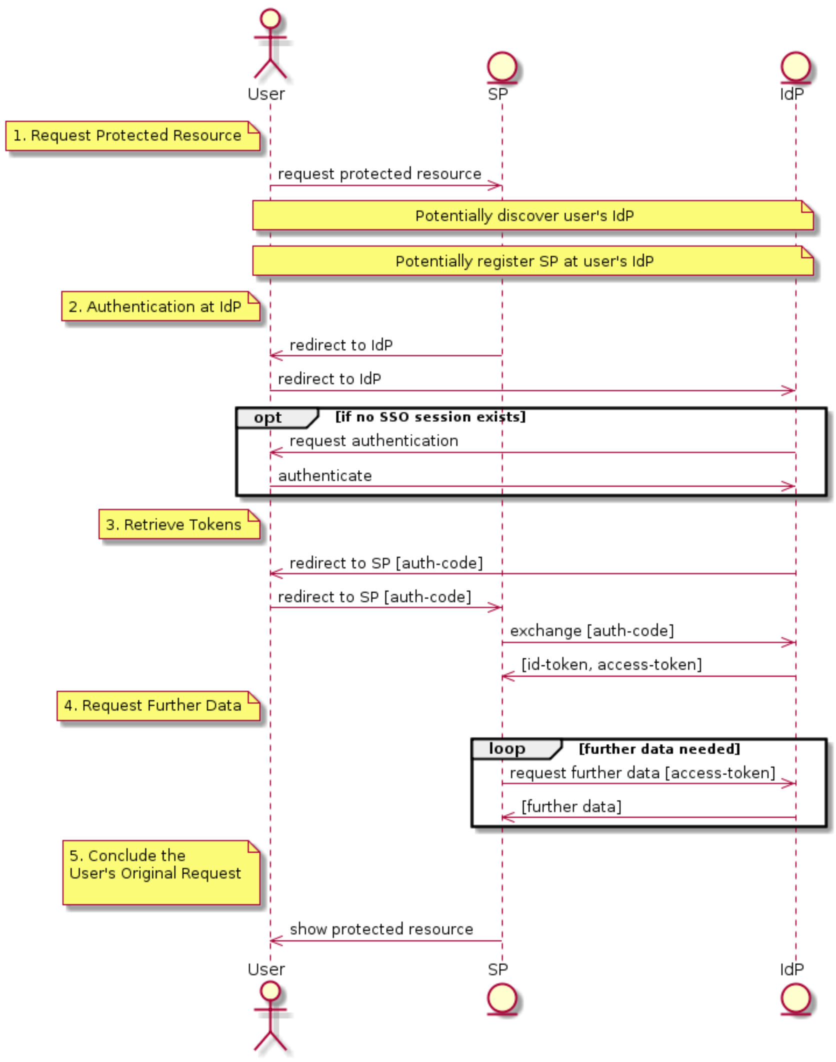

User
SP
IdP
1. Request Protected Resource
request protected resource
Potentially discover user's IdP
Potentially register SP at user's IdP
2. Authentication at IdP
redirect to IdP
redirect to IdP
opt
[if no SSO session exists]
request authentication
authenticate
3. Retrieve Tokens
redirect to SP [auth-code]
redirect to SP [auth-code]
exchange [auth-code]
[id-token, access-token]
4. Request Further Data
loop
[further data needed]
request further data [access-token]
[further data]
5. Conclude the User's Original Request
show protected resource
User
SP
IdP

Figure 23: Authentication with OpenID Connect

1. **Request Protected Resource:** When a user tries to access a protected resource at the service provider, this service provider has to make an authorization decision based on the user's authenticated identity or her attributes. With OpenID Connect, this process of user authentication and attribute acquisition can be delegated to an identity provider. In order to perform such a delegation, the service and identity provider have to be associated, which can be established with extensions for discovery and dynamic registration.

2. **Authenticate at Identity Provider:** The service provider redirects the user's browser to the identity provider with a request for authentication and a list of required permissions. The authentication is performed out of band at the identity provider, which allows the integration of a multitude of authentication methods. The resulting proof of authentication and required attributes are compiled into an id-token. In addition, an access-token representing authorization to access further resources at the identity provider is generated.

3. **Retrieve Tokens:** There are three methods, called flows, that define how these id- and access-tokens can be transmitted back to the service provider. Firstly, in the authorization code flow, the identity provider redirects the user's browser to the service provider. The authentication code is added as GET parameter to the destination URL pointing at the service provider, which then exchanges this code for tokens. This authorization code flow is illustrated in Figure 23. Secondly, in the implicit flow, the id-token and access-token are sent directly back to the service provider again as parameter in a redirect. Thirdly, in hybrid flows, one token is sent back directly, whereas the other token has to be obtained by exchanging the authorization code.

4. **Request Further Data:** In some scenarios, the service provider requires further data associated with the user. To access these data, the user has to prove her previously obtained authorization. Of particular interest regarding the OpenID Connect process are the OAuth 2.0 Bearer Token [108] specification as well as the JSON Web Token profile for OAuth 2.0 [107]. According to the OAuth 2.0 Bearer Token specification, the access-token can be included in the request to demonstrate authorization. Figure 23 illustrates this Bearer Token usage. In the JSON Web Token profile for OAuth 2.0, authorization can be demonstrated with JWT tokens, such as the id-token.

5. **Conclusion of the User's Request:** After the service provider's requirements have been fulfilled, the initial request by the user for a protected resource can be fulfilled. The authenticity of the user and her attributes have been presented by the obtained id-token. In addition, the service provider was able to request further services offered by the identity provider.

Discovery Process

The discovery extension of the OpenID Connect specification allows to determine the user's preferred identity provider and its configuration. This discovery process is based on usernames that reference hosts. By querying such a specified host, the service provider learns the URL of the user's preferred identity provider. With this URL, the service provider is able to obtain the identity provider's configuration. The following lines describe the individual steps of the discovery process, as illustrated in Figure 24.

1. **Obtain Username:** In order to perform the authentication process triggered by the user's attempt to access a protected resource, the service provider first has to obtain the username. Therefore, the service provider presents the user a form asking for her username. Valid usernames follow schemes that include a hostname, such as `joe@example.com:8080` or `https://example.com/joe`.

2. **Discover User's Identity Provider:** The service provider sends a *WebFinger* [110] request to the host extracted from the username in order to determine the user's preferred identity provider. This request is sent to a standardized endpoint and contains the username. Based on this information, the *WebFinger* service responds with a URL to the user's preferred identity provider.

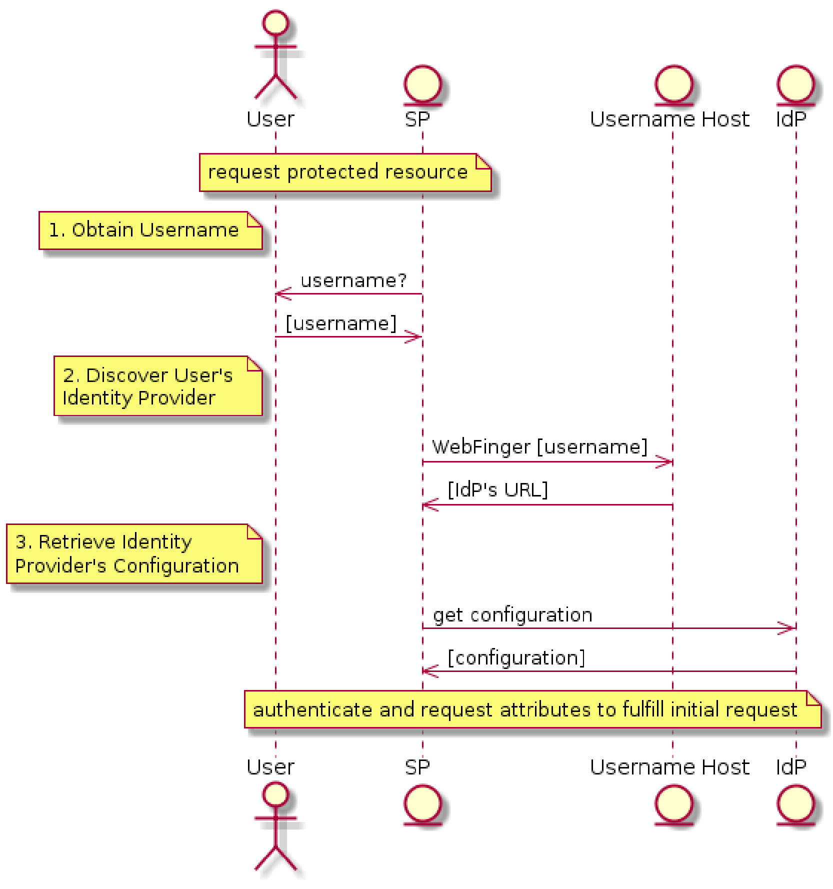

User
SP
Username Host
IdP
request protected resource
1. Obtain Username
username?
[username]
2. Discover User's Identity Provider
WebFinger [username]
[IdP's URL]
3. Retrieve Identity Provider's Configuration
get configuration
[configuration]
authenticate and request attributes to fulfill initial request

Figure 24: Discovery of Identity Providers with OpenID Connect

3. **Retrieve Identity Provider's Configuration:** In the final step, the service provider retrieves the identity provider's configuration. The previously obtained URL specifies the identity provider's host. A request to a standardized path at this host returns configuration data, such as necessary endpoints, supported cryptographic algorithms, and the locations of key material.

Dynamic Registration Process

The OpenID Connect extension for dynamic registration allows to establish an association between service providers and identity providers. During this process the service provider presents information about itself to the identity provider and in turn obtains information required to use the identity provider. The exchanged information includes supported optional functionality as well as cryptographic algorithms and key material. As illustrated in Figure 25, the extension for dynamic registration specifies two steps, which are described in further detail below.

1. **Register Service Provider at Identity Provider:** In order to register, the service provider sends information about its configuration to the identity provider. Additionally,

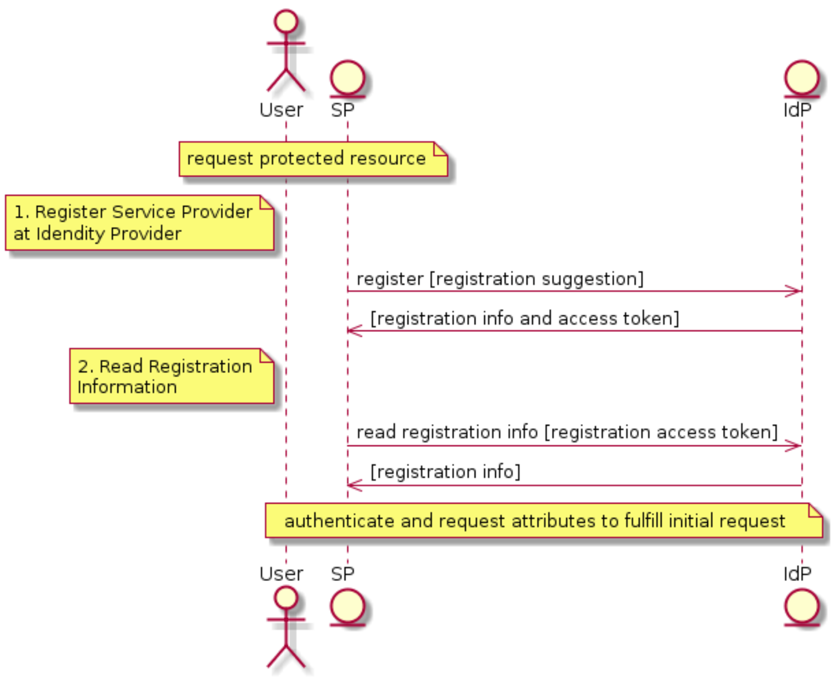

User
SP
IdP
request protected resource
1. Register Service Provider at Idendity Provider
register [registration suggestion]
[registration info and access token]
2. Read Registration Information
read registration info [registration access token]
[registration info]
authenticate and request attributes to fulfill initial request
User
SP
IdP

Figure 25: Registration of Service Providers with OpenID Connect

based on the previously obtained identity provider's configuration, the service provider is able to select supported cryptographic algorithms. The identity provider takes these options into account, but ultimately defines which methods are used. As a result, the service provider receives a document specifying the parameters of the association, as well as a registration access token, which can be used to obtain the registration information at a later point in time.

2. **Read Registration Information:** With the previously obtained registration access token, the service provider is able to get a new copy of the registration information.

Session Management

Even though identity and service provider maintain their own sessions, an overarching concept of session management allows to coordinate these sessions. The identity provider might offer a long-term single sign-on session, which is opened when the user first authenticates, in order to skip the need for user interaction during immediate further requests. Service providers typically open a session after receiving and verifying the id-token issued by the identity provider. However, the service provider's session is derived from the identity provider's session, which exists because of the user's actual authentication. Therefore, identity and service provider have reason to coordinate their sessions.

The issues regarding session management and coordination were considered in further specifications for OpenID Connect. OpenID Connect Session Management [48] defines how iframes can communicate through HTML5 Cross-Document Messaging [85], in order to check the current session state and track changes. In addition, specifications for front-channel [104] and back-channel [105] logout propose ways to close the identity and service provider's session simultaneously, indirectly via the user agent or, respectively, directly between identity and service provider.

D.1.2 SAML

Security Assertion Markup Language (SAML) is an XML-based framework for communicating user authentication and attribute information developed by Security Services Technical Committee of OASIS [83]. In this document we refer to the latest version of SAML, version 2.0, which was released in 2005.

SAML has been developed to solve difficulties in the authentication and authorization procedure. Moreover, it has been developed for use cases such as Single Sign-On (SSO), authorization service and back office transaction. The way these use cases are realized is based on a standard message format (XML), a standard message exchange protocol and rules for the message transport. This standards and rules are increasing the interoperability.

Figure 26 shows the **typical usage** of an authentication procedure utilizing SAML. In the scenario of the basic concept is a user trying to access a service. This service is provided by the so-called Service Provider (SP), which can provide different kinds of services. The first step the SP takes after receiving the access request from the user, is verifying if the user is already authenticated or not. In this scenario we assume that the user hasn't been authenticated yet. In this case the SP is trying to identify the user's Identity Provider (IdP). An IdP is an entity which is able to authenticate and authorize the user. Next, the SP requests a user authentication from the identity provider. At this point, the identity provider is trying to authenticate and authorize the user. If this procedure is successful, a token will be created with information about the user, including authentication and authorization information related to the specific user. The SP gets the token from the IdP. The two providers in this scenario know and trust each other. Therefore, the service provider can use the SAML token created by the identity provider to allow or deny the user access to the requested service.

The **advantage** towards utilizing SAML is having the possibility to create tokens for different services on the same service provider as well as having the possibility to authenticate the user from other service providers. This means that we have a centralized place, the identity provider, for authentication and authorization of users. Another advantage is that the service provider does not have to care about how to authenticate the user, such as storing the user's username, password and other additional information.

One of the main reasons for using SAML is the usage of Single Sign-On (SSO). The SSO principle enables the user to only sign in once to gain access to more than one application for a predefined period of time. SSO is enabled by using for example a cookie in the web browser which identifies the sign on session.

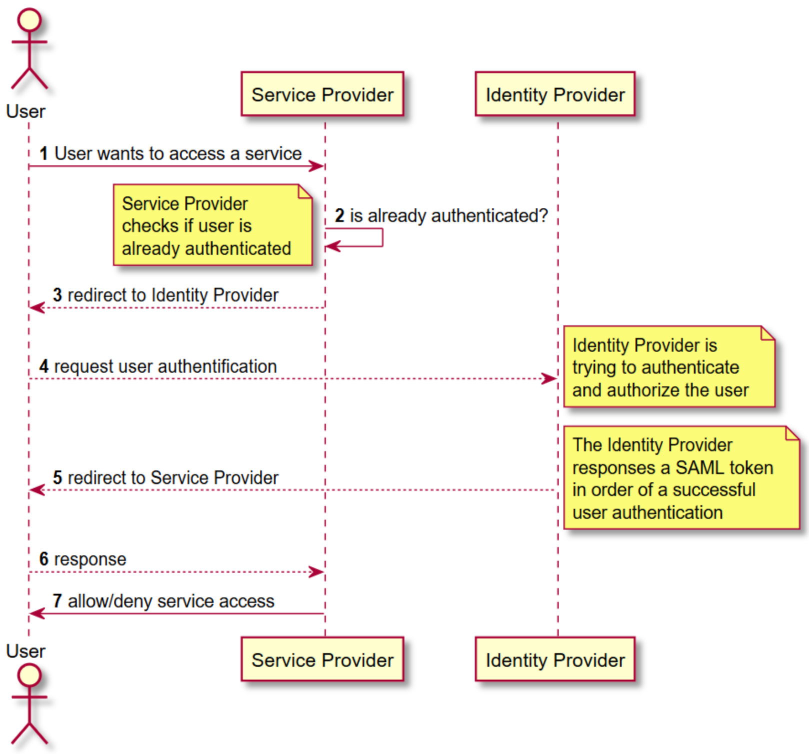

User
Service Provider
Identity Provider
1 User wants to access a service
Service Provider checks if user is already authenticated
2 is already authenticated?
3 redirect to Identity Provider
4 request user authentification
Identity Provider is trying to authenticate and authorize the user
The Identity Provider responses a SAML token in order of a successful user authentication
5 redirect to Service Provider
6 response
7 allow/deny service access
User
Service Provider
Identity Provider

Figure 26: SAML Basic Concept

Available SAML **open source implementations** are listed on the OASIS [9] website. These implementations are available in different programming languages not to mention different combinations with other protocols or software tools such as a GUI.

Specification

The SAML architecture is split into various components which are shown in Figure 27. Putting these components together allows SAML the support of several use cases. Each component is described in detail in the following.

Assertions: Assertions are one of the core parts in SAML which can enclose several statements such as properties and attributes of a subject. Assertions, issued by the identity provider, are used for access decisions from the service provider. The assertion structure and content is defined by an XML schema. The three possible statements enclosed in the assertion are detailed below.

- **Authentication Statements:** This statement is typically created by an entity such as an identity provider, which is in charge of the user's authentication information.

[9] `http://saml.xml.org/wiki/saml-open-source-implementations`

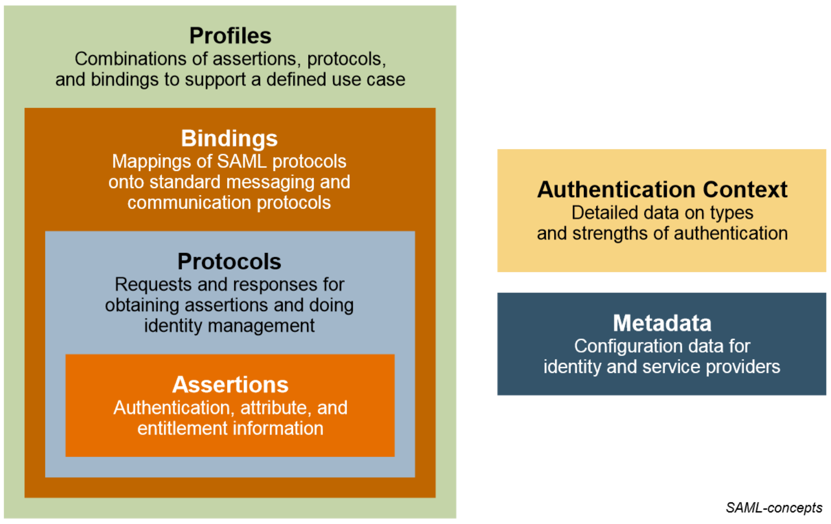

Profiles
Combinations of assertions, protocols, and bindings to support a defined use case
Bindings
Mappings of SAML protocols onto standard messaging and communication protocols
Protocols
Requests and responses for obtaining assertions and doing identity management
Assertions
Authentication, attribute, and entitlement information
Authentication Context
Detailed data on types and strengths of authentication
Metadata
Configuration data for identity and service providers
SAML-concepts

Figure 27: SAML's Main Components [133]

The authentication statement is part of an assertion which is created after a user has successfully been authenticated.

- **Attribute Statements:** Attribute statements contain properties and attributes used for defining access control to an application. The specified subject is related to the identifying attributes and properties.
- **Authorization Decision Statements:** Authorization statements have been developed for authorization. The authorization refers to whether a user is permitted to access a service.

Protocols: The SAML protocols define the structure of SAML request messages and SAML response messages. Assertions are either requested from the service provider or are pushed from the identity provider and exchanged using various SAML protocols. SAML defines a number of protocols such as the following:

- **Authentication Request Protocol:** This protocol defines an authentication request message that triggers a response message which contains one or more assertions. Typically, the issuer of the authentication request is the SP and the response is issued by the IdP.
- **Single Logout Protocol:** This is a mechanism to log out of active sessions which can be triggered by a session time out, the user, the IdP or the SP.
- **Assertion Query and Request Protocol:** This protocol defines a set of queries which can either be based on a reference, subject or the statement type.
- **Artifact Resolution Protocol:** The Artifact Resolution Protocol provides a mechanism which is used to refer to an assertion by an artifact. This artifact is typically used by the service provider to obtain the actual assertion using this protocol.

- **Name Identifier Management Protocol:** The issuer of a name identifier request can be both the service provider and the identity provider. The protocol provides a mechanism to terminate an association of a name between the IdP and SP.
- **Name Identifier Mapping Protocol:** This protocol provides a mechanism to enable "account linking". Account linking can be used in browser based SSO where a user maintains separate accounts.

Bindings: The concept of the SAML protocol bindings describes mappings of request-response message exchanges to communication and standard messaging protocols. A so-called SAML binding is for example a mapping of SAML request-response message exchanges into a specific communication protocol. The possible SAML bindings are listed and detailed as follows:

- **SAML SOAP Binding:** Simple Object Access Protocol (SOAP) is a network protocol which allows a decentralized exchanging of structured information in a distributed environment. SOAP is based on XML technology and follows two main principles which are simplicity and extensibility. This binding defines how SOAP is used to send and receive SAML requests and responses.
- **Reverse SOAP (PAOS) Binding:** In the PAOS binding, the HTTP requester acts like a SOAP responder or SOAP intermediary which is able to process SOAP messages which contain SAML messages.
- **HTTP Redirect Binding:** This binding describes a mechanism where SAML protocol messages can be sent within URL parameters.
- **HTTP Post Binding:** The HTTP POST binding is utilized for more complex SAML protocol messages. This is necessary for long messages because even when the URL length is theoretically infinite, in practice it is limited. Therefore, the messages are transferred base64-encoded within a HTTP POST.
- **HTTP Artifact Binding:** In this binding, the SAML requests, responses or both are transmitted by reference which is the so-called artifact. This binding can be composed with the HTTP redirect binding or the HTTP POST binding.
- **SAML URI Binding:** The SAML URI binding is used to support encapsulation of a SAML assertion request message with a single assertion reference utilizing one unique resource identifier (URI). This is based on the principle that URIs are referring to a resource in a protocol independent way.

Profiles: SAML Profiles basically describe a formulation on how SAML assertions, protocols and bindings are combined. Moreover, a SAML profile defines extensions and constrains of the usage of SAML for a specific application. Possible SAML profiles are listed as follows:

- Web Browser SSO Profile
- Enhanced Client and Proxy (ECP) Profile
- Identity Provider Discovery Profile

- Single Logout Profile
- Name Identifier Management Profile
- Artifact Resolution Profile
- Assertion Query/Request Profile
- Name Identifier Mapping Profile

SAML profiles support two general message flows in relation to where the message flow was initiated. The former and more common scenario is shown in Figure 26 and the latter scenario is shown in Figure 28.

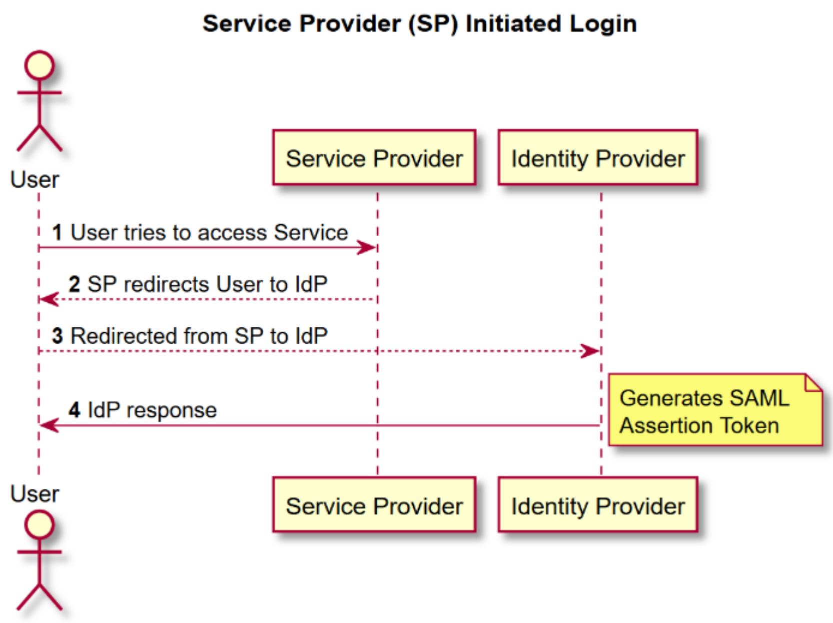

Service Provider (SP) Initiated Login
User
Service Provider
Identity Provider
1 User tries to access Service
2 SP redirects User to IdP
3 Redirected from SP to IdP
4 IdP response
Generates SAML Assertion Token
User
Service Provider
Identity Provider

Figure 28: Identity Provider Initiated Login

SAML **metadata** defines a specification of an extensible metadata format utilized in a standardized way by SAML system entities. Metadata may contain URLs, supported SAML profiles, unique identifiers such as the provider ID or associated digital certificates.

Privacy in SAML

SAML is often deployed in cases where privacy is very important. It supports various mechanisms which enables the deployment in a privacy preserving way.

- Pseudonyms are used in SAML which are ascertained by a particular identity provider and a service provider.
- SAML supports ephemeral one-time identifiers which ensures that the service provider is unable to recognize a certain user access utilizing a single sign-on operation.

- The authentication context mechanisms provided by SAML allow the user to be authenticated at a sufficient assurance level.

Security in SAML

SAML defines various security mechanisms to detect and protect against attacks such as the "man-in-the-middle" attack. The primary mechanism is a pre-existing trust relationship between the relying party and the asserting party. This trust relationship is typically related to a Public Key Infrastructure (PKI) whose usage is not mandatory in SAML but highly recommended. Previously mentioned security mechanisms utilized in SAML would include TLS 1.0 or SSL 3.0 for transport level security. As for the message level security, different techniques are utilized such as XML signature (XMLDsig) or XML encryption.

D.2 Authorization Protocols

This section completes the fact sheets of OAuth, UMA, and WS-Trust from Section 6.1.1.

D.2.1 OAuth

OAuth [84] is a standardized protocol that is used to authorize a third-party application to access a web service in the name of a user. OAuth operates in the web environment. For example, by using OAuth a user grants an application access to the user's files, which reside on a cloud storage provider. This section describes the version 1.0 of OAuth. Figure 29 shows the OAuth authentication process. The following paragraphs describe this process in more detail.

The OAuth authorization process is performed if a user visits a web application that requires the user's authorization to access a web service. This process runs through the following four steps:

In the first step, the web application sends a request for a temporary token to the web service. This request contains a client id, which was obtained at the registration of the web application at the web service. The web service responds with a newly generated temporary token and saves the association between the received client id and the temporary token.

In the second step, the web application responds to the initial request by redirecting the user's browser to the web service. This redirected request contains the previously gained temporary token. At the web service, the user is asked to authenticate. As the web service saved the client id for the received temporary token in the first step, the service is able to identify the web application. After authentication, the user is asked if authorization for the web application should be granted. If the user approves, the browser is redirected back to the web application.

In the third step, the web application is notified that the authorization process is complete by the redirect. It sends the web service a request to exchange the temporary token with an access token. The temporary token is only used for the authorization process. In contrast, the access

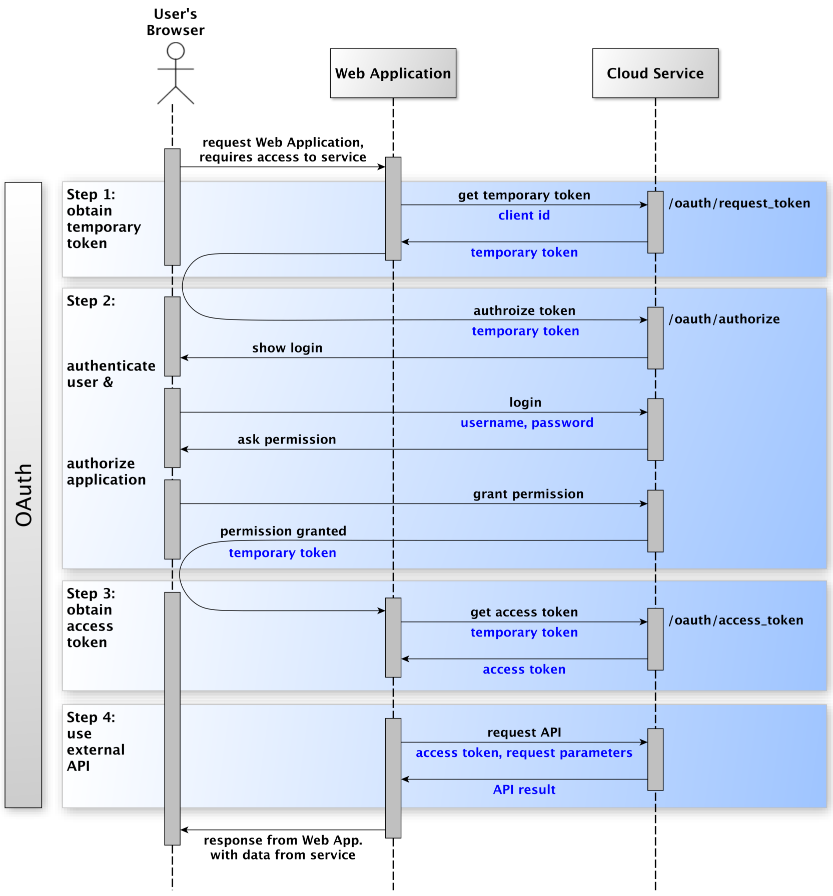


Figure 29: OAuth Process Flow

token is used to perform the operations of the web service that require authentication. After the web application received the access token, the authorization process is completed.

Finally, in the fourth step, the web application is allowed to use the functions provided by the web service in the name of the user. As proof of the permission, the application includes the access token in each request to the external API. With the functions of the web service and the data the user stored on it, the web application is able to fulfill the user's requests.

D.2.2 UMA

On 23rd March 2015, the Kantara Initiative [10] has approved the version 1.0 of the User-Managed Access (UMA) protocol standard. UMA is an access management protocol based on OAuth 2. Utilizing the UMA protocol allows users to authorize access to their online resources. These online resources can consist of personal data such as identity attributes, general content such

[10] `http://kantarainitiative.org`

as pictures, and services. In other words, utilizing UMA enables the user to precisely manage the access to his/her resources. [114]

This section is organized as follows: We first describe new terms the Kantara Initiative has introduced in the context of UMA. We explain the UMA protocol standard specification and provide a high level overview of the UMA flow. Then, we explain the connection of UMA to other standards. Finally, we give details about the features of UMA.

UMA Terminology

The Kantara Initiative has introduced new terms in context of UMA. These terms are used in the UMA specification, therefore, the important terms are described as follows. [114]

- **Resource Owner (RO):** The RO is the owner of an OAuth resource in a User-Managed Access (UMA).
- **Resource Server (RS):** The RS describes the server where the resources are located managed by the resource owner (RO).
- **Authorization Server (AS):** The AS is a server which is responsible to authorize the resource access for requesting parties.
- **Requesting Party (RqP):** The RqP describes the end user, which is using a client to request access a protected resource. The RqP is usually not the resource owner it is a user who wants to access a resource which belongs to the resource owner.
- **Client:** The client is an application used to request access to a protected resource.
- **Protection API Token (PAT):** The PAT is an OAuth access token utilized by the Resource Server (RS) at the protection API.
- **Authorization API Token (AAT):** The AAT is an OAuth token used for accessing the authorization API by the Client.
- **Requesting Party Token (RPT):** The RPT is an UMA access token which can be used, together with the related authorization data, by the client to get access to a resource on the resource server.
- **Authorization Data:** Authorization Data are data related to a RPT which are used to by the client to access a resource at the RS. Authorization data are created by the authorization server.
- **Policy:** A policy is the configuration of the authorization server consisting of configuration parameter.
- **Protection API:** The protection API offers an interface used to access the authorization server. Only authorized entities can communicate with the protection API using the protection API token.

- **Authorization API:** The authorization API is utilized by the authorization server to get authorization after authentication. Clients need credentials at the authorization API together with the authorization API token (AAT).

High Level Flow of UMA

The UMA protocol specifies how the Resource Owner (RO) controls the protected resource access by the clients. It is a profile of OAuth 2 and consists of a set authorization and consent APIs. UMA basically allows a user to precisely define rules on how to share an online resource and with who using policies. A so-called Requesting Party (RqP) can then request access to the resource.

UMA consists of three phases which are illustrated in Figure 30 and described as follows. [114]

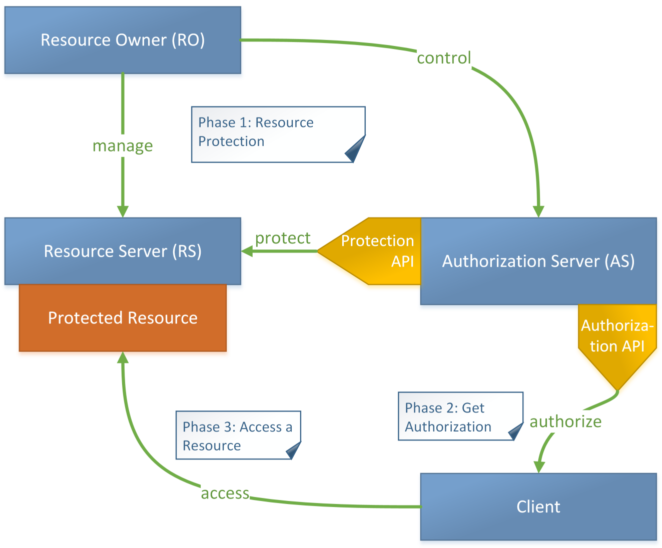


Figure 30: UMA's Three Different Phases [112]

The high level flow diagram shown in Figure 31 is explained in detail as follows [59] [113]. Moreover, the list bellow identifies the single steps of the high level flow of UMA as well as the three phases of the process flow.

Phase 1 - Protect a Resource:

- The Resource Owner (RO) selects the resources to protect at the Resource Server (RS) – this is out of band.
- The RO configures the Authorization Server (AS) using policies. These policies are related to a resource on the RS – this is out of band.

1. In the first step the RS registers the resource sets and scopes. To be able to communicate with the protection API a Protection API Token (PAT) is required. The PAT must be included in all protection API calls.

Phase 2 - Get Authorization:

2. A Requesting Party (RqP) requests access to a resource at the RS by using a client.
3. RS checks permission for attempted request at the AS.
4. After successfully checking the request a permission ticket is created by the AS and transmitted to the RS.
5. The RS returns an error 403 including the URI of the AS and the permission ticket.
6. The client is using the permission ticket together with AS's URI to request authorization data. Furthermore, the client needs an Authorization API Token (AAT) when communicating with the authorization API.
7. After successfully verify the request from the client by the AS, the AS generates and returns a Requesting Party Token (RPT) together with the authorization data. The verification is successful when the RO's policies have been met.

Phase 3 - Access a Resource:

8. The client utilizes the received data to request the resource at the RS.
9. The RS introspects the RPT at the AS.
10. The AS returns the status of the token to the RS.
11. Finally, the RS returns the status of the token and access to resource on success otherwise not.

Relation to other Technologies

UMA is based on OAuth 2 and may be used with OpenID Connect or SAML as identity protocol. Figure 32 shows a Venn-diagram of the UMA protocol standard and symbolizes the interconnection between the different technologies with the overlapping parts.

The different parts of the Venn-diagram are detailed as follows:

- **OAuth 2** is utilized to control access to web APIs. Furthermore, OAuth 2 is used for calling applications, which is based on authenticated identity.
- **OpenID Connect** can be used in this context for single sign-on (SSO). Moreover, the OpenID Connect protocol can be used to manage the session management and the identity claims retrieval.
- **User-Managed Access** allows the user to manage the access to his/her online resources stored on the resource server (RS).

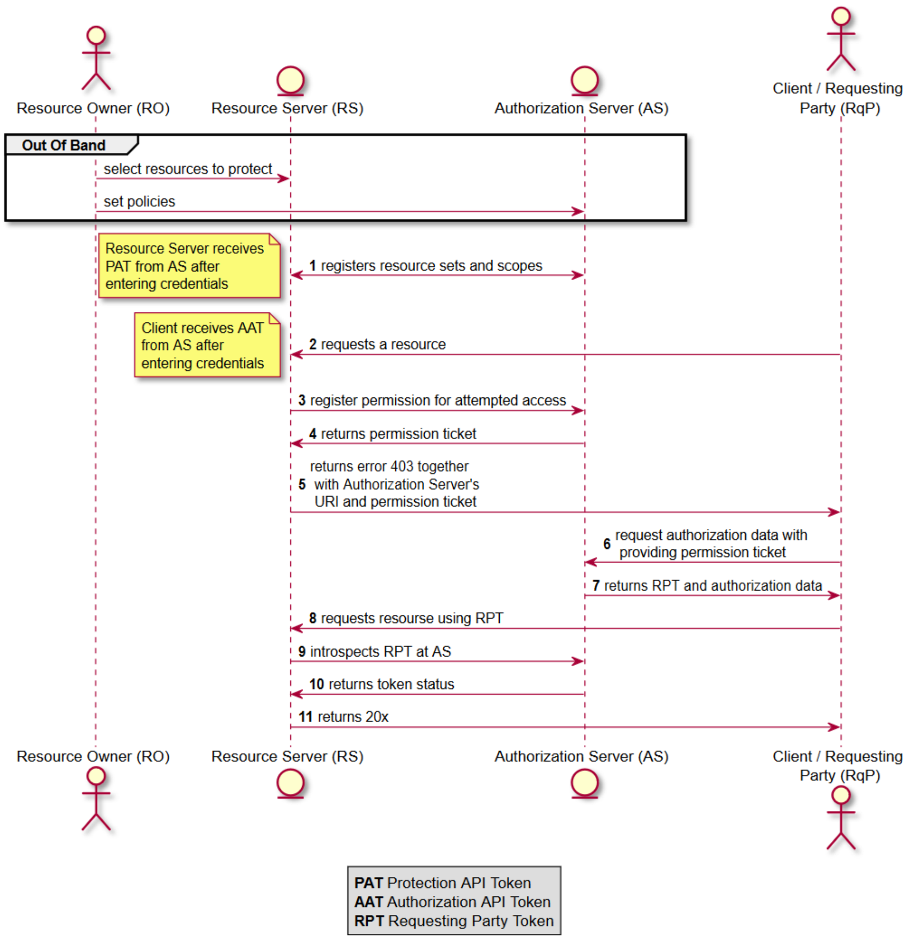

Resource Owner (RO)
Resource Server (RS)
Authorization Server (AS)
Client / Requesting Party (RqP)
Out Of Band
select resources to protect
set policies
Resource Server receives PAT from AS after entering credentials
1 registers resource sets and scopes
Client receives AAT from AS after entering credentials
2 requests a resource
3 register permission for attempted access
4 returns permission ticket
5 returns error 403 together with Authorization Server's URI and permission ticket
6 request authorization data with providing permission ticket
7 returns RPT and authorization data
8 requests resourse using RPT
9 introspects RPT at AS
10 returns token status
11 returns 20x
PAT Protection API Token
AAT Authorization API Token
RPT Requesting Party Token

Figure 31: UMA's High Level Flow [59, 113]

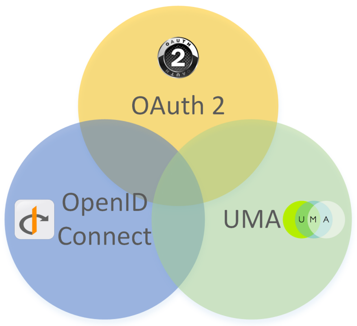


Figure 32: Venn-Diagram of the UMA Protocol Standard

Furthermore, additional technologies can also be included:

- **SAML** could be used instead of OpenID Connect as identity protocol.

- **XACML** could be utilized as extension to process the access policies. XACML standard contains the authorization policies, the XACML engine to process the policies and the definition how policies are being processed.

Features

In OAuth the connection between client and authorization server as well as the connection between client and resource server is unstated whereas UMA fully defines a standard interface between these components. The authentication of the requesting party at the authorization server is out of band for the UMA standard which leaves space to include any authentication method or protocol.

Utilizing UMA offers different features such as enabling the person-to-person sharing as well as the person-to-organization sharing where OAuth typically only person-to-self sharing allows. UMA allows that many pairwise connections can be managed, controlled and revoked at the same time.

In OAuth the authorization is usually based on simple authentication whereas UMA allows to create fine-grained policies. Utilizing these authorization policies drives to fine-grained and claim-based authorization decision. In OAuth are usually scopes defined and utilizing UMA instead allow the user to create arbitrary rules for the authorization policies.

D.2.3 WS-Trust

WS-Trust is an OASIS standard with the goal to construct trusted SOAP message exchanges. Trust relies in this standard on the use of security tokens which are introduced through WS-Security. It describes how security tokens can be issued, renewed, and validated by the communication partners.

The basic model of WS-Trust is described in Figure 33. It starts with the security token requester which requests a security token from a security token service. The security token service verifies the request, issues a security token and returns it to the requester. The requester can now provide this security token to the security token consumer, which is usually the service that the security token requester wants to access. By using the security mechanisms introduced by WS-Trust the security token consumer can verify the claims in the security token since it is issued by the security token service.

WS-Trust defines four types of bindings. Issuance Binding, Renewal Binding, Cancel Binding, and Validation Binding:

- The **Issuance Binding** describes how a client can request one or multiple Security Tokens and how the Security Token Service should respond those tokens. The response is usually a combination of the requested security token and a proof-of-possession token. The proof-of-possession token is cryptographically bounded to the security token. Thus the requester can proof with this token that he really possesses the security token.

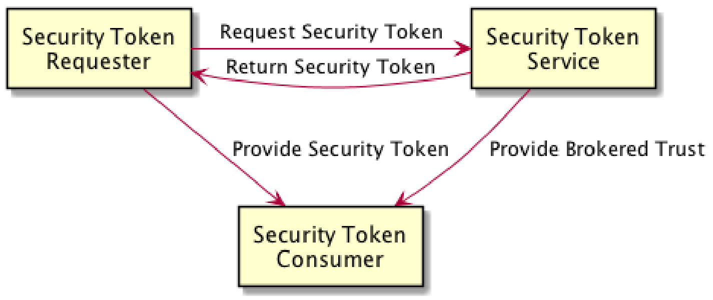


Figure 33: WS-Trust Model

- The **Renewal Binding** describes how an already issued token can be renewed. The requester has to provide the token or at least a reference to the token that has to be renewed. The renew operation can be applied to token that are already expired. It is up to the security token service to define a time limit for how long expired tokens can be renewed.
- The **Cancel Binding** describes how to disable already issued tokens. For example, a client has finished his business application and no longer needs the issued token. To prevent further use of it, the client invalidates it at the security token service. After a token is cancelled it can no longer be validated or renewed at the security token service.
- The **Validation Binding** describes how a security token can be validated at the security token service.

Before security tokens can be exchanged it is often necessary to negotiate and transmit cryptographic challenges between requester and security token service. The negotiation and challenge process used in WS-Trust is described in Figure 34: The Service A requests a token at Service B. This service responds with a challenge. The Service A computes an answer for the challenge and returns it to Service B. After verifying the answer, the Service B can issue and return a security token. Please note, that it is possible that Service A itself challenges the Service B at his first request. Thus Service B has to provide an answer itself.

WS-Trust specifies the following challenging modes:

- **Signature Challenge:** In this challenge a string is included in the Request Security Token Response or Request Security Token. The recipient of this challenge has to sign this string and returns it to the requester. It is up to the recipient and requester to negotiate a proper sign algorithm.
- **User Interaction Challenge:** In this challenge a user interaction should be integrated in the answer to this challenge. The default challenge can be either a textual information by the user or a choice selection between multiple alternatives. Further extensions of this challenge can include a secret PIN or a one-time-password (OTP) input by the user.

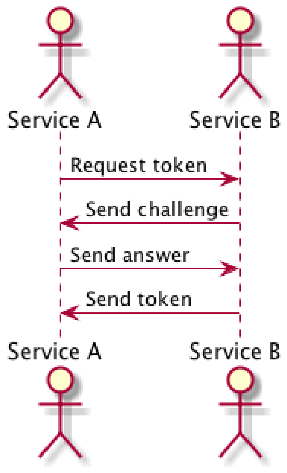


Figure 34: Negotiation Process in WS-Trust

- **Binary Exchanges:** In this challenge additional binary information can be passed through the negotiation phase. It is a good practice to use binary exchange information to establish a secure channel to secure the token and proof-of-possession token that should be issued later on.
- **Key Exchange Token:** Within this token a requester or issuer can exchange entropy or a key to the other party.
- **Custom Exchanges:** An extension point where custom XML-based exchanges can be specified in an own profile.

D.3 Policies

WS-Security and WS-Trust allows a highly flexible secure communication between partners. In order to establish a secure and trusted communication each partner should have knowledge about the desired level of security that is needed for the communication. For example, a service needs to know the authenticity of a requester that requires a signature of a message. In contrast a client wants to hide information in a request for potential attackers and thus wants to encrypt the message for the recipient. In both cases the server and the client needs to know these requirement thus that they can exchange for example key material in order to sign and encrypt the message parts. To express these needs the W3C consortium created the WS-Policy specification [11]. WS-Policy is a framework to express constraints or requirements for a web service or a client who can access web services.

Another policy technology is the eXtensible Access Control Markup Language (XACML). XACML provides a framework for a high flexible access control management system. Policies are used within the framework to express access control rules on resources. The following sections describe the WS-Policy and XACML technologies in detail.

[11]https://www.w3.org/TR/ws-policy/

D.3.1 WS-Policy / WS-SecurityPolicy

Figure 35 describes the underlying model for WS-Policy. The root element of a WS-Policy is a policy. It contains of zero or multiple policy alternatives. Each policy alternative expresses a choice while evaluating the policy. A policy alternative contains of zero or multiple policy assertions. Each policy alternative describes the effective behavior only implemented by the including policy assertions. A policy assertion finally is a concrete requirement, capability or other functional behavior that is stated about an entity. This entity is called the policy subject. A policy assertion can be further specified by a policy expression.

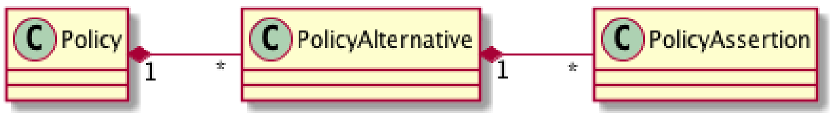


Figure 35: WS-Policy Model

Policies are defined in the XML Infoset representation straightforward to their model structure. The namespace for a WS-Policy is `http://www.w3.org/ns/ws-policy` and is abbreviated as „wsp“. The normal form of a policy uses the following three nested elements:

- **wsp:Policy:** The root element of a policy. All subsequent elements describe the characteristic of this policy.
- **wsp:ExactlyOne:** The list of policy alternatives. It is valid to have an empty list.
- **wsp:All:** Represents a policy alternative which itself is a list of policy assertions that have to be fulfilled.

Policy Intersection

Using policies between multiple parties often results in different policies between all participants. Since policies are used to express the requirements and needs by each different partner for a successful and secure communication both parties have to negotiate a reasonable policy which results in an agreement where both parties are satisfied. The process to determine such a policy is called policy intersection and depends on finding a compatible policy. Two policies A and B are compatible if (according to the WS-Policy spec):

- A and B have the same policy type
- either of them contains a nested policy expression, they are compatible if both of them have a nested policy expression and the alternative in the nested policy expression of one is compatible with the policy alternative in the nested policy expression of the other.

Keep in mind that this definition only covers the fundamental ruling of how the intersection of two policies works. Further specifications of various policy types should take into account to describe whether or not two policy types are compatible. In addition, WS-Policy does not describe the compatibility between policy assertions that use parameters. It is up to the specification to describe how the intersection functions between policies that use parameters.

WS-Security Policy

WS-Security Policy is an OASIS [12] standard which extends the WS-Policy model with security policies that defines security requirements for WS-SOAP Message model[13], WS-Trust[14] and WS-SecureConversation[15] for messages on a communication path.

WS-Security Policy defines a Security Assertion Model which consists of the following type of security assertions:

- Protection Assertion
- Conditional Assertion
- Security Binding Assertions
- Supporting Token Assertions
- WSS and Trust Assertions

Security assertions are associated with a scope of protection which identifies message parts that are protected in the specified way by a security assertion. Security assertions can be used in combinations to each other in order to qualify a specific assertion further.

Security policies targeting a subject during the message conversation. These subjects are identified by WS-PolicyAttachment[16] and can be:

- **Message Policy Subject:** Specifies that a policy is attached to a message which can be a wsdl:message itself or a wsdl:input, wsdl:output, or wsdl:fault inside of a wsdl:operation of a wsdl:portType or wsdl:binding.
- **Operation Policy Subject:** Tokens that are bound to an operation which can be on a wsdl:portType or a wsdl:binding.
- **Endpoint Policy Subject:** Tokens that are bound to an entire endpoint. They can be specified on a wsdl:portType, wsdl:binding or wsdl:port element.

[12] `http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.html`
[13] `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf`
[14] `http://docs.oasis-open.org/ws-sx/ws-trust/200512`
[15] `http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512`
[16] `http://www.w3.org/TR/2007/CR-ws-policy-attach-20070228/`

Protection Assertions

A Protection Assertion defines what is being protected and how strong the level of protection is. They should usually be defined for message policy subjects but can also be defined for operation and endpoint policy subjects meaning that either all messages of an operation or all operations of an endpoint are protected by this assertion.

There are three types of Protection Assertions:

- **Integrity Assertion:** Specifies how parts of a message are protected by integrity constraints. The protection can range from SOAP Message security mechanisms as well as external mechanisms like sending a message via HTTPS. The exact mechanisms are determined by the security binding assertion. Integrity Assertions can be a SignedParts Assertion which uses QNames to identify parts of a message that have to be signed as well as the SignedElements Assertion which uses XPath expressions to specify what has to be signed.
- **Confidentiality Assertion:** Specifies how parts of a messages are protected by confidentiality constraints. The protection can range from SOAP Message security mechanisms as well as external mechanisms like sending a message via HTTPS. The exact mechanisms are determined by the security binding assertion. Confidentiality Assertions can be a EncryptedPartsAssertion which uses QNames to identify parts of a message that have to be encrypted. It can be an EncryptedElements Assertion which uses XPath expressions to specify what has to be encrypted. It can be a ContentEncryptedElements Assertion which specifies elements which content needs to be protected by encryption mechanisms.
- **Required Elements Assertion:** Specifies which header elements in a message have to be present during the communication. It can be specified via XPath as well as QName references.

Token Assertions

A Token Assertion is used to specify the type of token that will protect a message. The following properties can be specified:

- **Token Inclusion:** Specifies if the token should be included in the message or, if not, that cryptographic mechanism should be used to retrieve the token.
- **Token Issuer:** This element describes the issuing authority of the token. Usually its pointing to the issuers endpoint address.
- **Required Claims:** Specifies the claims about an entity that a token of an issuer should carry. It is defined in the WS-Trust namespace. Claims are bound by issuing entities on tokens. Thus it is not necessary to carry all claims in one token but split them across multiple tokens.

- **Token Properties:** The Token Properties are used to specify if and how WS-SecureConversation should use derived keys.
- **Token Assertion Types:** Specifies the token that has to be used. WS-SecurityPolicy defines the following types of tokens: Username Token, Issued Token, X509 Token, Kerberos Token, SpnegoContext Token, Security Context Token, Secure Conversation Token, SAML Token, REL Token, HTTPS Token, KeyValue Token.

Security Binding

All of these information is bundled together in security bindings that describe how the security is performed during the communication. A binding specifies what tokens are used and how they are bound to messages, how keys are exchanged, what message elements are required, and additional parameters for various algorithms (for example canonicalization etc.). WS-SecurityPolicy defines three types of security bindings. The TransportBinding defines that a communication is secured by other means than WS-Security, for example HTTPS. The SymmetricBinding defines message protection by means of WS-Security. The Assertion allows to define Encryption and Signature Tokens. Both are used for the communication from sender to receiver and from receiver to sender. The AsymmetricBinding defines message protection by means of WS-Security. Signature and Encryption Tokens are split into tokens that are used by the Initiator and tokens that are used by the recipient.

Supporting Tokens

The Security Binding defines which tokens are used for a message signature and message encryption. However, if those tokens do not provide enough claims about the entity additional tokens can be defined for a message exchange. Those tokens are usually attached to the security header and are protected by the means of the security binding's token. Nevertheless, a supporting token can carry cryptographic material and thus is able to sign or encrypt parts of a message in addition to the security binding's token.

D.3.2 XACML - eXtensible Access Control Markup Language

eXtensible Access Control Markup Language (XACML) is an OASIS[17] standard which specifies schemas for authorization policies. Besides the policies defines the XACML standard and also the policy decision engine [134].

A typical scenario would be that a user tries to access a service or resource. To get access to the requested service/resource, the request hast to be evaluated against policies. These policies have to be created before the request. The policies contain rules which are used in the decision making process. The computed authorization response can be “Permit”, “Deny”, “Indeterminate” or “Not Applicable” [174].

[17] `https://www.oasis-open.org/`

The XACML standard follows the decoupling principle which is realized with separating the enforcement part from the decision part and also the management part. Each of these parts are realized with the corresponding XACML system point. For the enforcement part, the Policy Enforcement Point (PEP) is used, for the decision making process the Policy Decision Point (PDP) and for the policy management the Policy Administration Point (PAP).

We first provide a high-level description of the together with the different system parts. Then, we identify parts and process flows of the XACML engine. Subsequently, the XACML Policy with its elements and attributes are detailed.

Architectural Overview

This section gives an overview of the XACML architecture. Figure 36 illustrates the high level overview of the XACML architecture.

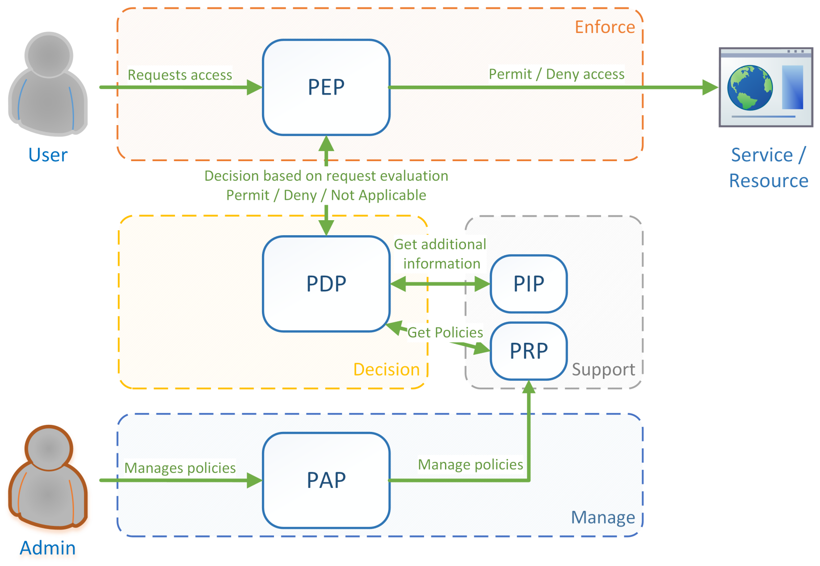


Figure 36: XACML High Level Architecture [134]

The **Policy Enforcement Point (PEP)** is responsible for protecting the access to a resource. This PEP receives the request to access a resource send by a user. After receiving the request, it forwards the request in form of an authorization decision request to the so-called Policy Decision Point.

The **Policy Decision Point (PDP)** evaluates authorization decision request against the policies in order to compute an access decision. Policies consists of a set of rules which are used to calculate authorization decisions. The PDP is using the so-called XACML engine to compute the access decision.

The PDP may requires additional information to compute the decision. It is getting additional information from the **Policy Information Point (PIP)** located in the support part shown in Figure 29. The PDP has to include all applicable policies in the decision making process. To get all policies which apply to a request the so-called **Policy Retrieval Point (PRP)** is used. This PRP can be an internal database as well as external/online storage places where policies are stored.

The admin is responsible for managing the policies using the **Policy Administration Point (PAP)**. This PAP allows the admin to create, delete or modify policies on internal and external storage.

XACML Engine

In this section, the XACML engine is described in detail. The XACML engine is used by the PDP to compute an access decision by evaluating the access authorization request against policies. Figure 37 depicts the process flow together with the main components of the XACML engine [134].

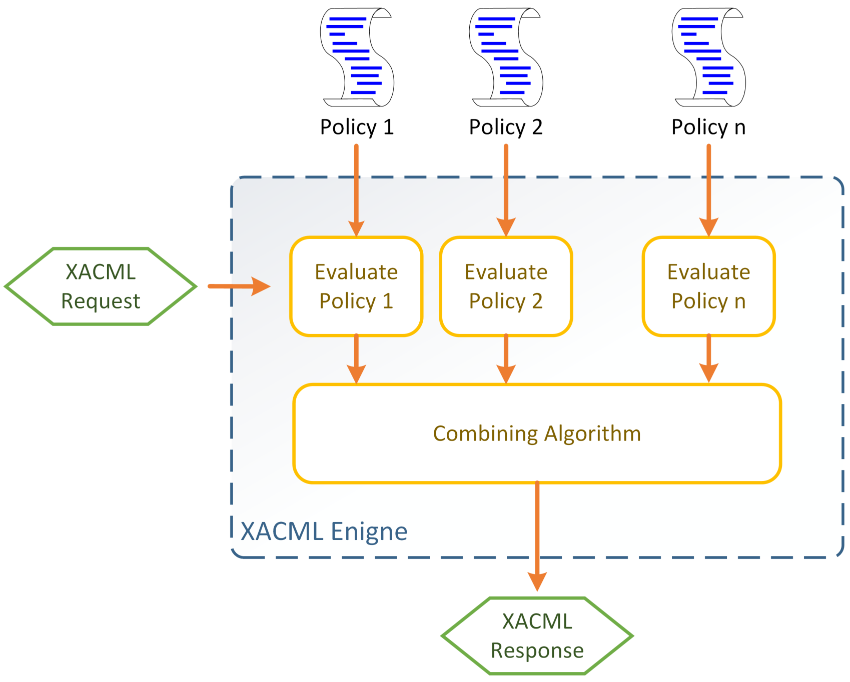


Figure 37: XACML Engine Process Flow [134]

The XACML engine takes as input the authorization decision request and also the applicable policies. One request can apply to more than one policy the so-called policy set where each policy can have more than one rule. The engine is using a combining algorithm to combine applicable policies and rules. This **combining algorithm** describes the logical way how the policies have to be combined and processed. There are different combining algorithms defined which are identified in the XACML specification [134]. Two examples of possible algorithms are:

- **Permit-Overrides:** This algorithm will return permit if any decision of a single rule is permit. Even if another applicable rule results deny, the decision is still permit. The permit overrides all other decisions and the algorithm will stop after the first permit is computed. To provide a simpler view, by disregarding the special effects indeterminate and not applicable, this combining algorithm acts like a logical OR.
- **Deny-Overrides:** This is the opposite of the permit-override algorithm which returns deny after the first deny has been computed. This combining algorithm acts also like a logical OR, when not considering the indeterminate and not applicable effect.

After evaluating the request, the engine has three different return values which are identified as follows.

- **Permit:** The request is evaluated successfully and authorized to continue the operation.
- **Deny:** The engine tried evaluating the request against all applicable polices using the given combining algorithm and has computed that the request is not authorized to access the requested resource/service.
- **Indeterminate:** The PDP is not able to evaluate the requested access because exception have been occurred such as missing attributes, network errors while retrieving policies, division by zero, syntax errors in the request or policy, etc.
- **Not Applicable:** The XACML engine could not find an applicable policy/rule/target for the given request, therefore, it could not be evaluated.

XACML Policy

This section details the elements and attributes of the XACML policy and its structure, which is shown in Figure 38. Furthermore, this section should give a short introduction to the main elements and attributes because there are a more optional elements available as described in this section which can be found in the XACML specification [134].

The XACML policy is one of the main components of the XACML standard. Policies are created by an administrator or user to define the access rules for a specific resource. Each of these rules identifies the target on which rule applies. A target consists of simplified conditions for the subject, resource and action that must be met to apply to a given request. The rule contains besides the target a condition. This condition is the heart of a policy because it describes a predicate which has to be satisfied to perform the effect defined in the rule.

XACML Policy Element:

- **Policy ID** [Required]: The policy identifier is an attribute which has to be unique for a specific PDP. The PAP is responsible to maintain the uniqueness. The policy ID possibly consist of the URI or URL which can solve this issue.
- **Version** [Required]: The version identifies the version of the policy.

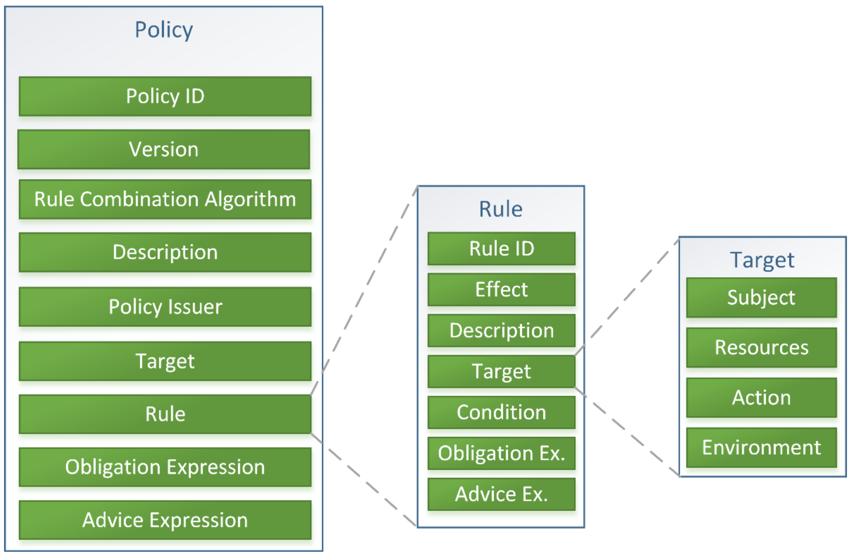


Figure 38: XACML Policy Elements [174]

- **Rule Combination Algorithm** [Required]: The rule combination algorithm is an attribute used by the XACML engine to identify the used combining algorithm. Combination algorithms are utilized to combine policies and rules if more than one policy or rule is applicable to a request. More details about these algorithms can be found in Figure 37.
- **Description** [Optional]: The description element gives space for a personal description or comments.
- **Policy Issuer** [Optional]: This attribute describes the policy issuer.
- **Target** [Required]: The applicability of the policy is defined in the target element. It is also possible that the target element is not entered by the creator but rather computed by the referenced rule elements. The target element has nested elements which are described below.
- **Rule** [Required]: The rule element contains a sequence of rules which must be combined according to the rule combining algorithm. Rules whose target elements and conditions match the request must be considered otherwise they should be ignored. The rule element contains nested elements which are described below.
- **Obligation Expressions** [Optional]: The obligation expression element contains a conjunctive sequence of obligation expressions which must be evaluated by the PDP. The resulting obligation must be fulfilled by the PEP together with the authorization decision. An example would be an expression which defines that after unauthorized requesting access to a resource this event is being logged.
- **Advice Expressions** [Optional]: The advice expression element contains a conjunctive sequence of advice expressions which must be evaluated by the PDP. The resulting advice provides additional information to the PEP. An example would be an expression which

defines that after successfully accessing a resource a predefined user will get an email notification about this event.

XACML Rule Element:

- **Rule ID** [Required]: The rule identifier is a string identifying the rule.
- **Effect** [Required]: The attribute effect represents the resulting effect after evaluating and matching the request. This effect can either be "Permit" or "Deny".
- **Description** [Optional]: The description element gives space for personal description or comments.
- **Target** [Optional]: Identifies the set of requests which shall be evaluated. If the target element is omitted, the target element of the enclosing policy takes place.
- **Condition** [Required]: The condition element describes a predicate which must be satisfied for the rule to get its effect value.

XACML Target Element:

- **Subject**: The subject defines the target for which this rule applies. It consists of a subject attribute name and a subject attribute value.
- **Resource**: The resource element defines the location of the requested resource which might be a URI or a URL.
- **Action**: The action element identifies the requested activity on a resource such as read, write execute, etc.
- **Environment**: The environment element specifies the system width where this policy/rule applies. For example, it is possible to restrict policies/rules to specific domains with utilizing this element.

D.4 Cryptographic Protocols and APIs

This section completes the fact sheets of the W3C Web Crypto API and KMIP from Section 6.4.

D.4.1 W3C Web Crypto API

The Web Cryptography API specification [171] defines a JavaScript API to access the cryptographic functions that are implemented in the browser. As this API was designed for JavaScript, it follows an asynchronous event-based approach to model the execution of long-running cryptographic operations. The provided functions can be grouped in two classes. The first class are the cryptographic operations, which perform encryption and decryption, signature creation and verification, and hashing. As these functions are natively implemented in the browser, they are more efficient than JavaScript implementations. The second class are key operations, which can be used to generate or derive keys, as well as to export and import key material. The browser manages the actual key material and can enforce access restrictions on them, like for which cryptographic functions a key may be used, or if the raw key material may be extracted.

D.4.2 KMIP – Key Management Interoperability Protocol

Key Management Interoperability Protocol (KMIP) [137] is a communication protocol standard developed by OASIS. The current version is the version 1.2 of the specification standard approved by OASIS on 19th May 2015. This protocol standard has been developed for the storage and maintenance of key, certificate and secret objects. It can be seen as interoperability protocol used for secure communication between server and client. The KMIP standard supports different crypto principles such as symmetric key encryption, asymmetric key encryption and digital signatures. The supported technologies make the KMIP to a powerful standard for example for the usage in enterprises.

The issues which KMIP addresses are the communication between the clients and the key management system as well as the management of the key material itself. It is possible, utilizing KMIP, to perform operations such as creating key material. KMIP uses network security mechanisms such as HTTPS and TLS to perform authenticated communication between client and key management system. KMIP relies on already existing standards for encryption, key derivation and other operations.

This section is structured as follows: First, we provide a high-level overview as well as the motivation for utilizing this protocol. Then, we detail the different elements of the KMIP standard.

KMIP High-Level Description

KMIP [137] tries to resolve the problem of missing interoperability between different cryptographic systems. Figure 39 depicts the key management system in an enterprise without utilizing KMIP. The obvious issue in this system is that each system has its own key management system. The key management systems used cannot be combined most of the time because of missing interoperability. Different key material or different protocols used in the systems are possible reasons for the missing interoperability. Therefore, each system is simply using its own key management system including key material, ciphers, protocols etc. This separation of the key management systems creates obstacles within an enterprise such as administration overhead or difficulties in adding new systems.

The KMIP is trying to solve the interoperability issue of enterprises by specifying a standard protocol for clients which require key material. Figure 40 illustrates how KMIP solves this interoperability issue using different systems. The protocol defines a low-level protocol which can be used for requesting and delivering keys between clients and key management system and enables fully interoperable key management. Utilizing KMIP allows enterprises to deploy a single

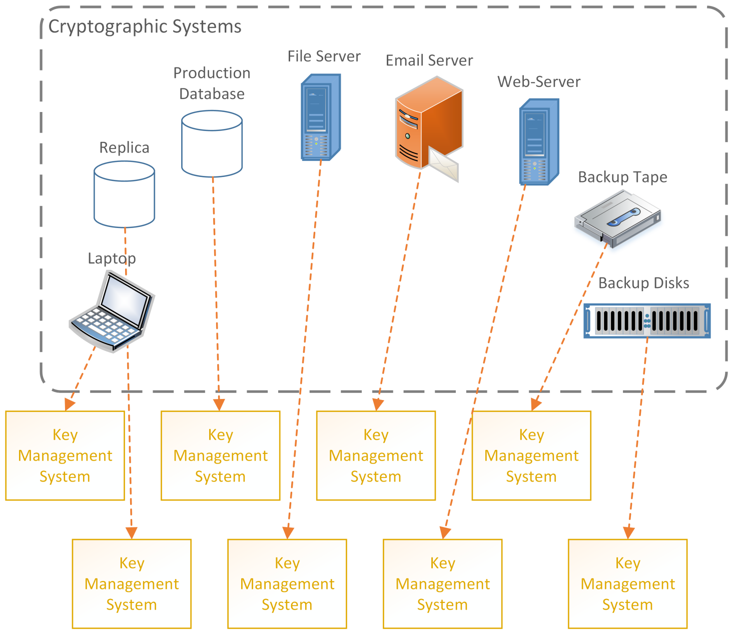


Figure 39: IT-Infrastructure in an Enterprise not using KMIP

key management infrastructure which is used to manage the key material for all applications, devices and systems within an enterprise. These components require different cryptographic material such as symmetric keys, asymmetric keys, certificates or other cryptographic objects, which are supported by the KMIP. A consistent model of cryptographic objects, attributes and operations is used to be able to create interoperability between various clients. Utilizing KMIP can reduce operational and key management costs, and reduce the risk in deploying cryptographic capabilities. [137]

KMIP tries to offer a secure interoperable solution to perform key operations. The usage of cryptographic operations requires key material which can with KMIP be managed on a single management point.

Specification

The KMIP standard consists basically of three primary elements which are the objects, the attributes and the operations. Each of these elements are detailed as follows.

Objects: The objects in the KMIP standard consists of two different types, namely the base objects and the managed objects.

First, the base objects are used within a message of the protocol, but these objects are not managed by the key management system. To base objects belong objects such as attribute objects, credential objects, key block objects, key value objects, key wrapping data objects, data objects, data length objects, signature data objects etc [137].

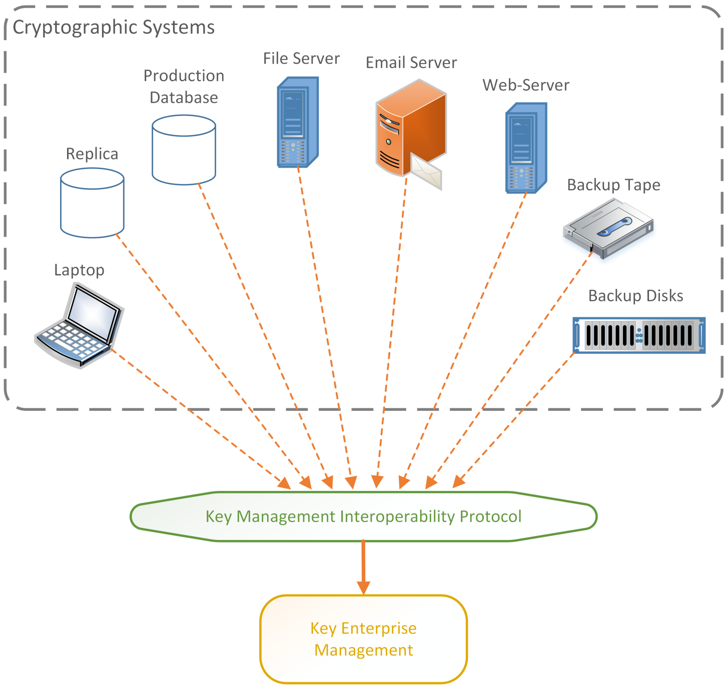


Figure 40: IT-Infrastructure in an Enterprise Utilizing KMIP

Second, managed objects are objects that are used as subject for key management operations. The managed cryptographic objects are a subset of the managed objects and describe these objects which contain cryptographic material such as certificates, keys or secret data. Examples for managed objects are certificate objects, symmetric key objects, public key objects, private key objects, split key objects, secret data objects, PGP key objects and more [137].

Attributes: Attributes in the KMIP standard are identifying attributes of the associated managed objects. Objects can have multiple attributes. Attributes can be obtained by the client or the server using the “Get Attribute” operation. Attributes can be set, modified or deleted with the corresponding operations “Add Attribute”, “Modify Attribute” or “Delete Attribute”. It might be that an attribute is read-only, which cannot be modified or deleted by either the client or the server. Examples for attributes are unique identifiers, names, object types, cryptographic algorithms, cryptographic lengths, cryptographic parameters, certificate types, certificate lengths, digital signature algorithms, operation policy names and more.

Operations: This section describes the operations which can be requested by either a client or the key management system. Not all clients have to support all kind of operation. KMIP operations can be distinguished between Client-to-Server operations and Server-to-Client operations. Basically, the operations can be clustered into different operation categories such as cryptographic operations, attribute manipulation operations, certificate operations etc. A subset of the Client-to-Server operations is detailed as follows [137].

- **Create:** The “Create” operation is utilized to generate a new symmetric key and after success a managed cryptographic object is returned.
- **Create Key Pair:** The “Create Key Pair” operations request the server to create a new public-private key-pair and register the two new managed cryptographic objects.
- **Certify:** The “Certify” operation is used to generate a certificate object for a public key. Locate: This operation is used to search for one or more managed objects depending on the given attributes.
- **Check:** This operation requests the server to check for the use of a specific managed object according to the given values specified in the request.
- **Get:** The “Get” operation is used to request a managed object by its unique identifier.
- **Get Attributes:** This operation is utilized to request one or more attributes associated with a managed object.
- **Add Attribute:** Using this operation allows a client to add an attribute to a managed object.
- **Modify Attribute:** This operation requests a modification of an already existing attribute from a specific managed object.
- **Delete Attribute:** This operation deletes an attribute associated with a managed object.
- **Activate:** The “Activate” operation requests the server to activate a managed object. The operation has to be performed on an object which has the “Pre-Active” state.
- **Destroy:** This operation is used to tell the server to destroy key material related to a managed object. The meta-data related to the key material may be retained by the server.
- **Validate:** This operation is used to validate a certificate chain and return the information about its validity.
- **Encrypt:** This operation requests an encryption operation on the provided data using a managed cryptographic object.
- **Decrypt:** Using the “Decrypt” operation performs the decryption of the provided data using a managed cryptographic object.
- **Sign:** The “Sign” operation requests the server to perform a signature operation on the given data by using a specific managed cryptographic object.

The counterpart to the Client-to-Server operations are the Server-to-Client operations which consists of two operations, the "Notify" and the "Put" operation.

- **Notify:** The “Notify” operation is used to notify a client of an event which resulted by attribute change of an object.
- **Put:** The “Put” operation is used to push managed cryptographic objects to the client.

D.5 Other Technologies

This section completes the fact sheet from Section 6.5.

D.5.1 SCIM

The System for Cross-Domain Identity Management is a standardized protocol developed by the Internet Engineering Task Force (IETF). The latest version 2.0 has been released in September 2015. The system was designed to simplify the user identity management in cloud based applications and services. In particular, the SCIM system defines a protocol standard together with a core schema and an object model. This offers a standardized way to create, read, update and delete (CRUD) identity data. The SCIM system is designed for the usage within enterprises as well as for consumer based digital identities.

Furthermore, this standard aims to solve one of the hardest challenges in identity management, namely interoperability. Interoperability issues emerge for example by using different technologies such as different identity protocols. SCIM tries to solve this issue and simplifies the identity management. Summarizing, SCIM is a standard designed to enable identity provisioning in cloud based applications and web services, basically, SCIM aims are: "make it fast, cheap, and easy to move users in to, out of, and around the cloud" [99].

SCIM allows to manage identities of multiple cloud service providers. When an identity resource has been created, updated or deleted, the identity resources are synchronized with multiple cloud service provider.

Authentication and authorization is out of band from the SCIM specification. This gives the developer freedom to implement authentication and authorization mechanisms based on the system requirements. The choice of authentication mechanism can impact the interoperability. Additionally to the authentication and authorization mechanisms are the non-identity related resources are out of band too.

The main advantages of using SCIM are:

- User identities re managed in a single place.
- SCIM uses standards for communication such as REST API and JSON.
- The SCIM standard is easy to extend.
- SCIM aims to solve interoperability issues.
- User identities can be synchronized between different cloud service providers.

This section is structured as follows: First, we detail the main actors according to the SCIM standard. Then, we explain the SCIM schemas together with the object model. Finally, we describe SCIM operations.

Actors

The actors in SCIM [98] describe the operating parties on both sides of the SCIM protocol. We can identify three main actors which are shown in Figure 41 and described as follows.

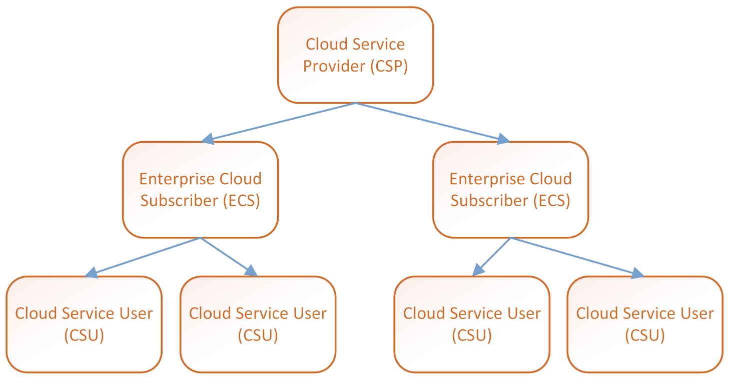


Figure 41: SCIM Actors [98]

- **Cloud Service Provider (CSP):** The cloud service is operated by the cloud service provider, which is in a Software as a Service (SaaS) scenario basically an application provider. The CSP can be seen as the part of a system which holds the identity information being operated, furthermore, it is the services which the user interacts with.
- **Enterprise Cloud Subscriber (ECS):** The enterprise cloud subscriber can be seen as middle tier of aggregation for identity records. An ECS is managing multiple cloud service users, which can be grouped together to administer as part of some broader agreement or operational exchange. For example, an ECS represents an enterprise which has bought a service from a service provider and administer it for the cloud service users within this organization.
- **Cloud Service User (CSU):** The cloud service end user is represented by the so-called cloud service user. This is for example a person who is logging into and using a cloud service.

Schema

The SCIM core schema is based on an object model depict in Figure 42. In this model is the main object the resource object and other objects are derived from it. All objects have the same common attributes which are an identifier, and external identifier and meta-data except the service provider configuration object. This service provider configuration object differs from the other objects because it does not contain any user information and is used to discovery.

The attribute structure of the objects is based on the relating schemas where each attribute has a different type, cardinality or mutability. Furthermore, the schema specifies which data are required and which data are optional in an object. Additionally, the schemas define also the data type of the attributes as well as the way the attributes are being processed.

The SCIM schema specification provides a minimal core schema, which can be used to represent resources such as users and groups, as well as common attributes. The schema specification also defines a standardized way how service providers possibly extend schemas. This extension is used to define new resources and attributes.

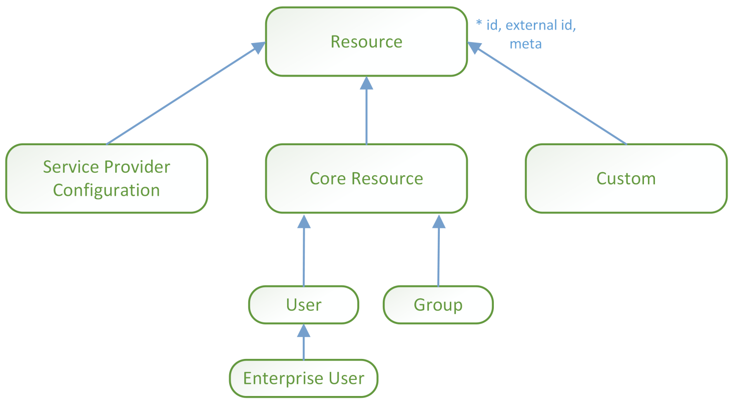


Figure 42: SCIM Object Model [99]

The resource object and its attributes are detailed as bellow. The specification of the other objects can be found in the SCIM specification [99]. The SCIM resource object, which can be the Users, Groups, etc., contains common attributes. These attributes are listed and detailed as follows.

- **Id:** The id describes a unique identifier for a SCIM resource, which is defined by the service provider. Each representation of the SCIM resource must contain an id value.
- **External id:** The external id is set by the provisioning client and is an identifier for the resource. This id is used to simplify identification of a resource between client and service provider by allowing the client to use filter.
- **Meta:** The meta attribute is a so-called complex attribute containing resource meta data. All sub attributes are assigned by the service provider and listed as follows.
  - **Resource type:** The resource type name of a resource.
  - **Created:** The date when the resource has been added to the service provider.
  - **Last Modified:** This attribute describes the last time when the resource was modified by the service provider.
  - **Location:** The location attribute contains the URI of the resource being returned.
  - **Version:** The version of the resource being returned is described by this attribute.

Operations

The SCIM protocol defines HTTP methods which are used to manage resources such as users and groups. The requests using the SCIM protocol are transferred in JSON or XML format over HTTP using a REST (Representational State Transfer) API.

All requests to the service provider are performed using HTTP methods. These methods are using a URL derived from the base URL. The responses are returned within the body of the HTTP response and formatted as JSON. If an error occurs, it will be transmitted using HTTP status response code. The following lines describe the use of different HTTP methods within KMIP:

- **GET:** Retrieves one or more resources (partial or complete).
- **POST:** Creates new resources, a search request or maybe is used to create a new resource.
- **PUT:** This method modifies a resource by replacing existing attributes with a specific set of replacement attributes. The PUT method is not used to create a new resource.
- **PATCH:** The patch method modifies a resource with a set of client specified changes.
- **DELETE:** This method is used to delete a resource.

The following list describes SCIM's resources and endpoints together with the corresponding HTTP operation supported. The description details what exactly is possible on a resource and related endpoint using the supported HTTP methods.

- /Users (GET, POST, PUT, PATCH, DELETE): Retrieve, add and modify Users.
- /Groups (GET, POST, PUT, PATCH, DELETE): Retrieve, add and modify Groups.
- /Service-Provider-Config (GET): Retrieve the configuration of the service provider.
- /ResourceTypes (GET): Retrieve the supported resource types.
- /Schemas (GET): Retrieve the supported schemas.
- /Bulk (POST): Bulk updates one or more resources.
- [prefix]/.search (POST): The search can be performed from the root or within a resource endpoint for one or more resource types.

E Details for eGovernment Technologies

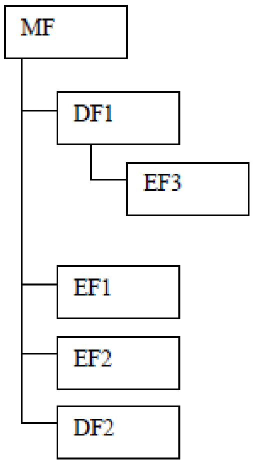
In the following sections, we provide a detailed description of eGovernment technologies which were previously evaluated and assessed in Section 7. First, Appendix E.1 provides details on CNS, the National Smartcard Standard, whilst Appendix E.2 describes CSP-Cryptographic Service Provider which is used to interface smartcard to establish an SSL client authentication. Appendix E.3 describes the Public-Key Cryptographic Standards PKCS #11 that provides guidelines and application programming interfaces (APIs) for the usage of cryptographic methods. Appendix E.4 gives details on ISO 7816, the international standard focused on contact id card, especially smartcard. Finally, Appendix E.5 describes the framework of STORK/STORK 2.0 for enabling secure eID federation across European countries whilst Appendix E.6 provides details on eIDAS Interoperability Framework which can be considered as an extension and a modern generalization of the core of STORK Framework.

E.1 CNS (Carta Nazionale dei Servizi)

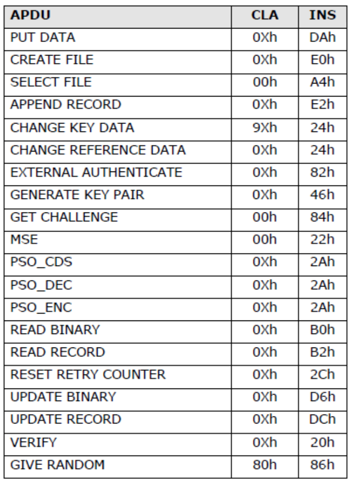
CNS (Carta Nazionale dei Servizi) is a national smartcard standard widely used in Italy. CNS is a ISO 7816 smartcard with a client authentication X509 certificate on-board to allow digital authentication over the Internet. CNS does not contain user photo or other sensitive data (such as biometric data). CNS contains several personal data such as name, surname, fiscal code, etc. Data are stored in different files and format (mainly ASN.1 and tag-length-value coding) in order to assure large compatibility with existing software. It also contains a Netlink structure that is dedicated to store some medical data known when the card is issued and that can be modify by other “authorized” healthcare professional cards (HPC). Netlink derived keys are stored on CNS in order to allow mutual authentication with HPC. It is possible to create and store digital signature keys, using a file system section reserved to authorized Certification Authorities. There are several CNS producers in Italy. Each of them must be compliant with CNS rules (details are below) and must have a certification released by Italian Digital Agency.

The CNS file system can be described with a tree-representation as depicted in Figure 43a: Where MF (Master File) is the smartcard root; DF (Dedicated File) is a directory, which can contain EF (Elementary File); EF can also be stored under MF. Every file (MF, DF, EF) has certain permissions (read, write, update, etc). CNS smartcard also contains BSO (Base Security Object), which is a container for secret or sensitive data. BSO are used in cryptographic operation and for the verification of the access conditions to the resources of the smart card. Name, length and content of every file and BSO are precisely defined in CNS rules. CNS contains approximately 12 DF, 21 EF and 25 BSO. CNS chip is capable of store from 16 to 64 kBytes of additional user data – depending on manufacturer.

One of the most important characteristic of CNS smartcards is ATR – Answer to Reset. The Answer To Reset byte string returns information about the communication protocol, the card and application type, the life cycle status, etc. While some of the bytes are defined by ISO, as they are used by the communication protocol, there is the possibility to define the value of other bytes, called Historical Bytes. To avoid possible interoperability problems, only the



(a) CNS File System



APDU	CLA	INS
PUT DATA	0Xh	DAh
CREATE FILE	0Xh	E0h
SELECT FILE	00h	A4h
APPEND RECORD	0Xh	E2h
CHANGE KEY DATA	9Xh	24h
CHANGE REFERENCE DATA	0Xh	24h
EXTERNAL AUTHENTICATE	0Xh	82h
GENERATE KEY PAIR	0Xh	46h
GET CHALLENGE	00h	84h
MSE	00h	22h
PSO_CDS	0Xh	2Ah
PSO_DEC	0Xh	2Ah
PSO_ENC	0Xh	2Ah
READ BINARY	0Xh	B0h
READ RECORD	0Xh	B2h
RESET RETRY COUNTER	0Xh	2Ch
UPDATE BINARY	0Xh	D6h
UPDATE RECORD	0Xh	DCh
VERIFY	0Xh	20h
GIVE RANDOM	80h	86h

(b) CNS APDU Commands

Figure 43: CNS

historical bytes are mandated, while the value (and, if applicable, the presence) of all the interface characters is not fixed in CNS specification.

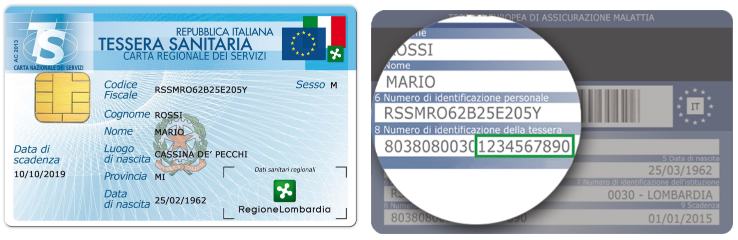
The card has to support the ISO protocol T=1. It may support also T=0, but protocol T=1 has to be used during the use phase. The maximum communication speed is 115200bps 3.5712MHz, typically used during the use phase to allow optimal performances. The value for Vcc during the operational phase is 5 Volts.

The CNS needs a subset of the APDU (Application Protocol Data Unit) specified by the ISO norms. Not all the ISO specified operating modes of the supported commands are needed. The complete list of APDU commands used in the CNS is shown in Figure 43b.

CNS support the Secure Messaging (SM) protocol. Secure Messaging is used to protect the communication between the interface device (IFD) and the smartcard. Secure Messaging is typically used when an external actor (another smartcard, or a library) need to perform a sensitive operation on CNS (i.e. create a digital signature structure). There are two ways to communicate data in SM format:

- SM_ENC: APDU commands with enciphered data (data confidentiality)
- SM_SIG: APDU commands with cryptographic checksum (data authentication and integrity)

As mentioned above, every CNS is equipped with a X.509 digital certificate, for authentication use. Certification authorities, which want to produce CNS certificate, must follow certain rules and obtain a formal authorization from Italian Digital Agency. Of course every CNS certificate, regardless of its issuer, has well defined characteristic, in order to guarantee interoperability between applications. CNS is also equipped with magnetic stripe, typically used in legacy applications, which read anagrafic data from magnetic band and not from CNS chip.


REPUBBLICA ITALIANA
TESSERA SANITARIA
CARTA REGIONALE DEI SERVIZI
CARTA NAZIONALE DEI SERVIZI
Codice Fiscale RSSMRO62B25E205Y
Sesso M
Cognome ROSSI
Nome MARIO
Data di scadenza 10/10/2019
Luogo di nascita CASSINA DE' PECCHI
Provincia MI
Data di nascita 25/02/1962
Dati sanitari regionali
RegioneLombardia
OSSI
MARIO
6 Numero di identificazione personale
RSSMRO62B25E205Y
8 Numero di identificazione della tessera
80380800301234567890
25/03/1962
0030 - LOMBARDIA
01/01/2015

(a) Front (b) Back

Figure 44: CNS Card

CNS duration is six years. Authentication certificate loaded into CNS has the same duration of the card (expiry date is printed on it). The card is also used to store citizen's fiscal code – that is also printed both on front and back side of the card, as shown in Figures 44a and 44b. CNS is not an id card: it does not contain any user photo or biometric info. Many millions of CNS have been deployed in Italy since 2004 – almost 10 million are currently used in the Lombardy Region. This has created a "digital ecosytem" of CNS-compliant smartcard reader, cryptographic libraries and – at the end – client and web applications (including Identity Providers), which use CNS smartcard for a lot of functionality (user authentication over the Internet, user authentication on desktop client, fiscal code and age recognition in shops or postal offices, digital signatures of documents, etc.).

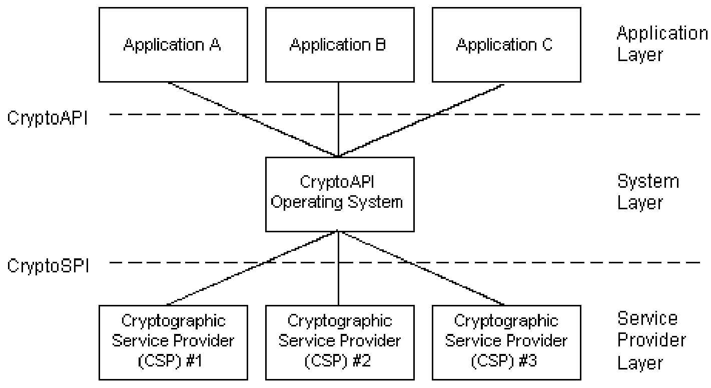
E.2 CSP (Cryptographic Service Provider)

CSP (Cryptographic Service Provider) is a Windows software library that implements Microsoft CryptoAPI (CAPI). CSP is typically use to perform cryptographic operations, such as strong user authentication and secure email. In the LISPA eGovernment pilot, CSP is used to interface smartcard to establish an SSL client authentication, using smartcard private key. This usage needs certain browser, like Internet Explorer or Chrome. To perform the same cryptographic operation on Mozilla Firefox, you need another, similar component, called PKCS#11 (out of scope of this description). CSP is responsible for implementing cryptographic algorithms and standards, so applications don't need to be concerned about security details.

When used in a browser, CSP imports user authentication certificate into internal browser store as soon as smartcard is inserted into a reader. From this moment, CSP is ready to react to a request of SSL client authentication coming from a secure server, following the behavior specified below: 1. When client authentication is required, CSP checks if root CA that issued the SSL server certificate is trusted by client certificate store; 2. If so, CSP checks how many user client certificate are issued by a CA configured as trusted on server side; 3. If more than one user certificate is available, user is asked to choice one; 4. When user certificate is selected, CSP – and SSL protocol itself – create a random challenge and sign it with user's private key; 5. Smartcard typically reacts asking PIN associate with private key; 6. User inserts smartcard

PIN into a form shown by CSP; 7. If PIN is correct, challenge is signed and send to server – from now on, SSL protocol continues without using CSP

From an architectural point of view, CSPs are independent modules that can be used by different applications. A software calls CryptoAPI functions and these find an implementation into CSPs functions. An important note is that application can define which CSP it is going to be used on its calls to CryptoAPI. In other words, all cryptographic activity is implemented in CSP, and CryptoAPI only works as a middleware or a wrapper between the application and the CSP.


Application A
Application B
Application C
Application Layer
CryptoAPI
CryptoAPI Operating System
System Layer
CryptoSPI
Cryptographic Service Provider (CSP) #1
Cryptographic Service Provider (CSP) #2
Cryptographic Service Provider (CSP) #3
Service Provider Layer

Application layer has several constraints:

- Cannot directly access keys – keys are usable via handle only;
- Cannot specify some cryptographic details – CSP only let some commands are send to it;
- Cannot directly manage digital signature

In a smartcard scenario, several smartcards might have several CSP capable of using them. To detect the correct CSP to use, operating system detects smartcard ATR (Answer to Reset) and looks for an association into Windows Registry. Association is a match between an ATR and a CSP capable of using that smartcard. In Italy there are several CSP compatible with CNS smartcard – the main differences between them are about potentially user PIN caching, ATR filtering, and PIN form request.

CSP implementation is typically based on DLL, with some special capabilities and restrictions: due to these aspects, CSP must be digitally signed by Microsoft in order to be correctly used, and signature verification must be done on load. Some CSP are currently available from Microsoft. They are mainly focused on basic cryptographic functions, common to several countries or regions, and on particular cryptographic support (i.e. Diffie-Hellman key exchange). The basic implementation of CSP deployed by Microsoft is the so-called “base CSP”.

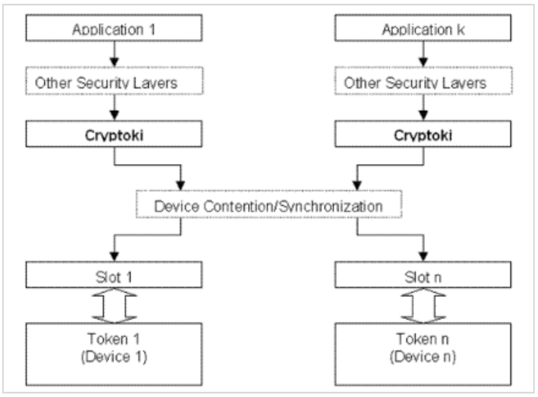
E.3 PKCS #11

The Public-Key Cryptographic Standards (PKCS) include a group of cryptographic standards that provide guidelines and application programming interfaces (APIs) for the usage of cryptographic methods. As the name PKCS suggests, these standards put an emphasis on the usage of public key – asymmetric - cryptography.

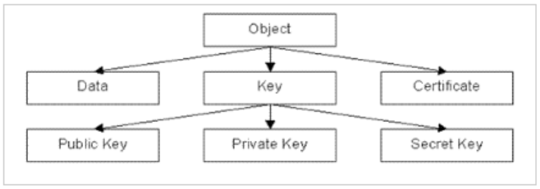
PKCS #11 is a cryptographic token interface standard, which specifies an API, called Cryptoki. With this API, applications can address cryptographic devices as tokens and can perform cryptographic functions as implemented by these tokens (i.e. smart cards or Hardware Secure Modules – HSM). This standard, formerly developed by the RSA Laboratories in cooperation with representatives from industry, science, and governments, is now an open standard lead-managed by the OASIS PKCS #11 Technical Committee.

It follows an object-based approach, addressing the goals of technology independence (any kind of hardware device) and resource sharing. It also presents to applications a common, logical view of the device that is called a cryptographic token. PKCS #11 , or Cryptoki, assigns a slot ID to each token. An application identifies the token that it wants to access by specifying the appropriate slot ID.

Most commercial certificate authority (CA) software uses PKCS #11 to access the CA signing key or to enroll user certificates. Cross-platform software that needs to use smart cards uses PKCS #11, such as Mozilla Firefox and OpenSSL (using an extension). As already mentioned, it is also used to access smart cards and HSMs. Software written for Microsoft Windows may use the platform specific MS-CAPI API instead. Both Oracle Solaris and Red Hat Linux contain implementations for use by applications, as well.



(a) Architecture of a PKCS System



(b) Objects of a PKCS System

Figure 45: PKCS #11

A typical Cryptoki-based system architecture is depicted in Figure 45a. The cryptographic device (aka token) is connected to the system via a slot. Typically, a slot corresponds to a smart card reader or a specific card terminal. However, because Cryptoki offers a purely logical view of the system, it could happen that different slots point to the same physical reader device or, vice versa, a single slot could have more than one device.

A specific Cryptoki implementation maps the token's physical structure, typically composed by memory zones in which data, cryptographic keys and their digital certificates are stored, into a logical structure that adheres to the hierarchical model shown in Figure 45b. Cryptoki's objects are classified depending on their visibility in public objects (i.e. accessible by all applications), and private objects (visible only after granting access permissions typically performed via PIN-verification as described later), and on their persistency in: token objects which persist when the token is plugged-out from the slot and in session objects which don't persist. For each class of objects the specifications define a set of attributes (as described later) characterizing all instances of the class, which, are inherited by derived classes, similarly to an object-oriented model (for example, the Private Key class inherits all attributes from the Key class etc.). The specifications define three main object classes:

- **Data objects** host generic data which semantics is defined by the application who created them.
- **Key objects** contain a public, private or secret cryptographic key.
- **Certificate** objects store digital certificates.

Lombardy Region and Italian Government have designed CNS in order to be used by a PKCS #11 module. Some typical use of a PKCS #11 library with CNS are: Firstly, in several browsers such as Mozilla Firefox, a PKCS #11 dll is used to perform a SSL client authentication with a smart card; Secondly, many stand-alone software uses PKCS #11 dll to create a digital signature of a document; signing process involve CNS private key. Both open source PKCS #11 module and vendor dependent are available on the market. A vendor implementation is needed when vendor token implements specific functionalities.

E.4 ISO 7816

ISO 7816 is an international standard focused on contact id card, especially smartcard, managed and published by ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission). The standard was published for the first time in 1987 and faced a number of amendment, until 2013. It was initially written for contact-only cards (cards with the pins exposed in the plastic).

ISO 7816 is a collection of fourteen chapters, described below. Parts 1 to 3 are the description of the physical / link layer part of the protocol, part 4 and above describe the standardized commands on the upper layer.

- Part 1: Physical characteristics - this part of ISO7816 is important for card manufacturers. They are the ones that choose the materials and establish a process that embeds the integrated circuit into the card
- Part 2: Dimensions and location of the contacts - this part includes standards about number, function and position of the electrical contacts
- Part 3: Electronic signals and transmission protocols
- Part 4: Interindustry commands for interchange

- Part 5: Numbering system and registration procedure for application identifiers
- Part 6: Interindustry data elements
- Part 7: Interindustry commands for Structured Card Query Language (SCQL)
- Part 8: Security related interindustry commands
- Part 9: Additional interindustry commands and security attributes
- Part 10: Electronic signals and answer to reset for synchronous cards
- Part 11: Personal verification through biometric methods
- Part 12: USB electrical interface and operating procedures
- Part 13: Commands for application management in multi-application environment
- Part 15: Cryptographic information application

The entire standard covers a wide variety of card characteristic. The most famous and used part is probably part 4 that specifies:

- contents of command-response pairs exchanged at the interface;
- means of retrieval of data elements and data objects in the card;
- structures and contents of historical bytes to describe operating characteristics of the card;
- structures for applications and data in the card, as seen at the interface when processing commands;
- access methods to files and data in the card;
- a security architecture defining access rights to files and data in the card;
- means and mechanisms for identifying and addressing applications in the card;
- methods for secure messaging;
- access methods to the algorithms processed by the card. It does not describe these algorithms.

Conformity to ISO 7816-3 and -4 is requested to smartcard reader which wants to be Italy CNS compliant, for example. Most of ISO 7816-3 is important for reader manufacturers or developers who want to establish a communication with a smart card on a very low level, the signal level. In fact, conformity to these parts of the standard was one of the main request in a public tender issued by Lombardy Region in 2006, with the target of acquire some millions of devices to let citizen use CNS to authenticate on the internet to e-Gov and e-Health web service. ISO 7816, as others ISO rules, are not freely available – a specific or subscription fee is required to access this kind of documentation.

Several parts of ISO 7816 have been reviewed (amended) due to technology and market innovations, for example in contact-less card. Usually, contact-less cards comply with ISO 14443, and optionally with ISO 7816 part 4. There are also cards that have dual interface.

E.5 STORK/STORK 2.0 Framework

The STORK framework – enabling secure eID federation across European countries – has been designed to be the dominant identification and authentication framework across Europe in the future. The project is based on two lines of thoughts on how an interoperability framework can be build: In the first approach, the service provider integrates all foreign eID tokens using a middleware. We refer to this approach as "middleware model". In the second approach, cross-border eID transactions are delegated to a national gateway – a proxy – that hides the specifics of national eID tokens and infrastructure from other countries. We refer to this as "proxy model". This can easily lead to scalability issues, especially for the proxy-based (PEPS) approach in STORK, which relies on a central gateway being responsible for managing and handling citizen authentications.

Middleware Model

The middleware model of STORK framework can be described as follow: A citizen directly authenticates at a service provider. The citizen remains the owner of the data and the service provider is the data controller. Identity data is usually stored on a secure token, e.g. smart cards, and will only be released if the user gives his consent to do so, e.g. by entering a PIN. No intermediary is in the path between the citizen and the service provider.

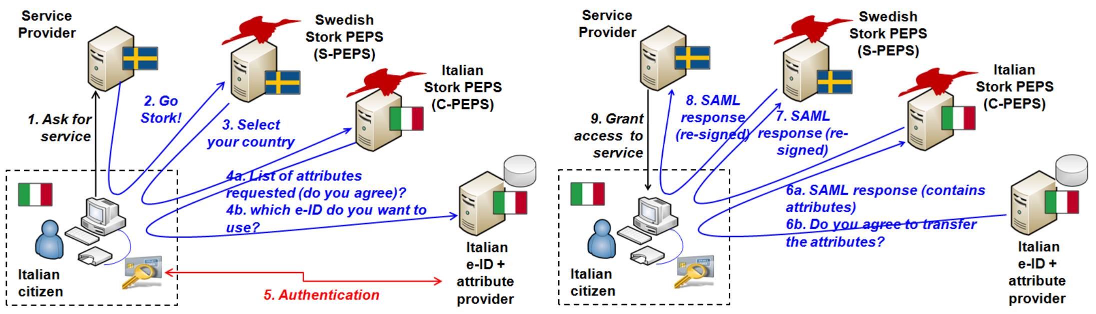
The "middleware model" consists of two separate software, one running on the user's and one running on the service provider's system (server-side middleware – a sort of virtual identity provider, VIDP). Generally, the client-middleware handles the communication with the secure token and the server-side middleware. The server-side middleware is responsible for transmitting the identity information retrieved from the token to the SP application. The "middleware model" can ensure end-to-end security.

Proxy Model

The proxy model can be described as follows: A component called PEPS is widely used in this scenario. This component bundles several services required for a cross-border eID solution and hides the complexity and specifics of national solutions from other countries. The services provided by a PEPS include the identification and authentication at identity providers, the additional retrieval of identity attributes, or the secure transfer of the identity information to service providers (SP).

Steps involved in the scenario are the following: A citizen from a PEPS country wants to authenticate at a service provider in another PEPS country. The SP delegates the authentication process to its S-PEPS which in turn forwards the request to the citizen's origin C-PEPS. The

C-PEPS triggers the actual authentication process with the user by invoking the appropriate identity and attribute provider (this can involve user in a sort of “IdP selection” if more than one national IdP are available). If authentication is successful, the C-PEPS assembles a so-called SAML assertion containing the requested identity data, wraps it into a SAML Response message and returns it to the S-PEPS. The S-PEPS verifies the assertion and forwards the citizen to the SP that grants or denies access to the requested resource. Following the point-to-point trust relationship, the messages are validated at each receiver, re-signed and forwarded to the next component. As depicted in Figures 46a and 46b, an Italian citizen tries to access a Swedish service provider.


Service Provider
Swedish Stork PEPS (S-PEPS)
Italian Stork PEPS (C-PEPS)
1. Ask for service
2. Go Stork!
3. Select your country
4a. List of attributes requested (do you agree)?
4b. which e-ID do you want to use?
Italian citizen
5. Authentication
Italian e-ID + attribute provider
Service Provider
Swedish Stork PEPS (S-PEPS)
Italian Stork PEPS (C-PEPS)
9. Grant access to service
8. SAML response (re-signed)
7. SAML response (re-signed)
6a. SAML response (contains attributes)
6b. Do you agree to transfer the attributes?
Italian citizen
Italian e-ID + attribute provider

(a) Request (b) Response

Figure 46: STORK Process

Note that: Authentication scheme between C-PEPS and national IdP can be proprietary; S-PEPS is responsible for SAML 2.0 assertion creation; During SAML 2.0 assertion creation, S-PEPS uses well defined tags, in order to assure interoperability. The use of STORK/STORK 2.0 Framework, mostly “proxy model” section, is an essential part of LISPA e-Gov pilot, in order to assure cross-border interoperability between SP and IdP located in different countries.

E.6 eIDAS Interoperability Framework

eIDAS interoperability Framework can be simplified as an extension and a “modern” generalization of the core of STORK Framework.

First of all, we need to identify the actors which join eIDAS-Network - which is made of eIDAS-nodes. The main components of the eIDAS-Network are: Firstly, the relying party (i.e. a SP), which requires authenticity/integrity of the received personal identification data. Also, in order to fulfill the data protection obligations, it requires confidentiality of the received personal identification data. Secondly, the citizen, who expects confidentiality of his personal identification data.

Within the eIDAS Interoperability Framework, communication between eIDAS-nodes (i.e. eIDAS-Services and eIDAS-Connectors) is performed via the citizen’s browser. Here, the content of the communication between eIDAS nodes is performed using cryptographically protected SAML messages. To secure the transport layer of this communication between these components and the citizen’s browser, a recent version of TLS is used.

Several Member States (MB) can join eIDAS Network, so cross-border interoperability is a major goal. Interoperability between different eID-schemes is achieved via defining the technical interfaces between eIDAS-Connectors and eIDAS-Services, collectively eIDAS-Nodes. Sending Member State can choose between two integration scenarios for their eID-scheme; they are derived from STORK/STORK2 Framework and are the following:

- Proxy-based: The Sending Member State operates an eIDAS-Proxy-Service, relaying authentication requests and authentication assertions between an eIDAS-Connector operated by the Receiving Member State and the eID scheme of the Sending MS.
- Middleware-based: In this scenario the Sending MS does not operate a Proxy for the purpose of authentication of persons to relying parties of other MS. The Sending MS provides a Middleware to other MS, which is operated by the operator(s) of the eID-Connector(s) of the Receiving MS.

Each Receiving Member State shall operate one or more eIDAS-Connectors. It is up to the Receiving Member State to decide the national deployment of Connectors. Connectors need not to be operated by the Member State itself, but can also be operated by public and/or private relying parties established in that Member State. Typically, MSs operating exactly one Connector are called Centralized MSs (analog to C-PEPS in STORK scenario – see figure below), while MSs operating several Connectors are called Decentralized MSs. An eIDAS-Connector is operated together with eIDAS-Middleware-Services for communication with middleware-based eID schemes. Italian eID project SPID is an example of IdP federation compliant with eIDAS directives.

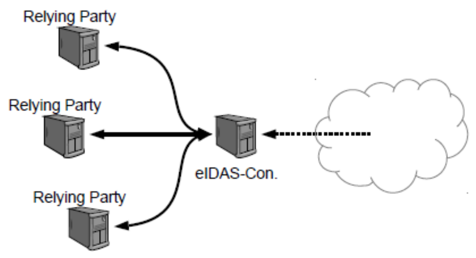


Figure 47: eIDAS

The authentication experience – from a user point of view – is very similar to STORK approach. One of the most important change is about the concept of Level Of Assurance: if the requested (or higher) Level of Assurance cannot be fulfilled by the eIDAS-Service, the Request must be rejected. Another major difference is about the metadata exchange: to provide an uninterrupted chain of trust for authentications, as well as an uninterrupted chain of responsibility for integrity/authenticity and confidentiality for personal identification data, Nodes must be securely identified. This identification is made by (signed) XML metadata files. In some cases, metadata can be cached by a party. Metadata must be verified by the parties involved in the authentication process.

F Details for eHealth Technologies

Healthcare organizations are rather productive in defining healthcare specific standards for healthcare specific issues such as medical content structuring and encoding. Nevertheless, when it comes to more generic functionalities such as data sharing and security, there is a strong tendency to adopt existing standards to the specific needs of healthcare use cases. This adaption is usually referred to as “profiling” with the resulting “profiles” defining constraints on existing standards in order to foster interoperability when using these standards in healthcare.

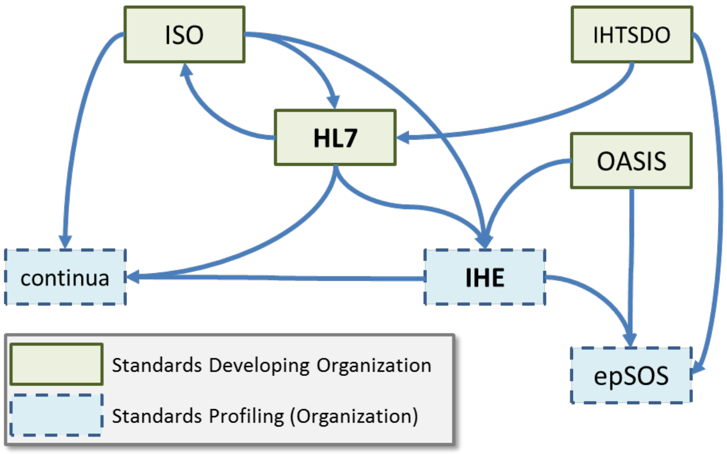

ISO
IHTSDO
HL7
OASIS
continua
IHE
epSOS
Standards Developing Organization
Standards Profiling (Organization)

Figure 48: Prominent Standardization Bodies

Figure 48 sketches some of the most prominent standardization bodies and profiling initiatives which are relevant for healthcare-IT:

- Especially in the domain of mobile health and medical devices healthcare-IT adapts existing ISO standards (i.e. ISO/IEEE 11073).
- HL7 is most prominent SDO that solely focusses on healthcare. HL7 standards range from content representation formats to permission catalogues. Through a co-operation with ANSI some HL7 standards are as well ISO norms.
- IHTSDO is a non-for-profit organization that releases the most comprehensive terminology in healthcare which is about to substitute many of the existing specialized terminologies.
- IHE (Integrating the Healthcare Enterprise) defines profiles on existing standards which allow implementing eHealth use cases using existing COTS. Typical origins of IHE profiles are HL7 (e.g. document type specifications), OASIS (e.g. ebXML-based sharing of health data) and DICOM (radiology).
- Continua Alliance is another profiling organization that defines interoperability profiles for medical devices on top of ISO/IEEE 11073. Continua Alliance as well provides a reference architecture for connecting personal health devices to health records by using IHE profiles.

- epSOS is an example for an initiative (European FP7 LSP project) that further profiled existing IHE profiles. By this existing international profiles are further constrained to common European requirements (e.g. as derived from the data protection directive) and to cross-border sharing of health data.

In the following sections some established healthcare-IT standards and profiles are sketched which either may be relevant for *CREDENTIAL* or even shall be considered for the *CREDENTIAL* eHealth use case due to their wide acceptance with vendors and clinics.

F.1 Health Information Exchange

This section describes document exchange protocols. It covers basic protocols for sharing medical data as well as protocols for exchanging metadata like patient and healthcare professional identifiers.

F.1.1 Cross-Enterprise Document Sharing (XDS)

The IHE profile "Cross-Enterprise Document Sharing (XDS)" defines interfaces, metadata and data flow protocols for sharing medical documents among healthcare professionals [94]. IHE XDS builds upon the OASIS ebXML standard and provides a specific configuration and deployment of this standard's components to reflect typical use case within regional healthcare networks.

Actors and Transactions

Figure 49 sketches the actors (logical building blocks) and transactions (services) as defined by IHE XDS.

HE XDS implements the ebXML separation of a document registry and a document repository while integrating these with further building blocks:

- The Document Registry takes responsibility for the management of document metadata and provides services for document query and registration
- The Document Repository implements a content-agnostic store for medical documents. It provides services for storing documents (which will then be registered at the Document Registry by the Document Repository) and for retrieving a set of identified documents.
- The Document Source actor is the origin of medical documents that are shared through IHE XDS. IHE XDS defines two different flavors of sources: (Static) Document Sources that upload copies of locally managed document to the XDS Document Repository and On-Demand Document Sources which generate documents on demand out of whatever kind of patient data (e. g. generating a lab report from a clinical database).

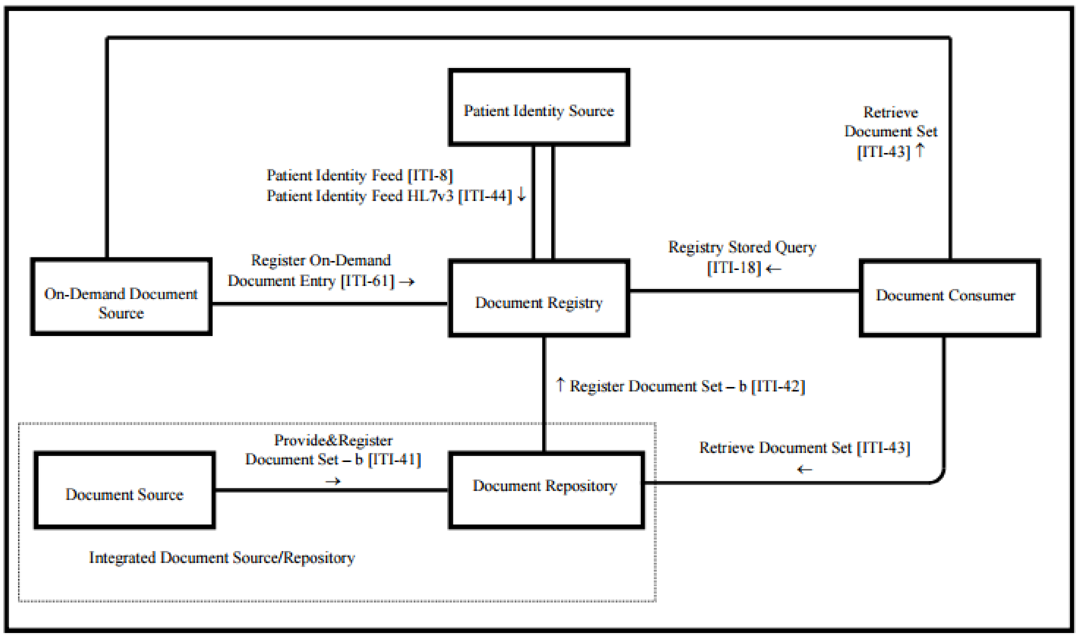

Patient Identity Source
Patient Identity Feed [ITI-8]
Patient Identity Feed HL7v3 [ITI-44] ↓
Retrieve Document Set [ITI-43] ↑
Register On-Demand Document Entry [ITI-61] →
On-Demand Document Source
Document Registry
Registry Stored Query [ITI-18] ←
Document Consumer
↑ Register Document Set – b [ITI-42]
Provide&Register Document Set – b [ITI-41] →
Document Source
Document Repository
Retrieve Document Set [ITI-43] ←
Integrated Document Source/Repository

Figure 49: IHE XDS Actors and Transactions (from IHE ITI TF-1)

- The Patient Identity Source takes responsibility for announcing new patient identifiers to the XDS Document Registry (e.g. when a patient is admitted to a hospital) and provides the connected systems with information about patient ID related events (e.g. it had been discovered that a patient was a duplicate to an already registered patient).
- The Document Consumer actor is implemented by any component that utilizes document sharing services of the Document Repository and Document Registry.

The data model implemented by IHE XDS consists of five major kinds of entities:

- DocumentEntry ebXML registry objects capsule the metadata associated to a medical document. These metadata reflect the author, affected patient and creation context of the document together with some technical information such as the document file size and a hash value.
- Each DocumentEntry refers to a medical document that is treated by IHE XDS as a BLOB.
- Medical data is provided to a Repository as a SubmissionSet registry object which capsules a set of medical documents together with their belonging DocumentEntry objects. SubmissionSets have metadata of their own and cannot be modified after they have been uploaded to the Document Registry.
- Folder registry objects allow bracing a set of DocumentEntry objects. In XDS folders cannot be nested. Nevertheless, any DocumentEntry can be a member to multiple folders.

As DocumentEntries, Folder objects can only be registered to the Document Registry as part of a SubmissionSet.

- Relationships among objects are always explicit. They are expressed through Association registry objects that link one object to another. The most important kind of relationship is the hasMember-association which places DocumentEntries into SubmissionSets and/or Folders. Figure 50 sketches which Associations need to be explicitly defined to upload a single DocumentEntry within a single Folder into a single SubmissionSet. For making the lifecycle of documents visible, documentEntries may be linked to other document entries using Associations for document replacement, document transformation and document addendum.

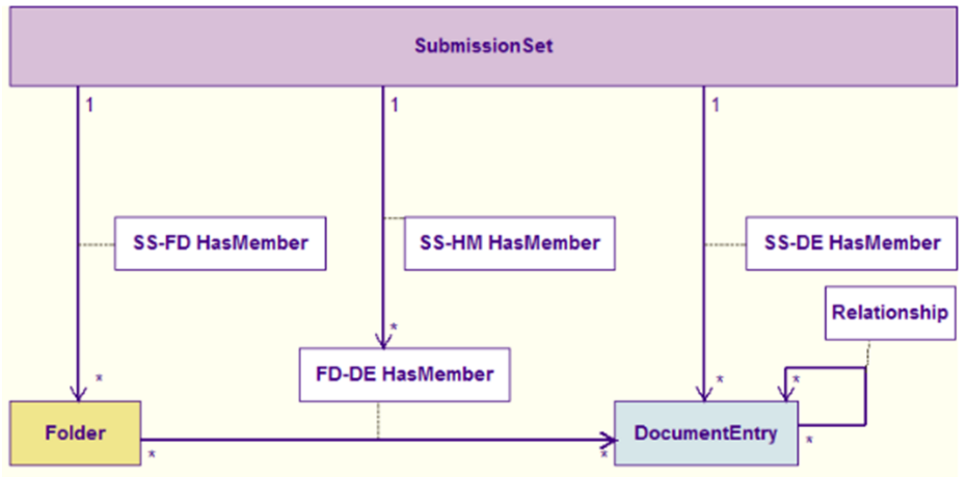


Figure 50: IHE XDS Registry Objects [IHE ITI TF-3]

Use Cases and Functional Features

IHE XDS actors implement an Affinity Domain. An Affinity Domain is made up from a single Document Registry and one or more Document Repositories. This e. g. allows for scenarios where medical data is stored within hospitals while the Document Registry implements a centrally accessible service for discovering requested data within these decentralized Document Repositories. In addition, the organizations working together in an Affinity Domain have to agree on common means for patient identification, network security, provider identification, document metadata details, data formats, etc.

A typical use case for such an Affinity Domain is a regional care network, where e.g. a community hospital operates a central Document Registry to enable sharing of medical documents among all participants of the care network. By using Associations between DocumentEntry objects and by manipulating document metadata, functionalities for updating and invalidating documents can be implemented on top of the core services for uploading, searching and downloading documents. This as well is the strength of XDS; it focuses on core services for a Health Information Exchange among co-operating organizations which in advance defined common rules on how they want to share data.

Nevertheless, with every installation of XDS, new demands for enhanced features are coming up. IHE serves these requests not only by continuously adjusting the core of XDS but as well by specifying additional functionalities as separate profiles which may even define additional actors and transactions. Among these profiles are:

- Cross-Community Access (XCA): XDS is restricted to sharing data within a single Affinity Domain. Nevertheless, patients may visit doctors which across such rather artificially domain boundaries. For such settings the IHE XDS profile defines gateways for mediating data query and retrieval across Affinity Domains
- Document Metadata Subscription (DSUB): A common use case for XDS is to enable the sharing of data that is needed by the participating organization in cases of admission-discharge and referral / re-referral care sequences. In such setting doctors may want to receive automatic notification if a needed document for an identified is made available or if addenda had been added to a formerly retrieved document. The IHE DSUB profile provides actors and transactions that allow doctors to subscribe to events and to receive notification in case that event takes place.
- XDS-i: This profile provides an integration of XDS paradigms and services with the DICOM protocol which is the established international standard for sharing image data.
- Mobile Access to Health Documents (MHD): Being based on ebXML, XDS messages are very verbose and by this rather unsuitable for document consumer and source actors which are located on mobile devices. The recently published MHD profile defines actors and transactions for capsuling an XDS infrastructures by a RESTful proxy that can be connected through JSON and FHIR by mobile devices.

Relevance

IHE XDS is the de facto healthcare-IT standard for record based infrastructures. It is widely used worldwide and has broad support from healthcare-IT-vendors. In Europe many health information exchange networks build upon IHE XDS and most of the public tenders for such networks request for IHE XDS as the base standard. The following list gives some of the most prominent XDS activities in Europe:

- Elektronische Gesundheitsakte ELGA (Austria): The Austrian national eHealth infrastructure is made up from multiple, interconnected IHE Affinity Domains. It went operational in December 2015.
- Electronic CaseRecord EFA (Germany): EFA is a specific configuration of IHE XDS that pays specific attention to European Privacy legislation. Seven IHE XDS vendor implementations have been successfully tested for EFA compatibility at the IHE European Connectathon 2016.

- European Patients Smart Open Services epSOS (European Commission): epSOS defines means for transcoding and translating medical documents while sharing them across borders through IHE XCA gateways (with the relevant transactions being more or less identical to the IHE XDS Registry Stored Query and Retrieve Document Set transactions).

Consideration for *CREDENTIAL* eHealth Pilot

The *CREDENTIAL* eHealth pilot shall use an IHE XDS compliant backend for managing medical data. Only this ensures that the solution can be properly integrated with existing infrastructures and data flows in healthcare-IT.

IHE XDS is security agnostic by only defining business level transactions. Therefore, IHE XDS actors may be capsuled by *CREDENTIAL* security proxies for safeguarding access to medical data within an XDS registry/repository. The IHE MHD profile gives an idea how this could be implemented.

A specific of IHE XDS that needs to be considered by *CREDENTIAL* is that a document consumer needs to first query the Document Registry for requested documents through a search on document metadata. In the second step the discovered documents can be retrieved by providing their document IDs to the Document Repository. This imposes some issues that need to be solved by *CREDENTIAL* in order to not weaken *CREDENTIAL*'s high level of privacy on the application level:

- Information in IHE XDS document and submission set metadata discloses (too) much protected information about the patient. E. g. as XDS allows querying for documents by document type, each DocumentEntry holds the document type together with the patient identifier. In many cases, knowing that a specific type of a document has been created for a patient, discloses which procedure the patient has undergone or with suspect diagnose leaded to that specific report. *CREDENTIAL* needs to carefully assess if pseudonymization, metadata hiding, metadata encryption or any combination of these is sufficient to protect the patient's privacy while not requiring users to always download the full set of documents for a patient in order to find a single document that could have been easily discovered by a metadata search.
- The request message for retrieving a set of documents does not contain any patient ID, just the IDs of the requested documents. *CREDENTIAL* needs to define means for enforcing the patient's privacy policy on such a request in an efficient yet secure manner.

Given the diversity of objects defined by XDS for managing medical data and the rather complex integration of these objects, *CREDENTIAL* must carefully consider that any extension or modification on XDS transactions may have impact on the consistency of the document store. E.g. XDS defines clear rules on how e.g. an update to a document that has an addendum shall be performed and how a document query must deal with such a cluster of related documents. Encrypting too much information may hinder XDS implementations in automatically performing the required processing of such operations.

F.1.2 Patient Identifier Cross-Referencing (PIX) & Patient Demographics Query (PDQ)

A common problem in healthcare IT is that the same patient is registered with different identifiers at different care providers and IT-systems. This not only holds for cross-enterprise use cases (e.g. a hospital and a resident physician referencing the same patient using different IDs) but even within enterprises (e.g. radiology systems introducing their own patient IDs). IHE defines two integration profiles that deal with such problems:

- **Patient Demographics Query (PDQ)** defines actors and transactions for discovering a patient identifier from known demographics (e.g. name, date of birth). The profile as well covers scenarios, where manual intervention is needed because available demographics are not sufficient for univocally identifying a patient.
- **Patient Identifier Cross-referencing (PIX)** defines actors and transactions for linking identifiers of the same patient across multiple enterprises. By this data about the same patient from different systems can be correlated.

Vendors usually provide Patient ID Management solutions (e.g. Master Patient Indices for hospitals and regional care networks) that implement both profiles as typical use cases require a very close integration of ID discovery (PDQ) and ID matching (PIX).

Actors and Transactions

In hospitals most IT systems communicate through HL7v2 messages with each other. Whenever an event occurs (e.g. a patient has been admitted or moved to another room, a document has been released or revoked, a lab has been ordered) the system that triggered or captured that event sends out an HL7v2 message to a defined address. This address is usually a communication server that accepts the message and forwards it to all connected systems that may be affected by the event. HL7v2 is based on the EDI standard that separates messages into segments and fields using pipes and roofs as delimiters.

The IHE PIX profile specifies actors and transaction that enable a hospital or care network to synchronize patient identifiers from different devices and organizations based on such events. Among these events are

- A patient is admitted at a care organization where an identifier is assigned to that patient. Usually multiple identifiers are assigned, e.g. to differentiate between the patient and the visit.
- Patient data (e.g. address) changed.
- At admission it was not recognized that an identifier was already assigned to the patient so that the patient now has two files under different IDs. Through an event connected systems are advised to merge two identifiers into one.

Figure 51 sketches the actors (logical components) and transactions (services) as defined by the IHE PIX integration profile.

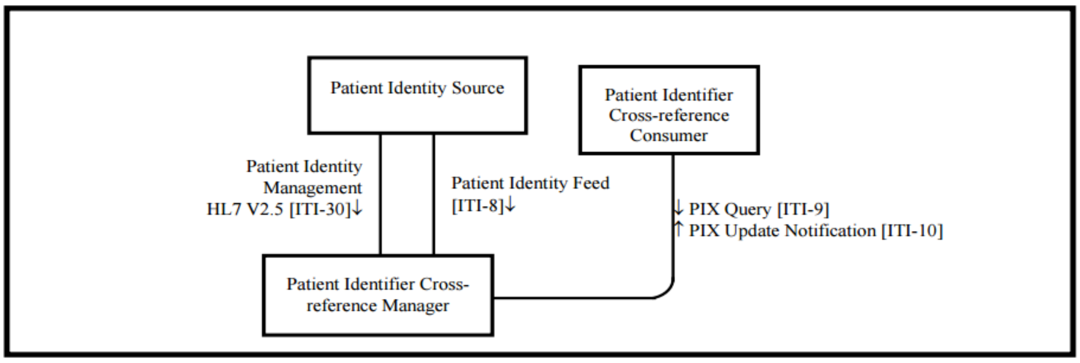

Patient Identity Source
Patient Identifier Cross-reference Consumer
Patient Identity Management HL7 V2.5 [ITI-30]↓
Patient Identity Feed [ITI-8]↓
↓ PIX Query [ITI-9]
↑ PIX Update Notification [ITI-10]
Patient Identifier Cross-reference Manager

Figure 51: IHE Actors and Transactions

Patient-ID related messages as the ones sketched above are called Patient Identity Feeds. Systems that trigger these events act as Patient Identity Sources. Identity sources shall send all patient-ID relevant events to a central Patient Identifier Cross-reference Manager, which accepts the information within these events and assembles a cross-reference table. The cross-reference table records the identifier each patient is assigned in each identity domain (patient-ID managing system/organization). Using the PIX Query transaction, client systems as Patient Identifier Cross-reference Consumers can query for the ID that is assigned to the patient in a given domain.

Closely related to the PIX profile is the Patient Demographics Query (PDQ) profile which defines a service for querying for a patient identifier by demographic data. The major difference to PIX Query is that a PIX Query demands for a domain patient ID as input while PDQ also works solely on demographics. In addition, the PDQ Patient Demographics Supplier usually implements sophisticated algorithms to cope with slightly wrong-spelled names or to discover the ID of a person even if that person had married and changed her name.

Use Cases and Functional Features

A network of cooperating doctors – as will be set up for the *CREDENTIAL* eHealth demonstrator – cannot be set up without having PIX and PDQ services available. Even in countries were all citizens are assigned a national (healthcare) ID, patients may have forgotten their respective ID cards or are unable to show these. Therefore, such networks usually operate a single Master Patient Index based on PIX and PDQ services that allow each doctor to:

- Register the identifiers he uses for the participating patients.
- Obtain the identifier a cooperating doctor uses for the same patient.
- Obtain a domain PID for the patient that shall be known to all doctors.

The latest functionality is very important, because data managing systems like an IHE XDS Document Registry usually only allow for a single, unique identifier per patient. Therefore, a new patient ID is generated and PIX used for mapping the care organizations' local patient IDs onto this domain PID. Data Source and Consumer actors in hospitals and practices automatically connect to the Affinity Domains Patient Identifier Cross-reference Manager before connecting to the XDS health record in order to map the local patient ID onto the patient ID used by the XDS Affinity Domain.

In order to better serve such cross-enterprise use cases, IHE has released HL7v3-versions of the PIX and PDQ profiles which are based on XML instead of EDI. For *CREDENTIAL* only the versions of the profiles will be considered.

Relevance

IHE PIX/PDQ is the de facto standard for managing patient identifiers in cross-enterprise use cases. It interplays well with IHE XDS and is supported by most of the existing health record solutions (commercial and open source). Therefore, *CREDENTIAL* identity management components shall be able to cope with IHE PIX/PDQ actors and transactions for the eHealth use case. In case that *CREDENTIAL* eHealth use case introduces additional IDs for patients, IHE PIX compliant Patient Identifier Cross-reference Manager shall be used to match these with existing patient identifiers.

Considerations for *CREDENTIAL* eHealth Pilot

The use cases defined for the *CREDENTIAL* eHealth Pilot require several care organizations to share medical data about the same patient. In addition, the patient himself and personal health devices may contribute data. While the central CRENDENTIAL data managing infrastructure will accept only a single unique patient ID for linking all patient data, the distributed data sources and consumers will operate on their own ID management. E.g. a fitness tracking device will not be aware of the patient at all and only place its device identifier as a unique key into each data set.

In *CREDENTIAL* a Patient Identifier Cross-reference Manager will be deployed as a core component of the Personal Health record infrastructure. It takes responsibility for maintaining a central Patient Identifier Cross-reference table. This imposes several challenges with respect to patient privacy, as the defined IHE messages for registering patient identifiers are directly derived from hospitals' internal events and therefore disclose much information about the patient, his environment and reasons for coming to hospital. Therefore, *CREDENTIAL* will take over the general protocol patterns of IHE PIXv3 but define its own data set for patient registration and patient ID query. This will in particular consider personal device identifiers as additional IDs to be linked with a patient.

The *CREDENTIAL* use case analysis identified a need for a PDQ-alike functionality: For sending event notifications to the patient (e.g. someone provided a new document) the *CREDENTIAL* notification service needs to obtain contact data for the patient. In addition, there may even be scenarios where a patient advised that such notifications should be send to a relative who takes care of the patient. All this information (relatives, contact data, etc.) is typically provided through a Patient Demographics Supplier. Therefore, a respective component will be set up for *CREDENTIAL* as part of the notification subsystem. Ideally this component only acts as a proxy to the common *CREDENTIAL* Participant Index and just provides a healthcare-aware API that wraps the existing *CREDENTIAL* Participant Search Service. In this deployment the common *CREDENTIAL* Participant Registration Service can be fully re-used for the eHealth pilot.

F.1.3 Audit Trail and Node Authentication (ATNA)

IHE recommends that every Affinity Domain (see section on IHE XDS) shall implement four baseline security means:

1. All hosts within the domain are able to mutually authenticate each other. Successful mutual node authentication is a prerequisite for any access to protected data.
2. The host identification is used to determine which data is accessible to automated processes on that host, and/or persons under the direction of that host's access controls. This considers that many systems provide pre-fetching mechanisms for synchronizing local and shared data over night so that medical data is available to doctors when needed.
3. The node that hosts protected data is responsible for reasonable access controls, including user authentication and authorization.
4. Every access to protected data and every other security related event is logged to a secure audit protocol. There may be scenarios (e.g. emergency access) where only a log is written without prior access control.

Hosts that implement all these mechanisms for the full software stack on a machine are called Secure Nodes in the IHE nomenclature. Products that implement these mechanisms – but rely on the underlying operating system, database, etc. to be protected as well – are called Secure Applications.

While mechanisms 1-3 are either within the core scope of *CREDENTIAL* or part of the common *CREDENTIAL* security infrastructure, the need for an audit trail is a rather specific for the *CREDENTIAL* eHealth pilot.

Actors and Transactions

The IHE Audit Trail and Node Authentication (ATNA) integration profile defines actors (components) and transactions (services) for setting up secure domains based on mutual node authentication an audit logs.

IHE Node Authentication defines specific configurations of the TLS and S/MIME protocols that ensure a unique level of security and interoperability in connecting communicating nodes. As

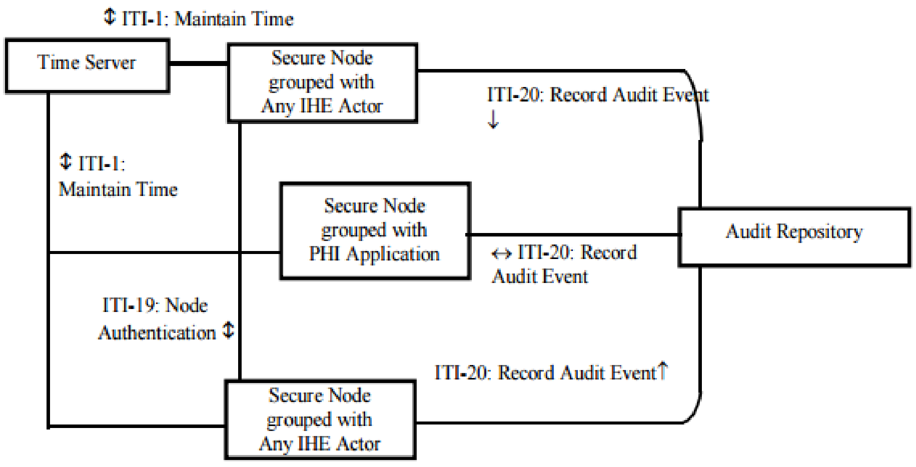

⇕ ITI-1: Maintain Time
Time Server
Secure Node grouped with Any IHE Actor
ITI-20: Record Audit Event ↓
⇕ ITI-1: Maintain Time
Secure Node grouped with PHI Application
Audit Repository
↔ ITI-20: Record Audit Event
ITI-19: Node Authentication ⇕
ITI-20: Record Audit Event↑
Secure Node grouped with Any IHE Actor

Figure 52: IHE Audit Trail and Node Authentication (ATNA)

IHE ATNA is still based on TLSv1.0, this part of the profile will not be used for *CREDENTIAL* which builds upon TLSv1.2 as a baseline for transport layer security. Therefore, the focus for the rest of this section will be on the Record Audit Event transaction and the Audit Repository Actor.

IHE ATNA audit messages are based on the respective specifications from the DICOM standard which again are based on the – now deprecated – RFC 3881 "Security Audit and Access Accountability Message - XML Data Definitions for Healthcare Applications". The figure below sketches the core information model of a single audit trail entry that is sent as a message to the Audit Repository.

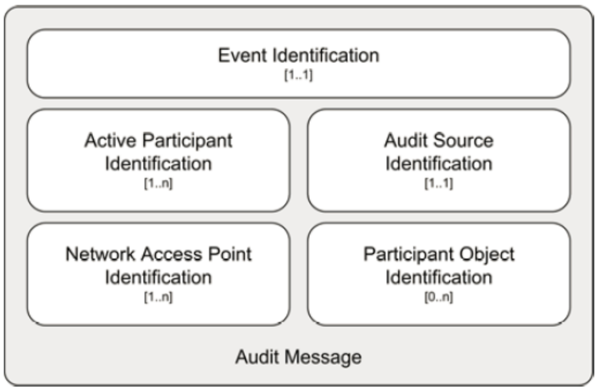

Event Identification
[1..1]
Active Participant Identification
[1..n]
Audit Source Identification
[1..1]
Network Access Point Identification
[1..n]
Participant Object Identification
[0..n]
Audit Message

Figure 53: Five Major Building Blocks of the Message

The five major building blocks of the message give answers to the most relevant questions when non-repudiation of an activity is requested:

- Event Identification: What activity has been performed?

- Active Participant Identification: Which actors (humans and/or IT-services) performed the activity?
- Network Access Point Identification: Which technical systems was the originator of the event?
- Audit Source Identification: Which system requested that an audit trail entry is written for the event (e.g. the service that accepted the event and released protected data in response to the event)?
- Participant Object Identification: Which resources where subject to the event?

The Audit Repository takes responsibility that all reported events are logged in a secure way. This includes mechanisms to protect the integrity and confidentiality of the audit log. In particular, there must be mechanisms in place to prevent any manipulation of a once written audit trail.

Use Cases and Functional Features

Writing an audit trail entry is part of almost all business use within a health information exchange. Ever access to medical data is to be logged as well as security related events such as the issuance of an assertion or the grant of an access permission.

While IHE originally only specified how data is pushed into an audit trail using reliable syslog, the recently published profile "Add RESTful Query to ATNA" adds a further actor and transactions for securely reading data from an audit trail repository through a defined REST interface.

Relevance

Non-Repudiation is a rather rigid requirement for most eHealth use cases (including the ones specified for *CREDENTIAL*). IHE ATNA is the most elaborated profile on top of RFC3881 which again is the only standard in this respect that gained wider acceptance.

Therefore, an ATNA compliant audit trail repository shall be considered as a required component for the implementation of the *CREDENTIAL* eHealth use case.

Considerations for *CREDENTIAL* eHealth Pilot

IHE ATNA does not define normative means for safeguarding audit trail entries. Nevertheless, it will be a rigid requirement for *CREDENTIAL* to fully protect the integrity and confidentiality of the audit trail as otherwise this would impose a weak spot to the overall *CREDENTIAL* security shield.

For the eHealth pilot it shall be investigated if cryptographic means defined by *CREDENTIAL* are applicable to ATNA audit trails for technically protecting the confidentiality and authenticity of audit trail entries. One solution could be to encrypt the critical parts of all audit messages for the affected patient. Patients could then grant trusted persons access to their audit data (e.g. privacy commissioners) through proxy-reencrypting that data for these persons.

F.1.4 Cross-Enterprise User Assertion (XUA) & Cross-Enterprise User Assertion - Attribute Extension (XUA++)

IHE XUA is a profile on OASIS SAML v2.0 that defines constraints on the SAML assertion format and on the use of attribute statements. The flow of control is only roughly sketched and implements a specific integration of SAML and WS Trust. The only part that is normative for conformance to the XUA profile is the provisioning of a conformant SAML Identity Assertion to a Service Provider within a SOAP message header (see Figure 54).

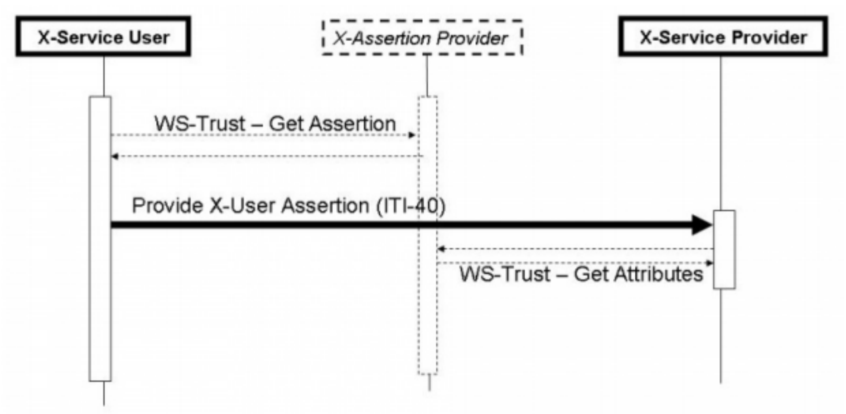


Figure 54: Flow of Control

Options and Definitions on Attribute Statements

The major purpose of the XUA profile is to take up the increasing use of federated authentication in healthcare-IT projects and to define useful constraints in order to make SAML aware systems of different vendors interoperable.

For this several SAML elements and attribute definitions are constrained:

- Sender-vouches subject confirmation method is not allowed. Bearer is considered as a default (assuming that ATNA compliant TLS transport layer security is used).
- An AudienceRestriction shall be provided. It shall either refer to the URI of an Affinity Domain (see section on IHE XDS) or the URI of the service that is requested.
- Subject ID and Subject Organization attributes shall include plain text names of the user and the organization.
- Only assertions which are digitally signed by the issuer are accepted.

In addition, XUA specifies various options which implementers may support. Even though *CREDENTIAL* may decide to not support these options, *CREDENTIAL* shall not specify the same semantic in a way that is not compliant to these XUA options.

The Subject-Role option defines how a subject's role is to be encoded as an attribute statement. The most relevant constraints include the use of a defined name and namespace which enforce the use of HL7 data types and SNOMED CT for encoding role values. The following example from [IHE ITI TF-2b] shows how a XUA-compliant role attribute looks like:

```
<saml:Attribute Name="urn:oasis:names:tc:xacml:2.0:subject:role">
  <saml:AttributeValue>
    <Role xmlns="urn:hl7-org:v3" xsi:type="CE" code="46255001"
          codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED_CT"
          displayName="Pharmacist"/>
  </saml:AttributeValue>
</saml:Attribute>
```

The patient ID may be provided as a resource ID within a SAML attribute statement. If this option is implemented, the HL7v2 CX data type must be used for encoding the patient ID and its assigning authority:

```
<saml:Attribute Name="urn:oasis:names:tc:xacml:2.0:resource:resource-id">
  <saml:AttributeValue>543797436^^^&amp;1.2.840.113619.6.197&amp;ISO
  </saml:AttributeValue>
</saml:Attribute>
```

Relevance

Role- or Attribute Based Access Control in healthcare requires the definition of constraints on the use of subject attributes and resource attributes if systems from different vendors are required to be interoperable. Given the low adoption rate of international standards within the different sectors and the high diversity of standards across sectors this is a tough challenge. This the more as information from identity assertions issued by one system need to be written to audit trails by another system in a way that long-term understandability pf this information can be guaranteed.

Considerations for *CREDENTIAL* eHealth Pilot

The *CREDENTIAL* Identity Provider service set up and configured for the *CREDENTIAL* eHealth pilot shall only issue XUA compliant assertions. The respective constraints should not be that severe considering that *CREDENTIAL* proxy re-encryption implies a DAC-style access control model which only requires subject and organization IDs for enforcing policies. As XUA defines healthcare specific attribute names for these attributes, there will be no conflicts with semantically equivalent attribute definitions from other domains.

F.1.5 IHE Advanced Patient Privacy Consent (APPC)

The collection, use and/or processing of personal health information is only legitimate if either requested by legislation or authorized through a freely given, informed consent of the individual concerned. This rule is one of the core principles of the European privacy directive [Directive 95/46/EC] and basis of all derived European national privacy legislations.

In eHealth and telemedicine, a patient consent guarantees the individual's civil rights, strengthens her informational self-determination and provides the data processor with the legal basis for collecting, using and processing personal health information.

The existing decoupling of written consents and electronically enforceable permissions has some obstacles especially in larger settings, where multiple eHealth services are operated and where consents need to be adapted to specific and individual care scenarios. The IHE Advances Patient Privacy Consent (APPC) integration profile defines how to express a patient's consent as a machine processable privacy policy. This allows for a continuous flow of authorization information, where a patient's given consent can be translated into machine processable rules that are enforced whenever that patient's personal health data is processed by an electronic service that is governed by that consent.

Contents and Encoding

An Advanced Patient Privacy Policy allows for expressing all information that is required to be governed through a consent:

- Individual whose health information is protected by the policy (subject of consent): This individual can be univocally identified through a machine readable, registered ID.
- Resources that are protected by the policy (target of consent): These resources may either be identified through their unique identifiers or through resource attribute filters (e.g. type of resource, healthcare event associated with resource). Resource attributes used within a patient privacy policy shall be univocally defined and identified and the processing system shall be able to discover all resources that match the given attribute filters.
- Individuals and organizations that are authorized to collect, use and/or process the protected resources: These organizations and individuals may either be identified through their unique identifiers or through entity attribute filters (e.g. administrative role, class of healthcare entity). Entity attributes used within a policy shall be univocally defined and identified and the processing system shall be able to decide whether a resource requestor matches the given attribute filters

APPC is specified as a profile on the OASIS eXtensible Access Control Language (XACML) by imposing constraints on the structure of XACML policies and by restricting the set of attributes that may be used for expressing these policies. Such restrictions allow for a seamless interplay of XDS, XUA and APPC. E. g. by this the policy processing services are able to match authorized organizations or individuals who are identified by their IDs with the ID of the resource requestor as provided with the request through an XUA compliant identity assertion.

Considerations for *CREDENTIAL* eHealth Pilot

CREDENTIAL requires that patients who participate in the eHealth pilot give written consent to their doctor before any data is collected or processed within the context of *CREDENTIAL*. This consent will be given on paper and it will be signed with the patient's wet signature. For making the patient's will processable in *CREDENTIAL* the doctor will document the relevant consent information electronically. It shall be assessed if pure APPC is sufficient or if a wrapper will be needed (e.g. a XML document where the APPC is embedded).

The other scenario where APPC may be helpful is the ad-hoc-authorization of physicians, e.g. for one-time access to the health record or for assigning them a role within the care team. If such an authorization is capsuled as an access policy, APPC shall be considered as a good basis for gaining technical and semantic interoperability across all means that may be used for ad-hoc authorization.

F.1.6 Document Digital Signature (DSG)

IHE Document Digital Signature (DSG) specifies how digital signatures of medical documents are used while they are shared across multiple organizations. The profile supports three different methods for digital signature. An enveloped signature is a signature which is packed along with the document content itself. A detached signature is a signature with a reference to the signed document content. A submission set signature is a detached signature which points to a set of documents that are signed by this signature.

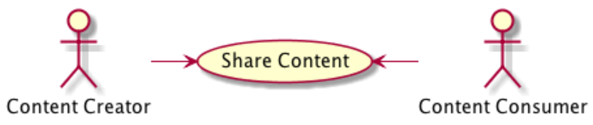


Figure 55: DSG Actor Model

DSG uses a two actor model as described in Figure 55. The Content Creator is the entity who creates a digital signature document. On the other side the Content Consumer is the entity who is responsible to verify the signature of those documents. The documents are shared from the Content Creator to the Content Consumer.

The goal of DSG is to establish document integrity. The signature is used to guarantee that the signed document is not modified by error or intent and to vouch for the identity of the signer. In addition, a practical use case would be to verify the clinical content of a document by a practitioner.

The Document Consumer can perform the following operations:

- Search for signatures of a document: With this operation a Document Consumer is in possession of a document and needs to find the document's signature.
- Search for documents of a signature: The signature contains a reference to the document(s) that are signed by it. Thus the Document Consumer is able to search for these documents.
- Search signatures: Signature can be queries by time range, specific participant, signature purpose, etc.
- Ignore signature in document query: If only the source document is needed, the Document Consumer can suppress the document signature.

Considerations for *CREDENTIAL* eHealth Pilot

DSG shall be used as a guidance about how document signature may be integrated with typical health data sharing scenarios.

It is still possible to use different XAdES profile, hashing algorithms, policy identifiers, or signature purpose vocabulary. *CREDENTIAL* opportunities for processing signed data may impose additional requirements on DSG implementations.

F.1.7 Document Encryption (DEN)

IHE DEN is a profile on IETF Cryptographic Message Syntax (CMS). It defines how health data shall be encrypted for transmission and storage. By this DEN supports the notion of end-to-end encryption where medical data is only disclosed to the creator and consumer of medical content while all intermediary actors may only process encrypted content.

Considerations for *CREDENTIAL* eHealth Pilot

HE DEN shall be used as guidance about how data encryption may be integrated with typical health data sharing scenarios. Especially definitions about how encrypted data is to be handled in conjunction with other healthcare-IT standards should be considered. While keeping the overall DEN flow of control and data, *CREDENTIAL* may define its own protocols and mechanisms for medical data encryption. *CREDENTIAL* opportunities for processing encrypted data may impose additional requirements on DEN implementations.

F.2 Clinical Content Representation

This section describes the two standards for defining medical documents. First, the Clinical Document Architecture (CDA) is introduced. Second, the Fast Healthcare Interoperability Resources (FHIR) standard is explained.

F.2.1 Clinical Document Architecture (CDA)

Clinical Document Architecture (CDA) is a HL7 standard for structured markup language in clinical documents. It is expressed in an XML-encoded format and can contain text, images, sounds, and other content. A CDA uses the HL7 Reference Model [89] to represent the document's content and achieve a machine processable meaning.

A CDA document has a hierarchical structure with the following elements:

- Clinical Document: The root element of a CDA. It contains a CDA Header and a Structured Body.
- CDA Header: The header contains meta-information about the medical document. It identifies the participants who are involved in this document and which roles they hold. Relationships between other documents are described.
- Structured Body: The body can be an unstructured blob or a structured document. Every CDA has exactly one structured body. A body can have multiple sections which are used to organize the document.

The main focus of a CDA document is an authenticated, human readable form of medical documents which can be transmitted from a sender to a receiver. The human readability is achieved through three different levels. In level 1, only text, images, and similar media information is contained inside a CDA. By using level 2, structured XML information is added to a document. Level 3 is a complete machine interpretable form of the medical document which is conform to the CDA-RIM specification.

F.2.2 Fast Healthcare Interoperability Resources (FHIR)

For better reflecting recent tendencies in healthcare IT such as mobile health devices, REST service interfaces and the use of web standards (e.g. JSON, OAuth) the HL7 standardization organization developed from scratch a new standard for sharing health data. This standard, named Fast Healthcare Interoperability Resources (FHIR), builds upon modular resource definitions (e.g. "patient", "care plan", "medication") that can be easily combined to implement arbitrary complex and interoperable information objects. Resource definitions themselves build upon a small set of primitive types that again can be combined into complex types such as codeable concepts or identifiers which are bound to a specific semantics.

A strong focus of FHIR is on implementability and reduction of complexity. Each resource definition is specified based on abstract data types and comes with normative bindings to XML and JSON. FHIR follows the approach to consider a clinical information as a network of connected resources which may be managed independently and follow their own lifecycle.

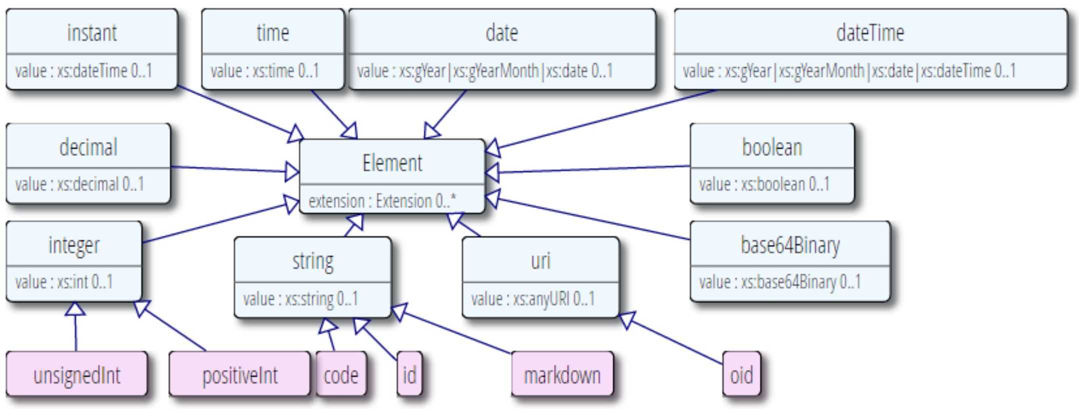


Figure 56: FHIR Primitive Types [from FHIR homepage]

Considerations for *CREDENTIAL* eHealth Pilot

For the *CREDENTIAL* eHealth pilot FHIR shall be the first option of choice for representing medical documents and data sharing among professionals. In addition, FHIR will be used as the external representation format for terminologies, value sets and concepts (see next section on CTS2).

Data provided by the patient will almost be time series of sensor data. HL7 FHIR only provides rudimentary support for such kind of data and does not well integrate with data base systems which are optimized for managing and processing that kind of data. Therefore, *CREDENTIAL* will use more optimized formats for sharing time series data.

F.3 Outline

This chapter explained the various types of eHealth standards that are relevant for the eHealth pilot. The technologies were categorized in two clusters clinical content representation and health information exchange. With health information exchange we covered the main standards in the eHealth domain that are used for sharing medical data across healthcare professionals. With respect to these standards we are able to integrate an eHealth solution easily with other products and technologies. Clinical content representation fulfills the same role, except it focuses on standardized data formats.

Most of the technologies are mandatory for the eHealth pilot and should simply be used because they are the de facto standard. Nevertheless, the technology analysis for CDA and FHIR will be made, since they are competitive standards.

G Details for eBusiness Technologies

The eBusiness pilot mainly uses three technologies: PEC, S/MIME and SPID. S/MIME is a well-known, international standard, while PEC and SPID are specific to Italian scenario. SPID is essentially an Italian implementation of eIDAS specifics about secure authentication on the net, whereas PEC is peculiar and no similar initiatives are known outside Italy. PEC is widely used in Italy, and since 2013 all communications among enterprises, public administration and local/central Government must be done using PEC. In 2015, 766 million PEC messages were managed by 25 authorized PEC providers. Those messages were exchanged among almost 8 million PEC addresses. SPID is a more recent initiatives of Italian Government, and aims to create an ecosystem of certified Identity Providers. Every public administration and public company is obliged to use a SPID IdP for its user authentication process at the latest in April 2018. At the date, there are five certified Identity Providers and over 1 million digital identities have been released.

G.1 PEC (Posta Elettronica Certificata)

After two years of technical tests, during 2005 the DPR 68 (Decree of the President of the Republic) defined the characteristics of an official electronic delivery service, named certified electronic mail (in Italian Posta Elettronica Certificata, PEC) giving it legal value. PEC characteristics are well explained in the DM November 2005 (Decree of Ministers) and in its attached technical rules. DPR 68 established a public register of PEC providers giving to AgID public agency selective enrollment and monitoring duties. DPR 68 stated that an e-mail is considered:

- **sent** when the sender's provider, after several checks, accepts the e-mail and returns an acceptance receipt to the sender;
- **received** when it is stored in the e-mail account of the receiver. Then, the receiver's provider returns a receipt of delivery to the sender.

PEC, compared to traditional e-mail, ensures: 1. recognition of the sender (that is to say the mail account); 2. integrity of sent message; 3. no delivery refusal; 4. matching between the delivery receipt and the message sent by the user. Providers are required to have a logging system, which tracks and stores all system events for 30 months, except for the mails written by the sender.

PEC Platform Components

Access Point (Punto di accesso): The Access Point 1. provides services to access, send and read PEC messages, 2. provides authentication service, 3. scans for viruses in the PEC message, 4. issues a receipt of acceptance (ricevuta di accettazione), and 5. creates the transportation envelope for the original message.

Receiving Point (Punto di ricezione): The Receiving Point 1. receives the message sent with an address that is registered with a PEC domain, 2. checks for the source and correctness of the PEC message, 3. issues the receipt for taking charge (ricevuta di presa in carico), 4. envelopes incorrect messages in an Anomaly envelope (Busta di anomalia), and 5. scans for viruses in the regular/ordinary email messages (posta ordinaria) and in the transportation envelope received.

Delivery Point (Punto di consegna): The Delivery Point 1. delivers the message in the recipient's PEC mailbox, 2. checks for the source and correctness of the message, and 3. issues the receipt of delivery (ricevuta di consegna) or the Notice of non-delivery (avviso di mancata consegna)

Acceptance Receipt (Ricevuta di accettazione): It contains the certification data and it's released from the Access Point when a PEC message is sent. It is signed with the sender PEC provider's (Gestore di posta elettronica certificata) private key.

Notice of non-acceptance (Avviso di non accettazione): It's the notice issued when the sender PEC provider cannot accept the incoming message. The non-acceptance cause is written in the text of the notice. It is signed with the sender PEC provider's private key.

Receipt for taking charge (Ricevuta di presa in carico): The receipt is issued from the recipient's PEC provider Receiving Point and sent to the sender's PEC provider. It contains certification data to be able to link it with the PEC message received It is signed with the receipt's PEC provider's private key.

Delivery receipt (Ricevuta di avvenuta consegna): The Delivery point provides to the sender the delivery receipt when the message is saved in the recipient's mailbox. One receipt is issued for each recipient of the message. It is signed with the recipient's PEC provider's private key.

Notice of non-delivery (Avviso di mancata consegna): If the PEC provider cannot deliver the message, a notice of non-delivery is sent to the message sender

Original message (Messaggio originale): It's the message sent by the PEC user before the deliver to the Access Point. The original message is delivered to the recipient through a transportation envelope (busta di trasporto) that contains it

Transportation envelope (Busta di trasporto): It's the message created by the Access Point, it contains the original message sent by the PEC user and the certification data. It is signed with the sender's PEC provider private key.

Anomaly envelope (Busta di anomalia): When an incorrect message or an ordinary mail message must be delivered to a user, the PEC provider insert it in an Anomaly envelope. The envelope is signed with the recipient's PEC provider's private key.

Certification Data (Dati di certificazione): A group of data that describe original message and are certified by the sender's PEC provider. They are delivered to the recipient with the original message inside the transportation envelope. The data contain: timestamp of the sending, sender, recipient, subject, ID of the message.

PEC provider (Gestore di posta elettronica certificata): It manages one or more PEC domains. It is the owner of the key used to sign receipt and envelopes. It cooperates with other PEC providers to send and deliver PEC messages.

PEC providers Index (Indice dei gestori di posta elettronica certificata): An LDAP Server containing all the PEC providers name, public keys certificates and PEC domains managed.

PEC mailbox (Casella di posta elettronica certificata): An electronic mail mailbox defined inside a PEC domain and managed by a PEC provider.

Generic Protocol Overview

The PEC system create messages (receipt, notice and envelope) in MIME format. The messages are composed by text and some attachments, like the original message and certification data. The message is composed in an S/MIME v3 structure signed with the private key of the PEC provider, the certificate with the PEC provider's public key is also put inside this structure. The format o S/MIME used in the message signature is "multipart/signature" (.p7s) as described in RFC 2633 § 3.4.3. Messages are transferred with a 7-bit encoding. The hashing algorithm used is SHA256.

To be verified, the sender of the PEC message (transportation envelope) must be the same as the one specified in the certificate of the S/MIME sign. This means that the sender of the PEC message will be different from the one of the original message. The original message will have

```
From: "John Doe" <john.doe@legalmail.it>
```

The transportation envelop will have

```
From: "On behalf of: john.doe@legalmail.it" posta-certificata@legalmail.it
```

The PEC messages are created by the PEC providers using special headers that identify the type of the message and give some other relevant data, like the ID of the message. The PEC message must be always delivered within 24 hours. If a sender's PEC provider won't receive a delivery receipt from the recipient's PEC provider within 24 hours, it will issue a Notice on non-delivery to its user (the original sender).

Interaction between two PEC providers

1a. User send an e-mail to the Access Point.

1b. Access Point return to the sender a Receipt of Acceptance.

2a. Access Point create a Transportation Envelope and forward it to the Receiving Point of the recipient's PEC provider.

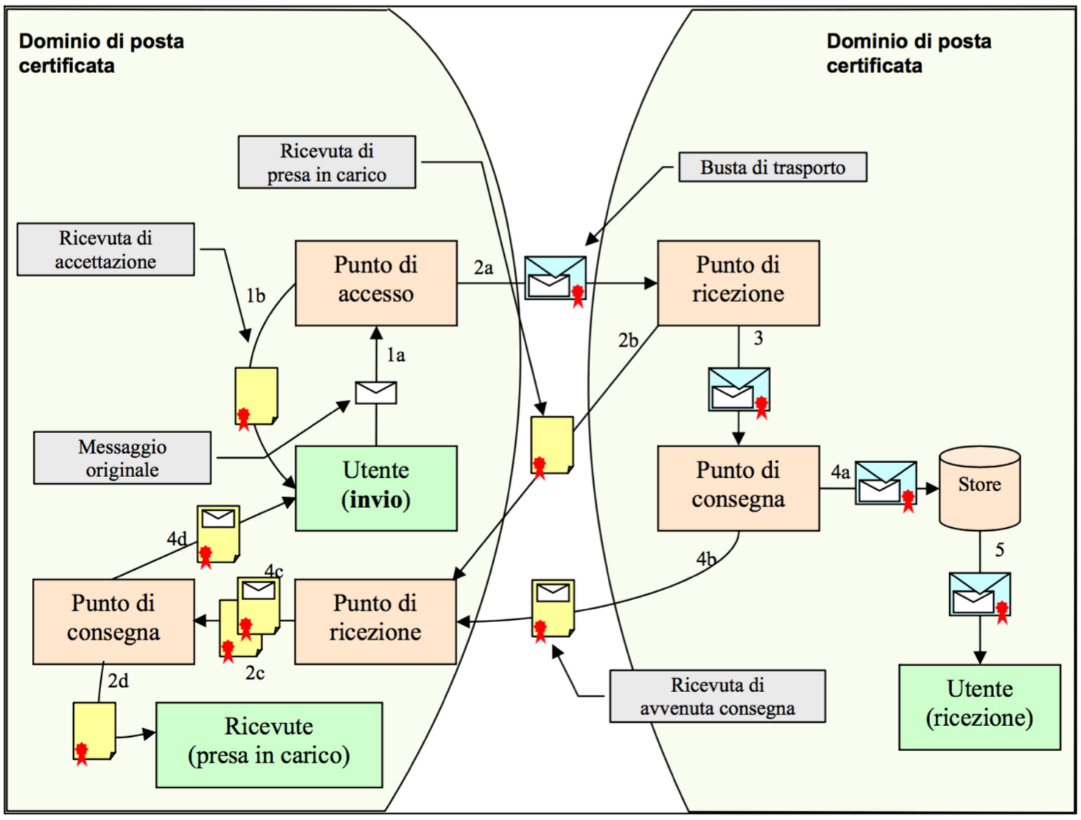


Figure 57: Correct Transportation Envelope with Successful Delivery

2b. Receiving Point verify the Transportation Envelope and generate a Receipt for taking charge that is forwarded to the Receiving Point of the sender's PEC provider.

2c. Receiving Point verify the validity of the Receipt for taking charge and forward it to the Delivery Point.

2d. Delivery Point save the Receipt for taking charge in the PEC provider store.

3. The Receiving Point forward the Transportation to the Delivery Point.

4a. Delivery Point verify the content of the Transportation Envelope and save it to the recipient's mailbox.

4b. Delivery Point create an Acceptance Receipt and forward it to the Receiving Point of the sender's PEC Provider.

4c. Receiving Point verify the validity of the Acceptance Receipt and forwards it to the Delivery Point.

4d. Delivery Point save the Acceptance Receipt to the sender's mailbox.

5. Recipients have now the e-mail sent to him.

G.2 S/MIME

S/MIME is It is a standard defined in a number of documents, most importantly RFCs 3369, 3370, 3850 and 3851, It stands for Secure/Multipurpose Internet Mail Extensions and is a standard for public key encryption and signing of MIME data (an email message). S/MIME allows you to: 1. Ensure to your email recipients that YOU actually sent the email 2. Allows the possibility of sending and/or receiving email encrypted

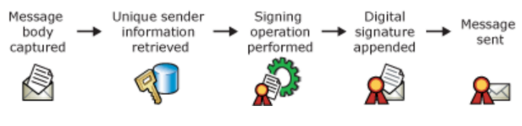
S/MIME specifies the MIME type application/pkcs7-mime (smime-type "enveloped-data") for data enveloping (encrypting) where the whole (prepared) MIME entity to be enveloped is encrypted and packed into an object which subsequently is inserted into an application/pkcs7-mime MIME entity.

Before S/MIME can be used in any of the above applications, one must obtain and install an individual key/certificate either from one's in-house certificate authority (CA) or from a public CA. The accepted best practice is to use separate private keys (and associated certificates) for signature and for encryption.

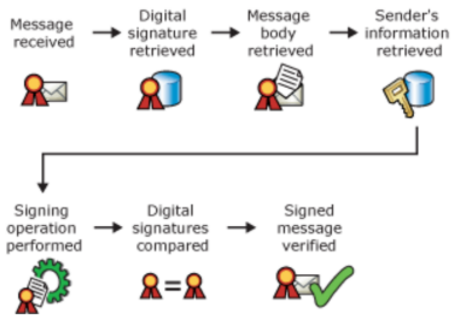
Digital Signature

The signing operation (cf. Figure 58a) that is performed when the message is sent requires information that can only be supplied by the sender. This information is used in a signing operation by capturing the e-mail message and performing a signing operation on the message. This operation produces the actual digital signature. This signature is then appended to the e-mail message and included with the message when it is sent.

For signature verification (cf. Figure 58b) the following steps are taken: 1. The message is received. 2. Digital signature is retrieved from the message. 3. Message is retrieved. 4. Information identifying the sender is retrieved. 5. Signing operation is performed on the message. 6. Digital signature included with the message is compared against the digital signature produced on receipt. 7. If the digital signatures match, the message is valid.


Message body captured
Unique sender information retrieved
Signing operation performed
Digital signature appended
Message sent

(a) Message Signing


Message received
Digital signature retrieved
Message body retrieved
Sender's information retrieved
Signing operation performed
Digital signatures compared
Signed message verified

(b) Verifying Message Signature

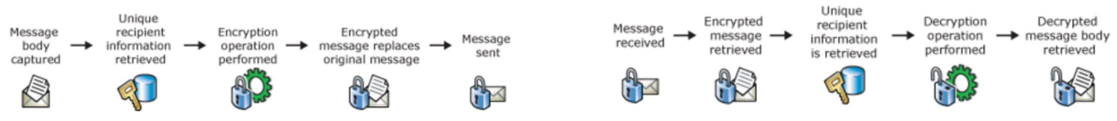
Figure 58: S/MIME Signatures

Encryption

Message encryption provides a solution to information disclosure. SMTP-based Internet e-mail does not secure messages. An SMTP Internet e-mail message can be read by anyone who sees it as it travels or views it where it is stored. These problems are addressed by S/MIME through the use of encryption.

The encryption operation (cf. Figure 59a) that is performed when the message is sent captures the e-mail message and encrypts it using information that is specific to the intended recipient. The encrypted message replaces the original message, and then the message is sent to the recipient. The following figure shows the sequence of encrypting an e-mail message.

For decryption (cf. Figure 59b) the following steps are taken: 1. The message is received. 2. Encrypted message is retrieved. 3. Information uniquely identifying the recipient is retrieved. 4. Decryption operation is performed on the encrypted message using the recipient's unique information to produce an unencrypted message. 5. Unencrypted message is returned to the recipient.


Message body captured
Unique recipient information retrieved
Encryption operation performed
Encrypted message replaces original message
Message sent
Message received
Encrypted message retrieved
Unique recipient information is retrieved
Decryption operation performed
Decrypted message body retrieved

(a) S/MIME Encryption

(b) S/MIME Decryption

Figure 59: S/MIME Encryption

G.3 SPID

SPID is the solution that allows you to access all the online services of public administration with a single Digital Identity. The SPID identity is formed by a pair of credentials (username and password) through which you can access the services from any device: computer, tablet and smartphone. There are 3 levels of SPID identity that correspond to Kantara[18], each with an increasing degree of security:

- **SPID Level 1** (Kantara AL2): allows access to services with a username and password.
- **SPID Level 2** (Kantara AL3): allows access to services with username, password and a temporary access code.
- **SPID Level 3** (Kantara AL4): allows access to services with username, password and the use of an access device.

To get a digital identity SPID you have to choose an Identity Provider among those accredited by AgID (Agency for digital Italy), then the Identity Provider must verify with certainty your identity through digital recognition (for example with qualified digital signature) or physical recognition (in person or with webcam recognition procedure).

[18]https://kantarainitiative.org/idassurance

Protocol and Interfaces

The mode of operation of the Identity providers will be those provided by SAML v2 for the profile "Web Browser SSO" [133] . The two versions "SP-Initiated": "Redirect / POST binding" and "POST / POST binding " must be provided. In these versions the authentication mechanism is triggered by the service provider, which is directed to the Identity Provider in “Pull” mode. The SAML authentication request (based on the construct <AuthnRequest>) can be submitted by a Service Provider to the Identity Provider using the HTTP Redirect binding or http binding POST. The SAML response (based on the <Response> construct) may be sent from the Identity Provider to Service Provider only via HTTP POST binding.

The logical interface of the Identity Provider are described as follows:

- The **IIDPUserInterface** allows users to interact via web with the component through User Agent under challenge authentication.
- The **IAuthnRequest** receives a SAML authentication requests.
- The **IMetadataRetrieve** allows the retrieval of Identity Provider’s metadata.

The logical interfaces of Service Provider are described as follows:

- The **IAuthnResponse** receives the SAML authentication responses.
- The **IMetadataRetrieve** allows the retrieval of Service Provider’s metadata.
- The **IDSResponse** receives feedback from the Discovery Service.

Single Sign-On Interaction

Figure 60 depicts the steps of a single sign-on process, which is further described as follows:

1. The user request access to a resource using the browser (User Agent).
2. This step is split in two: Firstly, the Service Provider send to the User Agent an authentication request to be provided to the Identity Provider. Secondly, the User Agent forward the authentication request to the Identity Provider. This interaction relies either on HTTP Redirect or HTTP Post to transmit SAML’s AuthnRequest.
3. The Identity provider verifies the received request and, if necessary, challenges the user for authentication.
4. The Identity Provider, with a successful authentication, prepares the Assertion with the authentication statement to be sent to the Service Provider (additional attributes may be present).
5. The Identity Provider replies to the User Agent returning the SAML Response with the assertion created at Step 4 (via HTTP POST).

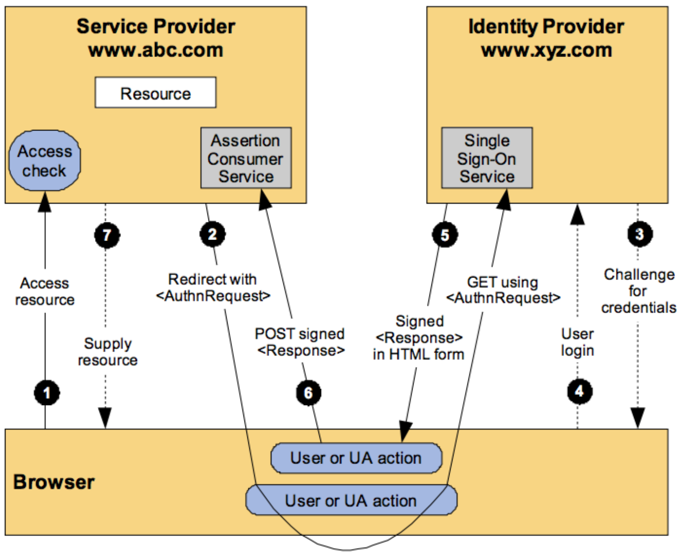


Figura 1 - SSO SP-Initiated Redirect/POST binding

Figure 60: SSO SP-Initiated Redirect/POST binding

6. The User Agent forwards the Response issued by the Identity Provider to the Service Provider.

7. Finally, the initially requested resource can be supplied.