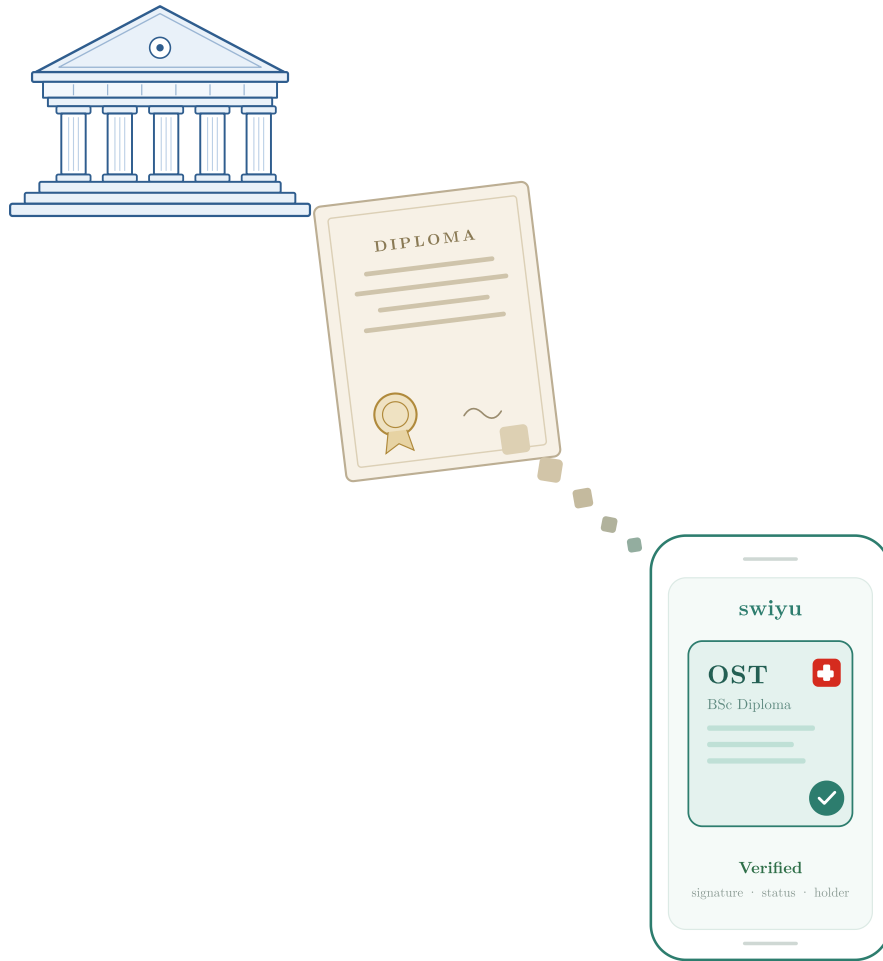


Ostschweizer Fachhochschule

Studiengang Informatik



Verifiable Credentials for Diplomas

Path to MVP

Bachelor's Thesis

Author Simon Dietschi

Supervisor Dr. Daniel Tschudi

Date 12 June 2026

Abstract

Academic credentials are still issued on paper or as PDF, formats that are easy to forge and slow to verify, since a recipient must contact the issuing institution and await a manual reply. The Swiss Trust Infrastructure (swiyu), operated by the Federal Office of Information Technology and approaching productive use, lets accredited organisations issue tamper-evident credentials that citizens hold in a government wallet and present without contacting the issuer. Higher-education institutions can issue academic credentials on it, but no reference implementation has been documented that integrates it with a university study-administration system at the granularity of individual modules.

This thesis specifies, implements and evaluates an issuer system for OST Eastern Switzerland University of Applied Sciences, built on the swiyu Generic Issuer reference container. It issues two credential types as SD-JWT VC: a Module Credential per completed module and a standalone Diploma Credential. Issuance is gated by a four-eyes approval bound to hardware security keys and coupled, in the same transaction, to a tamper-evident, append-only audit log; operators authenticate through the institutional Switch edu-ID with required multi-factor authentication.

Issuance, revocation and re-issuance were exercised end-to-end against the live swiyu beta infrastructure using the official Swiss wallet, and nearly all functional and most non-functional requirements were met. To the author's knowledge, no comparable self-hosted, integration-oriented implementation on the Swiss platform has been documented.

Future work centres on the path to a productive institutional deployment: protecting the issuer signing key with a production Hardware Security Module, replacing the mock service with an integration into the institution's live study-administration system, both technically and organisationally so that issuance fits the real operational workflow of the study administration, and a legal and data-protection review under the upcoming Federal Act on Electronic Identity and the Federal Act on Data Protection.

Management Summary

Initial Situation

Universities still issue diplomas on paper and transcripts as PDF. Both formats are easy to forge and hard to verify: a recipient who wants to confirm an applicant's academic record today has to contact the issuing institution by post or email and wait for a manual response. The Swiss Confederation addresses this problem at a national level with the Swiss Trust Infrastructure, marketed as *swiyu* and operated by the Federal Office of Information Technology. The platform allows accredited organisations to issue tamper-evident digital credentials that citizens hold in a government-issued wallet on their phone and present to any third party without contacting the issuing institution.

OST Eastern Switzerland University of Applied Sciences has expressed interest in becoming an issuer on this infrastructure but does not yet operate the technical components needed to do so. A preceding semester project in HS 2025, carried out by the author together with two fellow students, established a proof of concept that an academic credential can in principle be issued through *swiyu*. That proof of concept was not production-ready: it had no operator workflow, no audit trail, no approval ceremony, no re-issuance path, and no production-grade operator-side security controls.

This Bachelor's thesis closes that gap. It delivers a reference implementation of an academic credential issuer for OST, designed against the operational, legal and security constraints of a Swiss university of applied sciences. The system issues two distinct credential types: a *Module Credential* for each completed module and a standalone *Diploma Credential* for a finished degree. Reported commercial alternatives in the Swiss higher-education sector (educa with Procvivis, the Bern University of Applied Sciences with Accredify) issue one credential per qualification through software-as-a-service issuers; this work is, at the time of writing, the only documented implementation that models module-level credentials independently of diploma credentials and runs on self-hosted infrastructure.

Approach and Technologies

The solution is composed of three loosely coupled subsystems. A newly built *credential system* handles all issuer logic, operator workflows, approvals and the audit trail. A newly built *mock service* stands in for the institutional study-administration system myOST so that the system can be developed and demonstrated without touching production data. The third subsystem is the unmodified *Generic Issuer* reference container provided by the Federal Office of Information Technology, which performs the

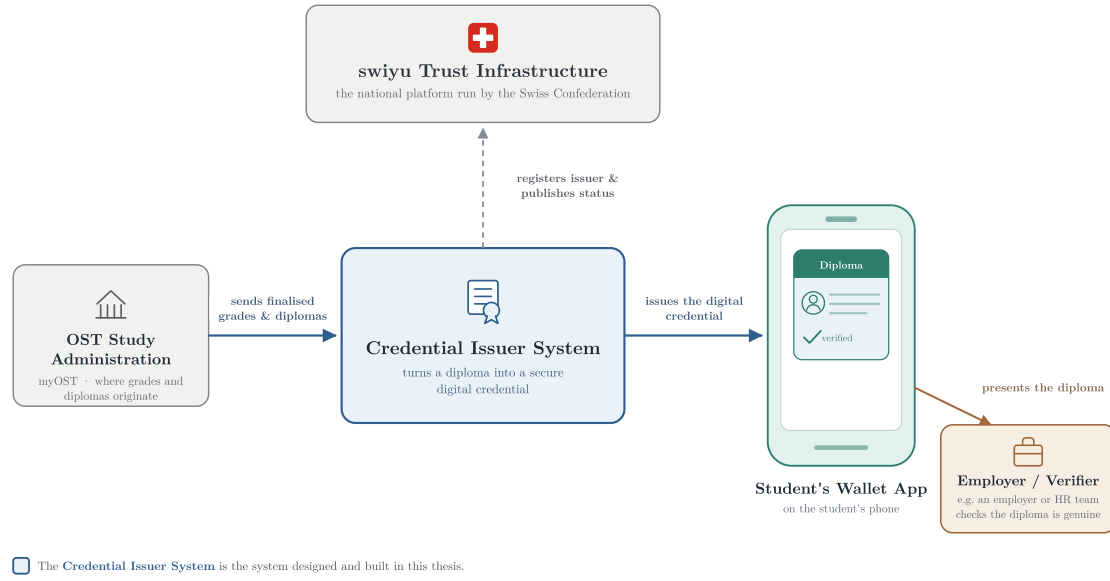


Figure 1.: High-level system context of the credential issuer system, which turns the study administration’s grades and diplomas into digital credentials.

cryptographic interaction with the swiyu base registry. Overall, the thesis consolidates three open-source third-party services (swiyu-issuer, swiyu-verifier, didtoolbox) and authors two in-house services.

The credential system is implemented in TypeScript on Node.js with a PostgreSQL database. Credentials are signed with a standard elliptic-curve signature scheme (ES256) and packaged as Selective Disclosure JSON Web Token Verifiable Credentials (SD-JWT VC), the format used on swiyu. Communication between the mock service and the credential system is event-driven and push-based; a separate pull path is used only to backfill current master data on demand, and a failed event delivery is recovered through a documented operator-triggered procedure.

Three controls go beyond a typical web application and reflect the sensitivity of issuing academic records:

- **Four-eyes approval bound to hardware keys.** Every credential issuance requires two distinct operators to approve on physical YubiKey security keys. The cryptographic signature covers the exact credential payload shown on screen, so an attacker who compromises the browser cannot substitute different data. Distinct approvers are enforced in three independent layers: the application logic, a database uniqueness constraint, and the WebAuthn protocol itself.
- **Tamper-evident audit log.** Every state-changing action is recorded in the database and, in the same transaction, in an append-only file on a separate volume. Each entry contains the cryptographic hash of the previous entry, so any retroactive modification is detectable. If the append-only volume is unavailable, the system refuses to write at all rather than silently losing the audit record.

- **Federated operator login.** Operators authenticate with their existing OST identity via Switch edu-ID and are required to complete multi-factor authentication (REFEDS MFA). No separate user database is maintained.

The system additionally supports two re-issuance flows, one for correcting master-data errors and one for students who have lost their wallet device. It also operates its own internal certificate authority for service-to-service encryption, so that no host on the public internet has to be trusted for traffic between OST-internal components.

Figure 2 summarises the credential lifecycle: a credential is issued only after a two-person approval, remains in use in the student’s wallet, and can later be revoked, suspended, or re-issued.

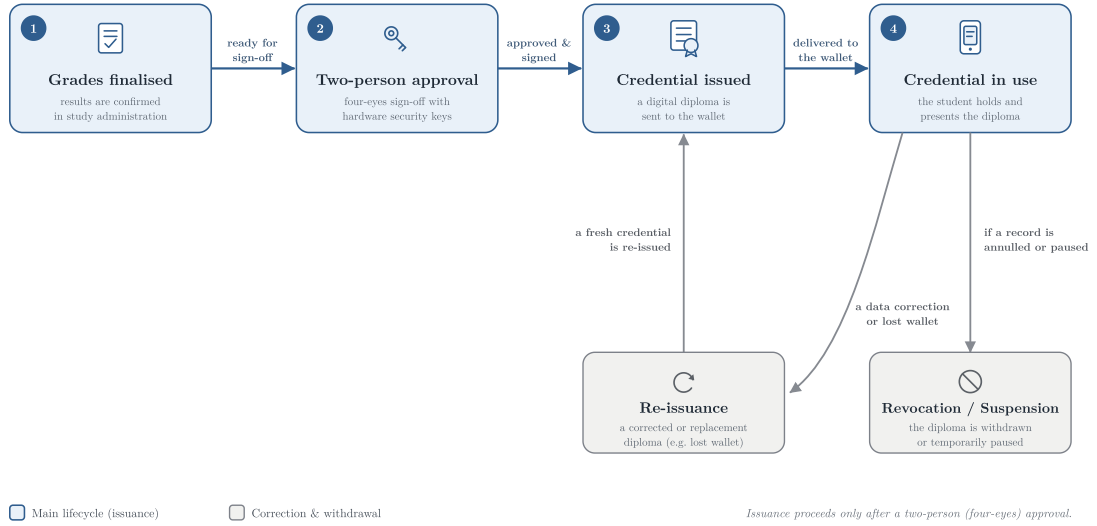


Figure 2.: Credential lifecycle, from approval and issuance to later status changes.

Results

The system was deployed against the live swiyu beta infrastructure of the Confederation and successfully issued credentials end-to-end into the official Swiss wallet application. Figure 3 shows the result from the student’s perspective: the OST diploma as a digital credential on the phone, ready to be presented to any verifier. The Demo watermark on the card marks the swiyu beta environment. Of 25 functional requirements defined for this thesis, 24 are fully achieved and one was descoped during the project (productive Hardware Security Module integration; the Securosys CloudHSM Sandbox account was requested but not activated in time). Of 18 non-functional requirements, 11 are achieved, 2 are partial, and 5 (performance, availability, and usability) were not measured under load and remain open; the diploma-batch issuance timing was the exception and met its two-minute target in a single measured run of 500 students. The twenty-four-hour

session lifetime and the ten-year credential validity are deliberate design decisions with documented compensating controls rather than gaps.

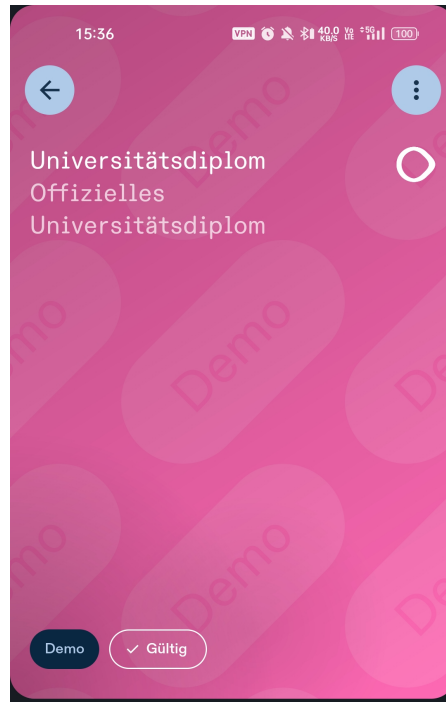


Figure 3.: The issued OST diploma in the official swiyu wallet.

The work was presented to the OST study administration on 2026-05-28. The presentation served as a stakeholder demonstration and as input for a future productive rollout decision; it was explicitly not a user acceptance test.

The deliverable is a self-contained codebase together with an accompanying operational concept that OST can adopt directly or hand to a third-party operator. Two areas remain for follow-up work beyond the descoped HSM integration: a legal review with the data protection office of OST to confirm compliance with the Swiss Federal Act on Data Protection (FADP) and the upcoming Federal Act on Electronic Identity, and a measurement campaign for the performance and availability requirements that this thesis defined but did not quantify.

Acknowledgements

Writing this thesis has been a genuinely rewarding chapter, and I am grateful to the people who made it so.

My thanks go to my supervisor, Dr. Daniel Tschudi, for his thoughtful guidance throughout this work and, just as much, for the trust he placed in me to pursue it independently. That balance of support and freedom shaped both the result and how much I enjoyed reaching it.

I am grateful to André Clerc, the expert accompanying this project, for his time, his careful feedback, and his assessment of the work. His perspective sharpened the thesis in ways I came to appreciate more with every iteration.

My thanks also go to Prof. Laurent Metzger, head of the Department of Computer Science at OST, for the warm interest he took in the project and for opening the door to the study administration and to the team behind the myOST platform. The conversations with them made the project feel real rather than hypothetical, and I would be glad to stay involved as it moves forward.

Finally, a big thank you goes to my colleagues Micha Harzbecker and Polina Lisetska, who started this project with me. Together we laid its foundations in the preceding semester project, and the enthusiasm and team spirit of that collaboration carried me through this thesis. It was a privilege to begin this journey with them.

Contents

Abstract	II
Management Summary	III
Acknowledgements	VII
Glossary and Abbreviations	XVI
1. Initial Situation and Problem Statement	1
1.1. Motivation	1
1.2. Problem Statement	1
1.3. State of the Art	2
1.4. Delimitation	2
2. Assignment and Objectives	4
2.1. Vision	4
2.2. Objectives and Deliverables	4
2.3. Success Factors	6
3. Constraints	7
3.1. Type of Work, Credits, and Time Budget	7
3.2. Scope and Methodology Constraints	7
4. System Context and Domain	9
4.1. Self-Sovereign Identity and Verifiable Credentials	9
4.1.1. The Three-Party Model	9
4.1.2. Verifiable Credentials and Selective Disclosure	9
4.1.3. Issuance and Presentation Protocols	10
4.1.4. Decentralised Identifiers and Status Information	11
4.2. Swiss Trust Infrastructure	11
4.2.1. Platform Components	12
4.2.2. Current Maturity and Limitations	12
4.2.3. Implications for This Thesis	13
4.3. Academic Credentials at OST	14
4.3.1. Existing Process for Diploma Issuance	14
4.3.2. Document Types and Data Items	15
4.3.3. Integration Constraints	16

5. Requirements	17
5.1. Stakeholders and Actors	17
5.2. Functional Requirements	17
5.2.1. Requirements Carried Forward from the Predecessor Project . .	18
5.2.2. New Requirements: Must Have	19
5.2.3. New Requirements: Should Have	20
5.2.4. Complex Workflows	21
5.3. Non-Functional Requirements	21
6. Market Analysis	23
6.1. educa and Switch: Maturity-Certificate Proof of Concept	23
6.2. Berner Fachhochschule IPST: CAS Continuing-Education Certificate Proof of Concept	24
6.3. Electronic Learner Driving Permit (eLFA)	24
6.4. Positioning of This Thesis	25
7. Prior Work	27
7.1. Outcomes of the Predecessor Project	27
7.2. Limitations of the Predecessor Project	27
7.3. Delimitation of the Present Thesis	28
8. Design and Architecture	29
8.1. Architecture Overview	29
8.1.1. Deployment Topology	29
8.1.2. Principal Data Flows	29
8.2. Design Decisions and Trade-offs	30
8.2.1. System Decomposition into Three Subsystems	30
8.2.2. Ownership of the Study-Administration Logic	31
8.2.3. Event-Driven Communication with Master-Data Backfill	31
8.2.4. Unified Batch Abstraction with Immutability after First Approval	32
8.2.5. Credential and Offer Validity	32
8.2.6. Identity Federation through Switch edu-ID	33
8.3. Subsystem Design	33
8.3.1. Mock Service	33
8.3.2. Credential System	34
8.3.3. Self-Hosted swiyu Generic Issuer	34
8.3.4. Verifier Subsystems	34
8.3.5. Expiry Worker	35
8.4. Credential Schemas	36
8.4.1. Common Format and Holder Binding	36
8.4.2. Module Credential	36
8.4.3. Diploma Credential	37
8.4.4. Selective Disclosure and Verifier-Side Presentation	38
8.4.5. Independence of Module and Diploma Credentials	39

8.5.	Lifecycle Flows	39
8.5.1.	Issuance Flow	40
8.5.2.	Re-Issuance after Master-Data Correction	41
8.5.3.	Re-Issuance after Wallet Loss	41
8.5.4.	Revocation and Suspension	42
8.5.5.	Systemic Expiry versus Status-List Revocation	42
8.5.6.	Webhook Delivery Semantics	43
8.6.	Four-Eyes Approval and Identity	43
8.6.1.	Role Concept and Authorisation Middleware	43
8.6.2.	Identity Federation and Local User Binding	44
8.6.3.	WebAuthn Registration and Authenticator Policy	44
8.6.4.	Cryptographic Four-Eyes Binding	45
8.6.5.	Bootstrapping and Open Limitations	46
8.7.	Internal Public Key Infrastructure	47
8.7.1.	Motivation	47
8.7.2.	Certificate-Authority Hierarchy and Provisioners	47
8.7.3.	Leaf-Certificate Lifecycle	48
8.7.4.	Trust Distribution and the swiyu TLS Proxy	49
8.8.	Audit Log Integrity	50
8.8.1.	Motivation and Scope	50
8.8.2.	Write-Once-Read-Many Storage	50
8.8.3.	Canonical-JSON Hash Chain	51
8.8.4.	Atomic Coupling and Coverage	51
8.8.5.	Verification and Failure Handling	52
8.9.	Status-List Partitioning	53
8.9.1.	Motivation and Design Drivers	53
8.9.2.	Partitioning Scheme and Mapping	53
8.9.3.	Lifecycle of a Partition	54
8.10.	Key Management	55
8.10.1.	Overview and Scope	55
8.10.2.	Key Inventory	56
8.10.3.	Storage and Protection	56
8.10.4.	Rotation, Bootstrap, and Re-keying	58
8.10.5.	Hardware Procurement and Residual Risks	58
9.	Security Concept	60
9.1.	Scope and Analysis Method	60
9.2.	Threat Model	60
9.2.1.	Actors	60
9.2.2.	Assets and Trust Boundaries	61
9.2.3.	Threat Surfaces	62
9.2.4.	The ITA Attack Cascade	62
9.3.	Security Controls and Mapping to Design	64
9.4.	Residual Risks Accepted in the MVP	64

9.5. Compliance Context	65
9.6. Open Items and Future Work	67
10. Implementation	68
10.1. Technology Stack and Dependencies	68
10.2. Repository Layout and Service Topology	69
10.3. Selected Implementation Aspects	69
10.3.1. Append-only WORM Storage Coupled to the Hash Chain . . .	69
10.3.2. Transactional Four-Eyes Approval Ceremony	70
10.3.3. Inter-Service Signature with Body-in-Claim Envelope	71
10.3.4. Leaf-Certificate Renewal with Hot Reload through SIGHUP . .	72
10.3.5. Idempotent Background Worker	73
10.3.6. Idempotent Inbound Webhook with a Composite Deduplication Key	73
10.4. Testing Approach	74
10.5. Deployment and Operational Interface	75
10.6. Implementation Limitations	76
11. Results	77
11.1. Delivered System	77
11.2. Functional Requirements Fulfilment	80
11.3. Non-Functional Requirements Fulfilment	81
11.4. Verification and Test Results	82
11.5. Encountered Platform Limitations	83
11.6. What Was Not Achieved and Why	83
12. Conclusion and Outlook	85
12.1. Conclusion	85
12.2. Outlook	86
Bibliography	88
A. Project and Time Management	93
A.1. Project Approach	93
A.2. Project Plan	93
A.3. Quality Assurance	94
A.4. Risk Analysis	95
A.5. Time Tracking	96
B. Operational Notes	98
B.1. Tools Used and Disclosure of AI Tools	98
B.2. Continuous Integration and Continuous Delivery	99
B.3. Installation Guide	99
B.4. User Guide	100

C. Supplementary Diagrams and Screenshots	101
C.1. System Walkthrough	103

List of Figures

1.	High-level system context of the credential issuer system, which turns the study administration's grades and diplomas into digital credentials.	IV
2.	Credential lifecycle, from approval and issuance to later status changes.	V
3.	The issued OST diploma in the official swiyu wallet.	VI
2.1.	System context (C4 Level 1) of the credential system and mock service.	5
4.1.	The three-party Self-Sovereign Identity model, showing the issuer, holder, and verifier.	10
4.2.	Anatomy of an SD-JWT Verifiable Credential.	11
4.3.	The swiyu platform components and their communication.	12
4.4.	High-level overview of the existing OST diploma issuance process. . . .	14
4.5.	Document types relevant to the credential system and their data items.	15
5.1.	Actors and system interactions of the credential platform.	18
8.1.	Container diagram (C4 Level 2) of the deployed single-host Docker Compose composition.	30
8.2.	Verifier subsystems and their interactions	35
8.3.	Hourly pass of the expiry worker	36
8.4.	State machine of a credential record.	40
8.5.	State machine of a wallet-loss re-issuance request.	42
8.6.	Activation and recovery of the YubiKey for the study administrator. . .	45
8.7.	Four-eyes approval ceremony over the fingerprint of the approval data.	46
8.8.	Trust hierarchy and provisioners of the internal certificate authority. . .	48
8.9.	Write-Once-Read-Many storage for the audit log.	50
8.10.	Status-list partitioning by semester and credential type.	54
8.11.	Server-side storage locations of the key material.	57
8.12.	Storage of the operator-authentication keys.	57
9.1.	The four trust boundaries of the composition and the authentication at each crossing.	62
9.2.	The four break points of the IT administrator attack cascade.	63
10.1.	Single database transaction of the four-eyes approval insert.	70
10.2.	Four-eyes approval ceremony in the operator interface.	71
10.3.	One renewal pass of the internal leaf certificates with hot reload.	73
10.4.	The nine compose stacks with the startup order of <code>make start</code>	75
11.1.	Operator interface of the credential system.	78

11.2. Diploma issuance on the demonstration deployment.	79
11.3. Diploma verification from presentation request to result.	79
A.1. Gantt-style overview of the project from the Jira Advanced Roadmap. .	94
A.2. Risk matrix of the project risk register.	96
A.3. Recorded work hours per project week.	97
C.1. End-to-end BPMN 2.0 collaboration diagram of the diploma issuance (rotated for legibility).	102
C.2. Landing page of the credential system with the edu-ID login.	103
C.3. The myOST mock service in its three role-specific views.	104
C.4. Public and administrative views of the verifier web application.	105
C.5. Module credential in the swiyu wallet.	105

List of Tables

2.1. Critical success criteria with associated targets.	6
3.1. Framework dates of this bachelor's thesis	7
5.1. Actors of the system and their responsibilities.	17
5.2. Functional requirements carried forward from the predecessor project. .	18
5.3. New functional requirements with priority must-have.	19
5.4. New functional requirements with priority should-have.	20
5.5. Non-functional requirements grouped by category.	21
6.1. Comparison of design choices: educa maturity-certificate proof of concept versus this thesis.	24
8.1. Claims of the module credential.	37
8.2. Claims of the diploma credential.	38
8.3. Claim sets of the standard Presentation Definitions.	39
8.4. Cryptographic keys of the composition.	56
9.1. Threat surfaces, with one representative threat per surface and the STRIDE classification.	63
9.2. Security controls of the composition and the design sections that docu- ment them.	64
9.3. Residual risks of the MVP, with compensating controls and future-work paths.	65
9.4. Mapping of the legal provisions to the answers of the credential system.	66
10.1. Top-level layout of the implementation repository.	69
10.2. Automated test suites of the implementation.	74
11.1. Functional-requirement fulfilment.	80
11.2. Non-functional-requirement fulfilment.	81
A.1. Principal project milestones.	94
A.2. Project risk register.	95
A.3. Recorded work hours per month.	97
B.1. Tool inventory by task area, per the OST departmental guideline on the use of AI tools.	98

Glossary and Abbreviations

Glossary

Approval (four-eyes principle) Mechanism that requires two distinct authenticated identities to confirm an action before it takes effect. Used in the credential system before any issuance batch is released, where a batch may contain a single credential or many, so that every issuance is covered.

Base Registry Central component of the Swiss Trust Infrastructure that holds the published DID Documents of issuers and verifiers.

Diploma Credential Verifiable Credential representing the awarding of an academic degree by a higher-education institution. In this thesis the diploma credential is a standalone proof that the degree was awarded; it carries the full degree statement directly and is independent of the module credentials, without aggregating or listing them.

did:tdw / did:webvh DID method used by the Swiss Trust Infrastructure, carried in the live issuer identifiers under the label `did:tdw` (version 0.3) and since renamed `did:webvh` in its upstream specification. The identifier resolves to a JSON document that, in the Swiss Trust Infrastructure, is served from the central Base Registry, accompanied by a verifiable history that protects against silent key rotation.

Generic Issuer Containerised reference component provided by the Swiss Trust Infrastructure that exposes the OpenID for Verifiable Credential Issuance endpoints.

Generic Verifier Containerised reference component provided by the Swiss Trust Infrastructure that exposes the OpenID for Verifiable Presentations endpoints.

Hashchain Linear chain of records in which each entry includes the hash of the previous entry, used to make tampering of an append-only log detectable.

Holder Natural person to whom a Verifiable Credential refers and who stores it under their own control in a wallet.

I14Y Swiss federal interoperability data catalogue at i14y.admin.ch, where data schemas may be registered for cross-organisation use.

Issuer Organisation that signs and issues a Verifiable Credential. In this thesis, OST in the role of the higher-education institution awarding the credential.

Mock service In this thesis, a stand-in application that represents the institutional study-administration system in the demonstration scenario in place of myOST.

Module Credential Verifiable Credential representing the successful completion of a single module of a study programme.

myOST Internal study-administration system of OST, successor of an earlier system named Adunis, based on Microsoft SQL Server with SQL Server Reporting Services for document generation.

Public Beta Pre-production phase of the Swiss Trust Infrastructure during which institutions may issue and verify Verifiable Credentials but the full legal framework of the productive identity does not yet apply.

Selective Disclosure Mechanism by which a Holder reveals only a chosen subset of the claims contained in a Verifiable Credential to a Verifier. The unrevealed claims remain bound to the issuer signature through their salted hashes.

Self-Sovereign Identity Paradigm for digital identity in which the Holder stores credentials and controls their disclosure, without depending on a centralised identity provider at the moment of presentation.

Status List Signed JWT published at a stable URL that encodes, for each issued credential, its current status. The encoding used in the Swiss Trust Infrastructure is two bits per entry, supporting the values valid, revoked, suspended, and unknown.

swiyu Brand name of the Swiss Trust Infrastructure operated by the Federal Office of Information Technology.

Trust Protocol 2.0 Mechanism introduced in swiyu Generic Issuer 3.1.0 by which an issuer publishes machine-readable trust statements about itself, complementing the cryptographic trust path with operational metadata.

Trust Registry Central component of the Swiss Trust Infrastructure that holds the trust statements about issuers and verifiers, complementing the cryptographic trust anchors of the Base Registry.

Verifiable Credential Tamper-evident set of claims about a Holder, signed by the Issuer in a format that the Verifier can validate cryptographically.

Verifier Party that receives a presented credential together with cryptographic evidence of the Holder's possession and validates both against the trust anchors.

Wallet Software, typically on the Holder's mobile device, that stores credentials and mediates their presentation to a Verifier.

WORM storage Write-Once-Read-Many storage, used to make the audit log of security-relevant operations tamper-resistant by physically preventing modification after the initial write.

Abbreviations

Abbreviation	Meaning
API	Application Programming Interface
BA	Bachelor's Thesis
BFH	Berner Fachhochschule
BGEID	Bundesgesetz über elektronische Identifizierungsmittel und andere elektronische Nachweise (Federal Act on Electronic Identity and Other Electronic Credentials)
BIT	Bundesamt für Informatik und Telekommunikation (Federal Office of Information Technology)
BPMN	Business Process Model and Notation
C4	Context, Container, Component, Code (software-architecture diagram method)
CAS	Certificate of Advanced Studies
CI/CD	Continuous Integration / Continuous Delivery
DCO	Digital Credentials OST (Jira project key of this thesis)
DCQL	Digital Credentials Query Language
DID	Decentralised Identifier
DIF	Decentralised Identity Foundation
DOZ	Lecturer (role identifier in this thesis; the mock-service dozent role)
DVS	Digital Public Services Switzerland (Digitale Verwaltung Schweiz)
ECDSA	Elliptic Curve Digital Signature Algorithm
ECTS	European Credit Transfer and Accumulation System
edu-ID	Switch edu-ID, federated identity for Swiss higher education
EPFL	École Polytechnique Fédérale de Lausanne
ES256	ECDSA on the NIST P-256 curve with SHA-256 (JOSE signature algorithm identifier)
FADP	Federal Act on Data Protection (Switzerland)
FR	Functional Requirement
FS26	Frühlingssemester 2026 (Spring Semester 2026)
HFKG	Bundesgesetz über die Förderung der Hochschulen und die Koordination im schweizerischen Hochschulbereich (Federal Act on the Funding and Coordination of the Higher Education Sector)
HSM	Hardware Security Module
IETF	Internet Engineering Task Force
IPST	Institute for Public Sector Transformation, Berner Fachhochschule
ITA	IT Administrator (role identifier in this thesis)
JWE	JSON Web Encryption
JWK	JSON Web Key
JWS	JSON Web Signature
JWT	JSON Web Token

Continued on next page

Continued from previous page

Abbreviation	Meaning
MAR	Reglement und Verordnung über die Anerkennung von gymnasialen Maturitätsausweisen (Swiss instrument on the recognition of maturity certificates)
MS	Mock service (role identifier in this thesis)
MVP	Minimum Viable Product
NFR	Non-Functional Requirement
OCA	Overlays Capture Architecture
OpenID4VCI	OpenID for Verifiable Credential Issuance
OpenID4VP	OpenID for Verifiable Presentations
OST	Ostschweizer Fachhochschule (Eastern Switzerland University of Applied Sciences)
PDF	Portable Document Format
PoC	Proof of Concept
QR	Quick Response (code)
REST	Representational State Transfer
RFC	Request for Comments
SA	Study Administration (role identifier in this thesis)
SD-JWT	Selective Disclosure JWT (RFC 9901)
SD-JWT VC	SD-JWT-based Verifiable Credential
SSI	Self-Sovereign Identity
SSRS	SQL Server Reporting Services
STU	Student (role identifier in this thesis)
TLS	Transport Layer Security
UC	Use Case
UML	Unified Modelling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VC	Verifiable Credential
VER	Verifier (role identifier in this thesis)
W3C	World Wide Web Consortium
WORM	Write Once Read Many

1. Initial Situation and Problem Statement

1.1. Motivation

Academic credentials in Switzerland still travel on paper or as institutional PDFs; at OST, the myOST study-administration system produces them [OST26]. Whoever receives such a document, usually a university or an employer, must inspect it manually and, when in doubt, confirm it by telephone or email with the issuing institution. That path is slow, easy to forge, and a poor fit for the fully digital application processes now common in admission and recruitment.

The Confederation has meanwhile built the foundations for a national alternative: a new federal act on electronic identity provides the legal basis, and the Swiss Trust Infrastructure (swiyu) provides the technology. The e-ID launch is expected for 1 December 2026; the Federal Council puts the act into force once the Federal Supreme Court has decided the pending ballot appeals [Fed26]. The platform is already in public beta, where institutions can issue and verify Verifiable Credentials [Swig; Swi26b].

Together these two developments open an opportunity: express academic credentials as Verifiable Credentials on the national infrastructure. A holder then presents a diploma from the official Swiss wallet, and the receiver verifies it cryptographically against trust anchors maintained by the Confederation, without ever contacting the issuing institution. Verification becomes near-instant, forgery resistance reduces to the strength of the underlying cryptography, and the credential turns machine-readable for digital application processes.

1.2. Problem Statement

OST issues nothing through the Swiss Trust Infrastructure today; diplomas and per-module records remain in the paper-and-PDF form described above [OST26]. The predecessor semester project [DLH25] already established feasibility: it issued a single OST diploma as an SD-JWT Verifiable Credential on the swiyu Generic Issuer under a fixed schema. What that proof of concept did not address is everything an institutional deployment demands: diplomas and per-module credentials side by side, a role concept that separates administrative responsibilities, bulk issuance at semester end, detection of out-of-application modifications to credential data, an integration contract with the institutional study-administration software, and the operational documentation that lets another institution adopt the system.

This thesis takes the proof of concept to a Minimum Viable Product that meets those requirements (Chapter 5), specifies the resulting architecture (Chapter 8), and realises the system (Chapter 10).

1.3. State of the Art

Three developments have fallen into place since the predecessor project, and together they make this the moment to build beyond a proof of concept. First, the standards have settled. The formats and protocols behind Verifiable Credentials all reached stable releases in the meantime: Selective Disclosure for JSON Web Tokens became RFC 9901 in November 2025 [Int25], the SD-JWT Verifiable Credentials profile built on it has been requested for publication [Int26], and the two OpenID specifications for issuance and presentation reached Final 1.0 in 2025 [Ope25a; Ope25b].

Second, the platform has caught up. Swiyu advanced during the thesis itself: Its Generic Issuer gained Trust Protocol 2.0 in May 2026 and refined its batch issuance mechanism in February 2026 [Swi26b]. Chapter 4 takes stock of where the platform stands and where it still falls short.

Third, this thesis is no longer alone. Several Swiss initiatives put proof-of-concept educational credentials on swiyu during 2025 and early 2026, and Chapter 6 compares them with this work along four axes. At the time of writing, no other documented initiative combines a self-hosted issuer, module-level granularity, and an integration contract with the study administration.

1.4. Delimitation

In the terminology of the OST documentation guide [EF22], this is an implementation-oriented bachelor’s thesis: it delivers a Minimum Viable Product with accompanying operational documentation, not an exploratory feasibility study. Chapter 7 delimits it against the predecessor project that established that feasibility.

The scope is the issuer side of the platform: The credential system, the mock study-administration service that stands in for the institutional study-administration software in the demonstration scenario, and the documented REST interface between the two. The verifier from the predecessor project is extended rather than rebuilt: it now verifies module as well as diploma credentials and, through a configurable trusted-issuer list, can accept credentials from issuers other than OST, which makes it a practical tool for the HR department to check the correctness of the digital credentials presented to it.

Three neighbouring topics stay out of scope and are recorded as future work. A legal and data-protection analysis of issuing Verifiable Credentials at a Swiss university of applied sciences is deferred to a later pilot phase, to be carried out in coordination with the responsible person at OST. Productive integration with a Hardware Security Module depends on the availability of suitable hardware and is carried as a should-have requirement (FR-22) whose deferral does not jeopardise the MVP. A direct connection of the credential system to myOST is excluded because access to a myOST test

environment could not be arranged within the time and clearance budget of a bachelor's thesis [OST26].

2. Assignment and Objectives

2.1. Vision

This thesis develops the predecessor proof of concept [DLH25] into a Minimum Viable Product (MVP) for issuing and revoking academic Verifiable Credentials on swiyu. The maturity step runs along three axes.

First, besides the diploma credential carried over from the predecessor, the system can also issue a credential per completed module, in line with OST’s modular teaching structure.

Second, it replaces a single-actor administrative surface with a controlled operational environment. Two organisational roles separate credential issuance from system administration, and a four-eyes approval gates every issuance, including the bulk runs at the end of an academic period.

Third, the system is built so that another institution can run it with its own study-administration software. The credential system is kept separate from the study-administration side, and the two communicate over a documented REST interface, so an institution swaps the mock service for its own software and connects through that interface without changing the credential system’s core; deploying it needs Docker, a configured domain, and the one-time swiyu onboarding, whose DID-toolbox step requires a Java runtime.

Figure 2.1 shows the resulting system context: the credential system and the study-administration mock service form the OST in-house scope, surrounded by the human actors and the external systems they depend on.

2.2. Objectives and Deliverables

This thesis is a single-person project, where the predecessor was carried out by a group of three [DLH25]. The per-person budget grows only moderately (360 instead of 240 hours) while the total budget halves, so the scope is deliberately narrowed to the issuer side of the platform; Chapter 7 delimits the work against the predecessor.

Within this focus, the thesis pursues the following high-level deliverables.

Credential system. A web application on the swiyu Generic Issuer supporting two credential types, diploma and module, with the full lifecycle of issuance, suspension, revocation, and restoration behind a role-aware administrative interface.

Verifier. The verifier from the predecessor project, extended to verify module as well as diploma credentials and, through a configurable trusted-issuer list, not only

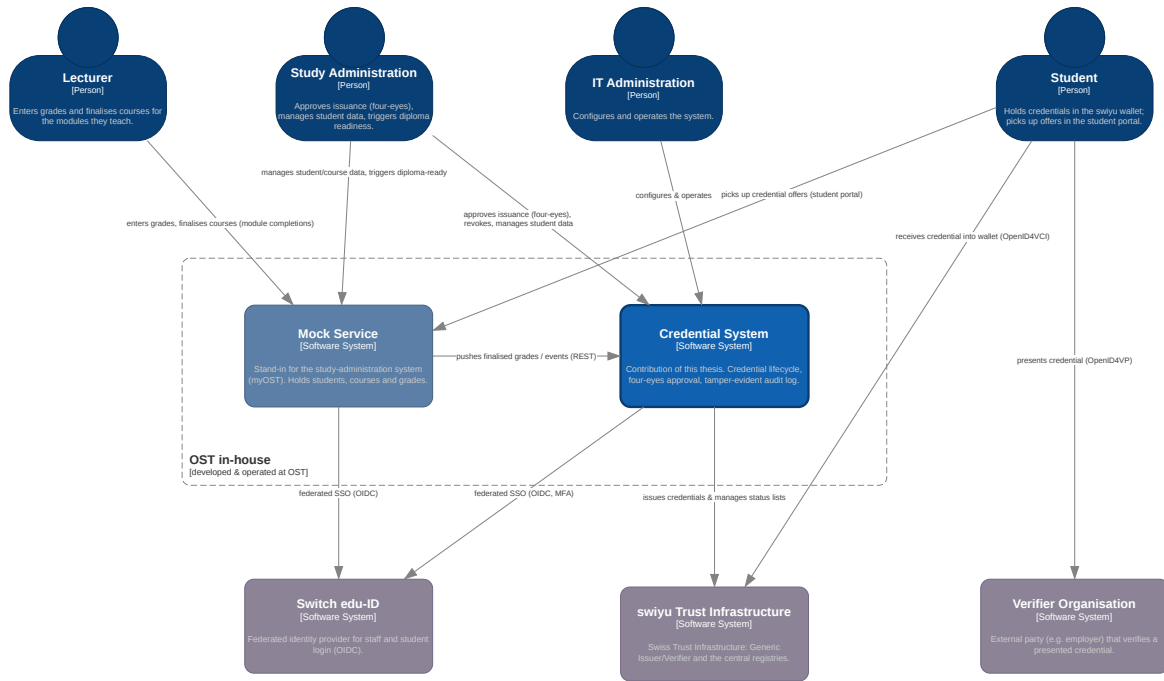


Figure 2.1.: System context (C4 Level 1) of the credential system and mock service.

those issued by OST, so that the HR department can check the correctness of digital credentials presented to it.

Mock service. A separate web application that represents the institutional study administration in the demonstration scenario. It holds student, course, and grade data, exposes a student portal for credential pick-up, and pushes finalised grades to the credential system over the documented REST interface.

REST interface specification. A documented contract between a study-administration system and the credential system, sufficient for an institution that already runs such a system to adopt the credential system without internal modifications.

Role concept and approval mechanism. An authentication and authorisation concept that distinguishes information-technology administration from study administration, integrates with Switch edu-ID single sign-on, and enforces a four-eyes approval before any issuance is triggered.

Security concept. A documented threat model, a key-management concept, and the integrity mechanism that detects out-of-application modification of credential data. The concept also names the risks left unmitigated within the MVP and records them as future work, including the deferred Hardware Security Module integration.

Operational documentation. Deployment guides, hardware and software requirements, and per-role instructions, sufficient for an IT-administrator at another institution to bring up the system independently.

The detailed functional and non-functional requirements that decompose these deliverables are documented in Chapter 5, the resulting architecture in Chapter 8.

The end-to-end demonstration scenario these deliverables realise, from the grade or diploma-readiness trigger in the mock service to credential pick-up by the student, is described in the design chapter (section 8.5.1) and documented as a BPMN collaboration diagram in appendix C (figure C.1).

2.3. Success Factors

Success is measured against quality criteria for the deliverables and the software, grouped into four categories: functional behaviour, operational behaviour, deployability, and transferability. Table 2.1 lists them with their targets.

Table 2.1.: Critical success criteria with associated targets.

Category	Criterion	Target
Functional	Module-credential batch issuance, up to 30 students per course	≤ 2 minutes
Functional	Diploma-credential batch issuance, up to 500 students per semester	≤ 2 minutes
Functional	Suspension, revocation, and restoration	2 to 3 clicks per operation
Functional	Detection of unauthorised data modification	required
Functional	Role-based access enforcement	both roles, scoped views
Operational	System uptime under met requirements	$\geq 99.9\%$
Deployability	Initial setup with Docker and Linux experience	≤ 4 hours
Deployability	Initial setup without prior experience	≤ 8 hours
Deployability	Instruction time per stakeholder role	≤ 1 hour
Transferability	Self-contained deployment documentation	required
Transferability	REST interface specification	published
Transferability	Template for additional credential schemas	provided

Beyond the table, the thesis addresses three shortcomings of the predecessor raised at the supervisor kick-off: the traceability of the path from proof of concept to MVP, the balance between depth and breadth of analysis, and the explicit treatment of non-functional requirements. These run through the requirements (Chapter 5), design (Chapter 8), and implementation (Chapter 10) chapters.

Work that exceeds the available budget is deferred to future work, in particular Hardware Security Module integration and a connection to a live study-administration system; both are delimited in section 1.4.

3. Constraints

This thesis is conducted as a bachelor’s thesis within the Bachelor of Science in Computer Science programme at OST Eastern Switzerland University of Applied Sciences. The work is governed by the binding programme guideline for bachelor’s and semester theses [RS25]. The constraints documented in this chapter follow directly from that guideline and from the framework dates of the spring semester 2026.

3.1. Type of Work, Credits, and Time Budget

The work is a single-person bachelor’s thesis at the end of the Bachelor of Science in Computer Science programme. The credit allocation is 12 ECTS credits, which corresponds to a work budget of 360 hours at a conversion factor of 30 hours per credit [RS25].

The 360 work hours are distributed across the spring semester 2026. The semester runs over 14 weeks, between calendar week 8 and calendar week 24, followed by an additional full-time block week before submission [RS25]. The reference distribution given by the programme guideline is approximately 40 hours during the block week and the remaining 320 hours spread over the 14 semester weeks, which corresponds to a weekly average of roughly 23 hours during the semester [RS25].

The concrete framework dates that apply to this thesis are summarised in table 3.1.

Table 3.1.: Framework dates of this bachelor’s thesis

Date	Milestone
16 February 2026	Start of the work (calendar week 8)
23 April 2026	Intermediate presentation with supervisor and expert
28 May 2026	Demonstration to the study administration of OST
12 June 2026 (17:00)	Submission of the bachelor’s thesis report (calendar week 24, Friday)
26 June 2026	Bachelor examination (final presentation and oral examination)

3.2. Scope and Methodology Constraints

Beyond the time and credit budget, the programme guideline imposes methodological constraints on the work [RS25]. The thesis must contain both a conceptual and a theoretical part as well as a practical part, and it must not consist of pure implementation work or of a purely theoretical treatment. Assumptions and decisions are to be

made plausible and their implications shown, missing knowledge is to be acquired independently, and contradictory requirements are to be discussed with the supervisor rather than used as a justification for unsatisfactory results. The work and its value must be positioned within its field, with related work presented at an appropriate level of detail. These methodological constraints are reflected in the structure of the present report and in the explicit treatment of related work in Chapter 6 and prior work in Chapter 7.

4. System Context and Domain

This chapter situates the thesis in three layers of external context, from general to specific: the international standards for self-sovereign identity and verifiable credentials that the platform builds on, the Swiss Trust Infrastructure (swiyu) that the credential system targets at runtime, and the academic-credential domain at OST where the system is deployed.

4.1. Self-Sovereign Identity and Verifiable Credentials

The platform follows the Self-Sovereign Identity (SSI) paradigm. The holder stores a credential in software under their own control and discloses it without going through a central identity provider. What lets a third party trust the credential is the issuer's signature and registry-based trust anchors, not an authenticated channel to an identity provider at presentation time.

4.1.1. The Three-Party Model

A credential exchange involves three principal roles. The *issuer* signs a credential that asserts a claim about the holder, the *holder* stores the credential in a digital wallet and decides if and when to present it, and the *verifier* receives the credential together with cryptographic evidence of the holder's possession and validates it against trust anchors [Wora]. Both issuer and verifier are typically organisations, while the holder is the natural person to whom the credential refers. The wallet is software operated under the holder's control, on a personal device. The three roles and the separation of the disclosure path from the trust path are shown in figure 4.1.

4.1.2. Verifiable Credentials and Selective Disclosure

A verifiable credential is a tamper-evident set of claims about a subject, signed by the issuer so that a verifier can check it cryptographically [Wora]. The Swiss Trust Infrastructure uses the SD-JWT Verifiable Credential profile [Int26], which builds on Selective Disclosure for JSON Web Tokens (RFC 9901) [Int25].

The format has two properties that matter here, both shown in figure 4.2. First, *selective disclosure*: the issuer signs only a hash of each claim, while the cleartext values travel separately, so the holder can reveal a subset of the claims and the issuer signature still holds. Second, *key binding*: the credential carries the holder's public key in the *cnf* claim (RFC 7800), and at presentation the holder proves possession with a short

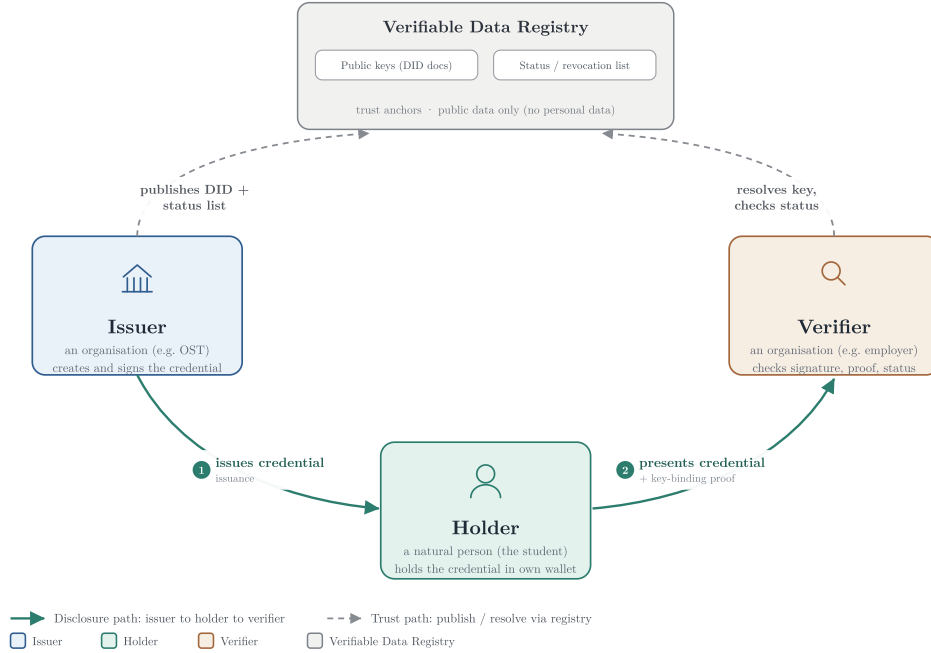


Figure 4.1.: The three-party Self-Sovereign Identity model, showing the issuer, holder, and verifier.

key-binding token. The Swiss Trust Infrastructure mandates ES256 for both the issuer signature and this proof [Swid].

In the swiyu realisation the holder has less control than the standard suggests: the Generic Issuer makes every custom claim disclosable regardless of issuer meta-data [Swi26c], and the wallet offers no per-claim selector, so the verifier’s presentation request decides what is actually revealed (Section 8.4.4).

4.1.3. Issuance and Presentation Protocols

Two OpenID specifications govern how credentials move between issuer, holder, and verifier. OpenID for Verifiable Credential Issuance covers how a holder requests and receives a credential, including format negotiation and request authorisation [Ope25a]. OpenID for Verifiable Presentations covers the inverse, how a verifier requests a presentation from a holder, including the verifier’s query [Ope25b]. Both reached Final in 2025: OpenID4VP 1.0 in July [Ope25b] and OpenID4VCI 1.0 in September [Ope25a].

A notable change in OpenID4VP 1.0 is the introduction of the Digital Credentials Query Language as the normative mechanism by which verifiers express their queries, replacing the earlier Presentation Exchange specification [Ope25b]. The swiyu Generic Verifier release used in this thesis documents the query language inconsistently: its

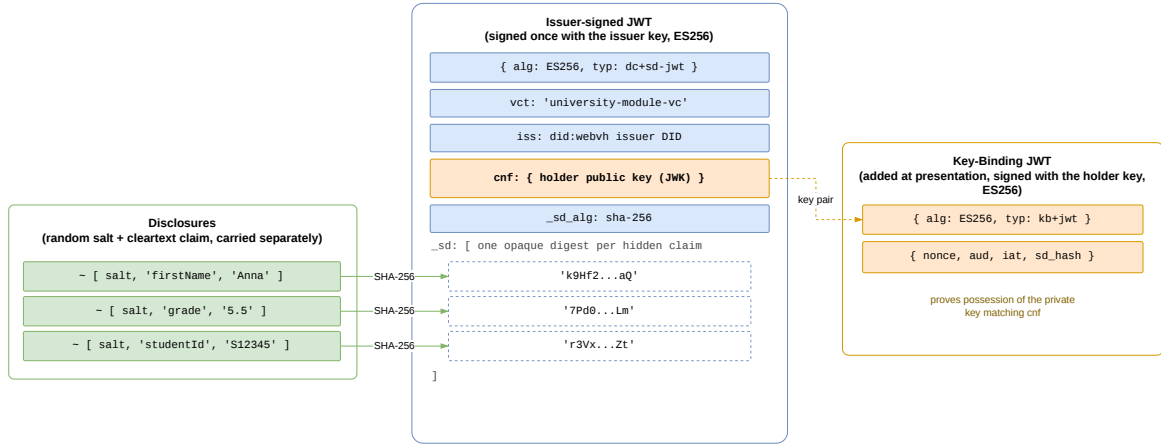


Figure 4.2.: Anatomy of an SD-JWT Verifiable Credential.

API documentation marks it as not yet implemented, and live requests that use it are rejected [Swi26d]. The official wallet has supported the query language since its update in May 2026 [Swi26a]. The verifier path of this thesis therefore relies on Presentation Exchange.

4.1.4. Decentralised Identifiers and Status Information

A Decentralised Identifier (DID) is a URI scheme that resolves to a DID Document holding the public keys for that identifier. The Swiss Trust Infrastructure standardised on the `did:tdw` method (version 0.3), since renamed `did:webvh` upstream. The DID resolves to a JSON document with a verifiable history that guards against silent key rotation, and that document is served from the central Base Registry (section 4.2.1) [Dec; Swi26b].

Revocation and suspension of issued credentials are expressed through the Token Status List specification, in which each issued credential is assigned an index into a publicly readable list whose entries encode the current status of the credential [Int]. The list is itself published as a signed JWT under a stable URL, and verifiers consult the list as part of credential validation. The Swiss Trust Infrastructure uses a two-bit encoding per entry, one bit for revocation and one for suspension, that supports the values *valid*, *revoked*, and *suspended*, and the present thesis introduces a partitioning scheme that maintains a separate list per semester and credential type, as defined in requirement FR-26.

4.2. Swiss Trust Infrastructure

The Swiss Trust Infrastructure, run by the Federal Office of Information Technology, Systems and Telecommunication (FOITT, German abbreviation BIT) under the brand *swiyu*, is the runtime platform the credential system depends on. It has been in public beta since March 2025 [Swic] and is set to go productive in 2026, once the Federal Act on Electronic Identity and Other Electronic Credentials (BGEID) enters into force [Swig].

4.2.1. Platform Components

The platform comprises reference components that institutions can deploy on their own infrastructure to issue or verify credentials, complemented by registries operated centrally by the Confederation. This thesis uses the Generic Issuer [Swi26b] and the Generic Verifier [Swi26d], which provide the OpenID4VCI and OpenID4VP endpoints respectively as containerised services. Three central registries support the trust model: the Base Registry stores issuer and verifier DID Documents, the Status Registry hosts the published Token Status Lists, and the Trust Registry adds human-readable information on the legitimacy of an issuer or verifier. The official Swiss wallet application is the holder-side counterpart that interacts with both endpoints. Figure 4.3 shows these components and their communication at issuance and verification time.

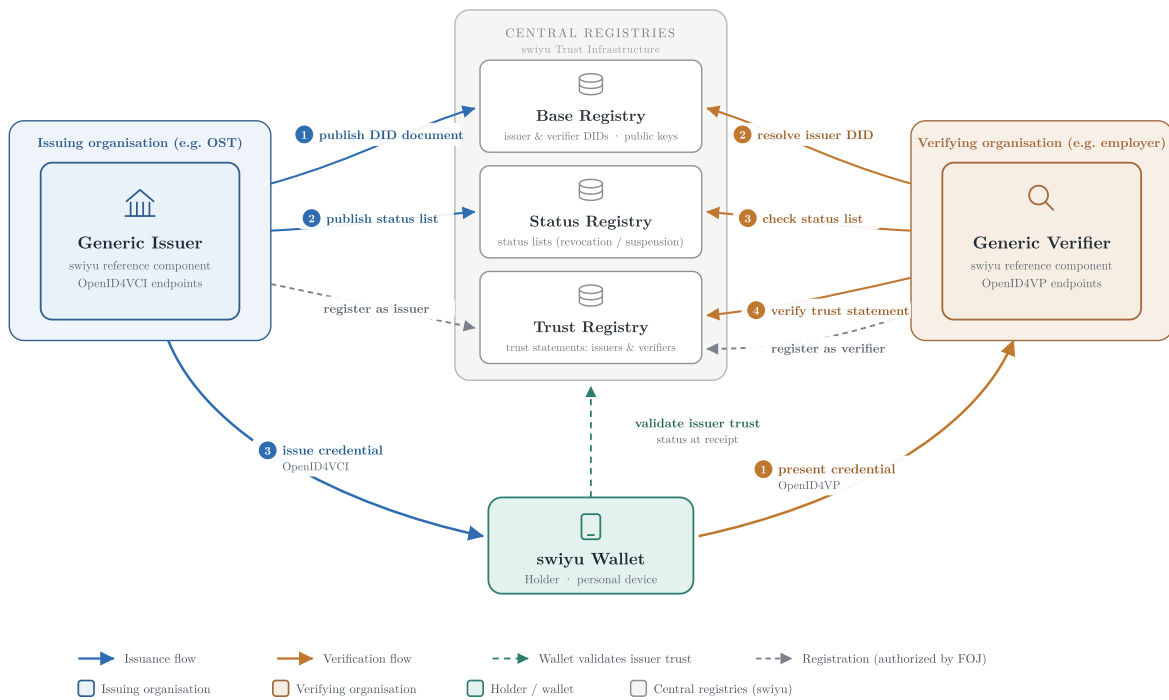


Figure 4.3.: The swiyu platform components and their communication.

4.2.2. Current Maturity and Limitations

As of May 2026 the platform is still in public beta [Swig]. The beta identity carries the same attribute set as the e-ID defined in Article 15 BGEID, but its technical features are restricted and the legal requirements of Section 3 BGEID do not apply, so it is more limited than the productive identity due at go-live in 2026 [Swig]. It has advanced considerably since the thesis began; three sets of advances matter here.

First, the Generic Issuer supports the OpenID4VCI batch issuance mechanism, refined in release 2.4.0 (February 2026) [Swi26b]. In this mechanism, a *single* wallet obtains several instances of the same credential in one request, each bound to a different holder key, so a fresh instance can be presented to each verifier and the holder cannot be correlated across presentations [Ope25a]. This operates per holder and is therefore distinct from the end-of-semester issuance to *many* holders required by FR-16 and the

performance targets NFR-01 and NFR-02 (Chapter 5), which the credential system realises through orchestrated per-holder calls to the Generic Issuer. The credential system deliberately does not use the batch mechanism for diplomas: a diploma is inherently identifying, so token-level unlinkability buys little, and the relevant privacy lever is the selective disclosure of the SD-JWT format instead. For now the platform offers no holder-side selection interface, so the system realises this lever through verifier-controlled presentation rather than holder choice (Section 8.4.4). Section 12.2 revisits the mechanism as future work, for a potential anonymous, frequently presented credential type.

Second, release 3.1.0 (May 2026) added Trust Protocol 2.0, including the identity (idTS) and protected issuance authorization (piaTS) trust statements that let a verifier obtain authoritative trust information about an issuer [Swi26b]. Third, JSON Web Encryption (JWE) of credential payloads and Overlays Capture Architecture (OCA) for wallet-side rendering are now supported standards in the public beta [Swig], both flagged as gaps in earlier platform analyses.

Limitations Affecting This Thesis

This subsection documents the limitations that remain and that directly shape the credential system’s design. They concern the Generic Verifier and the official wallet.

Software-based device binding, binding a credential to a wallet instance without a secure hardware module on the holder’s device, is listed on the roadmap as unsupported in the beta and scheduled for go-live in 2026 [Swig]. Verification of hardware key attestations (Android Key Attestation, iOS Client Attestation), which would prove a binding key sits in secure hardware such as the Apple Secure Enclave or the Android Trusted Execution Environment, is likewise not yet implemented [Swig]. Credentials issued during the beta therefore cannot be bound to the holder’s device in a way that survives wallet reinstallation, which informs the re-issuance design in Chapter 8.

Combined presentation of multiple credentials in one OpenID4VP exchange, and combined verification, was flagged as a gap at kick-off. The roadmap still lists both as unsupported as of February 2026 [Swig], so the design conservatively assumes diploma and module credentials are presented independently. If the feature lands before submission, the Chapter 8 design extends to it without architectural changes.

Offline presentation, a wallet history view, and wallet backup and restore were likewise flagged as gaps at kick-off. The roadmap lists each as unsupported as of February 2026 [Swig]; the thesis treats their absence as a constraint on the wallet user experience, not as a capability the credential system can rely on.

4.2.3. Implications for This Thesis

Two properties of the current platform shape the design that follows. Trust Protocol 2.0 enables a more mature design than the predecessor project, where authoritative issuer trust statements were unavailable or unstable [DLH25]. The absence of combined presentations, together with the limited software device binding, drives the choice to

issue diplomas and module credentials separately, with independent revocation paths (Section 8.4).

The wider security analysis carried out in Chapter 9 accounts for the platform limitations documented above. Limitations that are scheduled to be resolved during the lifetime of the platform are documented as known gaps of the current deployment environment rather than as shortcomings of the credential system itself.

4.3. Academic Credentials at OST

The credential system operates in the domain of issuing and verifying academic credentials at OST. Over a student's studies the institution issues several document types, from per-semester transcripts to the final diploma. Today these are produced by internal software and delivered on paper or as PDFs, with no externally accessible programmatic interface.

4.3.1. Existing Process for Diploma Issuance

The internal study-administration system at OST is named myOST and is the successor of an earlier system known as Adunis [OST26]. The data layer is a Microsoft SQL Server, and reports are generated through SQL Server Reporting Services (SSRS) directly from the database tables [OST26].

The process for a diploma proceeds in four conceptual steps, summarised in figure 4.4 [OST26]. Faculty members enter semester grades, examination grades, and module-final grades into myOST. Once all grades for a course are recorded, the corresponding semester certificates can be generated through SSRS. For diploma certificates, the system performs an automated compliance check (Leistungsüberprüfung) that evaluates the recorded grades against the rules of the study regulations of the relevant programme [OSTb]. If the check is positive, a diploma certificate is generated and the formal diploma confirmation (Diplomierung) is recorded. If the check is not positive, no diploma is issued; instead an interim certificate or performance report (Leistungsreport) is generated that documents the current standing of the student [OST26].

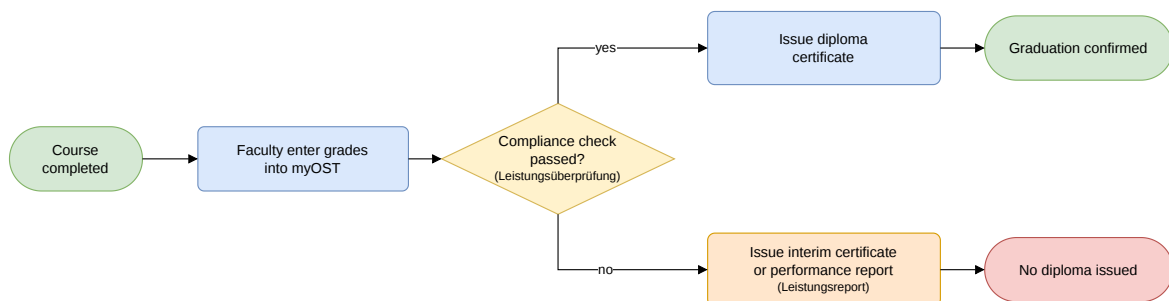


Figure 4.4.: High-level overview of the existing OST diploma issuance process.

4.3.2. Document Types and Data Items

OST issues two document types today: the semester certificate (Semesterzeugnis) and the diploma certificate (Diplomzeugnis). The semester certificate lists each module with its identifier, the courses it comprises, the grade with any sub-component grades, and the semester of completion; faculty names do not appear [OST26]. The module credential introduced by this thesis corresponds to a single such per-module entry. The diploma certificate lists every completed module with its ECTS credits and grade, sums them to the total credits and the final overall grade, breaks the credits down by the categories of the study regulations, and states the degree designation with major and specialisation together with the graduation date [OST24]. Its personal data is limited to the student's name, date of birth, matriculation number, and type of study [OST24]; further master data such as the postal address and the internal personal identifier (Pid) live in myOST but do not appear on the diploma certificate [OST26; OST24].

Every module also has a public description on the OST website, reachable through a stable URL linked from unterricht.ost.ch, which adds context such as the ECTS credit allocation [OST26]. The credential-schema design (Section 8.4) starts from these data items when deciding which claims the module and diploma credentials carry. Figure 4.5 summarises the data items of the two document types and the public module description.

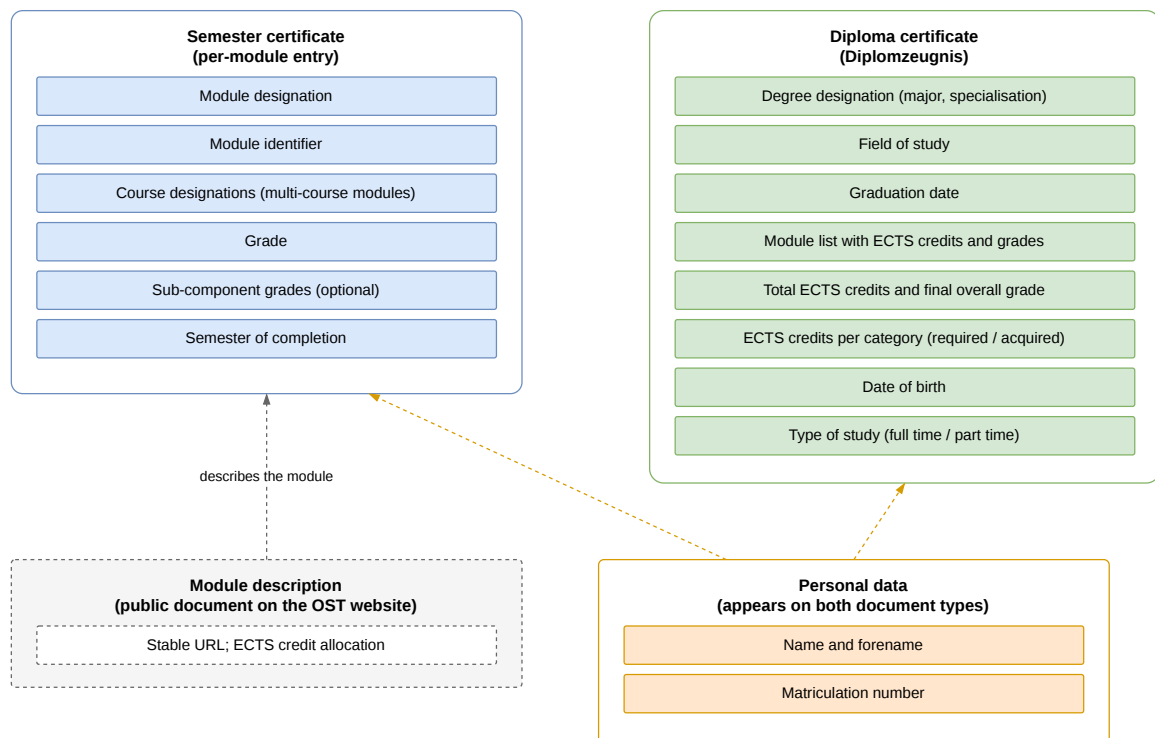


Figure 4.5.: Document types relevant to the credential system and their data items.

4.3.3. Integration Constraints

Programmatic access to myOST was requested early in the project. The written answer stated that the myOST system offers no programmatic interface for external access to grade or student data, and that opening such access involves a lengthy legal and data-protection clearance process, since grades are among the most sensitive data an institution holds [OST26]. A direct integration with myOST was therefore placed out of scope. Towards the end of the project, conversations with the team working with myOST opened a second avenue, an integration against the staging instance of myOST; this option surfaced too late to be realised within the thesis period.

This constraint shapes the architecture documented in Chapter 8 in three ways. The credential system itself is decoupled from any specific study-administration system through a documented REST interface (FR-23). The demonstration scenario is supported by a mock service that takes the place of myOST and exposes data in the shape required by the credential system. The connection of the credential system to myOST is documented as future work, with the staging instance as the first integration target; a productive connection remains contingent on the legal and data-protection clearances required by the institution.

5. Requirements

This chapter documents the functional and non-functional requirements that follow from the assignment and the early project analysis. They are formulated platform- and technology-neutrally and reference the actors of the system, which are introduced first. Requirement identifiers are stable across the lifetime of the project and are never reassigned, so the numbering carries gaps.

5.1. Stakeholders and Actors

The actors that interact with the system, either directly through a user interface or indirectly through a machine-to-machine interface, are summarised in table 5.1. The actor identifiers introduced here are used throughout the remainder of this chapter.

Table 5.1.: Actors of the system and their responsibilities.

Identifier	Actor	Responsibility
ITA	IT administrator	System configuration, user administration, maintenance.
SA	Study administration	Credential issuance, revocation, student data.
DOZ	Lecturer	Grade entry and course finalisation; may request a revocation.
STU	Student	Picks up credential offers via the student portal.
CS	Credential system	The system built in this thesis; fulfils automated requirements itself, without a human actor.
MS	Mock service	System that replaces the myOST environment.
VER	Verifier	Verifies presented credentials, for example an employer.

Figure 5.1 complements table 5.1 by making the topology of the interactions explicit: which actor interacts with which subsystem, and whether the interaction is human-to-system (solid associations) or system-to-system (dashed associations).

5.2. Functional Requirements

The functional requirements are grouped into three subgroups. The first subgroup contains requirements that describe functionality already implemented in the predecessor proof-of-concept [DLH25] and that is preserved, with adaptations, in the present thesis. The second and third subgroups contain new requirements introduced by the present thesis, separated by their priority into must-have and should-have requirements.

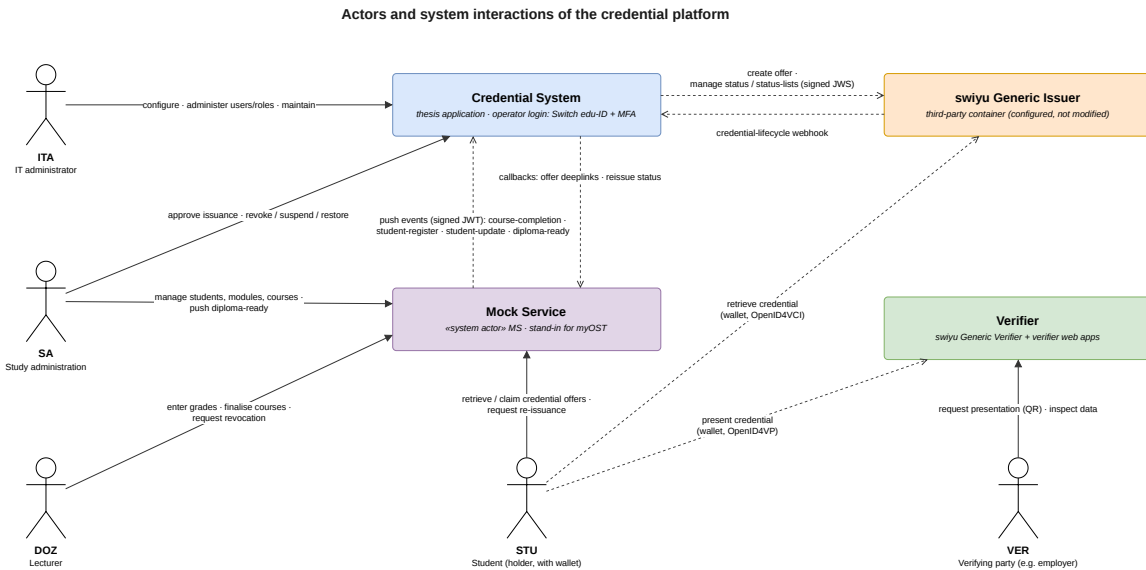


Figure 5.1.: Actors and system interactions of the credential platform.

5.2.1. Requirements Carried Forward from the Predecessor Project

The requirements in table 5.2 are adapted to the two-service architecture of credential system and mock service [DLH25].

Table 5.2.: Functional requirements carried forward from the predecessor project.

ID	Actor	Requirement
FR-01	SA	As a member of the study administration, I want to record and maintain students and their grades in the mock service, so that the data is available for credential issuance.
FR-02	SA	As a member of the study administration, I want to issue a diploma credential for a student in the credential system, so that the academic degree is digitally verifiable.
FR-03	SA	As a member of the study administration, I want to revoke an issued credential in the credential system, so that invalid degrees can no longer be verified.
FR-04	SA	As a member of the study administration, I want to suspend and reactivate an issued credential in the credential system, so that temporary blocks are possible.
FR-05	SA	As a member of the study administration, I want to view an overview of the status list in the credential system, so that I can see which credentials are assigned to which status list.
FR-06	VER	As a verifier, I want to request a credential from a holder via a QR code, so that I can verify an academic degree.
FR-07	VER	As a verifier, I want to inspect the data of a presented credential, so that I can validate the claims of an applicant.

5.2.2. New Requirements: Must Have

The must-have requirements in table 5.3 are the additions that distinguish the bachelor's thesis from the predecessor proof-of-concept. They are required to fulfil the vision documented in section 2.1.

Table 5.3.: New functional requirements with priority must-have.

ID	Actor	Requirement
FR-09	CS	The credential system shall receive student master data (name, matriculation number) and module assignments from the mock service and shall store them locally, so that credentials can be assigned, searched, and managed.
FR-10	SA	As a member of the study administration, I want to issue module credentials in addition to diploma credentials in the credential system, so that individual module completions are digitally verifiable.
FR-11	SA	As a member of the study administration, I want to manage both credential types, diploma and module, through the same revocation and suspension flow in the credential system, so that the process is uniform.
FR-12	ITA	As an IT administrator, I want to assign a role, IT administrator or study administration, to users in the credential system, so that only authorised persons can access the corresponding functions.
FR-13	SA	As a member of the study administration, I want to see only the functions corresponding to my role in the credential system, namely credential issuance, revocation, and student data, so that the interface remains clear and secure.
FR-14	ITA	As an IT administrator, I want to see only system configuration and user management, but not be able to issue credentials in the credential system, so that manipulation by IT staff is prevented.
FR-15	CS	The credential system shall detect unauthorised modifications to database entries through a hash-based mechanism and shall prevent the issuance of credentials based on manipulated data.
FR-16	SA	As a member of the study administration, I want to issue all passed module credentials for a finalised course in a single operation in the credential system, that is bulk issuance, so that the end-of-semester workflow is efficient.
FR-17	MS	The mock service shall push course data, namely grades, students, and module information, to the credential system as soon as the grades of a course are complete.
FR-18	CS	The credential system shall receive data from the mock service and shall prepare module credentials for finalisation without issuing them immediately.

Continued on next page

Continued from previous page

ID	Actor	Requirement
FR-19	MS	The mock service shall check whether a student fulfils all diploma prerequisites, that is the required ECTS per category according to the study regulations, and shall send a diploma-ready event to the credential system. The credential system shall thereupon prepare a diploma credential for finalisation. The diploma logic, including categories and thresholds, shall reside exclusively in the mock service.
FR-20	SA, ITA	As an authenticated user of the credential system, I want to authenticate via Switch edu-ID single sign-on with second-factor authentication, so that the authentication mechanism conforms to the institutional standard.
FR-26	CS	The credential system shall maintain a separate status list per semester and credential type. The assignment shall be realised relationally through the columns <code>semester</code> and <code>credential_type</code> of a dedicated status-list table under a uniqueness constraint, with the list identifier returned by the swiyu Generic Issuer at provisioning time serving as the identifier of the list.

5.2.3. New Requirements: Should Have

The should-have requirements in table 5.4 are pursued where time and external dependencies permit. They are not strictly required to deliver the MVP, but they substantially improve the institutional value and the transferability of the system.

Table 5.4.: New functional requirements with priority should-have.

ID	Actor	Requirement
FR-21	STU	As a student, I want to log into the student portal of the mock service via Switch edu-ID and to view my available credential offers, so that I can obtain my credentials.
FR-22	ITA	As an IT administrator, I want to store the issuer's private key in a Hardware Security Module, so that the key is protected against unauthorised access.
FR-23	CS	The credential system shall expose a documented REST interface through which any study-administration system can deliver data.
FR-24	MS	The mock service shall notify the credential system of master-data changes, for example name changes, so that affected credentials can be marked for review.

Continued on next page

Continued from previous page

ID	Actor	Requirement
FR-25	SA	As a member of the study administration, I want to view an overview of credentials affected by master-data changes in the credential system, so that I can decide whether revocation and re-issuance are required. The revocation triggers the four-eyes approval flow.

5.2.4. Complex Workflows

Two of the must-have requirements describe workflows that involve several actors and multiple sequential steps, and are therefore documented as use cases. UC-01 covers the four-eyes approval that precedes any bulk issuance (referenced by FR-16, FR-13, and FR-15); UC-02 covers the event-driven communication between the mock service and the credential system (referenced by FR-17, FR-18, FR-19, and FR-21). The detailed sequences of both are documented in Chapter 8.

5.3. Non-Functional Requirements

The non-functional requirements are grouped by category and listed in table 5.5. Each requirement carries a measurable target. The detailed design of the mechanisms required to satisfy the database-integrity requirement (NFR-06), the four-eyes-approval requirement (NFR-07), and the audit-log requirement (NFR-10) is documented in Chapter 8.

Table 5.5.: Non-functional requirements grouped by category.

ID	Category	Requirement and target
NFR-01	Performance	A module-credential batch issuance, arising from a finalised course and measured from the moment the second four-eyes confirmation is recorded, shall complete within 2 minutes for batches of up to 30 students.
NFR-02	Performance	A diploma-credential batch issuance, arising from an end-of-semester finalisation and measured from the moment the second four-eyes confirmation is recorded, shall complete within 2 minutes for batches of up to 500 students.
NFR-03	Performance	Credential preparation following data receipt from the mock service shall complete within 30 seconds.
NFR-04	Performance	ECTS aggregation following course finalisation shall complete within 10 seconds.
NFR-05	Availability	The overall system shall achieve an availability of at least 99.9 % under the documented infrastructure requirements.

Continued on next page

Continued from previous page

ID	Category	Requirement and target
NFR-06	Security	The hash-based mechanism of FR-15 shall detect 100 % of unauthorised database modifications.
NFR-07	Security	The four-eyes approval shall guarantee that two distinct identities confirm an action. The mechanism shall provide no path that allows a single identity to confirm on its own.
NFR-08	Security	Communication between the mock service and the credential system, and to external services, shall use TLS 1.2 or higher, with TLS 1.3 preferred.
NFR-09	Security	Operator sessions shall have an absolute lifetime of 24 hours, after which re-authentication is required.
NFR-10	Security	All security-relevant actions, including credential issuance, revocation, suspension, finalisation confirmations, login attempts, and integrity violations, shall be recorded in a tamper-evident audit log. Tamper evidence is achieved by a per-entry hashchain that links each entry to the hash of the previous entry, complemented by an external write-once-read-many append-only log that is held outside the application container and continuously cross-checked against the database. Each entry shall carry a timestamp and the identity of the acting user.
NFR-18	Security	Credentials shall have a configurable validity period, with a default of ten years.
NFR-11	Usability	A complete setup by an IT professional shall be possible in less than eight hours, or less than four hours if the IT professional has Docker and Linux experience.
NFR-12	Usability	The instruction of an additional user in the role of study administration shall require less than one hour per person.
NFR-13	Usability	The accompanying documentation shall be complete and comprehensible without external assistance.
NFR-14	Maintainability	The extension of the system by a new credential type shall be possible through a documented template, without modifications to the core code.
NFR-15	Maintainability	Backup and restoration of the database shall be documented as an operational procedure, with a recovery point objective of less than 24 hours.
NFR-16	Transferability	The integration of a new study-administration system through the REST interface shall be possible without modifications to the core code of the credential system.
NFR-17	Transferability	The deployment of the system on alternative infrastructure shall require only Docker and the configuration of a domain.

6. Market Analysis

This chapter analyses comparable initiatives that issue Verifiable Credentials in educational or governmental contexts in Switzerland and build on the Swiss Trust Infrastructure (swiyu) that this thesis also targets. Three initiatives are documented at the time of writing, and they cover complementary positions in the design space: a maturity certificate (educa and Switch), a continuing-education certificate (Berner Fachhochschule with the Institute for Public Sector Transformation), and a regulated governmental credential outside the educational domain (the electronic learner driving permit).

6.1. educa and Switch: Maturity-Certificate Proof of Concept

The DVS innovation project *Digitales Maturitätszeugnis*, led by educa and Switch in collaboration with the cantons of Neuchâtel and Fribourg and the EPFL, produced a proof of concept for issuing the gymnasium maturity certificate as a Verifiable Credential on the swiyu Trust Infrastructure [edu]. The credential schema derives from Article 27 of the Federal Ordinance on the Recognition of Maturity Certificates (MAR) and is registered with the federal interoperability catalogue I14Y [Fed].

In the envisaged use case, a gymnasium issues the digital maturity certificate, the holder presents it to a university at enrolment, and the university verifies it against the trust registry instead of inspecting a paper certificate by hand.

The schema published on I14Y models the maturity certificate as one single credential. The principal entity is the student, with personal data (forename, surname, date of birth) and relations to citizenship information [Fed]. Sub-entities are the issuing gymnasium (with location), the duration of education (start and end date), the individual subjects (with type, level, and grade, in a one-to-many relation), and the maturity-thesis title and grade. This all-in-one structure is a deliberate choice: all subject grades become available together at the end of the academic year and are validated jointly, and the schema is designed to mirror the structure of the legal document under MAR Article 27 [edu26].

For issuance, the project adopted Procivis One Desk, a commercial credential-management platform, letting the team focus on the process and the holder-facing user interface without implementing issuer logic [edu26]. The credential format is SD-JWT, the standard format of the swiyu Trust Infrastructure [edu26].

Table 6.1 contrasts the design choices of the educa project with those of this thesis.

Table 6.1.: Comparison of design choices: educa maturity-certificate proof of concept versus this thesis.

Aspect	educa proof of concept	This thesis
Credential type	Single maturity certificate (secondary II)	University-of-applied-sciences diploma plus module credentials
Issuer technology	Procivis One Desk (commercial SaaS)	Custom Node.js application on top of the swiyu Generic Issuer (self-hosted)
Credential format	SD-JWT VC	SD-JWT VC
Schema basis	MAR Article 27 (federal ordinance)	OST study regulations [OSTb] and concrete certificate fields of myOST [OST26]
Schema registration	I14Y federal catalogue	Application-internal; I14Y registration kept as a future option

6.2. Berner Fachhochschule IPST: CAS Continuing-Education Certificate Proof of Concept

The Institute for Public Sector Transformation (IPST) at the Berner Fachhochschule, together with the implementer ti&m, has issued the certificate of the continuing-education programme *Data Management and Open Government Data* as a Verifiable Credential on the swiyu Trust Infrastructure [tim]. The commercial provider Accredify operated the issuance on behalf of the institute, and a separate verifier component was built to validate the credential [Acc].

The IPST proof of concept shows that educational Verifiable Credentials are feasible on the swiyu platform at a small scale. The credential is a single, discrete continuing-education certificate, a commercial third party operates the issuer under mandate, and the proof of concept does not document any integration with the institution’s own learning-management or course-administration systems.

6.3. Electronic Learner Driving Permit (eLFA)

The electronic learner driving permit (eLFA) is one of the earliest production-oriented credentials on the swiyu Trust Infrastructure [Dig]. The Vereinigung der Strassenverkehrsämter (asa) and the Federal Roads Office (ASTRA) operate it as a pilot for issuing digital credentials to the wider population. The pilot precedes the planned introduction of an electronic driving licence (mDL Suisse) on the same infrastructure, which is itself contingent on the productive launch of the federal e-ID.

The eLFA is not an educational credential, but it shows how a regulated governmental credential operates on swiyu in practice. The parallels concern integration with an existing administrative system, governance of issuer credentials, and key management in an institutional context. Substantive design comparisons remain limited: the eLFA

targets a different domain and scale, and public implementation details are sparsely documented at the time of writing.

6.4. Positioning of This Thesis

The three initiatives share the swiyu Trust Infrastructure as their platform and the SD-JWT Verifiable Credential format as their credential profile. They differ from this thesis along four axes.

Self-hosted issuer. Both educational PoCs adopted commercial credential-management products, Procivis One Desk in the educa project [edu26] and Accredify in the IPST project [Acc]. This thesis builds its own self-hosted issuer application on top of the swiyu Generic Issuer reference container [Swi26b]. The choice trades the convenience of a managed product for control over the issuance logic, the audit-log mechanism, the role concept, and the data layer. Another institution can replicate the system without depending on a commercial credential-management platform.

Granularity of credentials. The educa project packs the entire maturity certificate, with all subject grades, into one credential. For the maturity certificate this is appropriate: all subject grades become available together at the end of the academic year, and MAR Article 27 fixes the structure of the document at the federal level [edu26; Fed]. At a university of applied sciences, module completions accumulate over six semesters, and the institutional value of a per-module credential lies precisely in its independent issuance. This thesis therefore models each module completion as a separate credential and adds a standalone diploma credential that is independent of them, as documented in Chapter 8.

Legal basis of the schema. MAR Article 27 gives the educa project a precise federal definition of what a maturity certificate must contain, which made deriving the schema a direct task [edu26]. No comparable federal definition exists for diplomas of universities of applied sciences. The Federal Act on the Funding and Coordination of the Higher Education Sector (HFKG) regulates admission, accreditation, and funding but does not prescribe the content of diplomas [Swib]. The content is instead governed by the study regulations of the issuing institution [OSTb], the coordination ordinance of the Swiss Higher Education Council on study levels and titles [Swi19], and the Diploma Supplement recommendations of the Kammer Fachhochschulen of swissuniversities [swi14]. The schema of this thesis therefore derives from the concrete certificate fields of myOST [OST26] and the study regulations of the OST Computer Science programme [OSTb].

Integration with the study-administration system. This thesis defines a documented REST interface (FR-23) and supplies a mock service that stands in for the

study-administration system in the demonstration scenario. The deliverable therefore comprises both an issuer application and the integration contract that another institution would need to connect its existing study-administration software.

Together, the four axes position this thesis as the only currently documented self-hosted, integration-oriented implementation of granular academic Verifiable Credentials for a university of applied sciences on the swiyu Trust Infrastructure. The documented REST interface and the deployment documentation give other institutions a concrete adoption path.

7. Prior Work

This thesis follows up on the semester project “Swiss eID for University Diplomas”, carried out in the autumn semester 2025 at OST by the author together with Polina Lisetska and Micha Harzbecker, under the supervision of Daniel Tschudi [DLH25].

7.1. Outcomes of the Predecessor Project

The predecessor project delivered an end-to-end proof-of-concept that integrated the swiyu Generic Issuer and Generic Verifier with custom administration components and demonstrated the full credential lifecycle from issuance through presentation to revocation [DLH25].

Its credential model was a single schema for university diplomas, `university-diploma-vc`: an SD-JWT Verifiable Credential with nine attributes covering personal information, academic information, and an institutional identifier, signed with ES256 and bound to the holder through a JSON Web Key proof [DLH25]. Revocation and suspension followed the IETF Token Status List specification, with one status list shared by all credentials regardless of type or issuance time [DLH25].

The proof-of-concept ran on an OST-internal server under Docker Compose and combined the swiyu Issuer and Verifier reference containers with three custom web applications: issuer administration, verifier administration, and a public verifier-facing application for credential presentation [DLH25]. The swiyu Issuer kept its persistent state in the PostgreSQL containers of the reference platform; the custom applications used embedded SQLite databases. HTTP Basic Authentication with shared credentials guarded administrative access to both web applications, without role-based access control [DLH25].

7.2. Limitations of the Predecessor Project

Five limitations remain. First, the credential model covers only diplomas. With a single schema, other artefacts produced during a course of studies, in particular individual module completions, cannot be issued as separate credentials.

Second, roles are not separated. All administrative actions share one credential, so the same actor can both manage student data and issue credentials. A controlled production environment normally separates these responsibilities.

Third, the data layer offers no integrity protection. The embedded SQLite database of the issuer administration application stores all attributes as untyped strings and cannot detect tampering.

Fourth, credentials are issued one at a time. Semester-end workflows, in which many credentials must be issued in a single coordinated action, are not efficiently supported.

Fifth, key management stays in software. The signing keys for credentials and for the status list are held as software key material, without a Hardware Security Module. This is acceptable for a proof-of-concept but does not satisfy the assurance requirements implied by the wider swiyu trust infrastructure (see Section 8.10).

7.3. Delimitation of the Present Thesis

This thesis focuses on the issuer side of the platform. The verifier from the predecessor project is nevertheless carried forward and extended: it verifies module as well as diploma credentials, can be configured through its trusted-issuer list to accept credentials from issuers other than OST, and thereby spares an HR department the manual checking of submitted qualifications.

Within this focus, the thesis advances the predecessor project in four directions. First, a second Verifiable Credential type for individual module completions extends the credential model. Second, an explicit role concept separates information-technology administration from study administration, and Switch edu-ID single sign-on replaces the shared credentials. Third, the data layer of the issuance side moves from embedded SQLite to PostgreSQL. Fourth, the architecture splits into two systems: a credential system that keeps the issuance responsibility, and a mock service that represents the institutional study administration. The two communicate over a REST interface, documented as an OpenAPI specification, which makes the credential system reusable for institutions that run study-administration software other than the one at OST.

One further extension is pursued where time and external dependencies permit: the issuer signing keys are to move into a Hardware Security Module, contingent on the availability of suitable hardware.

8. Design and Architecture

This chapter presents the architecture and the design decisions of the credential system. The security concept, including the threat model, follows in Chapter 9.

8.1. Architecture Overview

The system runs on a single host within the OST network zone and covers both issuance and verification. On the issuance side, two in-house applications developed in this thesis, the credential system and the mock service, combine with the self-hosted swiyu Generic Issuer [Swi26b]. On the verification side, the self-hosted swiyu Generic Verifier [Swi26d] operates alongside the two in-house verifier web applications carried forward from the predecessor project [DLH25].

In brief, the credential system holds the issuance business logic and the administrative views, the mock service stands in for the institutional study-administration system and runs its rules, and the Generic Issuer signs the credentials and serves the wallet-facing issuance endpoints. Section 8.3 describes each subsystem in detail. The actors that interact with the composition are those of table 5.1.

8.1.1. Deployment Topology

The composition runs as a set of cooperating Docker Compose stacks behind a single `make` entry point on the OST-internal host; figure 8.1 shows all containers and their connections. A Caddy reverse proxy terminates the edge with certificates from Let's Encrypt and routes requests by URL path to the application containers. The mock service, the credential system, and the Generic Issuer each own a PostgreSQL instance; the Generic Verifier keeps only ephemeral state. The supporting containers each serve a single purpose: an internal certificate authority with a renewal helper (section 8.7), a TLS proxy in front of the two Java-based swiyu containers (section 8.7.4), an hourly expiry worker (section 8.3.5), and a write-once storage container that protects the audit log (section 8.8).

8.1.2. Principal Data Flows

Four classes of data flow matter at the architectural level. The *issuance flow* starts with a course finalisation or a diploma-issuance trigger in the mock service and ends with the credential available to the holder in the wallet.

The *authentication flow* authenticates all administrative users against Switch edu-ID through OpenID Connect.

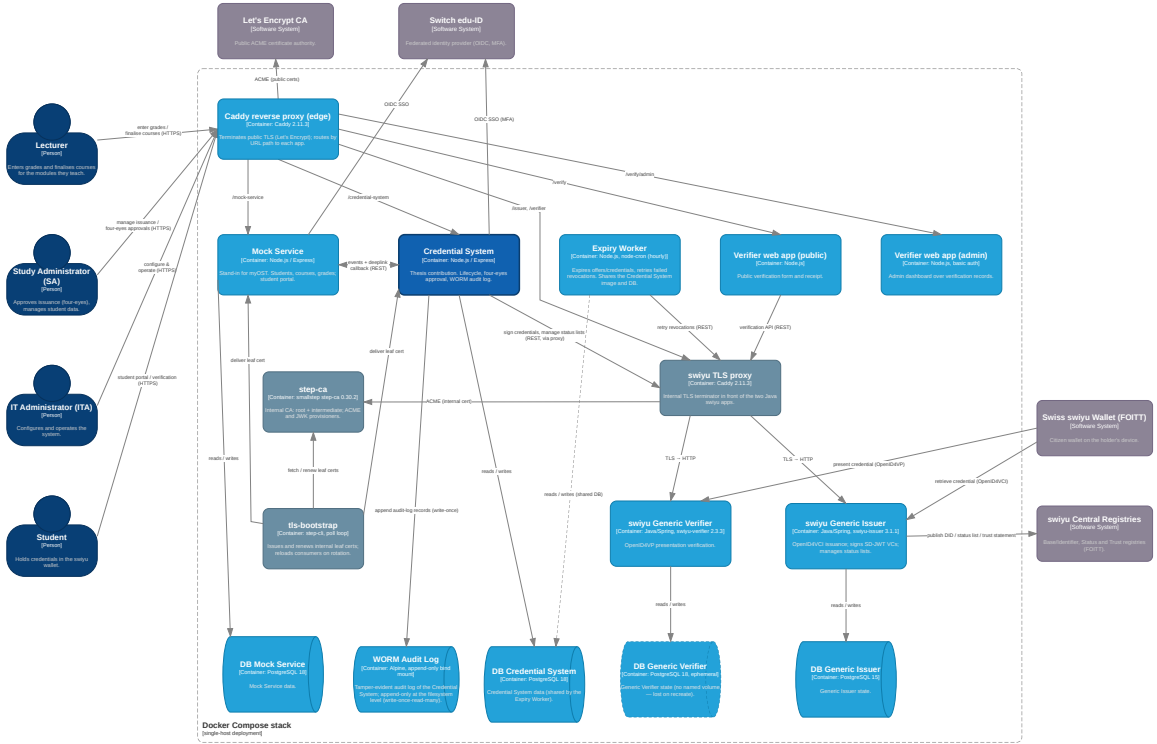


Figure 8.1.: Container diagram (C4 Level 2) of the deployed single-host Docker Compose composition.

The *registry traffic* connects the swiyyu Generic Issuer to the three central swiyyu registries: the DID document reaches the Base Registry through the one-time onboarding of section 8.10.4, the Generic Issuer pushes signed status-list JWTs to the Status Registry, and it retrieves trust statements from the Trust Registry under Trust Protocol 2.0 [Swi26b; Swig]. The credential system is not in the registry call path.

The *audit flow* records every security-relevant operation of the credential system in the audit log; a hash chain links the entries and makes any silent modification detectable.

8.2. Design Decisions and Trade-offs

Each decision below is stated with its rationale and the alternatives that were rejected in its place.

8.2.1. System Decomposition into Three Subsystems

The issuance side is deliberately split into three separately deployable subsystems that communicate exclusively over documented REST interfaces.

A university already operates a study-administration system; the credential system is an add-on to such a system, not a replacement. A university that adopts the credential system from another institution must be able to replace the mock service with its own study-administration software, without modifying the credential system itself.

The decomposition also confines the swiyu Generic Issuer to its role as an unmodified third-party container. Coupling the in-house business logic to a specific version of the Generic Issuer through code changes would prevent independent upgrades of the reference component as the Federal Office of Information Technology develops it further [Swi26b].

Rejected alternatives. Three alternatives were rejected: a direct integration with the staging myOST instance surfaced as an option only towards the end of the project, too late to be realised before submission (section 4.3.3); a monolithic single application would mix study data and credential issuance and would differ across institutions; and reimplementing the issuance endpoints in the credential system would duplicate functionality that the reference container already provides.

The decomposition has two operational consequences: three application services and three PostgreSQL instances to operate, and student master data in two places, the authoritative copy in the mock service and a working copy in the credential system. Both are accepted; the operational overhead is the price of transferability, and the master-data change notification of requirement FR-24 resolves the duplication.

8.2.2. Ownership of the Study-Administration Logic

The mock service owns all study-administration logic in the composition. The credential system never executes this logic.

The credential system takes the events from the mock service at face value: when notified that a student has completed a module, it prepares a credential without re-evaluating the institutional condition.

The trade-off is that the credential system trusts the mock service to enforce the institutional rules correctly. The trust is acceptable because the institution that operates the mock service also controls the issuance authorisation.

Rejected alternatives. Duplicating the study-administration logic in the credential system was rejected; it would tie the credential system to the curriculum and regulations of one institution and prevent reuse by other universities.

8.2.3. Event-Driven Communication with Master-Data Backfill

The mock service and the credential system communicate through pushed events. Each event results from an explicit operator action in the mock service, not from a background process: the lecturer finalising a fully graded course pushes the module-completion event (FR-17), and the study administration triggering the diploma issuance for one or all eligible students pushes the diploma-readiness event (FR-19). The institutional rules guard these actions; the mock service refuses a finalisation while grades are missing and a diploma trigger while the graduation date is not set.

Push delivery is near-real-time and avoids polling load. The event channel is push-only; an operator triggers the recovery of an event that the push channel failed to deliver. A complementary pull path serves one distinct purpose, the on-demand backfill

of master data: when the credential system needs the current values of a student record, it reads them from the mock service.

Rejected alternatives. Two alternatives were rejected: polling as the primary delivery mechanism, whose frequency either delays issuance or wastes resources, and the pure manual issuance carried over from the predecessor proof of concept [DLH25], which offers no integration value at the architectural level.

8.2.4. Unified Batch Abstraction with Immutability after First Approval

Issuance is processed through a unified batch abstraction with batch size at least one. The same code path issues a single credential and a batch of credentials.

After the first approver has confirmed, the batch becomes immutable until it is either fully issued (after the second approval) or rejected. A snapshot hash, captured at the first approval and re-verified before the issuance step, makes any modification of the batch contents detectable.

Rejected alternatives. Two alternatives were rejected: separate endpoints for single and bulk issuance would duplicate the approval, audit, and rollback semantics, and a batch that remains mutable after the first approval would let the first approver's review diverge from the credential set at issuance, defeating the four-eyes principle.

8.2.5. Credential and Offer Validity

A credential has a default validity of ten years from its issuance. A credential offer, which is the deeplink retrieved by the wallet to claim the credential, has a validity of seven days. Neither expiry is reflected in the status list.

Ten years reflect the typical span over which a graduate presents an academic degree in a working career. Seven days reflect the practical window in which a student is expected to retrieve a freshly issued credential into the wallet.

The status list is not modified when a credential or an offer expires through its own `exp` claim. The swiyu interoperability profile requires verifiers to check the `exp` claim [Swif], and the swiyu Generic Verifier does so in practice.

Rejected alternatives. Shorter validities, in particular a yearly renewal suggested during the kick-off phase, were rejected: they would impose recurring administrative overhead on the study administration without adding security beyond the existing revocation mechanism. Mirroring a systemic expiry as a revocation was likewise rejected; it would duplicate the verifier-side `exp` check that the swiyu interoperability profile mandates [Swif] and consume status-list capacity reserved for genuine revocations and suspensions.

8.2.6. Identity Federation through Switch edu-ID

Administrative users authenticate against Switch edu-ID through OpenID Connect. Each subsystem registers as a separate OpenID Connect client in the Switch Resource Registry. Within each subsystem, a local mapping table associates the stable subject identifier returned by Switch edu-ID with the local role of the user.

The stable identifier is the OpenID Connect `sub` claim, which is pairwise per relying party [SWI]. The authorisation request demands multi-factor authentication through a `refeds.org/profile/mfa` value in the authentication-class reference (`acr`) claim; Switch edu-ID enforces it during the authentication.

The credential system maintains no student role and is not exposed to students directly. Students authenticate only against the mock service and receive credential offers through its student portal (FR-13, FR-21). They self-register in the mock service on first login when their edu-ID affiliations contain `student@ost.ch`.

Rejected alternatives. Three alternatives were rejected: the `eduPerson` affiliation attributes are too coarse to separate the IT administration role from the study administration role, since the controlled vocabulary of `eduPersonAffiliation` knows no finer administrative category than `staff` [Res22] and both roles therefore arrive as `staff@ost.ch` in the `swissEduIDLinkedAffiliation` claim of Switch edu-ID [SWI26]; the email address is not guaranteed stable over the lifetime of an edu-ID user, unlike the `sub` claim; and a single OpenID Connect client shared by both subsystems would conflate the administrative role model of the credential system with the student-facing model of the mock service.

8.3. Subsystem Design

8.3.1. Mock Service

The mock service represents the institutional study-administration system in the demonstration scenario. It owns the authoritative copy of student master data, course and module definitions, and grades.

The mock service exposes two interfaces, both realised as a server-side-rendered Node.js application. Through the administration interface, the study-administration role enters and maintains grades, courses, and modules and initiates the events that trigger credential preparation. Through the student portal, students retrieve credential offers as QR codes or deeplinks. The portal supports authentication via Switch edu-ID, including self-registration on first login; in the demonstration scenario, the seeded demo students sign in through a simple username-and-password login instead.

The mock service emits three classes of events to the credential system: module-completion events (FR-17), diploma-readiness events (FR-19), and master-data change notifications (FR-24). The events arrive at a webhook endpoint on the credential system. The mock service in turn exposes the master-data pull endpoint for the on-demand

backfill of section 8.2.3 and a callback endpoint at which the credential system delivers the offer deeplinks for previously prepared credentials.

The mock service deliberately does not reproduce the full functionality of myOST; scheduling, room booking, library administration, and similar concerns are out of scope. It also does not authenticate against the credential system; the two services exchange events over an internal channel that is authenticated independently (Chapter 9).

8.3.2. Credential System

The credential system is a server-side-rendered Node.js monolith. Server-side rendering, rather than a separate frontend and backend, fits the operational scope: a small number of operators per institution uses the administrative interface, which does not benefit from a single-page-application architecture.

The credential system holds the business logic of credential issuance: the approval workflow that enforces the four-eyes principle (NFR-07), the batch management behind the unified batch abstraction of section 8.2.4, the role-based access control for the IT administration and study administration roles (FR-12, FR-13, FR-14), and the audit log of security-relevant operations (NFR-10). It also holds the local working copy of student master data (FR-09) required for credential assignment and lookup.

The credential system exposes two operator-facing interfaces: the administrative interface for the IT administration role and the issuance-and-approval interface for the study administration role.

8.3.3. Self-Hosted swiyu Generic Issuer

The swiyu Generic Issuer is a Java-based reference container provided by the Federal Office of Information Technology; it runs unmodified from the official upstream image [Swi26b], in the version recorded in section 10.1.

The Generic Issuer exposes the OpenID4VCI endpoints [Ope25a] that the official Swiss wallet calls during credential retrieval. It signs the SD-JWT Verifiable Credentials with ECDSA P-256; the key-provider strategy fixes the signing-key location at deployment time.

8.3.4. Verifier Subsystems

The swiyu Generic Verifier reference container [Swi26d] and two in-house web applications carried forward from the predecessor project [DLH25] fulfil the verifier responsibility; figure 8.2 shows their interactions. This thesis restructured the in-house applications to support both credential types under a shared user-interface design.

The swiyu Generic Verifier exposes the OpenID for Verifiable Presentations (OpenID4VP) endpoints [Ope25b] that a wallet calls during credential presentation: it validates the cryptographic signature, retrieves the relevant status list from the Status Registry, and reports the outcome to the calling application. Like the Generic Issuer, it runs unmodified in configuration-only mode.

The public verifier web application serves external verifying parties, for example an employer or a receiving university. The administrative verifier web application serves the institution itself, for example to inspect the data items of a presented credential during a support case.

The verifier subsystems do not communicate with the credential system.

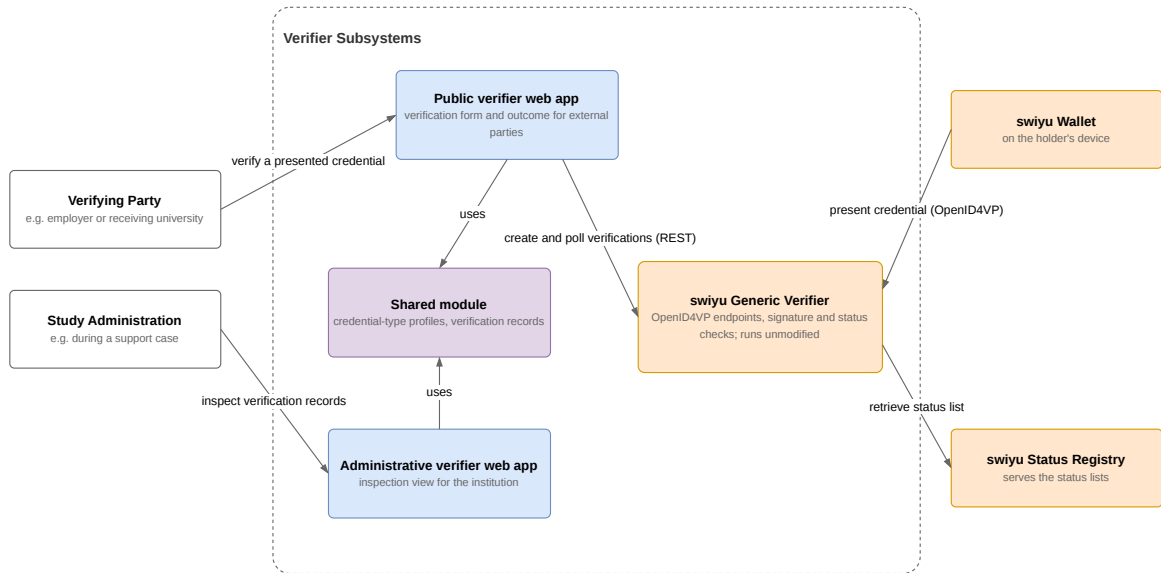


Figure 8.2.: Verifier subsystems and their interactions

8.3.5. Expiry Worker

The expiry worker is a small Node.js application that runs on an hourly cron schedule; figure 8.3 shows one pass. Each pass first executes the incremental audit-log verification of section 8.8 and aborts on any hash mismatch. It then marks every issued credential whose **exp** claim has elapsed as expired; the step is purely local bookkeeping, because verifiers enforce the **exp** claim themselves at presentation. It finally retries the revocations and suspensions whose status-list write had previously failed, so that a transient issuer outage does not leave a credential unrevoked; a credential whose retries keep failing is flagged for manual follow-up on the IT administration dashboard.

The worker does not handle offers that the holder never retrieves within the seven-day validity window of section 8.2.5; the swiyu Generic Issuer expires uncollected offers itself (section 8.5.5).

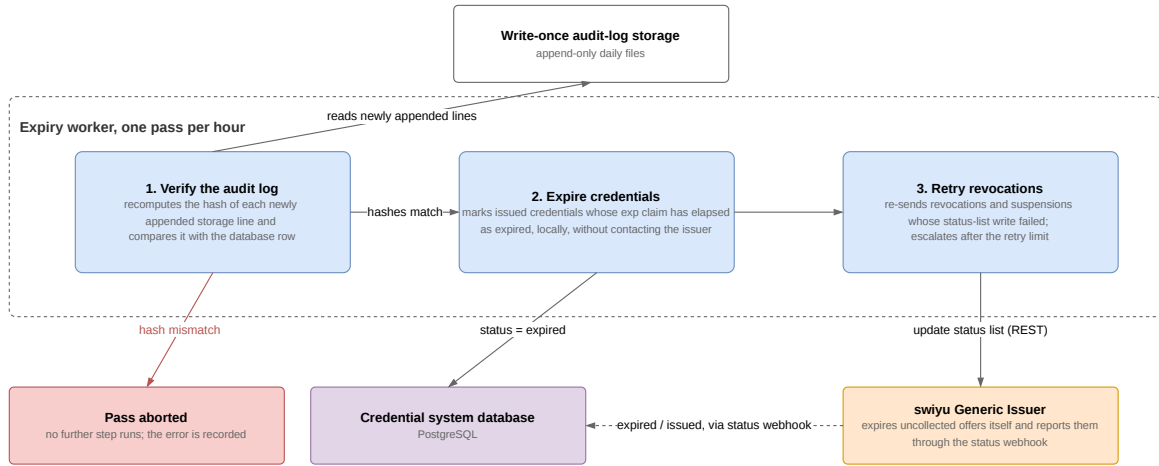


Figure 8.3.: Hourly pass of the expiry worker

8.4. Credential Schemas

The credential system issues two credential types, the module credential and the diploma credential. Both follow the SD-JWT Verifiable Credential profile [Int26], are signed with ECDSA P-256, and bind the credential to the holder through a JWK published under the `cnf` claim, in line with the swiyu interoperability requirements. The schema fields were confirmed in a correspondence with the OST study administration in February 2026 [OST26].

8.4.1. Common Format and Holder Binding

Both credentials carry the type-identifier scheme `vct: university-module-vc` for the module credential and `university-diploma-vc` for the diploma credential. The type identifiers are bare type strings rather than resolvable URIs; the bare form keeps the institutional configuration self-contained. The credential format is `vc+sd-jwt`, the signing algorithm is ES256, the cryptographic binding method is `jwk`, and the proof type is a JWT signed with ES256 [Swi26b].

All claim values are encoded as strings, including claims whose semantic type is numeric; the verifier converts them back at the application layer as required. The restriction responds to a tested limitation of the swiyu wallet, which fails to render claims whose value type is declared as a number or an object. The limitation was observed in March 2026 on the wallet versions available at the time [Swie].

8.4.2. Module Credential

The module credential is issued for each module completion of a student. One credential per module matches how holders typically present academic credentials: a student applying for a position or for credit transfer at another institution often needs a single completed module rather than the full academic record.

Two platform limitations leave no real alternative to this granularity. The swiyu wallet does not render object-typed claims (section 8.4.1), so several module completions cannot be nested inside one credential, and the platform offers no holder-side selective disclosure (section 8.4.4), so a holder can only reveal a single module if it is a credential of its own.

The module credential carries the twelve claims summarised in table 8.1: a base set with the identity of the holder and the institutional and curricular context of the module, and an additional set with the per-instance information of the completion and the institutional links.

Table 8.1.: Claims of the module credential.

Group	Claim	Source of value	Semantic type
Base	<code>firstName</code>	Student master data	string
Base	<code>lastName</code>	Student master data	string
Base	<code>moduleName</code>	OST module catalogue [OSTb]	string
Base	<code>semester</code>	Mock service course finalisation	string
Base	<code>ects</code>	OST module catalogue [OSTb]	number
Base	<code>degreeName</code>	Study programme record	string
Base	<code>fieldOfStudy</code>	Study programme record	string
Base	<code>university</code>	Mock service course finalisation (fixed value)	string
Additional	<code>grade</code>	Mock service grade record	number
Additional	<code>moduleId</code>	OST module catalogue [OSTb]	string
Additional	<code>moduleUrl</code>	OST module catalogue [OSTb]	URL
Additional	<code>studentId</code>	Student master data (matriculation number)	string

A module credential is issued when the mock service emits a course-finalisation event (FR-17) for a student who has passed the module; an explicit *passed* claim is unnecessary because the credential is only issued for passed modules. The credential is assigned to a status list named `module-{semester}`, with the semester taken from the corresponding claim; the partitioning scheme follows in Section 8.9.

The module credential omits items of the paper semester certificate that verification does not require: the internal personal identification number of the holder (PI_{id} in the myOST sense), the name of the lecturer, and the partial grades (*Teilnoten*). Birth date and citizenship live exclusively on the diploma credential, where the institutional contract motivates their presence.

8.4.3. Diploma Credential

The diploma credential is issued for the awarding of a degree. A standalone diploma credential keeps the diploma verifiable when the holder has not retained the module credentials in the wallet.

The diploma credential carries the nine claims summarised in table 8.2. All claims are mandatory at the issuer level.

Table 8.2.: Claims of the diploma credential.

Claim	Source of value	Semantic type
firstName	Student master data	string
lastName	Student master data	string
birthDate	Student master data	date
citizenship	Student master data	string
degreeName	Study programme record	string
fieldOfStudy	Study programme record	string
university	Study programme record	string
graduationDate	Student master data	date
studentId	Student master data (matriculation number)	string

A diploma credential is issued when the mock service emits a diploma-readiness event (FR-19) for a student. The mock service evaluates the institutional graduation conditions defined in the study regulations [OSTb]. As a defence-in-depth check, the credential system verifies that the `graduationDate` field of the student master data is set before preparing any diploma credential. The credential is assigned to a status list named after the pattern `diploma-{semester}`, with the semester derived from the graduation date.

8.4.4. Selective Disclosure and Verifier-Side Presentation

The SD-JWT Verifiable Credential profile [Int26] supports a per-claim selective-disclosure declaration at the issuer level, by which the issuer may mark each claim as always disclosed, optionally disclosable, or never disclosable. The swiyu Generic Issuer does not currently expose this declaration: all custom claims are automatically converted to selective-disclosure entries in the issued credential, regardless of the issuer-side meta-data [Swi26c]. The swiyu wallet, in turn, offers no holder-side selection interface for choosing which claims to reveal at presentation time.

These two limitations leave the control over disclosed claims entirely with the verifier. The verifier expresses the requested claim set through a Presentation Definition that the wallet evaluates against the credential held by the user. The Presentation Definition declares each requested claim as required or as optional via the `optional` marker; the wallet versions currently in use do not surface the marker to the holder and release all requested claims, so the marker does not yet translate into a holder-side choice (section 11.5).

The public and the administrative verifier web application each carry a single Presentation Definition per credential type. Table 8.3 summarises the claim sets of these standard Presentation Definitions. The omission of the `citizenship` claim is a data-minimisation choice consistent with the principle of necessity in Swiss data-protection law [Swia].

The deployed release of the swiyu Generic Verifier (section 10.1) documents the Digital Credentials Query Language of OpenID4VP 1.0 [Ope25b] but does not yet

Table 8.3.: Claim sets of the standard Presentation Definitions.

Credential	Required	Optional	Not requested
Module	firstName, lastName, moduleName, semester, grade, ects, university	moduleId, studentId	degreeName, fieldOfStudy, moduleUrl
Diploma	firstName, lastName, degreeName, fieldOfStudy, university, graduationDate	birthDate, studentId	citizenship

implement it: it accepts DCQL queries syntactically and rejects the request at creation time with an HTTP 400 error. The verifier web applications therefore fall back to the older DIF Presentation Exchange format, which remains supported and functional. The migration to DCQL is future work, contingent on a swiyu Generic Verifier release that activates the implementation.

8.4.5. Independence of Module and Diploma Credentials

The two credential types are independent in their issuance, in their status, and in their presentation.

The two types share several identity claims (`firstName`, `lastName`, `degreeName`, `fieldOfStudy`, `university`) by design, so that each credential is presentable without disclosing the other. Their temporal models differ: a module credential carries a `semester` claim for the semester of the module completion, a diploma credential a `graduationDate` claim for the date on which the degree was awarded.

Rejected alternative. A monolithic transcript credential, carrying all grades and the institutional context of a student in one Verifiable Credential, would prevent the per-module presentation that dominates the use cases here; the educa proof of concept adopted it for the maturity certificate [edu26], where all subject grades become available together and MAR Article 27 fixes the document structure at the federal level, so the trade-off is reversed there.

8.5. Lifecycle Flows

Four operator-driven flows govern the credential lifecycle: the issuance of new credentials, the re-issuance after a correction of the underlying master data, the re-issuance after a wallet loss reported by the holder, and the deliberate revocation or suspension of a credential. The expiry worker adds one housekeeping flow, the systemic-expiry processing. Figure 8.4 summarises the resulting state machine of a single credential record; the following subsections describe the flows that drive its transitions. For

legibility, the figure omits the transient technical failure state, in which a failed swiyu call parks a credential until an operator retries, and the revocation edges from **offered** and **review_needed**, which behave like the one from **issued**.

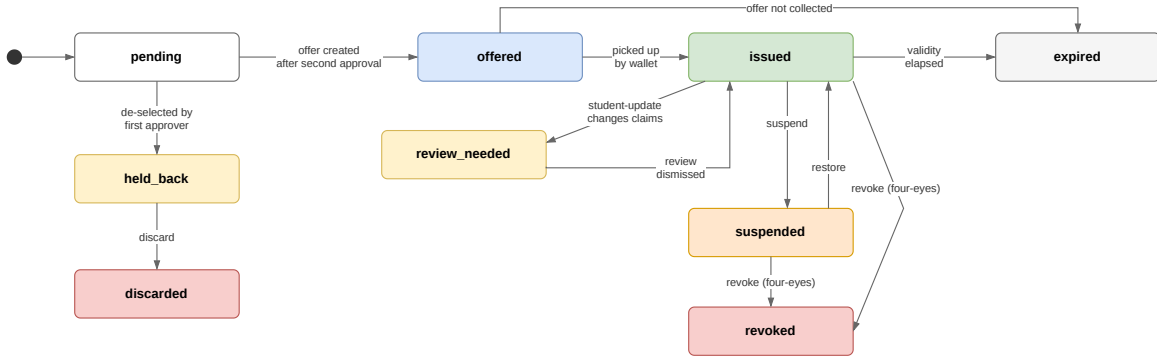


Figure 8.4.: State machine of a credential record.

8.5.1. Issuance Flow

The issuance flow starts with an institutional event from the mock service and ends with the new credential available to the holder in the wallet. The event is either a course finalisation for a module completion (FR-17) or a diploma readiness (FR-19); both arrive at a webhook endpoint of the credential system.

On receipt of an event, the credential system creates an issuance batch in the state **pending_approval**; the batch carries an internal identifier and the credentials that the institutional event triggers. In this chapter, *batch* refers exclusively to this credential-system-internal grouping, in contrast to the OpenID4VCI batch issuance mechanism of the Generic Issuer [Swi26b], which issues several unlinkable instances of a single credential to one holder and which the issuance flow does not use.

At batch creation, the credential system computes a snapshot hash over the contents of the batch. While the batch remains in **pending_approval**, the first approver, a study administrator, may de-select individual credentials; a de-selected credential is parked as **held_back** and is later issued individually or discarded, with each step recorded in the audit log together with the operator identity. On the first approval the batch transitions to **pending_second_approval**, the snapshot hash is recomputed, and the batch becomes immutable in the sense of section 8.2.4.

A second study administrator, distinct from the first approver, confirms the batch. On successful second approval, the credential system calls the swiyu Generic Issuer to create one credential offer per credential and receives the offer deeplinks, each carrying the seven-day validity of section 8.2.5.

The credential system posts the offer deeplinks to the dedicated callback endpoint of the mock service, which renders them as QR codes and inline links in the student portal; the holder retrieves the credentials into the wallet through the standard pull-based OpenID4VCI flow [Ope25a].

Figure C.1 in appendix C shows the complete issuance collaboration end-to-end, from the institutional trigger through the four-eyes approval loop and the per-credential offer

creation to the holder pick-up. The diagram depicts the diploma credential, whose readiness event triggers a bulk issuance of one credential to each graduating student; the module-credential issuance follows the same backbone with a course-finalisation trigger.

8.5.2. Re-Issuance after Master-Data Correction

The trigger of this flow is a `student-update` event, emitted by the mock service when one of seven whitelisted master-data fields of a student record changes: `firstName`, `lastName`, `birthDate`, `citizenship`, `degreeName`, `fieldOfStudy`, or `graduationDate`.

On receipt of the event, the credential system flips the affected student's credentials in state `issued` or `offered` to `review_needed` (figure 8.4) and records the previous and new values of the changed fields in the audit log. The credential remains visible in the wallet and verifies unchanged; from `review_needed`, the study administration can trigger a manual suspension or revocation.

The automated continuation is specified but not yet implemented: a first study administrator initiates the revoke-and-re-issue workflow on the batch of credentials that share the same correction, and a second, distinct study administrator inspects the diff of the changed fields and approves or rejects it. On approval, the credential system revokes the affected credentials on the swiyu status list and prepares a new batch whose claims are read fresh from the corrected master data, a deliberate deviation from the snapshot semantics of section 8.5.1 that reflects the intent of the correction; the new batch passes through the regular issuance flow.

The revocation strictly precedes the new issuance: the two steps are not atomic, since the swiyu status list is append-only and admits no rollback [Int], and the reverse order would leave two valid credentials in circulation during an outage. The accepted consequence is a short operational window in which the holder has no valid credential. Because OpenID4VCI [Ope25a] specifies no issuer-to-wallet push notification, informing the holder of the new credential out of band, for example by email, also remains future work for a productive deployment.

8.5.3. Re-Issuance after Wallet Loss

The wallet-loss flow covers holders who report that previously retrieved credentials are no longer accessible, typically after losing or replacing the device that held the wallet. The flow is holder-initiated through the student portal of the mock service; a study-administration-initiated variant for alumni who no longer hold an active institutional identity remains future work.

The holder authenticates through Switch edu-ID, selects the lost credential, and optionally adds a free-text justification. The mock service emits a dedicated `reissue-requested` event; the credential system creates a row in the dedicated table `reissue_requests` atomically with the corresponding audit-log entry.

The `reissue_requests` entity carries the state machine of the wallet-loss flow, shown in figure 8.5. It orchestrates two inner flows, the regular four-eyes revocation of the

lost credential followed by the regular four-eyes issuance of the new one, and enforces only the ordering between them; the row links the generated revocation request, the new issuance batch, and the new credential record, so the provenance of each artefact remains traceable.

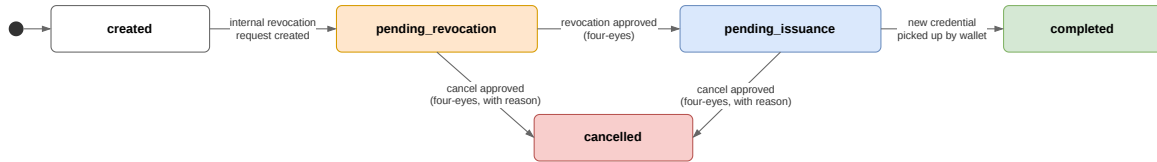


Figure 8.5.: State machine of a wallet-loss re-issuance request.

The re-issued credential carries claims identical to the lost one; refreshing master data is the job of the flow in section 8.5.2. The lost credential is revoked irreversibly on the status list before the new one is issued, so at most one valid credential per student and credential type exists at any time. Cancelling a pending request requires a documented `cancel_reason` and is itself subject to the four-eyes ceremony, so a single compromised study administrator cannot suppress a re-issuance request that the holder has legitimately submitted. A cancellation after the revocation has taken effect does not restore the lost credential, since the status list admits no rollback; it only stops the issuance of the replacement.

8.5.4. Revocation and Suspension

Revocation and suspension are the two deliberate, operator-initiated status-list operations that the credential system supports. Revocation is irreversible and invalidates a credential permanently, for example after the underlying study record has been annulled. Suspension is reversible and invalidates a credential temporarily while an institutional decision is pending, for example during a plagiarism investigation; a subsequent restore lifts it (figure 8.4).

A revocation is subject to the four-eyes ceremony in line with section 8.2.4; on the second approval, the credential system writes the revocation bit on the swiyu status list. Suspension and restore, being reversible, do not require the ceremony: the approver who initiates the suspension may also lift it. The audit log records every status operation together with the operator identities.

Failed writes to the swiyu status list, whether from a revocation, a suspension, or a restore, are tracked in dedicated retry columns of the credential record and retried by the hourly worker of section 8.3.5. The retry loop covers only these deliberate operations, not the systemic-expiry processing of section 8.5.5. A revoked credential remains visible in the holder's wallet, marked revoked, until the holder removes it manually.

8.5.5. Systemic Expiry versus Status-List Revocation

Systemic expiry covers the two cases in which a credential or an offer becomes invalid through the passage of time rather than through an operator action: offers that the

holder has not retrieved within the seven-day validity window of the offer deeplink, and issued credentials whose `exp` claim has passed. The hourly expiry worker of section 8.3.5 processes the credential case; the swiyu Generic Issuer expires uncollected offers itself and reports the transition through the status webhook of section 8.5.6. In neither case does the credential system write to the swiyu status list, for the reasons given in section 8.2.5; the local state flips to `expired` in a single transaction per credential and is recorded in the audit log.

8.5.6. Webhook Delivery Semantics

The lifecycle events of the previous subsections travel as HTTPS webhooks between the mock service and the credential system; chapter 10 documents the concrete delivery contract.

Inbound events from the mock service to the credential system cover the issuance triggers of section 8.5.1 (`course-completion`, FR-17; `diploma-ready`, FR-19), the master-data changes of section 8.5.2 (`student-update`, FR-24), the wallet-loss requests of section 8.5.3 (`reissue-requested`), and the revocation requests that the study administration raises in the mock-service portal (`revocation-request`, section 8.5.4). A third inbound channel, the status webhook, carries the lifecycle notifications of the swiyu Generic Issuer to the credential system, among them the expiry of an uncollected offer (section 8.5.5).

Outbound events from the credential system to the mock service cover the offer-deeplink callbacks of section 8.5.1 and the request-status callbacks of the wallet-loss flow of section 8.5.3. Callbacks are delivered on a single attempt and not retried automatically; a failed delivery is marked as failed, and an operator can trigger a resend.

The wallet pickup of the credential, in contrast, is not a webhook but a pull-based flow that the wallet initiates against the swiyu Generic Issuer, in line with the OpenID4VCI specification [Ope25a].

8.6. Four-Eyes Approval and Identity

The four-eyes mechanism currently applies to three workflows: batch approvals for credential issuance, revocation approvals, and the cancellation of a wallet-loss reissue request.

8.6.1. Role Concept and Authorisation Middleware

A database check constraint and an application middleware named `requireRole` enforce the two operator roles of the credential system, `ita` (IT administration) and `sa` (study administration).

The mock service maintains its own four-role model (`student`, `dozent`, `sa`, `admin`) that reflects the broader scope of an institutional study-administration system. The credential system role table and the mock service role table are separate; the same natural person may hold different roles in the two subsystems. The two role models

share the `sa` identifier intentionally, since a study administrator is the operator who acts in both subsystems at different points of the credential lifecycle.

Informally, the permission matrix reads as follows: the IT administration role owns user provisioning, including the invitation of new study administrators and the registration of their authenticators, and the operational endpoints of the system, but may not approve, revoke, suspend, or restore credentials. The study administration role owns all credential-state operations: the approval of issuance batches, the approval of revocations, and the single-approver suspension and restore.

8.6.2. Identity Federation and Local User Binding

On the first authenticated callback of a previously invited operator, the credential system creates a row in its `users` table with the stable subject identifier `edu_id_sub` returned by Switch edu-ID, the verified email address, and the role that the inviting IT administrator selected at invitation time. The subject identifier is the stable handle for all subsequent authentication and approval operations. Consuming the invitation requires the email address to match the open invitation token, the verified-email flag, and the multi-factor authentication-class reference; subsequent logins resolve the user record by the subject identifier alone.

The multi-factor enforcement uses the authentication-class-reference claim of OpenID Connect [Ope], with the value defined by the multi-factor authentication profile of the Research and Education Federations [Res]. The credential system requests the value as essential in the authorisation request and re-validates it on the callback.

Sessions live in a server-side store backed by the application's own PostgreSQL database. The session cookie is secure and HTTP-only, with a same-site policy of `lax` and a maximum age of twenty-four hours; the session identifier is regenerated on successful login to defend against session fixation. A single-logout mechanism with back-channel logout against the federation remains an open item for the productive deployment.

8.6.3. WebAuthn Registration and Authenticator Policy

The cryptographic four-eyes binding (section 8.6.4) requires each study administrator to hold an authenticator that signs data on demand. This thesis adopts the Web Authentication standard [Worb] together with the Client-to-Authenticator Protocol [FID] for this purpose.

Provisioning takes three steps. The IT administrator creates an invitation with an email address and a target role, which produces a row in a dedicated invitations table together with a single-use token, a cryptographically random value whose hash is persisted in the database. The IT administrator hands the hardware authenticator to the future study administrator, and the invitation email goes to the operator's institutional email address. The study administrator follows the invitation link, authenticates against Switch edu-ID with multi-factor authentication, and completes the WebAuthn registration ceremony; on success, the public key of the authenticator is stored in the

credential system database together with the metadata the authenticator returns.

The registration ceremony enforces the institutional policy: the authenticator-attachment parameter requires a cross-platform authenticator, the resident-key policy is discouraged, and the user-verification policy is preferred at registration time. The exclusion list passed in the registration options is populated system-wide from the existing public keys, so the same physical authenticator cannot be registered for two different operators. Assertions whose device-type metadata indicates a syncable, multi-device credential are rejected during verification, which binds the registered key to a single physical device rather than a synchronised credential set.

Recovery from a lost authenticator follows the same path as a fresh registration: the IT administrator deregisters the lost public key, hands over a new authenticator, and triggers a new registration ceremony. The new public key references its deregistered predecessor in the database, which preserves the audit trail across the recovery. Figure 8.6 summarises the provisioning and recovery path.

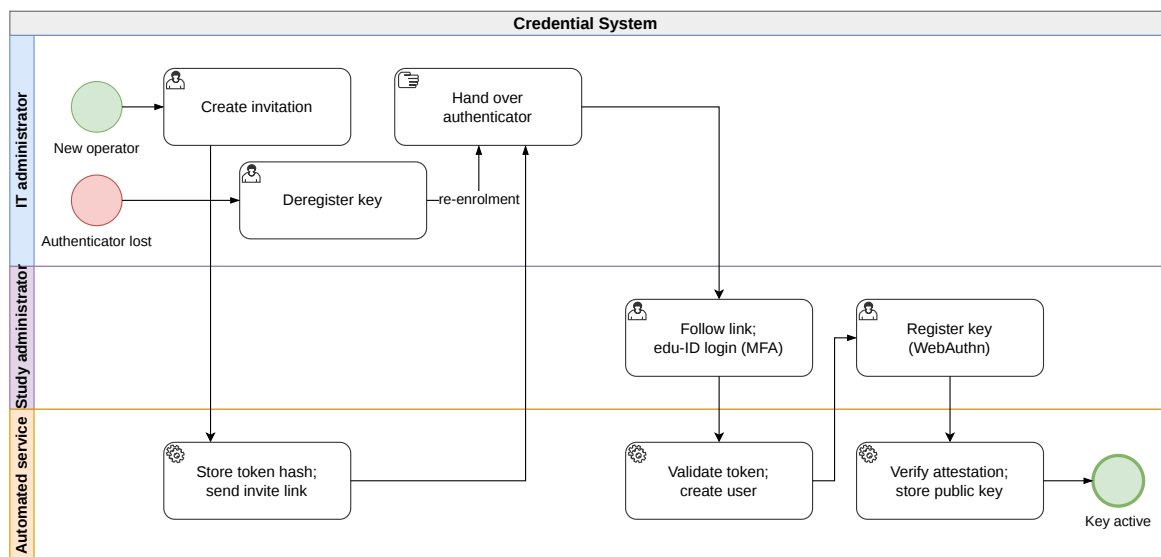


Figure 8.6.: Activation and recovery of the YubiKey for the study administrator.

8.6.4. Cryptographic Four-Eyes Binding

The four-eyes invariant requires that two distinct study administrators sign two distinct cryptographic statements about the same approval data. A Display-Binding hash that both signatures cover materialises this invariant: the approval data (for an issuance the batch, module, and semester identifiers plus the snapshot hash over the student and grade list; for the revocation and cancellation workflows the affected identifiers and the documented reason) is serialised through the JSON Canonicalization Scheme [Int20] and hashed with SHA-256.

Figure 8.7 shows the resulting ceremony; the fingerprint in the figure is this hash. The WebAuthn challenge is the raw hash itself, so the authenticator signs exactly what the approver saw (*what you see is what you sign*). Before accepting an approval, the server re-computes the hash from the current entity state and rejects on mismatch,

verifies the assertion against the stored public key of the operator, and increments the signature counter against replay.

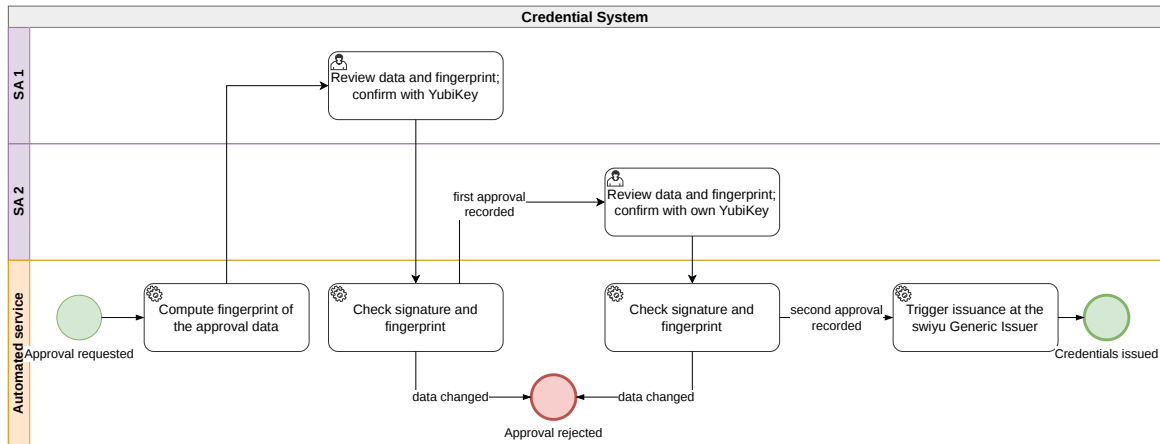


Figure 8.7.: Four-eyes approval ceremony over the fingerprint of the approval data.

Three layers keep the two approving identities distinct: the application rejects a second approval by the same operator, whose identity comes from the server-side session and never from the request body; a database unique constraint on the pair of workflow and operator closes the race window; and the system-wide WebAuthn exclusion list prevents one physical authenticator from being registered for two operators.

After the second approval, the credential system triggers the issuance at the swiyu Generic Issuer. The request carries a JSON Web Signature [Int15b] under the ECDSA P-256 key of the credential system, which the issuer verifies against the published JSON Web Key Set [Int15a]; this replaced an earlier shared-secret authentication that a compromise of the credential system process would have exposed.

The mechanism defends against an external attacker, a corrupt operator at the level of an SSH or container session, and a single stolen authenticator. It does not defend against the collusion of two study administrators, treated as an organisational control, nor against a root-level compromise of the credential system process itself; the documented future defence is a separate approval-verification service with a hardware-security-module-backed signing key and an external source for the operator public keys.

Rejected alternatives. Three alternatives were rejected, each forgeable by an actor with sufficient access: a user-interface confirmation backed only by a database row (database write access), a session-cookie-based approval (session theft), and an audit-log-only attestation (root-level tampering).

8.6.5. Bootstrapping and Open Limitations

Bootstrapping the credential system from a fresh deployment is the only operation that bypasses the regular invitation flow. The first IT administrator is created on the first authenticated callback whose verified email matches an environment variable configured

at deployment time; the match runs inside a transaction that re-checks that no IT administrator exists yet, which prevents a race between two simultaneous first-callback attempts.

After the bootstrap, the IT administrator invites the first study administrators through the regular invitation flow of section 8.6.3; an existing IT administrator provisions subsequent IT administrators through the same flow.

The mechanism accepts several open limitations as future work for the productive deployment. Recovery from a lost authenticator involves no four-eyes ceremony of its own, so a compromised IT administrator could provision a phantom authenticator for an existing operator; the audit log and the per-operator overview of registered authenticators mitigate this residual risk rather than a second technical control. Detecting a compromised operator authenticator within the 24-hour session depends on the user-verification configuration of the hardware authenticator; the present implementation does not require user verification on every assertion. The institutional escalation path for a compromised IT administrator key, including the role of the external master public-key source of the soll-architecture, remains future work alongside the separate approval-verification service.

8.7. Internal Public Key Infrastructure

An internal Public Key Infrastructure (PKI) issues short-lived X.509 leaf certificates to every service that exposes an HTTPS endpoint inside the composition. The PKI is rooted at a containerised certificate authority that runs alongside the application containers on the same OST-internal host introduced in section 8.1.1.

8.7.1. Motivation

Until the introduction of the internal PKI in late May 2026, every networked hop within the composition ran plain HTTP, in violation of NFR-08 and open to any process with access to the Docker bridge. The internal PKI raises the assurance of inter-service communication and draws a clean trust boundary between the containers that share the Docker bridge of the host.

8.7.2. Certificate-Authority Hierarchy and Provisioners

The certificate authority is the `smallstep/step-ca` reference container [Sma], deployed as a standalone authority with remote management and the public ACME endpoint both disabled at initialisation. `step-ca` was chosen primarily for its native ACME endpoint, which Caddy consumes directly.

The trust topology is the conventional three-layer hierarchy of a root certificate authority, a single intermediate certificate authority, and per-service leaf certificates. The standard initialisation command of `step-ca` generates the root and the intermediate on the first container start and persists them in the data volume of the authority.

The repository configuration does not explicitly pin validity periods or signature algorithms, so the `step-ca` defaults apply: the root and the intermediate certificate are valid for approximately ten years, leaf certificates for twenty-four hours, and all certificates use ECDSA over the NIST P-256 curve.

An idempotent initialisation script configures three provisioners on the certificate authority. The `admin` provisioner, a JSON Web Key created during the standard initialisation, is reserved for manual operator interventions; no component of the composition consumes it. The `bootstrap` provisioner, also a JSON Web Key, lets the bootstrap container issue the initial leaf certificates of the in-house services through a password-file-based command-line interaction. The `acme` provisioner implements the Automatic Certificate Management Environment [Int19]; the TLS proxy of the swiyu reference containers consumes it for self-service issuance and renewal. Figure 8.8 shows the hierarchy and the two issuance paths.

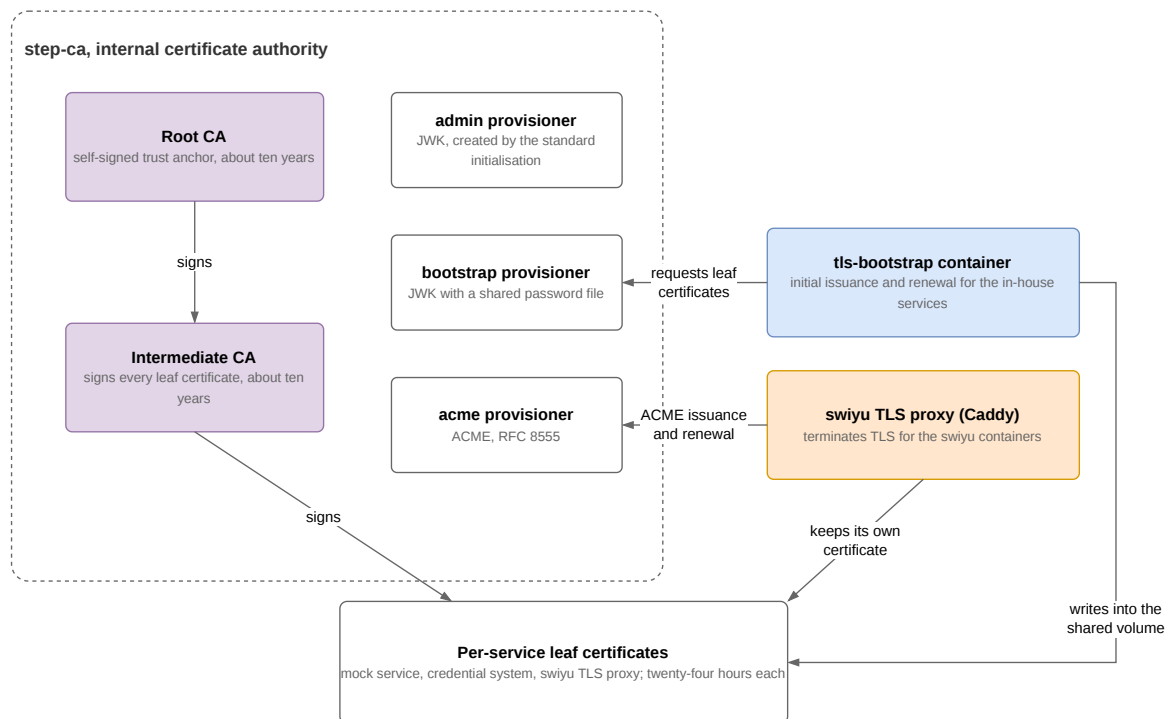


Figure 8.8.: Trust hierarchy and provisioners of the internal certificate authority.

8.7.3. Leaf-Certificate Lifecycle

The two in-house Node.js services, the credential system and the mock service, obtain their leaf certificates through a dedicated bootstrap container that handles the initial issuance, the periodic renewal, and the in-place reload of the consumer services after a renewal; section 10.3.4 documents the renewal mechanics. The two swiyu reference containers do not use it; the TLS proxy that fronts them obtains its certificate through the ACME provisioner of the authority directly and renews it at two thirds of the leaf lifetime, in line with the standard renewal policy of Caddy [Cadb].

The initial issuance proceeds in three steps: the bootstrap container waits for the

health endpoint of the certificate authority to become reachable, bootstraps the trust on the authority through a pre-pinned fingerprint of the root certificate, and issues a leaf certificate for every service listed in the service catalogue through the `bootstrap` provisioner. The root certificate is then copied into a shared volume from which all consumer containers read it. A renewal replaces a leaf certificate well before its expiry and reloads the consumer service without interrupting in-flight connections or restarting the container.

8.7.4. Trust Distribution and the *swiyu* TLS Proxy

No container trusts the certificates of the internal authority by default, so trust distribution complements the certificate-issuance path: every container that initiates an HTTPS connection to another in-house service mounts the shared volume with the root certificate of the authority and configures its runtime to trust it, the Node.js services through the `NODE_EXTRA_CA_CERTS` environment variable, the edge Caddy through the `tls_trusted_ca_certs` directive of its configuration.

The *swiyu* reference containers present a different challenge: the two Java applications, the Generic Issuer and the Generic Verifier, cannot terminate TLS without modifications to the upstream code, which the design decision of section 8.2.1 precludes. A dedicated *swiyu TLS proxy*, a separate Caddy instance, therefore terminates TLS on its public port and forwards the request as plain HTTP over the Docker bridge to the corresponding Java container. The proxy obtains its own leaf certificate through the ACME provisioner of the internal authority and requests Subject Alternative Names for both the Docker DNS name and the loopback name, so that callers reaching the proxy through either route obtain a valid certificate.

The edge Caddy that fronts the entire composition keeps Let's Encrypt for the public certificate of the institutional hostname, obtained through the standard automatic HTTPS mechanism of Caddy [Cada], and uses the internal certificate authority only on its upstream legs, where it verifies the internally signed certificate that the corresponding service container presents.

Rejected alternatives. Five alternatives were rejected. Let's Encrypt for the internal hops fails structurally, because the public ACME servers can neither resolve nor reach the internal Docker DNS names; it is retained only for the public edge certificate. Self-signed certificates per service would require a trust-store update on every consumer at every rotation. An initially built bootstrap container around the OpenSSL command line lacked an audit-grade issuance log, an ACME endpoint, and a documented key-recovery procedure, and was replaced by the step-ca design. HashiCorp Vault was set aside on account of the Shamir unsealing on every restart of the composition and the licensing uncertainty of its Business Source License 1.1; it is documented as the recommended migration target for a productive rollout, where these concerns are weighed differently. A single TLS termination at the edge, with plain HTTP on every internal hop, would conflate north-south and east-west traffic and make the edge proxy a single point of failure for the internal coordination.

8.8. Audit Log Integrity

8.8.1. Motivation and Scope

An audit log that lives only in the relational database does not stop an attacker with write access to that database: such an attacker can edit an entry, recompute the hash chain, and so restore internal consistency, or dispose of the persistent data volume altogether. The design therefore separates two concerns: a persistent named volume that survives the regular Compose lifecycle mitigates data loss, and an external Write-Once-Read-Many storage with a separate verification path that reconciles the relational state against it mitigates manipulation of the history. The adversary model and the accepted residual risks are documented in the consolidated threat model (section 9.2) and residual-risk register (section 9.4) of chapter 9.

The mechanism provides tamper evidence for six tables of the credential system: the hash chain itself, the application-level audit log, the four-eyes approvals of issuance batches, the four-eyes approvals of revocations, the wallet-loss re-issuance requests, and the four-eyes approvals of their cancellation. The mutable working tables for credentials, batches, and student master data are intentionally not covered: they describe the current state, not the history. The pseudonym-mapping table that resolves operator identifiers for the audit log is also not covered; it is a lookup helper, not an event.

8.8.2. Write-Once-Read-Many Storage

Figure 8.9 shows the design: the Write-Once-Read-Many storage is a separate Docker Compose stack, named `worm-storage`. The physical immutability of the storage is therefore decoupled from the application; an object store with object-lock semantics could substitute the implementation without modifications to the application.

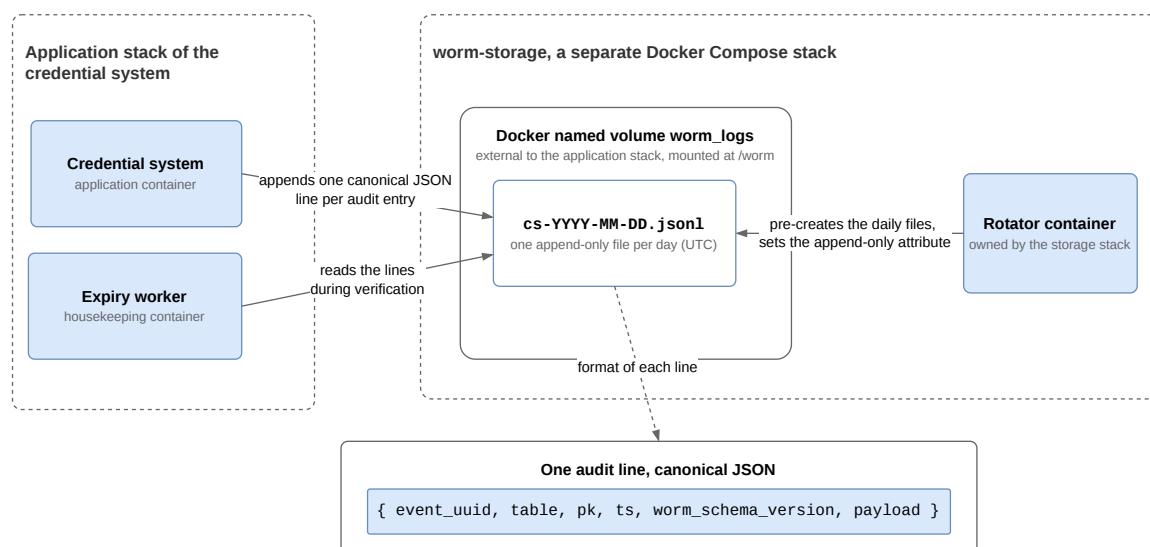


Figure 8.9.: Write-Once-Read-Many storage for the audit log.

The rotator container holds the Linux immutable-extension capability required for

the append-only filesystem attribute. Its 60-second rotation loop pre-creates the files for the current and the following day, so that the transition over midnight produces no window in which a write would land on an unprotected file.

The wire format of the daily files is JSON Lines; the `event_uuid` field of each envelope carries the idempotency-key UUID that links the line to its database row. Writes are synchronous and bounded by the atomic-write limit of the platform (section 10.3.1); the digest substitute for oversized payloads keeps the line on the storage consistent with the canonical-hash column of the database row.

8.8.3. Canonical-JSON Hash Chain

The hash of an entry is the SHA-256 over the concatenation, separated by a pipe character, of the hash of the previous entry, the canonical-JSON serialisation of the entry data, and the timestamp of the entry in the standard ISO-8601 format. The database stores the hash with the algorithm-identifier prefix `sha256:` followed by the 64 hexadecimal digits of the digest, 71 characters in total. The first entry of the chain uses the literal string `genesis` in place of the previous hash and is seeded in the baseline migration of the database schema.

The canonical-JSON serialisation follows the JSON Canonicalization Scheme of RFC 8785 [Int20]. An earlier implementation used the non-deterministic standard JSON serialiser of the runtime, which gave the same logical entry different hashes depending on insertion order and floating-point representation; the transition to the canonical scheme is recorded in a hash-algorithm version column on the hash-chain table, with the default value `v2`.

The chain table carries the entry identifier as a serial primary key, the hash and the previous hash as text columns, the hash-algorithm version, the idempotency-key UUID and the canonical hash that link the entry to the external storage, and the timestamp. A unique constraint on the previous-hash column enforces the linearity of the chain: no two entries may carry the same predecessor, so a fork cannot be committed.

8.8.4. Atomic Coupling and Coverage

A single transaction of the relational database couples the audit-log insert with the corresponding write to the Write-Once-Read-Many storage, in the order of database insert followed by storage append: the transaction begins, the application inserts the row into the appropriate audit-relevant table with the pre-computed canonical hash, calls the append function of the storage service inside the same transaction, and commits. If the storage append throws, the transaction rolls back, the database state remains unchanged, and the client receives an HTTP 503 response that signals the unavailability of the audit subsystem.

The reverse failure, a storage line written before a transaction commit that fails afterwards, is tolerated and surfaced by the verification path as an orphan warning. The asymmetry is conservative: the external storage may legitimately be a strict superset of the database, since an entry that did not enter the database state cannot have caused

an effect that the audit log would need to record.

Four entry points in the credential system code funnel the entire audit coverage, so the consistency between relational write and storage append holds by construction rather than by reviewing every call site of the audit log. They are the append function of the hash-chain service, the canonical path for credential-status transitions; the generic audit-log service that records master-data changes and user-mutation events; the approval-workflow service that inserts the four-eyes approval rows for issuance batches, revocations, and re-issuance cancellations; and the re-issuance-request service that creates and cancels the wallet-loss requests of section 8.5.3.

The audit subsystem shares its canonicalisation primitive with the Display-Binding hash of the cryptographic four-eyes binding (section 8.6.4); the canonicalisation contract is audited once and applied in two places.

8.8.5. Verification and Failure Handling

Verification runs in three modes. An incremental verification, run by the expiry worker on its hourly schedule and by the application as a pre-check before every batch approval, issuance, and revocation, walks the daily files of the external storage in chronological order from the last verified offset, persisted in a dedicated offset table. For each storage line, it looks up the database row by table name and primary key, recomputes the canonical hash of the stored payload, and compares the result against the corresponding column of the database row. Three outcomes are possible: a match advances the offset; an entry present on the storage and absent from the database is recorded as an orphan and tolerated; a mismatch raises a typed exception that the central error handler maps to an HTTP 503 response. A full verification, triggered on demand by the IT administration before an issuance session, after a security incident, or on an annual cadence, truncates the offset table and re-walks the entire history.

Before the application begins any transaction that would generate an audit-log entry, a pre-flight check on every critical route confirms that the external storage is reachable and writable; on failure, the application responds with an HTTP 503 before any database state is modified.

A documented future extension anchors the head hash of the chain periodically to an external time-stamping authority per RFC 3161 [Int01].

Rejected alternatives. The alternatives fall into three groups. Stronger cryptographic structures, a Merkle tree over the entries and a hardware-security-module-signed root hash, are not justified at the event rate of the application or depend on hardware access that remains future work; a periodic anchoring of the head hash was superseded by the per-entry coupling. Database-internal protections, a database-only audit log, separated application and administration database users, and triggers that block updates and deletes, all stop short of the PostgreSQL superuser and the server root, which is why the tamper protection sits outside the database. An application-level append-only enforcement without a filesystem-level signal yields tamper evidence that is not provable to an external reviewer, and an asynchronous queue towards the storage is unnecessary

because the synchronous transactional coupling is affordable at the present event rate; a hardware write-once medium requires institutional hardware and remains future work.

8.9. Status-List Partitioning

The status lists that carry the revocation and suspension bits of issued credentials follow the naming pattern `module-{semester}` and `diploma-{semester}` introduced in section 8.4. The partitioning scheme behind the pattern implements requirement FR-26.

8.9.1. Motivation and Design Drivers

The swiyu Token Status List specification [Int] encodes credential statuses in a fixed-length bit vector that the issuer publishes as a signed JSON Web Token. The credential system uses the two-bit encoding of section 4.1.4, one bit for revocation and one for suspension, and provisions each list with a maximum length of 100 000 entries, just below the upper bound of 102 400 entries that the swiyu Generic Issuer accepts at this encoding.

Three concerns motivate the partitioning. The first is bit scarcity at institutional scale: at a single department of approximately five hundred enrolled students, with seven module completions per student per semester, an undivided list of 100 000 entries would last approximately fourteen years; at the full OST scale of approximately three thousand eight hundred students across six departments [OSTa], the same list would be exhausted within approximately two years.

The second is the alignment of the credential lifecycle with the lifecycle of the status list that carries it. The institutional process produces and audits credential cohorts per semester; partitioning along the semester boundary aligns each status list with the cohort it tracks and makes the operational signals of the list, such as remaining capacity and issuer reachability, meaningful at the cohort level.

The third is the audit independence of the cohorts: a revocation on a module credential of the autumn semester 2026 should not perturb the status list of the diploma credentials of the spring semester 2027. The four-eyes revocation flow of section 8.5.4 writes on exactly one partition, which keeps the audit trail of a partition self-contained and limits the blast radius of an operational mistake or a key-compromise event to a single cohort.

8.9.2. Partitioning Scheme and Mapping

Figure 8.10 shows the scheme and the assignment of a credential batch to its partition at issuance time. A partition is identified by the pair of semester and credential type: the semester as `HS{year}` for an autumn semester or `FS{year}` for a spring semester, the credential type as one of `university-module-vc` and `university-diploma-vc`. A dedicated `status_lists` table holds the pair under a uniqueness constraint, the

credential type additionally restricted by a check constraint, and maps it to the identifier of the corresponding status list on the swiyu Generic Issuer.

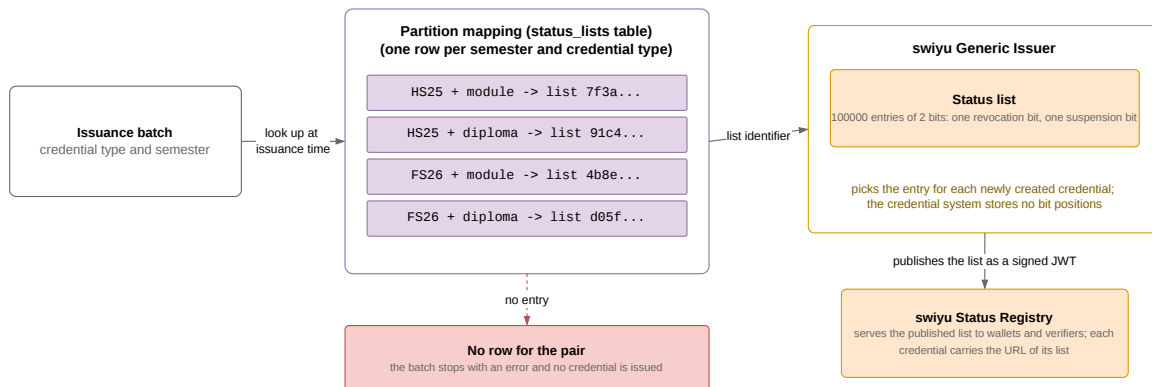


Figure 8.10.: Status-list partitioning by semester and credential type.

The naming pattern `module-{semester}` and `diploma-{semester}` of section 8.4 is a documentation convention that captures the business key in a single readable string; the credential system passes no name string to the swiyu Generic Issuer, and on the swiyu side a list is identified by the identifier that the Generic Issuer returns at provisioning time.

8.9.3. Lifecycle of a Partition

The IT administration provisions a partition through a single operation that creates the two lists of the same semester in parallel: the credential system calls the status-list endpoint of the swiyu Generic Issuer twice, with the two-bit encoding and the maximum length of 100 000 entries, and persists the returned identifiers as two rows of the `status_lists` table. The persistence uses the upsert operation of PostgreSQL on the uniqueness constraint of the business key, which lets the credential system re-point an existing row to a freshly provisioned list if a status-list recovery is required at the institutional level.

A partition is not actively sealed at the end of its semester. The semester boundary triggers the provisioning of the new partitions, but the existing partitions remain mutable for the full ten-year credential validity, so that revocations and suspensions on credentials of the closed semester continue to function. The archival of a partition, decommissioning the swiyu-side list and removing the database row, becomes possible once no credential of the partition is within its validity window; the archival procedure is documented as a future-work operational task and is not automated.

The administrative dashboard surfaces two operational signals per partition, both derived from a single query against the status-list endpoint of the swiyu Generic Issuer; the credential system reads the remaining-entry count that the issuer reports and does not decode the published JSON Web Token itself. The remaining-capacity indicator flags a partition once fewer than five percent of its entries remain free, which gives the IT administration a window to provision the next partition before the present one is

exhausted. The reachability indicator flags a partition whose list the issuer fails to return; an unreachable issuer suppresses the capacity signal of the partition, so that a missing count is never mistaken for a healthy list. Changes against a partition appear in the central audit log of section 8.8.

Rejected alternatives. Four alternatives were rejected: a single global list across all semesters and credential types would couple the blast radius of every revocation to the entire issued population and would run into the bit scarcity quantified in section 8.9.1; a per-semester partitioning without the credential-type split would conflate the frequent module credentials and the rare diploma credentials in the bit accounting and the audit segregation; a finer partitioning by department adds no operational value for a single study programme yet and is straightforward to add later through an additional column in the `status_lists` table; and list identifiers pinned in environment variables would prevent the per-semester rollover that motivates the scheme.

8.10. Key Management

The credential system, the mock service, the swiyu Generic Issuer container, and the internal Public Key Infrastructure rely on cryptographic keys whose creation, storage, rotation, and recovery shape the operational security of the composition. The recommendations for the productive deployment follow the vocabulary and life-cycle phases of NIST SP 800-57 [Bar20].

8.10.1. Overview and Scope

The keys fall into four trust layers. The credential-signing layer holds the keys with which the swiyu Generic Issuer signs credentials and status lists, together with the update key that authorises changes to the Decentralised Identifier (DID) document of the institution; it is anchored in the swiyu Base Registry. The service-to-service layer holds the keys with which the credential system and the mock service authenticate the requests of the lifecycle flows (section 8.5). The internal transport-security layer holds the keys that the internal Public Key Infrastructure of section 8.7 issues to the application containers. The operator-authentication layer holds the WebAuthn keys that bind an approval to a hardware authenticator (section 8.6.4).

With the exception of the Ed25519 DID update key, all asymmetric signing in the composition uses ECDSA over the NIST P-256 curve, in line with the swiyu interoperability profile [Swid] and the credential format of the SD-JWT Verifiable Credential profile [Int26; Ope25a]. The single curve simplifies hardware procurement: one authenticator family supports both the operator approvals and the issuer-signing operations. The external trust anchors of the swiyu Base Registry, the Trust Registry, and the Status Registry are operated by the Federal Office of Information Technology and lie outside the key-management responsibility of this thesis.

8.10.2. Key Inventory

Table 8.4 summarises the complete inventory of cryptographic keys in the composition, grouped by trust layer.

Table 8.4.: Cryptographic keys of the composition.

Layer	Key	Owner and use	Algorithm
Credential signing	Issuer DID update key	DID-toolbox, DID document updates	Ed25519
Credential signing	SD-JWT signing key	swiyu Generic Issuer, credential signature	ES256
Credential signing	Status-list signing key	swiyu Generic Issuer, status-list JWT signature	ES256
Credential signing	Verifier DID key	swiyu Generic Verifier, identity	ES256
Service-to-service	Credential-system JWS key	Credential system, request authentication	ES256
Service-to-service	Mock-service JWS key	Mock service, request authentication	ES256
Service-to-service	Webhook shared secret	Issuer-to-credential-system webhook	HMAC-style
Internal PKI	Root authority	step-ca, internal trust anchor	ECDSA
Internal PKI	Intermediate authority	step-ca, leaf issuance	ECDSA
Internal PKI	Service leaf certificates	Application containers, transport	ECDSA
Operator auth.	WebAuthn credentials	Per study administrator, approval signing	ES256

The webhook shared secret is the only key of the service-to-service layer that is not an asymmetric signature key. It remains because the swiyu Generic Issuer in the current release does not mint outbound JSON Web Signatures, so an ES256 verification cannot replace the shared secret on this path. A mutual-TLS configuration of the webhook was rejected because the issuer configuration exposes no first-class surface for rotating client certificates; it remains a plausible future migration, and the residual risk is recorded in section 9.4.

8.10.3. Storage and Protection

Figure 8.11 maps the server-side keys of table 8.4 to the locations where the key material rests at runtime; the leaf material in the shared certificate volume arrives there over the issuance paths of figure 8.8. Figure 8.12 shows the operator-authentication layer, whose private keys never leave the hardware authenticators.

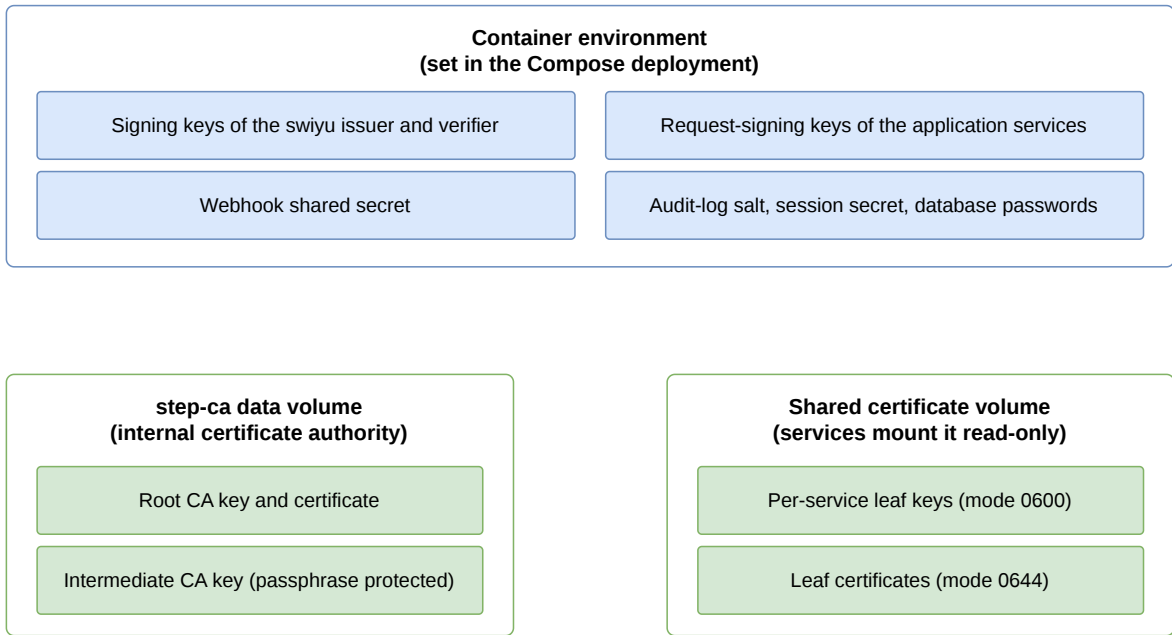


Figure 8.11.: Server-side storage locations of the key material.



Figure 8.12.: Storage of the operator-authentication keys.

The swiyu Generic Issuer loads no keys from a file or a hardware security module; the env-backed key provider is the baseline, the hardware-backed provider future work (section 8.10.5). The Ed25519 update key of the institution DID never enters the issuer container; it remains file material of the DID-toolbox bootstrap (section 8.10.4). The request-signing keys in figure 8.11 are the service-to-service JSON Web Signature keys of the lifecycle flows; the swiyu Generic Issuer validates the requests that the credential system signs against a JSON Web Key Set configured in the issuer environment. The webhook shared secret is transmitted as an HTTP header on each call; the receiving side compares a SHA-256 pre-hash in constant time to defend against timing-side-channel attacks.

The passphrase that encrypts the intermediate key of the certificate authority is persisted inside the step-ca container as a file under restrictive Unix permissions. The authenticator metadata in figure 8.12 comprises the authenticator-attestation-globally-unique identifier (AAGUID), the supported transports, and the de-registration and replacement links that preserve the audit trail across a re-enrolment (section 8.6.3); the WebAuthn credentials of the operators never reach the server in their private form. The audit-log salt in figure 8.11 is the pseudonymisation salt with which the audit log hashes operator identifiers; the session secret authenticates the session cookies. The

credential system and the mock service receive their PostgreSQL passwords through the same environment route and fail to start when any of these symmetric secrets is missing.

8.10.4. Rotation, Bootstrap, and Re-keying

The rotation cadence varies by trust layer. The keys of the credential-signing layer have no periodic rotation in the present implementation. The bootstrap of a credential-signing key proceeds through the DID-toolbox container of the swiyu reference platform: the generated key material is written to files, the credential-signing key is copied into the environment of the swiyu Generic Issuer deployment, and the corresponding DID document is published to the swiyu Base Registry. The DID method of the swiyu platform, carried in the live issuer identifiers as `did:tdw` version 0.3 and since renamed `did:webvh` upstream [Dec], permits subsequent updates of the DID document, but the present implementation does not yet author an operational rotation workflow.

The authentication of the issuer against the Status Registry follows a refresh-token chain: the refresh token renews automatically on every access, with an access-token lifetime of 24 hours, a refresh-token lifetime of 7 days, and a server-side reuse-detection mechanism that invalidates the chain on any anomaly. If the issuer database is wiped or the issuer is offline for more than seven days, the documented recovery procedure applies: the operator fetches a fresh bootstrap refresh token from the Self-Service Portal of the swiyu platform and re-injects it into the issuer configuration. The Trust Registry-side customer key is rotated manually through the same Self-Service Portal, followed by an update of the issuer environment and a restart of the container.

The service-to-service JSON Web Signature keys are rotated manually. Because the credential system exposes no JSON Web Key Set endpoint, a rotation requires a synchronous update of the issuer-side configuration that holds the corresponding public key. The operational overhead is the accepted price of a deployment without a dedicated key-distribution endpoint; the migration to a JSON Web Key Set endpoint that the issuer can poll remains future work.

The root and intermediate certificates have no regular rotation schedule; their rotation is an operational task for exceptional circumstances rather than a fixed cadence.

The re-enrolment of a hardware authenticator after the loss of a YubiKey follows the chained recovery path of section 8.6.3. The pseudonymisation salt and the session secret are rotated only under exceptional circumstances: rotating the former invalidates the re-identification of every historical audit-log entry, rotating the latter every active operator session.

8.10.5. Hardware Procurement and Residual Risks

The four-eyes binding of section 8.6.4 requires a hardware authenticator per study administrator. Two YubiKey 5C NFC tokens were procured for the role, in the firmware version 5.7 or later, certified at the FIDO2 Level 2 assurance level [Yub] and supporting the elliptic-curve signatures that the WebAuthn ceremonies require. The tokens have

been delivered and bound to the two study administrators of the demonstration scenario. No dedicated backup token was procured; the two tokens provide mutual redundancy in the sense that the IT administrator can re-provision an operator who has lost a token without an additional procurement step.

The hardware backing of the credential-signing key of the swiyu Generic Issuer follows a different timeline. A sandbox partition of the Securosys CloudHSM service, a FIPS 140-2 Level 3 hardware security module operated by Securosys in Zurich [Sec], has been requested and is in preparation at the time of writing; its activation is expected after the submission of this thesis. The demonstration scenario therefore ships the documented software-key configuration as the baseline. The residual risk that this configuration accepts, an actor with shell access on the host reading the credential-signing key from the container environment, is recorded together with its compensating controls in the consolidated residual-risk register of section 9.4; the migration of the key to the Securosys CloudHSM Sandbox is the next operational step.

The issuer key-management concept, tracked under a separate ticket and expected to define the productive rotation cadence and procedure for the credential-signing key, the status-list signing key, and the DID key, has not been closed at the time of writing; this section documents the configuration of the demonstration scenario rather than the productive concept that the open ticket is expected to deliver.

Rejected alternatives. Three further alternatives were rejected: shared-secret authentication across the service-to-service channels in general, since the `exp`, `iss`, and `aud` claims of the adopted JSON Web Signatures add a verifiable replay defence; a SoftHSM-based intermediate step for the credential-signing key, skipped because the Securosys CloudHSM Sandbox already provides the operational shape of the productive hardware security module and removes a second migration; and the donation of a physical hardware security module to OST, not pursued because the institutional acceptance and the physical installation cannot complete before the submission of this thesis, with the offer held open for the productive pilot.

9. Security Concept

Chapter 8 describes how the system is built; this chapter describes which threats it defends against, how the design decisions address them, and which residual risks the implementation knowingly accepts. The threat enumeration follows STRIDE [Sho14], the residual-risk vocabulary follows NIST SP 800-30 [Joi12].

9.1. Scope and Analysis Method

The analysis covers the composition of Chapter 8: the credential system, the mock service, the swiyu Generic Issuer and Generic Verifier reference containers, the internal PKI, and the audit-log infrastructure. The three central swiyu registries are trusted as documented in Section 4.2.2; the analysis treats them as upstream trust anchors and does not enumerate threats against them.

The analysis applies STRIDE to five attack surfaces: external (what the credential system exposes through the edge proxy), credential lifecycle (the issuance, revocation, and re-issuance paths), infrastructure (the container host and supporting services), privacy (the personal data of holders and operators), and availability.

9.2. Threat Model

9.2.1. Actors

The threat model considers seven actors. The IT administrator (ITA) is the most critical actor in this implementation: the role holds privileged access on the container host, the credential-system database, the mock-service database, and the audit-log directory. The study administrator (SA) is the four-eyes approver of credential issuance, revocation, and re-issuance cancellation; in the demonstration scenario the same role, together with the lecturers of the dozent role, also maintains the institutional study-administration data in the mock service. The holder is the natural person to whom a credential refers and who operates a swiyu wallet on a personal device. The verifier is an external party that requests a credential presentation. Three further actors are adversaries: the external attacker without credentials on the host, the supply-chain attacker who can place a modified dependency in the build chain, and the Docker-host root attacker with shell access on the composition host.

9.2.2. Assets and Trust Boundaries

The analysis protects the issued credentials, the audit log and its hash chain (Section 8.8), the WORM log, and the status lists (Section 8.9). It further protects the pseudonym-mapping table that resolves operator identifiers in the audit log, the Switch edu-ID identifiers and WebAuthn assertions of the operators, the cryptographic material of the internal PKI (Section 8.7), and the Docker volumes that persist the database state of the three application services.

The architecture makes four trust boundaries explicit; Figure 9.1 marks each crossing with the mechanism that authenticates it. Across the first boundary, the mock service pushes events to the credential system: every request carries an ES256 JSON Web Signature, and every transported record is verified against its hash. The second boundary is the public edge. All external traffic passes the edge proxy over TLS, and the proxy drops the management paths of the two swiyu reference containers before any routing; the privileged interfaces for creating credential offers and changing credential status therefore stay reachable only from inside the composition. The third boundary separates the services inside the composition; the service interfaces of the credential system, the mock service, and the swiyu TLS proxy authenticate with certificates from the internal PKI (Section 8.7). At the fourth boundary, the operator authenticates through Switch edu-ID with enforced multi-factor authentication (Section 8.2.6); approval actions are additionally bound to the hardware WebAuthn ceremony of Section 8.6.4.

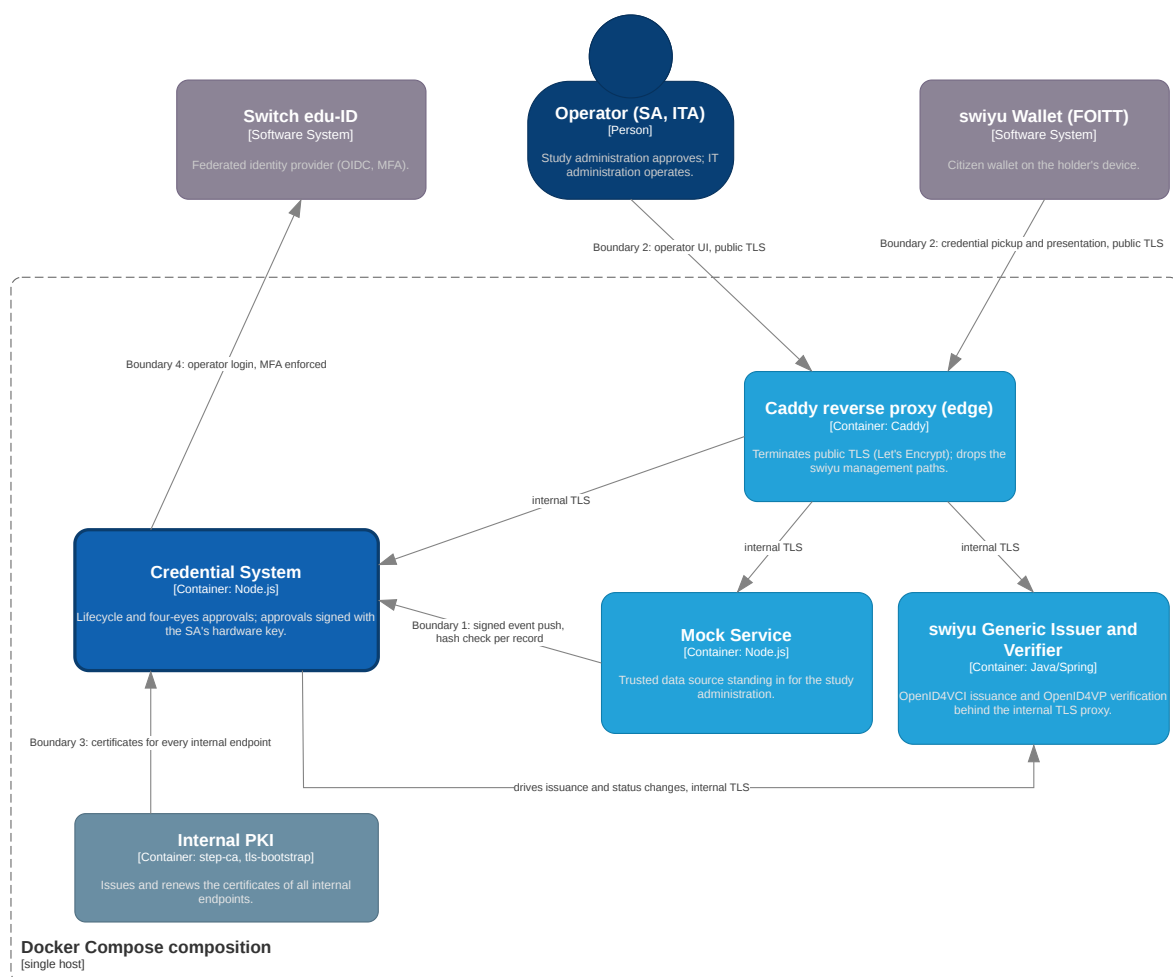


Figure 9.1.: The four trust boundaries of the composition and the authentication at each crossing.

Across the first boundary, the mock service counts as a trusted data source. It stands in for the institutional study-administration system, which lies outside the control of the credential system; the credential system therefore assumes the academic data it receives is authentic and does not re-validate it.

9.2.3. Threat Surfaces

The enumeration yields twenty-seven threats across the five surfaces of Section 9.1. Table 9.1 shows one representative threat per surface; the design-tracking documentation of the project records the full enumeration.

9.2.4. The ITA Attack Cascade

The most consequential threat scenario is the cascade of actions a compromised IT administrator could execute; Figure 9.2 pairs each move of the cascade with the control that stops it. Without the controls, the privileged access of the role (Section 9.2.1) would suffice to turn a forged grade into an issued credential that the holder did not earn and to align the stored audit rows with the forgery.

Table 9.1.: Threat surfaces, with one representative threat per surface and the STRIDE classification.

Surface	STRIDE	Representative threat
External (EX)	Spoofing	Phishing of operator credentials to bypass the four-eyes approval.
Credential (CR)	Tampering	Modification of credential data between batch creation and issuance.
Infrastructure (IN)	Elevation of priv.	Docker-host root extracts environment variables and signs arbitrary credentials.
Privacy (DS)	Information discl.	Re-identification of operators through the audit log over the retention horizon.
Availability (AV)	Denial of service	Status registry of the swiyu platform unreachable during a verification request.

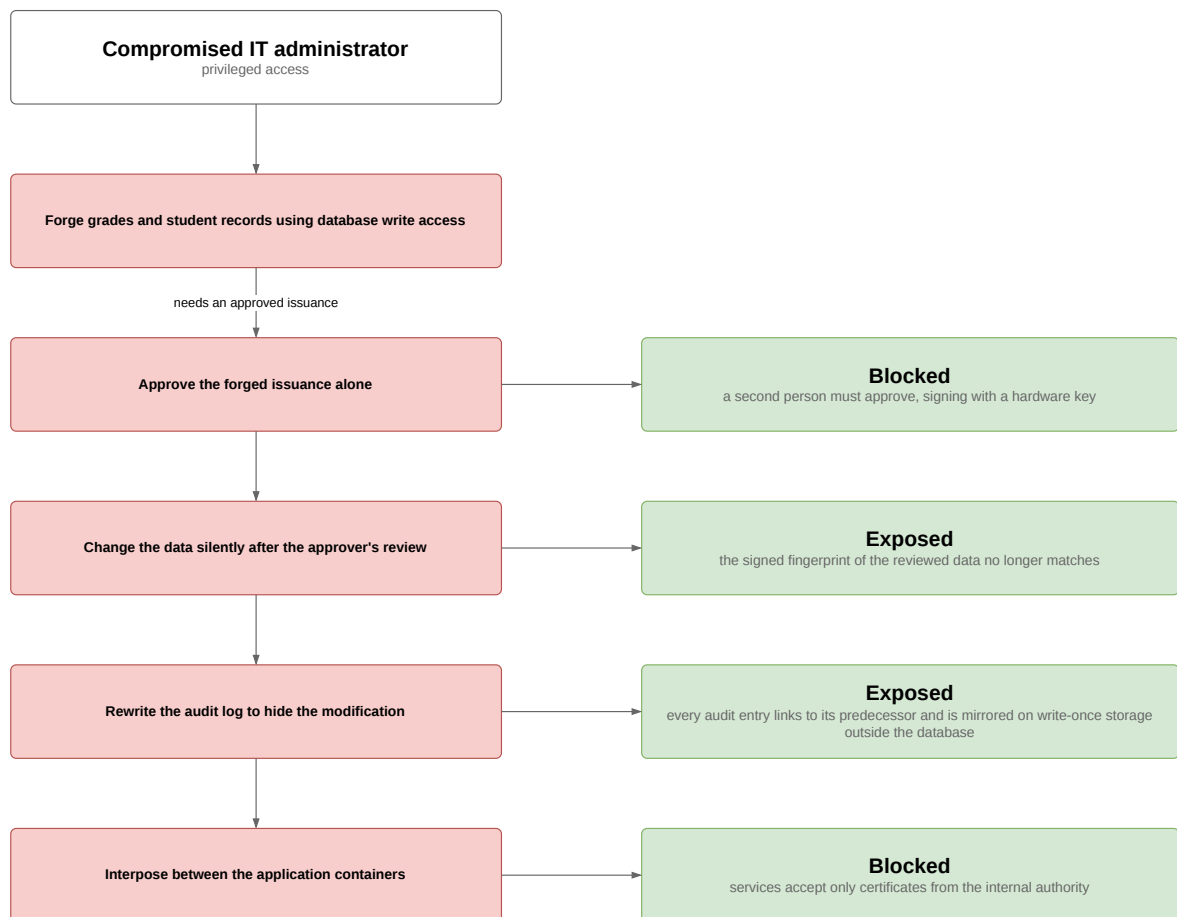


Figure 9.2.: The four break points of the IT administrator attack cascade.

The controls break the cascade at four points, in two ways. Two points block the move outright, even for an actor who controls the database: the four-eyes approval requires a second person whose hardware key signs over the approval data (Section 8.6.4), and the internal PKI (Section 8.7) denies an interposed endpoint a certificate that the consumers trust. The other two points do not prevent the write but make it detectable: the signed

fingerprint of the reviewed data is the Display-Binding hash inside the WebAuthn challenge, and the predecessor link of every audit entry is the hash chain (Section 8.8.3), whose write-once copy outside the database is the WORM storage of Section 8.8.2.

9.3. Security Controls and Mapping to Design

Table 9.2 maps each control to the threat class it addresses and to the design section that documents the implementation.

Table 9.2.: Security controls of the composition and the design sections that document them.

Control	Threat classes addressed	Design section
Four-eyes approval, hardware-bound WebAuthn	Spoofing of operator, ITA cascade step one	Sections 8.6 and 8.6.4
WORM audit log, external write-once	Repudiation, infrastructure tampering	Section 8.8.2
Canonical-JSON hash chain	Tampering of audit records	Section 8.8.3
Internal PKI, ES256 JSON Web Signatures	Man-in-the-middle, east-west spoofing	Sections 8.7 and 8.10.2
Switch edu-ID, REFEDS MFA	Spoofing of operator	Section 8.2.6
Fail-closed status-list verification	Stale revocation, credential-lifecycle tampering	Sections 8.5.4 and 8.9
Holder-key binding (cnf)	Theft of credential, replay	Section 8.4.1
Salted SHA-256 pseudonymisation	Re-identification through audit-log access	Sections 8.8 and 8.10.3
Key-management hierarchy	Compromise of signing material	Section 8.10

The controls of table 9.2 jointly address two non-functional requirements of Chapter 5. NFR-07 demands a four-eyes invariant that cannot be circumvented; the WebAuthn-bound approval ceremony, the application-level identity check, and the database-level uniqueness constraint on the approver identifier per workflow row address it together. NFR-10 demands a tamper-evident audit log; the canonical-JSON hash chain and the external WORM storage address it together. Both audit controls are wired fail closed: the incremental cross-check of section 8.8.5 runs before every approval, issuance, and revocation, and a mismatch between the external storage and the database stops the operation with an HTTP 503 response.

9.4. Residual Risks Accepted in the MVP

The MVP accepts residual risks that the demonstration scenario does not eliminate; table 9.3 summarises the register.

Table 9.3.: Residual risks of the MVP, with compensating controls and future-work paths.

Residual risk	Compensating control	Future work
No HSM root signing of the audit layer	WORM cross-check against the database (incremental verification, see section 8.8.5)	Activation of the Securosys CloudHSM Sandbox
Ten-year credential validity	Holder-key binding plus revocation plus re-issuance flow	Shorter validity not pursued; rationale documented in section 8.2.5
ITA also administers the mock service	Documented trust boundary, hash check at API, four-eyes for issuance only	Per-record signature of the dozent role on grade entries
No cryptographic unlinkability	Verifier-controlled selective-disclosure scope (section 8.4.4)	Awaits BBS+ in the swiyu interoperability profile
Per-claim issuer-side SD not enforceable	Verifier presentation-definition controls disclosure	Awaits resolution of the upstream limitation of the Generic Issuer
Docker-host root compromise	Container isolation, restrictive permissions on environment files, gitignore	Migration of the signing key to the HSM
Supply-chain and image hardening gaps	Manual review, pinned image versions, restricted operator population	Dependabot, image signing, encrypted volumes, automated backups
WebAuthn assertions retained 30+ years	Pseudonymised audit entries, separate mapping table, restrictive access	Formal data-protection review at OST, foreseen for a later pilot phase; cryptographic-erasure scheme or shorter truncation horizon as the mitigation path if the 30-year horizon is not admissible

9.5. Compliance Context

Two legal regimes meet in the credential system. As a cantonal public-law institution of higher education, OST falls primarily under the data-protection law of the Canton of St. Gallen, aligned with the revised Federal Act on Data Protection of the Confederation [Swia]. The European General Data Protection Regulation (GDPR) [Eur16] can reach the system as well: under the market-location principle of its Article 3(2), it can apply when the processing relates to offering services to data subjects in the European Union, for example when credentials are issued to or presented by persons in the EU.

This determination is a working assumption, not a confirmed legal position. The formal review with the responsible person at OST is foreseen for a later pilot phase, so the design had to hold under either outcome of that review. Table 9.4 maps each

provision to the answer of the credential system; the remainder of this section explains the mechanisms behind that mapping. The MVP itself is unaffected by the open confirmation.

Table 9.4.: Mapping of the legal provisions to the answers of the credential system.

Provision	Requirement	Answer of the system
Cantonal data-protection law of St. Gallen, aligned with the revised Federal Act on Data Protection [Swia]	Primary data-protection regime for a cantonal public-law institution	Applies as the working assumption of this thesis; formal review foreseen for the pilot phase
GDPR under the market-location principle	Applies when a credential of an EU resident is presented to a verifier in the EU	Covered by the same controls as the cantonal regime
GDPR Article 32, security of processing [Eur16]	Pseudonymisation and integrity of the processed audit data	Three-layer separation of the audit data (Section 8.8)
GDPR Article 17, right to erasure	Erasure of personal data once the retention horizon expires	Salt rotation and purge of the pseudonym-mapping table
Cantonal duty to retain examination records	Retention of the issuance-relevant records	Ten years for credential-bound entries, two years for operational entries
Cantonal admissibility of the 30-year WebAuthn retention	Open, falls to the formal review	Mitigation path of table 9.3; integrity-proof layer unaffected either way
Federal Act on Electronic Identity and Other Electronic Credentials [Swig]	Governs the Swiss Trust Infrastructure, not the credential system	Inherits the platform assurance level as a relying-party issuer

The audit subsystem (Section 8.8) was designed with two GDPR articles in view, and one mechanism answers both. The three-layer separation of the audit data addresses the security-of-processing requirement of Article 32 and the corresponding cantonal provisions: the hash chain holds no personal data and serves only the integrity proof, a salted hash pseudonymises the audit entries, and the pseudonym-mapping table is the single point at which an operator identifier can be re-identified. The same separation turns the right to erasure of Article 17 into a single operation: once the retention horizon expires, the salt is rotated and the mapping table purged, and what remains is an integrity-only proof of the historical events with no path back to the operator.

Retention follows the weight of the entry. Credential-bound entries (issuance, revocation, suspension, and the four-eyes approvals of the corresponding batches) are kept for ten years, matching the credential validity and the cantonal duty to retain examination records. Operational entries (operator logins, role-assignment changes,

configuration changes) are kept for two years. Neither horizon is baked into the system: every audit entry carries its retention category, so both periods are policy values that the data-protection review can adjust at any later point without changes to the audit data model.

The one genuinely open row of table 9.4 is the 30-year retention of the stored WebAuthn assertions. The design does not depend on the outcome of the review: if the horizon is not admissible, the mitigation path in the residual-risk register applies, and the integrity-proof layer remains unchanged either way.

The Federal Act on Electronic Identity and Other Electronic Credentials, finally, never addresses the credential system directly: as a relying-party issuer, the system inherits the assurance level of the Swiss Trust Infrastructure. What remains operationally is the platform-level transition from public beta to productive operation (Section 4.2.2).

9.6. Open Items and Future Work

The formal data-protection review with the responsible person at OST, foreseen for a later pilot phase, is the single hard precondition for the compliance position of Section 9.5. Its outcome changes the legal-confirmation paragraphs, not the technical analysis of this chapter, and it does not affect the MVP.

The remaining open items are internal to the credential system and the deployment infrastructure and do not block the demonstration scenario. Beyond the future-work column of table 9.3, future work for a productive deployment comprises a Merkle-tree representation of the audit data, the external master-public-key source documented as the soll-architecture in Section 8.6.4, and heartbeat alerting on the availability surface. It further comprises a content-delivery network and a web-application firewall in front of the edge proxy, and a hardware-authenticator-only login policy that prevents the fall-back to the standard Switch edu-ID multi-factor methods [Ope21].

10. Implementation

The implementation realises the design of Chapter 8 on the target deployment of section 8.1.1; it comprises the credential system, the mock service, and the supporting infrastructure.

10.1. Technology Stack and Dependencies

The credential system, the mock service, and the verifier web applications run as Node.js services on Node 24 [Nod26], pinned through the official `node:24-alpine` container image. The credential system is written in TypeScript and uses the native type-stripping of Node 24 to execute TypeScript directly, without a transpilation step in the container build; the mock service is written in JavaScript. The Spring Boot reference services of the Swiss Trust Infrastructure run unchanged from the upstream images `swiyu-issuer:3.1.1` [Swi26b] and `swiyu-verifier:2.3.3` [Swi26d].

The credential system and the mock service store their data in PostgreSQL 18 through the `postgres:18-alpine` container image; the verifier web applications keep their verification log in an embedded SQLite database. The upstream image pins the internal database of the swiyu Generic Issuer to PostgreSQL 15; the mixed major versions are accepted because each database is owned by a single service. The two PostgreSQL-backed services manage their schema with `node-pg-migrate`, which the service container invokes from its entry-point script on startup.

Caddy 2.11.3 [Cada] serves the TLS plane, both at the public edge and as the internal proxy that terminates the TLS hop in front of the swiyu reference containers; the open-source smallstep step-ca 0.30.2 [Sma] acts as the internal certificate authority. Section 8.7 documents the architectural role of each component.

Alongside the runtime, the credential system depends on Express 5.2 [Exp24] for the HTTP layer, `@simplewebauthn/server` 13.3 [Sim] for the WebAuthn ceremony of section 8.6.3, `openid-client` 6.8 for the OpenID Connect [Ope] flow against Switch edu-ID, and `jose` 5.9 for the JSON Web Signature operations against the swiyu Generic Issuer (section 8.6.4). The supporting dependencies are `pg` 8.20 as the PostgreSQL driver, `envalid` 8.1 for the strict validation of environment variables at startup, `csrf-csrf` 4.0 for the double-submit-cookie cross-site-request-forgery middleware, `connect-pg-simple` 10.0 for the PostgreSQL-backed session store, and `node-cron` 3.0 for the schedule of the background worker. The `canonicalize` 3.0 implementation of RFC 8785 [Int20], used by the audit-log integrity mechanism of section 8.8, is the only non-Node dependency that the implementation introduces.

10.2. Repository Layout and Service Topology

The implementation lives in a single Git repository whose top-level layout follows the deployment unit of the composition. Table 10.1 summarises the principal directories and their roles.

Table 10.1.: Top-level layout of the implementation repository.

Directory	Role
<code>credential-system/</code>	Credential-issuance service and background worker, written in TypeScript on Node 24.
<code>mock-service/</code>	Mock institutional study-administration service, written in JavaScript on Node 24.
<code>swiyu-issuer/</code>	Configuration of the upstream Generic Issuer reference container.
<code>swiyu-verifier/</code>	Configuration of the upstream Generic Verifier reference container.
<code>swiyu-tls-proxy/</code>	Internal Caddy proxy that terminates the TLS hop in front of the two Java services.
<code>verifier-webapp/</code>	In-house verifier web applications, public and administrative.
<code>caddy/</code>	Edge reverse-proxy configuration.
<code>step-ca/</code>	Internal certificate authority configuration.
<code>tls-bootstrap/</code>	Bootstrap and renewal loop for the internal leaf certificates.
<code>didtoolbox/</code>	DID-Toolbox helper, used once at deployment time to generate the DID document of the institution.
<code>Makefile</code>	Single orchestration entry point for the composition.

Each of the nine container stacks has its own `compose.yml` file in the corresponding directory; Figure 10.4 in section 10.5 groups the stacks and gives their startup order.

10.3. Selected Implementation Aspects

10.3.1. Append-only WORM Storage Coupled to the Hash Chain

The Write-Once-Read-Many storage of section 8.8.2 and the canonical-JSON hash chain of section 8.8.3 are implemented as two cooperating service modules, namely `wormService.ts` and `hashchainService.ts`, in the source directory `credential-system/src/services/`. The append function of the hash-chain service realises the atomic coupling of section 8.8.4: it reads the hash of the previous entry inside the transaction that the caller opens, computes the hash of the current entry, inserts the row, and calls the append function of the WORM service within the same transaction.

The storage append writes through a file descriptor opened in append mode, with an explicit `fsync` after every write. Lines that fit within the Linux pipe-buffer limit of 4096 bytes are atomic at the kernel level. The incremental-verification function of the storage takes a row-level lock on the offset table before walking new lines, so that two concurrent runs of the worker cannot advance the offset past the same line; the

offsets only commit when every line in the pass verifies, which makes the verification crash-safe.

A dedicated migration locks the canonical-hash columns of the database to the type `CHAR(71)`, which exactly matches the wire shape of the algorithm-prefixed digest of section 8.8.3. This schema-level invariant prevents a future change of the hash-storage format from desynchronising the storage line and the database row.

10.3.2. Transactional Four-Eyes Approval Ceremony

The cryptographic four-eyes binding of section 8.6.4 is implemented as a transactional ceremony in the service module `approvalWorkflow.ts`, supported by the binding-hash helpers of `approvalBinding.ts` and the WebAuthn assertion-verification helpers of `approvalCeremony.ts`. Figure 10.1 shows the single database transaction that the insert function of the approval workflow opens once the security key signature, the WebAuthn assertion, has been verified; the write-once audit storage in the figure is the WORM storage of section 10.3.1. Together, the row lock and the counter check defend against a race condition in which two concurrent approval attempts on the same authenticator advance the counter independently; the standard WebAuthn implementation alone does not prevent this.

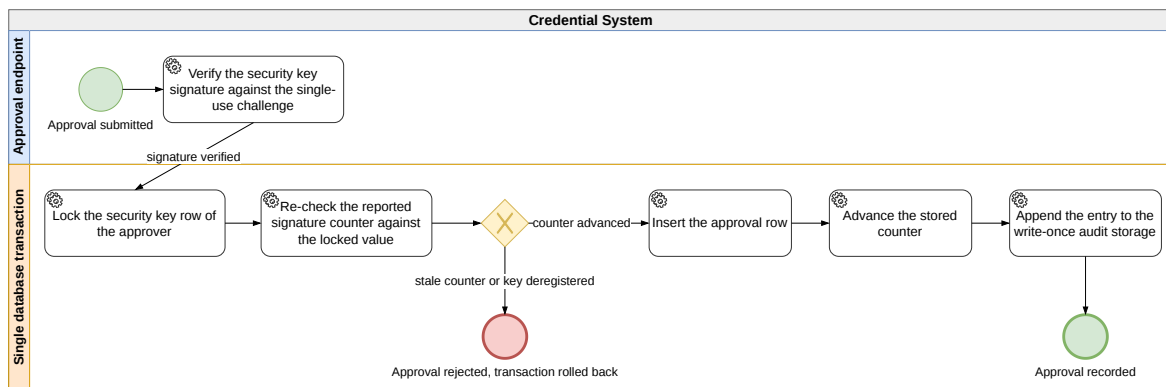
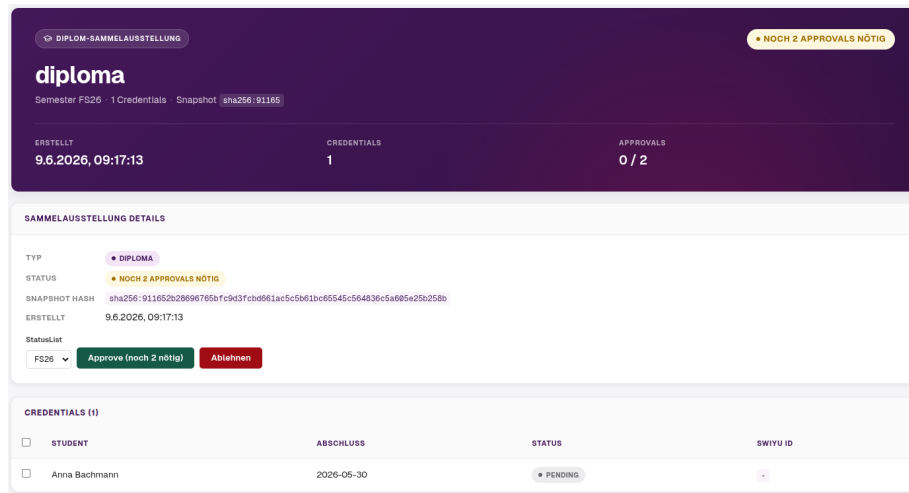
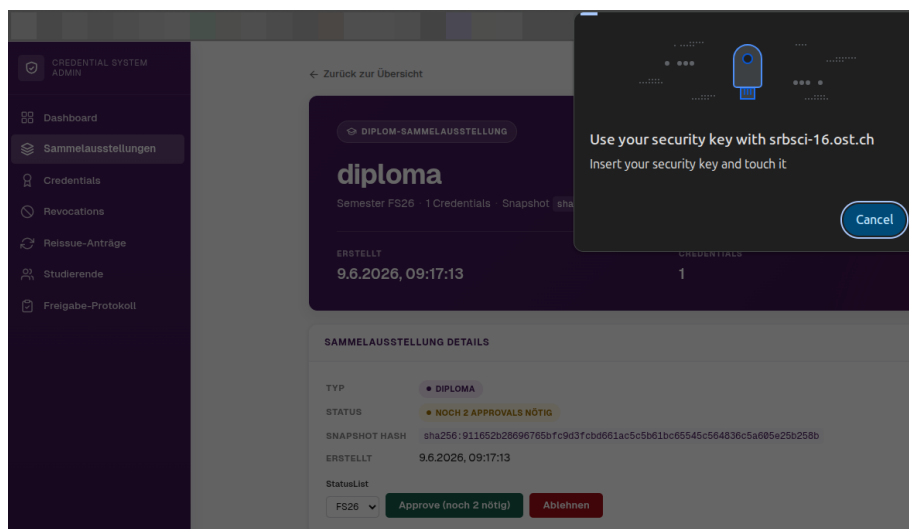


Figure 10.1.: Single database transaction of the four-eyes approval insert.

Figure 10.2 shows the ceremony from the operator’s perspective: the approval view presents the batch under approval together with its snapshot hash, and the browser prompts for the hardware security key whose assertion signs the Display-Binding hash of section 8.6.4 as the WebAuthn challenge.



(a) Approval view with the snapshot hash.



(b) Security-key prompt during the approval.

Figure 10.2.: Four-eyes approval ceremony in the operator interface.

10.3.3. Inter-Service Signature with Body-in-Claim Envelope

The credential-system module `jwtService.ts` and the issuer-side configuration of the `APPLICATION_AUTHENTICATION_JWKS` variable implement the service-to-service authentication of section 8.10.2. The implementation deviates from conventional bearer-token authentication in one operationally important respect: the helper function that signs a request to the swiyu Generic Issuer packs the request body into the `data` claim of the JSON Web Signature and sends the resulting compact representation as the HTTP request body, with the content-type header set to `application/json`. Listing 10.1 shows the signing helper and its call site in simplified form.

The request-filter chain of the swiyu Generic Issuer forces this deviation: it discards the wire body and replays the body from the `data` claim of the signature. The content type `application/json` would describe the request more accurately, but the content negotiation of the Spring framework runs before the filter chain of the swiyu Generic

Listing 10.1: Simplified signing helper with the request body in the data claim.

```
1 async function signForIssuerWithBody(sub, body) {
2   const data = JSON.stringify(body);
3   return new SignJWT({ data })    // request body as claim
4     .setProtectedHeader({ alg: 'ES256', kid: PRIVATE_JWK.kid })
5     .setIssuer('credential-system')
6     .setAudience('swiyu-issuer')
7     .setSubject(sub)              // operation type
8     .setIssuedAt()
9     .setExpirationTime('60s')
10    .sign(PRIVATE_KEY);
11 }
12
13 const jwt = await signForIssuerWithBody('create-credential-offer',
14   offer);
15 await fetch(url, {
16   method: 'POST',
17   headers: { 'Content-Type': 'application/json' },
18   body: jwt,              // compact JWS as the wire body
19 });
```

Issuer and rejects content types outside the configured set. The signature uses the ECDSA P-256 algorithm and a lifetime of 60 seconds, which bounds the replay window.

10.3.4. Leaf-Certificate Renewal with Hot Reload through SIGHUP

A small set of bash scripts in the `tls-bootstrap` container implements the initial issuance and the renewal loop of the internal leaf certificates of section 8.7.3 and maintains the shared certificate volume of figure 10.4. Figure 10.3 shows one renewal pass; the check interval and the renewal threshold are configurable, the figure shows the defaults. Proving itself with the current leaf is the X5C mechanism of the `step ca renew` command, which therefore needs no bootstrap password. The certificate swap in the consumer services is the `setSecureContext` method of the Node.js TLS server, which rotates the certificate and the private key atomically, without interrupting in-flight connections and without a container restart.

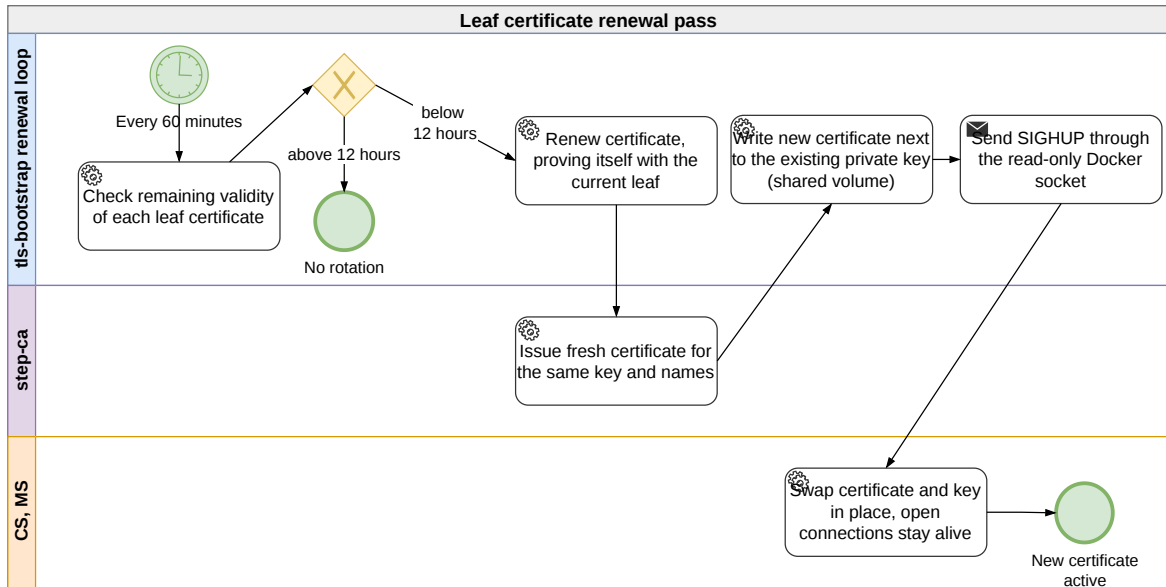


Figure 10.3.: One renewal pass of the internal leaf certificates with hot reload.

10.3.5. Idempotent Background Worker

A single Node.js process in the module `worker.ts` implements the expiry worker of section 8.3.5 on a configurable cron schedule, by default once per hour. The three steps of a pass run in the fixed order described there; a divergence found by the incremental audit-log verification raises a typed exception that aborts the entire pass before the expiry and retry steps run.

The retry loop enforces a per-credential retry budget through the corresponding columns of the credential row and emits a structured audit-log entry once the budget is exhausted. The expiry of unclaimed offers (section 8.5.5) needs no loop of its own: a dedicated route on the credential system receives the status webhook of the swiyu Generic Issuer whenever the state of an offer changes, which replaced the polling-based detection of an earlier implementation state by a push-based notification. Section 10.3.6 documents that route.

10.3.6. Idempotent Inbound Webhook with a Composite Deduplication Key

The lifecycle notifications of the swiyu Generic Issuer arrive on a dedicated listener that the credential system serves as a separate plain-HTTP application on the internal container network, because the upstream issuer image provides no mechanism to trust the internal certificate authority for its outbound webhook calls. The listener authenticates each delivery through a shared API key that the handler compares in constant time, hashing both values with SHA-256 before the comparison so that neither the content nor the length of the configured key leaks through the response timing.

The swiyu Generic Issuer delivers events at least once and re-sends a delivery that did not receive a success response. The handler therefore derives a deduplication key

from the subject identifier, the event type, and the event timestamp, and inserts the event row with an `ON CONFLICT DO NOTHING` clause against a unique constraint on that key. A repeated delivery inserts no row and is answered with a success status, which removes the event from the retry queue of the issuer without processing it a second time. The deduplication row is persisted before the type-specific processing runs; a processing failure marks the persisted row with the error message instead of releasing the key, so the subsequent retry of the issuer is short-circuited as a duplicate as well, and the recorded error remains for operator follow-up rather than triggering an automatic reprocessing. The constraint thereby turns the at-least-once delivery of the issuer into an at-most-once processing of every status mutation.

10.4. Testing Approach

The credential system and the mock service are tested with the test runner that Node 24 provides through the `node:test` and `node:assert` modules of the standard library [Nod; Nod26]; no third-party test framework is used. Table 10.2 summarises the automated suites at the time of writing.

Table 10.2.: Automated test suites of the implementation.

Service	Suite	Test files	Test cases
Credential system	Unit	7	53
Credential system	Integration	9	80
Mock service	Unit and contract parity	2	16
Total		18	149

The unit tests of the credential system cover the determinism of the hash construction and the canonical-JSON serialisation, the append and pre-flight check of the audit-log storage, and the binding hashes of the four-eyes approval. They further cover the truth table of the final batch status in the bulk-issuance flow, the single-use property of the WebAuthn challenge store, and the cross-site-request-forgery middleware. The integration tests run against a throwaway PostgreSQL database and a temporary directory for the audit-log storage. They cover the transactional coupling of the audit log with the database, the retry semantics of the bulk-issuance flow, and the persistence of the callback status, as well as the inbound-webhook handler for the lifecycle notifications of the swiyu Generic Issuer, the inter-service signature on cross-service requests, and the audit trail of the master-data changes. The integration runner enforces serial execution because the tests share the audit-log directory and the corresponding offset table.

The mock service contains a small set of unit tests; the principal one is a parity test that recomputes the hash of the cross-boundary contract and compares it against the hash that the credential system computes. The parity test prevents a silent divergence of the hash construction between the two services, which would manifest at runtime as a mass rejection of inbound events by the credential system.

Three coverage gaps remain. The verifier web applications contain no automated test; the user-facing verification flow is exercised manually during the demonstration scenario. The status-list partitioning of section 8.9 has no dedicated test; the issuance integration tests exercise it implicitly. The OpenID Connect callback path has no direct test; it is exercised against a live Switch edu-ID instance during the demonstration scenario.

10.5. Deployment and Operational Interface

The target `make start` brings up the composition from a clean host in the numbered order of Figure 10.4; the shared certificate volume in the figure is the compose volume `service_certs` that the renewal loop of section 10.3.4 maintains.

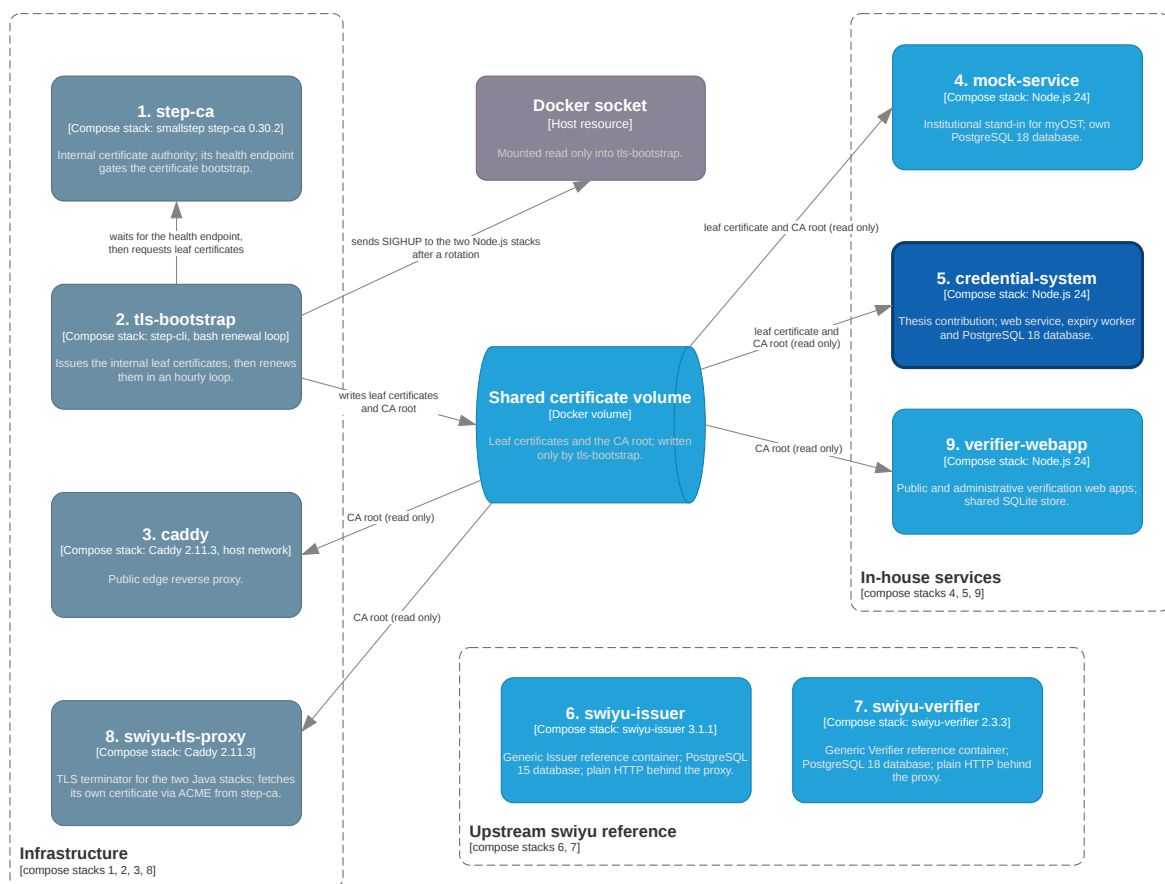


Figure 10.4.: The nine compose stacks with the startup order of `make start`.

Each service reads its configuration from a per-service `.env` file that the deployer populates before the first start. The principal inputs are the credentials with which the swiyu Generic Issuer authenticates against the central swiyu registries, including a one-shot bootstrap refresh token that the issuer-side configuration consumes on the first start and replaces by the rotating refresh token that the issuer maintains autonomously.

Further Makefile targets cover the operations: `make step-ca-status` inspects the internal certificate authority, `make tls-bootstrap` and `make tls-renew` bootstrap and

re-issue the leaf certificates, `make stop` stops the composition gracefully and preserves the database volumes, and `make stop-full` additionally wipes the database volumes of the in-house services. The destructive stop preserves the database volume of the swiyu Generic Issuer because a wipe of that volume would lose the persisted status-list identifiers and the rotated refresh token.

A single end-to-end smoke test validates the deployment after a clean start: the deployer issues a test credential through the administrative interface, receives it through the official Swiss wallet, verifies it through the public verifier interface, and suspends it through the administrative interface.

10.6. Implementation Limitations

The TypeScript configuration of the credential system does not enforce the strict mode of the compiler. A comment in the configuration file documents this as a deliberate baseline that the project tightens incrementally rather than imposes at the start. Beyond the type checker, the repository configures no automated linter, no formatter, and no pre-commit hook. The GitLab continuous-integration pipeline of the repository (appendix B.2) enforces the type check and the unit tests on every merge request and on every push to the default branch, with two exclusions: the integration-test suites require a live database and run locally through the containerised test runner, and one legacy unit-test suite of the mock service still targets the replaced API-key authentication and is excluded pending its migration to the inter-service signature.

A legacy refresh-token script at the repository root served as an emergency tool for the rare case in which the rotating refresh-token chain of the swiyu Generic Issuer becomes desynchronised; it was removed once the autonomous rotation had proven stable, because running it while the rotation is active triggers the reuse detection of the central swiyu registry and invalidates the chain.

Several operational decisions remain with the adopting institution rather than the implementation; the Confluence *Implementierungsanleitung* of the project documents them as open decisions. The principal one is the choice between the env-backed key provider of the present implementation and the hardware-security-module-backed provider that section 8.10.5 documents as future work.

11. Results

This chapter measures the delivered system against the requirements catalogue of Chapter 5. Judgements are labelled as such throughout.

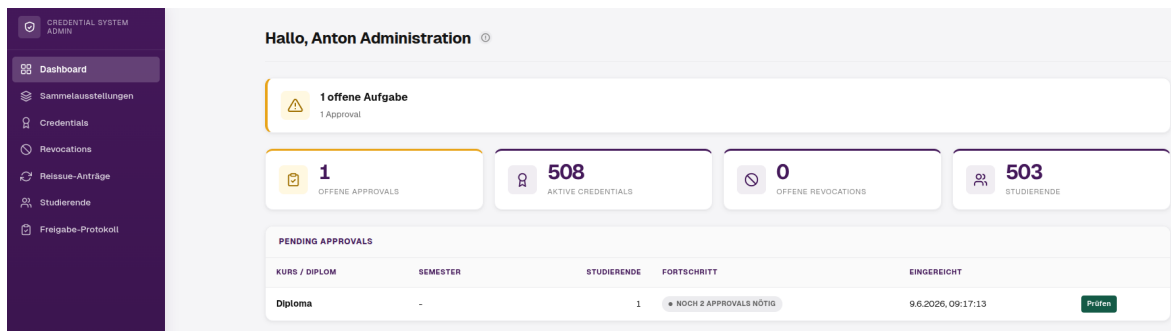
11.1. Delivered System

The thesis delivers a working Minimum Viable Product on an OST-internal host. It comprises three in-house services: the credential system (the issuance and lifecycle backend), the mock service (the stand-in for the OST study-administration system), and the verifier web application (a public-facing and an administrative server). Supporting infrastructure covers data, audit log, PKI, proxy, and authentication; figure 8.1 in Chapter 8 shows the full composition.

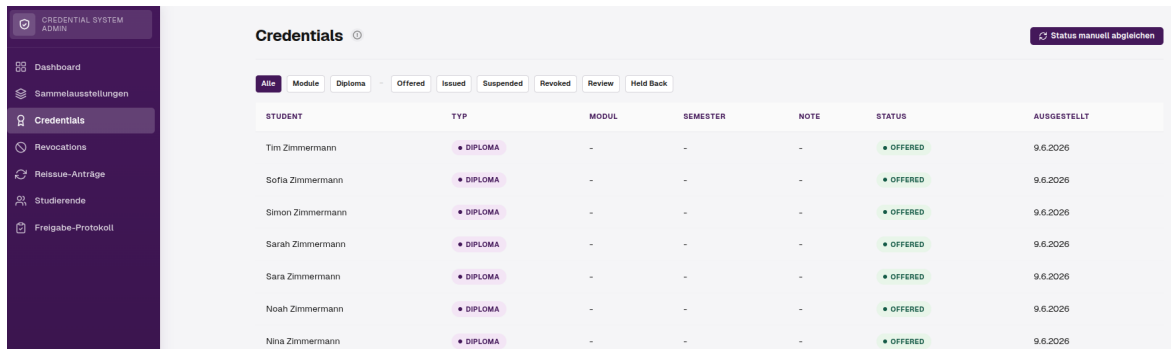
The system issues and verifies both credential types of Section 8.4, the diploma credential and the module credential, end-to-end against the swiyu Generic Issuer [Swi26b] and the swiyu Generic Verifier [Swi26d]. The credential system implements issuance, suspension, restore, revocation, and re-issuance after a wallet loss. A master-data correction flags the affected credential for review, from which an operator triggers a suspension or revocation manually; the fully automated revoke-and-re-issue chain for this case remains future work. A stakeholder demonstration with the OST study administration was held on 28 May 2026.

Three figures document the delivered system on the demonstration deployment, following a credential through its life: the operator backend that manages the lifecycle, the issuance into the wallet, and the verification by a relying party. All screenshots show synthetic test data; appendix C.1 in the appendix collects further screenshots of the portal interfaces.

Figure 11.1 shows the operator backend of the credential system. The dashboard summarises the open approvals, active credentials, open revocations, and registered students, and the credential list exposes the lifecycle states of figure 8.4 as filters on the issued credentials.



(a) Operator dashboard with a pending approval.



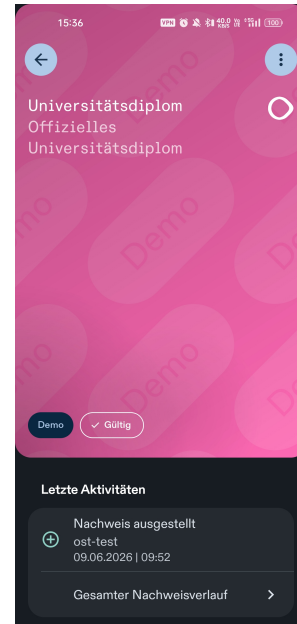
(b) Credential list with lifecycle-state filters.

Figure 11.1.: Operator interface of the credential system.

Figure 11.2 shows the issuance side. The student portal of the mock service presents the credential offer as a QR code, and the issued diploma credential appears in the swiyu wallet, whose activity log records the issuance by the OST test issuer. The Demo watermark on the wallet card marks credentials whose issuer is registered on the swiyu public beta rather than the productive trust infrastructure [Swic] (section 4.2.2); it is not part of the credential design.



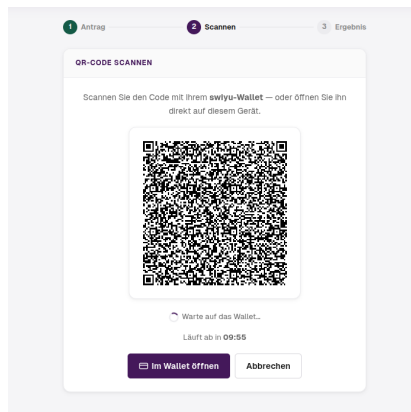
(a) Credential offer in the student portal.



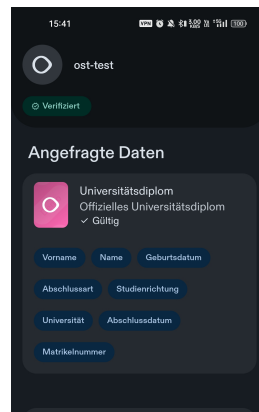
(b) Issued credential in the swiyu wallet.

Figure 11.2.: Diploma issuance on the demonstration deployment.

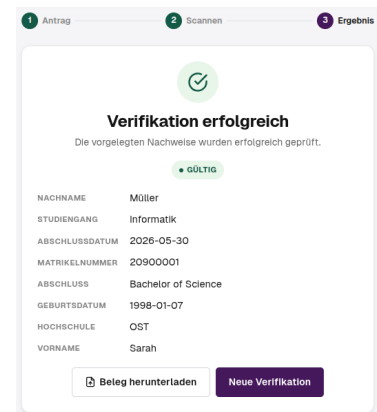
Figure 11.3 shows the verification side. The public verifier web application renders the presentation request as a QR code, the wallet lists the requested claims and asks the holder for consent before disclosing them, and the verifier reports the successful verification together with the disclosed claims.



(a) Presentation request.



(b) Disclosure consent.



(c) Verification result.

Figure 11.3.: Diploma verification from presentation request to result.

Chapter 10 documents the implementation, Chapter 8 the architecture, and Chapter 9 the security analysis.

11.2. Functional Requirements Fulfilment

Table 11.1 assesses the 25 functional requirements of Chapter 5. A requirement counts as *achieved* when the named code path exists in the implementation repository and an integration test or a manual walkthrough on the demonstration deployment confirms the flow, as *partially achieved* when the code path exists but a documented artefact is missing, and as *not achieved* when the implementation was descoped or not completed within the thesis.

Table 11.1.: Functional-requirement fulfilment.

ID	Status	Evidence (code path or artefact)
FR-01	Achieved	mock-service/src/routes/courses.js, enrollments.js
FR-02	Achieved	credentialBuilder.buildDiplomaCredential, bulkIssuanceService
FR-03	Achieved	credential-system/src/routes/revocations.ts
FR-04	Achieved	credentials.ts suspend and restore endpoints
FR-05	Achieved	routes/statuslists.ts, statusListService.ts
FR-06	Achieved	verifier-webapp/public-server/server-public.js
FR-07	Achieved	verifier-webapp/admin-server/server-admin.js
FR-09	Achieved	students table, routes/events.ts student-register
FR-10	Achieved	buildModuleCredential, credentialTypes.ts micro
FR-11	Achieved	shared bulk_issuances and revocation_requests tables
FR-12	Achieved	routes/users.ts, invitationService.ts
FR-13	Achieved	app.ts requireRole(Role.SA) mounts
FR-14	Achieved	app.ts IT-administrator mounts for users, audit, status lists
FR-15	Achieved	hashService.ts, source_hash and snapshot_hash columns
FR-16	Achieved	bulkIssuanceService bulk issuance per course and semester
FR-17	Achieved	mock-service/src/services/pushService.js
FR-18	Achieved	credentials.status='pending', webhookEventHandler
FR-19	Achieved	mock-service/src/services/diplomaService.js
FR-20	Achieved	routes/auth.ts, OIDC_REQUIRE_MFA=true
FR-21	Achieved	mock-service/src/routes/portal.js, edu-ID OIDC
FR-22	Not achieved	No HSM integration; descoping rationale in section 11.6
FR-23	Achieved	REST endpoints in routes/events.ts; OpenAPI 3.1 specification in docs/api/openapi.yaml
FR-24	Achieved	pushService.js student-update, routes/events.ts student-update
FR-25	Achieved	status='review_needed', routes/reissueRequests.ts
FR-26	Achieved	status_lists UNIQUE(semester, credential_type)

In total, 24 requirements are achieved and one is not achieved (FR-22).

11.3. Non-Functional Requirements Fulfilment

Table 11.2 assesses the 18 non-functional requirements of Chapter 5. Beyond the states of section 11.2, the assessment uses *not measured* for requirements whose verification would require a measurement harness that the thesis has not built.

Table 11.2.: Non-functional-requirement fulfilment.

ID	Status	Evidence or rationale
NFR-01	Not measured	No benchmark harness in <code>test/</code> or <code>scripts/</code>
NFR-02	Achieved	Single measured run: a 500-student diploma batch issued in about one minute from the second confirmation, within the 2-minute target
NFR-03	Not measured	No timing instrumentation on the preparation path
NFR-04	Not measured	No timing instrumentation on the ECTS aggregation
NFR-05	Not measured	No uptime monitoring in the repository
NFR-06	Achieved	<code>hashService.ts</code> , <code>hashchainService.ts</code> , <code>wormService.ts</code> , integration test <code>wormCoupling</code>
NFR-07	Achieved	<code>approvalCeremony.ts</code> , <code>bulk_issuance_approvals</code> <code>UNIQUE(bulk_issuance_id, approver_id)</code>
NFR-08	Achieved	<code>caddy-compose.yml</code> , <code>step-ca</code> , <code>swiyu-tls-proxy/Caddyfile</code>
NFR-09	Achieved	<code>app.ts</code> <code>cookie.maxAge: 24*60*60*1000</code> ; the 24-hour absolute session lifetime is the deliberate catalogue value
NFR-10	Achieved	<code>audit_log</code> , hash chain, WORM mount, integration test
NFR-11	Partially achieved	<code>SETUP.md</code> end-to-end setup guide and <code>Makefile</code> <code>start</code> target; no measured setup-time evidence
NFR-12	Not measured	Demonstration on 2026-05-28 is stakeholder confirmation, not a usability study
NFR-13	Achieved	<code>SETUP.md</code> setup guide, <code>docs/api/openapi.yaml</code> , per-component READMEs, and this thesis as the accompanying documentation
NFR-14	Partially achieved	<code>credentialTypes.ts</code> registry; new type still requires a new builder
NFR-15	Achieved	<code>scripts/backup.sh</code> and <code>scripts/revive.sh</code> (<code>make backup</code> , <code>make revive</code>), documented in the repository README; the online backup runs without downtime, so a daily invocation meets the 24-hour recovery point objective
NFR-16	Achieved	<code>routes/events.ts</code> REST, JWT-signed inter-service authentication

Continued on next page

Continued from previous page

ID	Status	Evidence or rationale
NFR-17	Achieved	Compose stacks behind a single <code>make start</code> entry point; per-service <code>OIDC_REDIRECT_URI</code> as the only domain-specific setting
NFR-18	Achieved	<code>CREDENTIAL_VALIDITY_YEARS=10</code> ; the ten-year default is the catalogue value, with rationale and compensation in section 8.2.5 and chapter 9

Eleven requirements are achieved (NFR-02, NFR-06, NFR-07, NFR-08, NFR-09, NFR-10, NFR-13, NFR-15, NFR-16, NFR-17, NFR-18), two partially achieved (NFR-11, NFR-14), and five not measured (NFR-01, NFR-03, NFR-04, NFR-05, and NFR-12). Two of the achieved requirements, the session lifetime of NFR-09 and the credential validity of NFR-18, rest on design decisions that set the catalogue value to the implemented behaviour (section 11.6).

11.4. Verification and Test Results

Nine unit-test suites across the credential system and the mock service run on the Node.js built-in test runner [Nod]. Nine integration-test suites in the credential system run in a containerised test runner against a throwaway PostgreSQL database and a temporary directory for the audit-log storage. Section 10.4 documents the per-suite coverage. A GitLab continuous-integration pipeline runs the type check and the self-contained unit tests on every merge request and on every push to the default branch (appendix B.2); the integration suites stay outside the pipeline because they require a live database.

Manual walkthroughs on the demonstration deployment exercise the full lifecycle: diploma issuance, module issuance, verification, revocation, and suspension with restore. Further walkthroughs cover the master-data correction, the WebAuthn registration of an authenticator, the Caddy well-known insertion of the RFC 8414 endpoint, and the trust-registry validation with the identifier trust statement of Trust Protocol 2.0. The project documentation records all walkthroughs as successful as of 25 May 2026.

Three classes of verification remain out of scope. Performance and availability are not measured, apart from the diploma-batch issuance timing reported in section 11.3; no benchmark harness exists. No formal user-acceptance test with external study-administration or student testers took place; the demonstration on 28 May 2026 is a stakeholder validation, not a user-acceptance test. No per-functional-requirement test mapping table exists; the qualitative coverage statements per lifecycle phase substitute for it.

A judgement: the test evidence supports the claim that the credential lifecycle works end-to-end against the federal swiyu infrastructure. It does not support a production-readiness claim; a productive deployment requires the measurements, user-acceptance testing, and formalised test cases that Section 11.6 discusses.

11.5. Encountered Platform Limitations

Eight limitations of the swiyu Trust Infrastructure constrained the design space. They are factual findings that shaped the design choices of Chapter 8.

First, the swiyu wallet does not render the claims of a credential whose payload includes nested JSON objects, even though the swiyu Generic Issuer accepts and signs such a credential. The finding, confirmed in March 2026, forced the flat schema of Section 8.4.2. Second, the swiyu Generic Issuer accepts non-string claim values and does not validate the declared `value_type` of the credential metadata, but the wallet renders such credentials with empty display fields. This forced the string-only encoding of all claim values in Section 8.4.1.

Third, the wallet user interface does not surface the DIF Presentation Exchange `optional` marker that the swiyu Generic Verifier accepts; the wallet releases all requested claims without an explicit holder choice. This forced the verifier-controlled selective-disclosure design of Section 8.4.4, with one Presentation Definition per credential type. Fourth, the swiyu Generic Verifier 2.3.3 marks the Digital Credentials Query Language of OpenID for Verifiable Presentations 1.0 [Ope25b] as not implemented in its API documentation and rejects requests that use the language. The verifier configuration therefore retains the older DIF Presentation Exchange format. Fifth, the swiyu Generic Issuer auto-converts every claim of a credential to a selective-disclosure entry and does not expose the per-claim declaration of the SD-JWT Verifiable Credential profile [Int26]; the behaviour is visible in the issuer source code [Swi26c].

Sixth, the Swiss interoperability profile does not yet support the simultaneous verification of multiple credentials in a single transaction; the verifier therefore requests the diploma credential and the module credential in independent presentation transactions. Seventh, since its May 2026 release the swiyu wallet validates Trust Protocol 2.0 identifier trust statements. The Generic Issuer delivers the credential technically correctly, but the wallet aborts immediately after receipt when the issuer's identifier trust statement is not registered in the central trust registry; the trust-registry record must therefore exist before any productive issuance. Eighth, the swiyu Generic Issuer 2.4.3 silently sets the `mandatory` field of the `credential_metadata` block to `false` when the field is omitted, which forced the explicit `mandatory: true` declaration on every claim of every credential type in the issuer configuration.

11.6. What Was Not Achieved and Why

This section consolidates the items left incomplete at submission.

The HSM integration of the issuer signing key (FR-22) is descoped. The descoping matches the MVP scope of the work and the procurement timeline of the Securosys CloudHSM Sandbox documented in Section 8.10.5.

Two non-functional requirements reflect deliberate design decisions rather than gaps. The session cookie has a fixed twenty-four-hour absolute lifetime; the catalogue value of NFR-09 adopts this value as an MVP decision, and the WebAuthn step-up required for privileged actions partially compensates the residual risk. The credential validity is

ten years; the catalogue value of NFR-18 adopts this value per the design decision of Section 8.2.5, compensated by holder-key binding, revocation, and re-issuance.

The remaining performance and availability requirements are not measured: the module-batch issuance latency (NFR-01), the preparation latency (NFR-03), the ECTS aggregation latency (NFR-04), and the system availability (NFR-05). The same holds for the operator-usability requirement (NFR-12); only the diploma-batch issuance latency (NFR-02) has a single timing run, reported in section 11.3. The implementation includes no benchmark harness, load-test harness, or uptime monitoring, and the demonstration on 28 May 2026 was not designed as a usability study.

Three legal-track items remain open at submission: the Federal Act on Electronic Identity analysis, the cantonal data-protection-law analysis, and the formal data-protection review with the responsible person at OST, which is foreseen for a later pilot phase. The compliance section (Section 9.5) therefore states its legal-confirmation paragraphs as working assumptions pending that review; the items do not affect the MVP and do not block submission.

A judgement: the realised system is a Minimum Viable Product that covers the credential lifecycle end-to-end against the federal swiyu Trust Infrastructure. The open items at submission concern measurement, preparation for a productive deployment, and legal confirmation, not missing core flows.

12. Conclusion and Outlook

12.1. Conclusion

This thesis advanced the diploma proof-of-concept of the predecessor semester project [DLH25] to a Minimum Viable Product aimed at a productive institutional deployment at OST. The predecessor project had shown that issuing an OST diploma as an SD-JWT Verifiable Credential on the swiyu Generic Issuer is feasible, but it issued a single credential under a fixed schema and left the broader requirements of an institutional deployment open (chapter 7).

To meet these requirements, the thesis developed the architecture of chapter 8. The system consists of a credential system, a mock study-administration service, and a self-hosted swiyu Generic Issuer (section 8.2.1), which communicate over a documented event-driven REST interface (section 8.2.3). The self-hosted swiyu Generic Verifier and two in-house verifier web applications cover the verification side (section 8.3.4). All issuance runs through a unified batch abstraction: a batch becomes immutable after the first of two approvals, which realises the four-eyes principle, and a snapshot hash makes any later modification of the approved credential set detectable (section 8.2.4). Diploma and module credentials are issued separately with independent revocation paths (section 8.4.5). Administrative access is federated through Switch edu-ID and split into an information-technology administration role and a study-administration role (section 8.2.6).

Against the critical success criteria of table 2.1, the system meets the structural and functional criteria but not yet those that require a measurement campaign. The diploma-credential batch issuance timing was measured in a single run and met its target; the remaining performance and usability targets are not measured (section 11.3). A systematic measurement campaign is therefore the most urgent follow-up to this work.

The requirements catalogue of chapter 5 contains 25 functional and 18 non-functional requirements. Of the functional requirements, 24 are achieved and 1, the Hardware Security Module integration, is descoped. Of the non-functional requirements, 11 are achieved, 2 are partially achieved, and 5 are not measured (section 11.2, section 11.3). Two of the achieved requirements were realigned to the implemented behaviour as deliberate design decisions: the twenty-four-hour absolute session lifetime of NFR-09 and the ten-year credential validity of NFR-18 (section 8.2.5, chapter 9).

In the candidate's judgement, the realised system demonstrates the credential lifecycle end-to-end against the federal swiyu Trust Infrastructure and adds the structural and security mechanisms that distinguish a Minimum Viable Product from a proof of concept. The open items at submission concern measurement, productive readiness, and

legal confirmation. The thesis therefore meets the vision of section 2.1: an integration-oriented Minimum Viable Product on which the institutional rollout decision can be based, with the steps that a productive rollout still requires documented openly.

12.2. Outlook

Unlinkable credential issuance. The batch credential issuance mechanism of the Generic Issuer, refined in release 2.4.0, issues several instances of the same credential to a single holder, each bound to a different key, so that the holder can present a fresh instance to every verifier and avoid being correlated across presentations [Ope25a]. This thesis issues a single instance per credential and deliberately leaves the mechanism inactive for diplomas (chapter 4). Per-holder batch issuance would nonetheless strengthen unlinkability for a future credential type that is anonymous and frequently presented.

Combined credential presentation. Diploma and module credentials are issued and presented independently, because the public beta does not support presenting multiple credentials in a single OpenID4VP exchange (chapter 4). Once the platform supports combined presentation and verification, a verifier could request a diploma together with its underlying module credentials in one exchange. The credential-schema design anticipates this extension and accommodates it without architectural changes (section 8.4.5).

Software-based device binding. Software-based device binding is listed as unsupported in the public beta and scheduled for the productive go-live (chapter 4). Once available, it could bind credentials issued to holders without compatible secure hardware to the device in a way that survives wallet reinstallation, which would reduce the reliance on the re-issuance path adopted in this thesis.

Productive institutional deployment. Three steps separate the Minimum Viable Product from a productive deployment at OST, all documented as out of scope in section 1.4. The signing keys are currently held as software key material; their protection through a Hardware Security Module is specified as the should-have requirement FR-22 and deferred pending suitable hardware. An integration with the live institutional system would replace the mock study-administration service; the documented REST interface is designed to enable this. Finally, a legal and data-protection analysis under the Federal Act on Electronic Identity and the Federal Act on Data Protection is a prerequisite for issuing credentials to real students.

Validation of transferability. The system is designed for transferability (NFR-16, NFR-17). Deploying it at a second institution would validate these properties in practice.

Extended credential catalogue. The thesis implements diploma and module credentials and provides a template for additional credential schemas (section 8.4). Further academic credential types, for instance micro-credentials, could be added on this basis.

Verifier-side extension. This thesis covers the issuer side; the verifier of the predecessor project was carried forward and extended to verify both credential types (section 1.4). Building out the verifier side into institution-facing scenarios, such as the verification of an OST credential by an employer, is a natural continuation of the work.

Bibliography

- [Acc] Accredify. *Pilot Digital Education Certificate as a Verifiable Credential*. URL: <https://www.accredify.io/education/pilot-digital-education-certificate-as-a-verifiable-credential> (visited on 05/27/2026).
- [Bar20] Elaine Barker. *Recommendation for Key Management: Part 1 – General*. Tech. rep. NIST Special Publication 800-57 Part 1, Revision 5. National Institute of Standards and Technology, May 2020. URL: <https://doi.org/10.6028/NIST.SP.800-57pt1r5> (visited on 06/03/2026).
- [Cada] Caddy Project. *Automatic HTTPS*. URL: <https://caddyserver.com/docs/automatic-https> (visited on 06/03/2026).
- [Cadb] Caddy Project. *Caddyfile directive: tls*. URL: <https://caddyserver.com/docs/caddyfile/directives/tls> (visited on 06/12/2026).
- [Dec] Decentralized Identity Foundation. *The Did:webvh (Web plus Verifiable History) Method 1.0*. URL: <https://identity.foundation/didwebvh/> (visited on 05/27/2026).
- [DLH25] Simon Dietschi, Polina Lisetska, and Micha Harzbecker. “Swiss eID for University Diplomas”. Advisor: Daniel Tschudi. Semester project. OST Eastern Switzerland University of Applied Sciences, Department of Computer Science, Dec. 2025.
- [Dig] Digital Public Services Switzerland. *Establishing cross-authority digital identification*. URL: <https://www.digital-public-services-switzerland.ch/en/implementation/agenda-dvs/Establishing-cross-authority-digital-identification> (visited on 05/27/2026).
- [edu26] educa, Fachagentur für den digitalen Bildungsraum Schweiz. “Personal communication on the digital Matura certificate proof-of-concept”. Email correspondence, 11 March 2026. Mar. 2026.
- [edu] educa, Fachagentur für den digitalen Bildungsraum Schweiz. *DVS-Innovationsprojekt digitales Maturitätszeugnis*. URL: <https://www.educa.ch/de/themen/digitale-identitaeten/dvs-innovationsprojekt-digitales-maturitaetszeugnis> (visited on 05/27/2026).
- [EF22] Timon Erhart and Claudia Furrer. *Orientierungshilfe für die Dokumentation einer Studien- oder Bachelorarbeit*. Version 1.1. OST – Ostschweizer Fachhochschule, Studiengang Informatik. 2022.

- [Eur16] European Union. *Regulation (EU) 2016/679 (General Data Protection Regulation)*. 2016. URL: <https://eur-lex.europa.eu/eli/reg/2016/679/oj> (visited on 06/12/2026).
- [Exp24] Express Project. *Express 5.0 Release Announcement*. Oct. 2024. URL: <https://expressjs.com/2024/10/15/v5-release.html> (visited on 06/03/2026).
- [Fed26] Federal Department of Justice and Police (EJPD). *Akzeptanz der E-ID mit zusätzlichen Massnahmen stärken*. Media release, 25 February 2026. Feb. 2026. URL: <https://www.ejpd.admin.ch/de/newsb/L6Nbf7gStnDm> (visited on 06/12/2026).
- [Fed] Federal Statistical Office (Switzerland). *Schema Digitales Maturitätszeugnis*. I14Y Datenkatalog. URL: <https://www.i14y.admin.ch/de/catalog/datasets/SchemaDMZ/description> (visited on 05/27/2026).
- [FID] FIDO Alliance. *Client to Authenticator Protocol (CTAP) 2.1*. URL: <https://fidoalliance.org/specs/fido-v2.1-ps-20210615/fido-client-to-authenticator-protocol-v2.1-ps-20210615.html> (visited on 06/01/2026).
- [Int01] Internet Engineering Task Force. *Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)*. RFC 3161. Aug. 2001. URL: <https://www.rfc-editor.org/rfc/rfc3161> (visited on 06/03/2026).
- [Int15a] Internet Engineering Task Force. *JSON Web Key (JWK)*. RFC 7517. May 2015. URL: <https://www.rfc-editor.org/rfc/rfc7517> (visited on 06/01/2026).
- [Int15b] Internet Engineering Task Force. *JSON Web Signature (JWS)*. RFC 7515. May 2015. URL: <https://www.rfc-editor.org/rfc/rfc7515> (visited on 06/01/2026).
- [Int19] Internet Engineering Task Force. *Automatic Certificate Management Environment (ACME)*. RFC 8555. Mar. 2019. URL: <https://www.rfc-editor.org/rfc/rfc8555> (visited on 06/03/2026).
- [Int20] Internet Engineering Task Force. *JSON Canonicalization Scheme (JCS)*. RFC 8785. June 2020. URL: <https://www.rfc-editor.org/rfc/rfc8785> (visited on 06/01/2026).
- [Int25] Internet Engineering Task Force. *Selective Disclosure for JSON Web Tokens (SD-JWT)*. RFC 9901. Nov. 2025. URL: <https://www.rfc-editor.org/rfc/rfc9901> (visited on 05/27/2026).
- [Int26] Internet Engineering Task Force. *SD-JWT-based Verifiable Credentials (SD-JWT VC)*. Internet-Draft draft-ietf-oauth-sd-jwt-vc-16, IESG Publication Requested. Apr. 2026. URL: <https://datatracker.ietf.org/doc/draft-ietf-oauth-sd-jwt-vc/> (visited on 05/27/2026).
- [Int] Internet Engineering Task Force. *Token Status List*. Internet-Draft draft-ietf-oauth-status-list. URL: <https://datatracker.ietf.org/doc/draft-ietf-oauth-status-list/> (visited on 05/27/2026).

- [Joi12] Joint Task Force Transformation Initiative. *Guide for Conducting Risk Assessments*. Tech. rep. NIST Special Publication 800-30 Revision 1. National Institute of Standards and Technology, Sept. 2012. URL: <https://doi.org/10.6028/NIST.SP.800-30r1> (visited on 06/03/2026).
- [Nod26] Node.js Project. *Node.js Version 24 Release Notes*. 2026. URL: <https://nodejs.org/en/blog/release/v24.0.0> (visited on 06/03/2026).
- [Nod] Node.js Project. *Test runner module (node:test)*. URL: <https://nodejs.org/api/test.html> (visited on 06/03/2026).
- [Ope21] Open Web Application Security Project. *Application Security Verification Standard 4.0.3*. Oct. 2021. URL: <https://owasp.org/www-project-application-security-verification-standard/> (visited on 06/03/2026).
- [Ope25a] OpenID Foundation. *OpenID for Verifiable Credential Issuance 1.0*. Final. Sept. 2025. URL: https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html (visited on 05/27/2026).
- [Ope25b] OpenID Foundation. *OpenID for Verifiable Presentations 1.0*. Final. July 2025. URL: https://openid.net/specs/openid-4-verifiable-presentations-1_0.html (visited on 05/27/2026).
- [Ope] OpenID Foundation. *OpenID Connect Core 1.0*. URL: https://openid.net/specs/openid-connect-core-1_0.html (visited on 06/01/2026).
- [OST24] OST Eastern Switzerland University of Applied Sciences. “Diplomzeugnis (Degree Certificate), specimen”. Anonymised specimen degree certificate, Bachelor of Science OST in Betriebsökonomie; document on file with the author. 2024.
- [OST26] OST Eastern Switzerland University of Applied Sciences. “Personal communication on myOST and the academic-credential workflow”. Email correspondence with the OST study administration, February 2026. Feb. 2026.
- [OSTa] OST Eastern Switzerland University of Applied Sciences. *OST – University of Applied Sciences*. URL: <https://www.ost.ch/en/university-of-applied-sciences> (visited on 06/12/2026).
- [OSTb] OST Eastern Switzerland University of Applied Sciences, Department of Computer Science. *Studienordnung Bachelor of Science in Informatik*. URL: https://studien.ost.ch/allStudies/10246_I.html (visited on 05/27/2026).
- [Res22] Research and Education Federations. *eduPerson Object Class Specification (202208), version 4.4.0*. 2022. URL: [https://wiki.refeds.org/display/STAN/eduPerson+\(202208\)+v4.4.0](https://wiki.refeds.org/display/STAN/eduPerson+(202208)+v4.4.0) (visited on 06/11/2026).
- [Res] Research and Education Federations. *REFEDS Multi-factor authentication profile*. URL: <https://refeds.org/profile/mfa> (visited on 06/01/2026).
- [RS25] Stefan Richter and Mirko Stocker. *Leitfaden für Bachelor- und Studienarbeiten*. Version 1.4, valid from FS 2026. OST – Ostschweizer Fachhochschule, Studiengang Informatik. Dec. 2025.

- [Sec] Securosys SA. *CloudHSM compliance and certifications*. URL: <https://docs.securosys.com/cloudhsm/Overview/compliance/> (visited on 06/12/2026).
- [Sho14] Adam Shostack. *Threat Modeling: Designing for Security*. Wiley, 2014. ISBN: 978-1-118-80999-0.
- [Sim] SimpleWebAuthn Project. *SimpleWebAuthn server library*. URL: <https://simplewebauthn.dev> (visited on 06/03/2026).
- [Sma] Smallstep Labs, Inc. *step-ca Documentation*. URL: <https://smallstep.com/docs/step-ca/> (visited on 06/03/2026).
- [Swia] Swiss Confederation. *Federal Act on Data Protection (FADP / DSG)*. Systematic Compilation SR 235.1. URL: <https://www.fedlex.admin.ch/eli/cc/2022/491/en> (visited on 05/27/2026).
- [Swib] Swiss Confederation. *Federal Act on Funding and Coordination of the Swiss Higher Education Sector (HEdA / HFKG)*. Systematic Compilation SR 414.20. URL: <https://www.fedlex.admin.ch/eli/cc/2014/691/de> (visited on 06/12/2026).
- [Swic] Swiss Confederation. *Public Beta of the swiyu Trust Infrastructure*. URL: <https://www.eid.admin.ch/en/public-beta-e> (visited on 06/12/2026).
- [Swid] Swiss Confederation, e-ID Program. *Swiss Profile*. Interoperability profile of the swiyu Trust Infrastructure. URL: <https://github.com/e-id-admin/open-source-community/blob/main/tech-roadmap/swiss-profile.md> (visited on 06/12/2026).
- [Swi26a] Swiss Federal Office of Information Technology. *swiyu Android wallet release v1.14.0*. May 2026. URL: <https://github.com/swiyu-admin-ch/eidch-android-wallet/releases/tag/v1.14.0> (visited on 06/12/2026).
- [Swi26b] Swiss Federal Office of Information Technology. *swiyu-issuer releases*. Release notes, versions 2.2.0 (November 2025) to 3.1.1 (May 2026). 2026. URL: <https://github.com/swiyu-admin-ch/swiyu-issuer/releases> (visited on 06/12/2026).
- [Swi26c] Swiss Federal Office of Information Technology. *swiyu-issuer source code, version 3.1.1*. Selective-disclosure handling in class `SdJwtCredential`. 2026. URL: <https://github.com/swiyu-admin-ch/swiyu-issuer/tree/3.1.1> (visited on 06/12/2026).
- [Swi26d] Swiss Federal Office of Information Technology. *swiyu-verifier release 2.3.3*. Mar. 2026. URL: <https://github.com/swiyu-admin-ch/swiyu-verifier/releases/tag/2.3.3> (visited on 05/27/2026).
- [Swie] Swiss Federal Office of Information Technology. *Breaking changes in next wallet versions*. swiyu Public Beta documentation. URL: <https://swiyu-admin-ch.github.io/publicbeta/breaking-changes-in-next-wallet-versions/> (visited on 05/27/2026).

- [Swif] Swiss Federal Office of Information Technology. *Swiss Profile for the swiyu Public Beta Trust Infrastructure*. Archived interoperability profile. URL: <https://swiyu-admin-ch.github.io/archive/interoperability-profile/> (visited on 06/12/2026).
- [Swig] Swiss Federal Office of Information Technology. *Swiyu Trust Infrastructure – Roadmap*. URL: <https://swiyu-admin-ch.github.io/roadmap/> (visited on 02/21/2026).
- [Swi19] Swiss Higher Education Council (Hochschulrat). *Verordnung über die Koordination der Lehre an den Schweizer Hochschulen*. Systematic Compilation SR 414.205.1. 2019. URL: <https://www.fedlex.admin.ch/eli/cc/2019/722/de> (visited on 06/12/2026).
- [swi14] swissuniversities, Kammer Fachhochschulen. *Das Diploma Supplement*. Best-practice report. 2014. URL: https://www.swissuniversities.ch/fileadmin/swissuniversities/Dokumente/Kammern/Kammer_FH/Best_practice/Bericht_Diploma_Supplement_2014.pdf (visited on 06/12/2026).
- [SWI26] SWITCH. *Attribute Specification for the Switch edu-ID Federation*. Mar. 2026. URL: https://help.switch.ch/aai/docs/Switch_edu-ID_Attribute_Specification.pdf (visited on 06/11/2026).
- [SWI] SWITCH. *Switch edu-ID documentation: OpenID Connect tokens*. URL: <https://help.switch.ch/eduid/docs/services/openid-connect/tokens/> (visited on 06/12/2026).
- [tim] ti&m. *Success Story: BFH IPST*. URL: <https://www.ti8m.com/en/success-stories/public-and-e-government/bfh-ipst> (visited on 05/27/2026).
- [Wora] World Wide Web Consortium. *Verifiable Credentials Data Model 2.0*. URL: <https://www.w3.org/TR/vc-data-model-2.0/> (visited on 05/27/2026).
- [Worb] World Wide Web Consortium. *Web Authentication: An API for accessing Public Key Credentials, Level 3*. URL: <https://www.w3.org/TR/webauthn-3/> (visited on 06/01/2026).
- [Yub] Yubico. *YubiKey 5.7 firmware specifics*. URL: <https://docs.yubico.com/hardware/yubikey/yk-tech-manual/5.7-firmware-specifics.html> (visited on 06/12/2026).

A. Project and Time Management

A.1. Project Approach

The thesis is conducted as a solo bachelor's-thesis project over the fourteen weeks of the spring semester 2026, between calendar week eight and calendar week twenty-four, followed by an additional full-time block week before submission, as documented in section 3.1. The project follows an agile-influenced process with weekly delivery checkpoints rather than a fixed waterfall plan. The work is structured into thematic sprints of one to two weeks each that follow the chapter structure of the present report, namely an analysis sprint that fixes the requirements and the system context, a design sprint that produces the architecture and the security concept, several implementation sprints that realise the lifecycle flows and the supporting infrastructure, a security sprint that introduces the four-eyes approval and the audit log, and a documentation sprint that produces the present report.

The project is tracked in a dedicated Jira project, identified by the key DCO (Digital Credentials OST). Confluence is used as the design-decision log: every architectural decision of the thesis is documented as a Confluence page with a status field and a date of code reconciliation, so that the relationship between the design intent and the implemented state is traceable across the lifetime of the project. The implementation work itself is kept in a single Git repository at the OST GitLab instance, with the repository structure documented in section 10.2.

The supervisor of the thesis is Daniel Tschudi, and the external expert is André Clerc. The weekly supervision meeting is held every Thursday over the fourteen weeks of the semester. A stakeholder demonstration with the OST study administration, represented by Damaris Hübner, was conducted on 28 May 2026 (Jira ticket DCO-85).

A.2. Project Plan

The project plan is maintained in Jira in the form of an Advanced Roadmap that lists the epics, the stories, the tasks, and the milestones of the project over the fourteen-week period. The plan is updated continuously as work progresses; the original plan and the effective execution are compared at the level of the closed-ticket count per week and at the level of the milestone dates. The principal milestones of the project are summarised in table A.1; the corresponding Gantt-style overview of the Jira Advanced Roadmap is reproduced in figure A.1.

Table A.1.: Principal project milestones.

Date	Milestone
2026-02-16	Start of the thesis (start of the spring semester)
2026-02-26	Kick-off meeting with the supervisor
2026-04-23	Intermediate presentation with supervisor and expert
2026-05-25	Internal Public Key Infrastructure activation (DCO-99)
2026-05-28	Stakeholder demonstration with the OST study administration (DCO-85)
2026-06-12	Submission of the thesis report (calendar week 24)
Calendar week 26 to 35	Bachelor examination with supervisor and expert

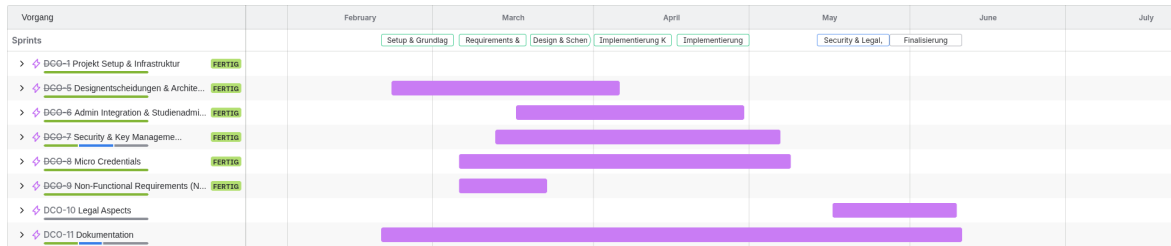


Figure A.1.: Gantt-style overview of the project from the Jira Advanced Roadmap.

A.3. Quality Assurance

The quality-assurance practice of the project rests on three layers, in line with the testing approach documented in section 10.4. The first layer is the automated test suite of the credential system and the mock service, executed through the Node.js built-in test runner across the unit and the integration levels. The second layer is the manual walkthrough of the principal lifecycle flows on the demonstration deployment, performed before every demonstration and after every substantial code change. The third layer is the design-decision reconciliation against the code, which the project documents on the corresponding Confluence pages as the *Code-Abgleich* note with the date of the reconciliation.

The definition of *done* that the project applies to a feature comprises four criteria: the code is merged on the main branch, the corresponding migration of the database schema is added if applicable, an automated test exists at the appropriate level, and the design-decision page on Confluence is updated with the resulting design state. The fourth criterion is the project-specific extension over a typical software-engineering definition of done; it ensures that the design documentation does not silently diverge from the implementation, an experience drawn from the predecessor semester project where the documentation lagged the implementation by several iterations.

The project operates a GitLab continuous-integration pipeline on the repository: every merge request and every push to the default branch runs the TypeScript type check and the self-contained unit tests (appendix B.2). The end-to-end validation against the swiyu Trust Infrastructure stays outside the pipeline because it depends on the central registries and the official wallet; the integration tests are executed locally through

the containerised test runner, and the manual walkthroughs serve as the end-to-end verification.

A.4. Risk Analysis

The risk register of the project is maintained from the start of the thesis and is updated continuously. Table A.2 summarises the risks that the project identified and the mitigations that were applied. The risk register adopts the categorisation of likelihood (low, medium, high) and impact (low, medium, high) of NIST SP 800-30 [Joi12]. Figure A.2 positions the five risks on the resulting risk matrix using the identifiers R1 to R5 of the register; the cell colour encodes the risk level that follows from the combination of likelihood and impact, from low (green) to critical (red).

Table A.2.: Project risk register.

ID	Risk	Class.	Mitigation
R1	Instability of the swiyu public beta during development	Medium / Medium	Pinning of the issuer and verifier image versions; cross-check of every release note against the implementation; weekly check of the swiyu roadmap.
R2	Sparse upstream documentation of the swiyu reference containers	Medium / Medium	Test-driven exploration of the API surface, captured in dedicated Confluence test-result pages.
R3	Solo-work continuity risk due to absence (illness, hardware failure)	Medium / High	Daily commit cadence; off-site backup of the repository; documented build-from-scratch procedure on the operational interface.
R4	Procurement delay of the Hardware Security Module hardware	Medium / Low	Documented software-key fallback; HSM integration descoped to a should-have requirement that does not block submission.
R5	Time slippage on the documentation phase relative to the implementation phase	Medium / High	Continuous documentation discipline: every closed design ticket triggers the corresponding documentation update in the same sprint.

None of the risks of the register materialised in a form that threatened the submission of the thesis. The procurement delay of the Hardware Security Module (R4) materialised as the descope documented in section 11.6, in line with the mitigation in the risk register.

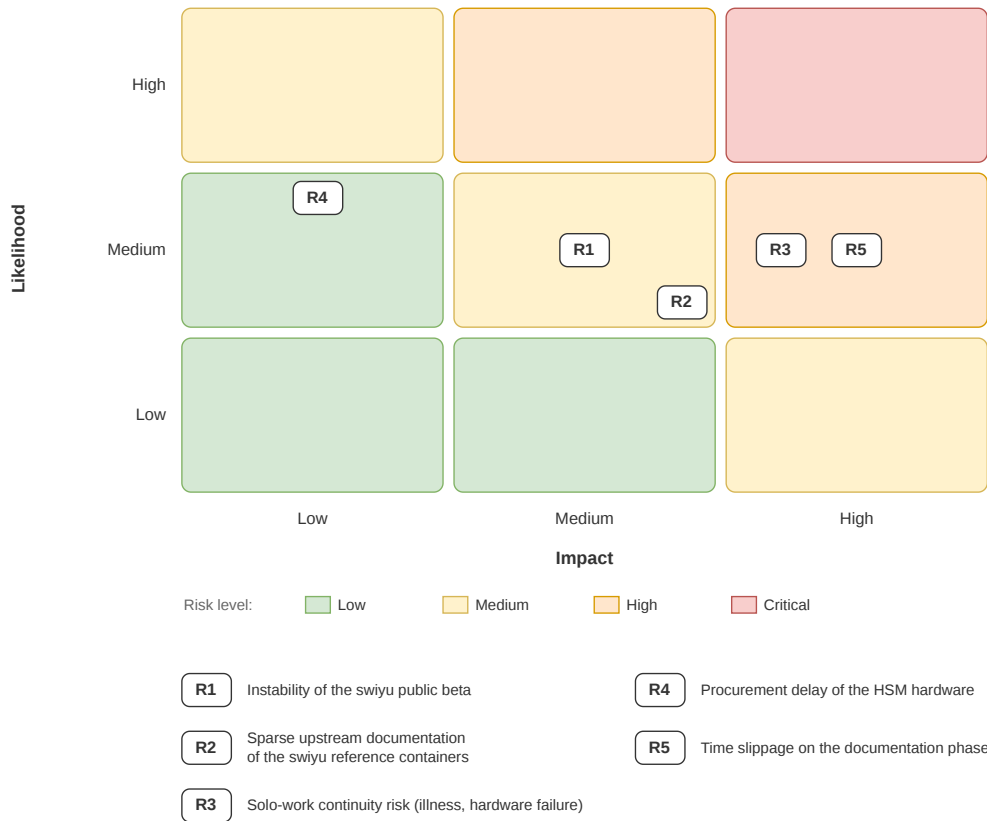


Figure A.2.: Risk matrix of the project risk register.

A.5. Time Tracking

The work budget for the bachelor’s thesis is 360 hours, corresponding to 12 ECTS credits at a conversion factor of 30 hours per credit [RS25] (see section 3.1). The reference distribution that the programme guideline gives, comprising approximately 23 hours per week over the fourteen semester weeks and 40 hours in the additional block week, is the target distribution that the project pursued.

The work was tracked through the Clockwork Lite Jira application, which records a per-Jira-issue worklog with the start time and the duration of each work session. The recorded period extends from 16 February 2026 to 12 June 2026 and comprises 121 worklogs that sum to 390 hours, approximately eight per cent above the reference budget of 360 hours. The distribution across the months of the project is shown in table A.3, and the distribution across the project weeks is shown in figure A.3. The per-issue breakdown is intentionally not reproduced, because in the later phase of the thesis the candidate increasingly recorded the documentation work as bulk entries on the documentation epic rather than as fine-grained per-task worklogs; the monthly and weekly aggregates remain meaningful even where the per-issue allocation lost its granularity.

Table A.3.: Recorded work hours per month.

Month	Hours
February 2026	24.25
March 2026	79.50
April 2026	94.07
May 2026	95.25
June 2026	96.75
Total	389.82

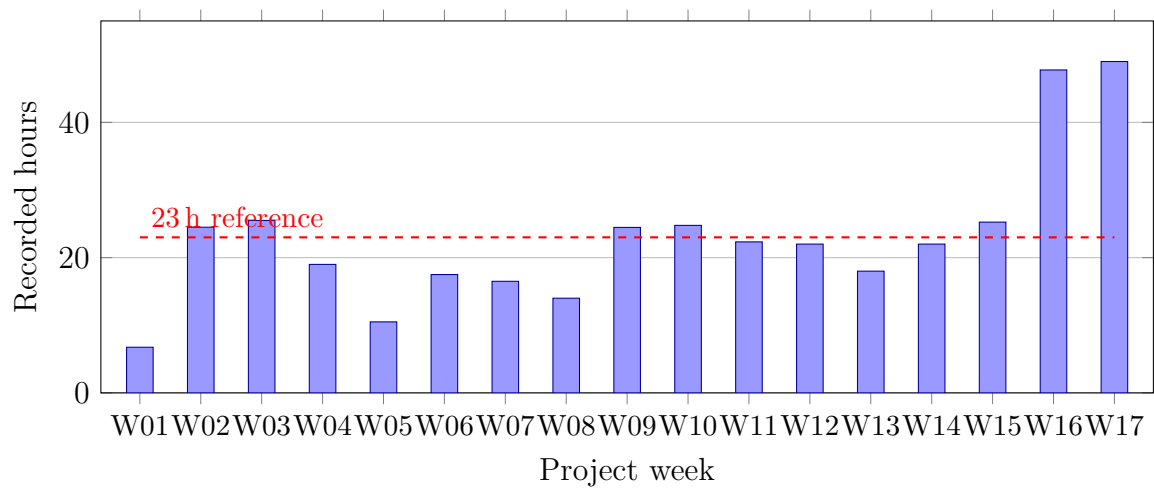


Figure A.3.: Recorded work hours per project week.

B. Operational Notes

B.1. Tools Used and Disclosure of AI Tools

The thesis follows the OST departmental guideline on the use of AI tools in written work, which permits and encourages the use of modern tools including AI tools both for the produced artefacts (code, diagrams) and for the report, and recommends a summary-level disclosure by task area rather than a per-chapter or per-paragraph annotation. The corresponding tool inventory of the present thesis is given in table B.1.

Table B.1.: Tool inventory by task area, per the OST departmental guideline on the use of AI tools.

Task area	Tools
Literature research and reference management	Google Scholar, IEEE Xplore, the federal Fedlex portal, the swiyu admin documentation, Claude
Idea generation and design exploration	Claude, draw.io
Translation and language polish	DeepL, Claude
Diagram authoring	Mermaid (in Confluence), draw.io
Coding	Visual Studio Code, Claude Code
Report drafting, language polish, and structural review	Claude (including the Claude Code agent harness for the integrated review of source code and report text), DeepL, LanguageTool
Collaboration and project management	Jira, Confluence, Microsoft Teams, Outlook
DevOps	Docker, Docker Compose, GitLab (including GitLab CI), smallstep step-ca, Caddy, GitHub Actions (for the swiyu reference repositories only)

The candidate retains responsibility for the quality of the produced artefacts, in line with the departmental guideline. In particular, the candidate has verified that the use of AI tools did not introduce any factually incorrect statement that the candidate

could not separately substantiate, that confidential information and personal data were not exposed through the cloud-based AI services, that no text passages constitute a plagiarism, and that all material claims of the report are supported by an explicit reference to a primary source.

B.2. Continuous Integration and Continuous Delivery

The repository of the credential system, the mock service, and the supporting infrastructure includes a GitLab continuous-integration pipeline. On every merge request and on every push to the default branch, the pipeline enforces the existing quality gates: the TypeScript type check of the credential system and the self-contained unit tests of both in-house services. The integration-test suites stay outside the pipeline because they require a live database; they run locally through the containerised test runner documented in section 10.4. The end-to-end validation against the swiyu Trust Infrastructure likewise remains a manual walkthrough, because it depends on the central registries and the official wallet. The remaining exclusions of the pipeline are documented in section 10.6.

Continuous delivery is deliberately out of scope: the composition targets a single host, and the build and the deployment are performed there through the Makefile and the Docker Compose tooling, as documented in section 10.5.

The deployment of the composition follows a single-step procedure: the target host is prepared with Docker and Docker Compose, the per-service environment files are populated with the configuration values that the operational interface documents, and the composition is started through the `make start` target. The first-run validation procedure that issues, retrieves, verifies, and suspends a test credential is the operational acceptance test of a fresh deployment. The validation procedure is documented in the Confluence *Implementierungsanleitung* of the project and is summarised in section 10.5.

B.3. Installation Guide

The installation of the composition on a target host follows the seven-step procedure that the setup guide at the repository root (`SETUP.md`) documents end to end, from a fresh clone to a credential-issuing system. The principal steps comprise the creation of the per-service environment files from their templates, the registration with the central swiyu registries (the creation of the DID through the swiyu DID Toolbox, which requires the Java Runtime Environment version 21, and the collection of the bootstrap refresh token), the generation of the inter-service signing keys through the `scripts/generate-service-keys.js` helper, the population of every environment field, the start of the external WORM audit-storage volume, the start of the composition through the `make start` target, and the first-run application setup that creates the first administrator account and the status lists and enrolls the issuing officers.

The pre-conditions on the target host are an installation of Docker and Docker Compose, an operating system that supports the Linux extended-attribute `chattr +a`

on the file system that hosts the audit-log volume (any common Linux distribution satisfies this), a public domain name with the corresponding DNS records that point to the host, and the ability of the host to reach the public Let's Encrypt and the central swiyu endpoints over the TCP ports 80 and 443. The full step-by-step procedure is part of the repository (`SETUP.md`); the recovery procedures, including the status-list recovery and the internal Public Key Infrastructure recovery, are persisted in the project documentation on Confluence.

The backup-and-restore procedure is likewise part of the repository and documented in its README: the `make backup` target takes an online backup without downtime (a `pg_dump` of all four databases together with archives of the non-regenerable volumes and of the environment files), and the `make revive` target restores a selected backup in one step.

B.4. User Guide

The day-to-day operation of the credential system through the two operator roles is exercised through the administrative interface of the credential system. The IT-administrator role provisions new operators through the invitation flow documented in section 8.6.3 and rotates the operator authentication of the lost or compromised hardware authenticators. The study-administration role triggers the issuance of credentials through the four-eyes approval ceremony documented in section 8.6.4, manages the revocation, the suspension, and the restoration of credentials through the corresponding routes of the administrative interface, and reviews the audit log on demand. The student-facing interface of the mock service permits the retrieval of credential offers through the QR-code-based handoff to the official Swiss wallet and the initiation of a re-issuance request after a wallet loss.

The principal walkthroughs of the operator interface, together with the screenshots that accompany them in the live deployment, are documented in the Confluence space of the project under the operational documentation subtree. The screenshots are not reproduced in the present appendix because the visual identity of the interface is expected to evolve during the productive rollout of the system, and a screenshot reproduction in a printed appendix would bind the report to a specific point in time of the visual identity.

C. Supplementary Diagrams and Screenshots

This appendix collects large-format diagrams and supplementary screenshots that are referenced from the main text. Figure C.1 shows the end-to-end issuance collaboration described in section 8.5.1, depicted for the diploma credential.



Figure C.1.: End-to-end BPMN 2.0 collaboration diagram of the diploma issuance (rotated for legibility).

C.1. System Walkthrough

The following screenshots complement the end-to-end evidence of section 11.1 with the operator and portal interfaces of the in-house services. All screenshots show synthetic test data; the user-interface language is German, matching the OST user base.

Figure C.2 shows the landing page of the credential system, the operator backend of section 8.3.2; the Switch edu-ID login is the only entry point. The operator dashboard and the credential list are shown in figure 11.1, the approval ceremony in figure 10.2.

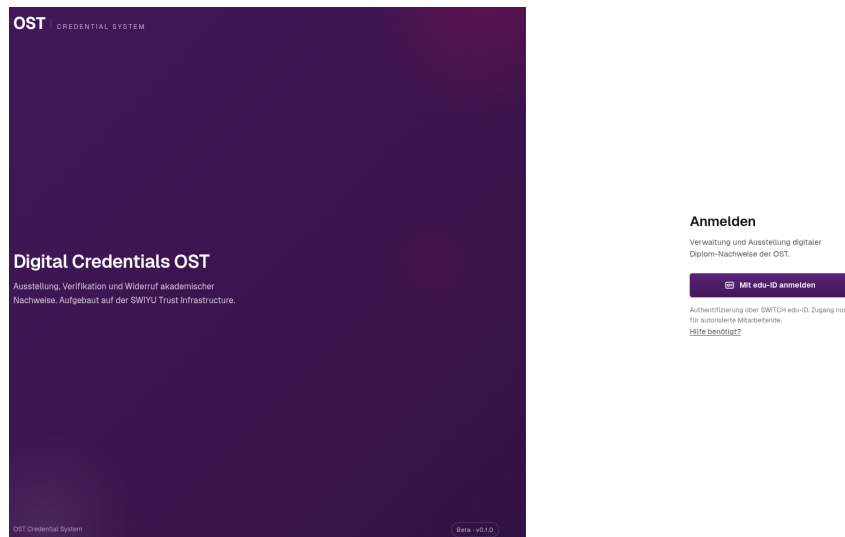
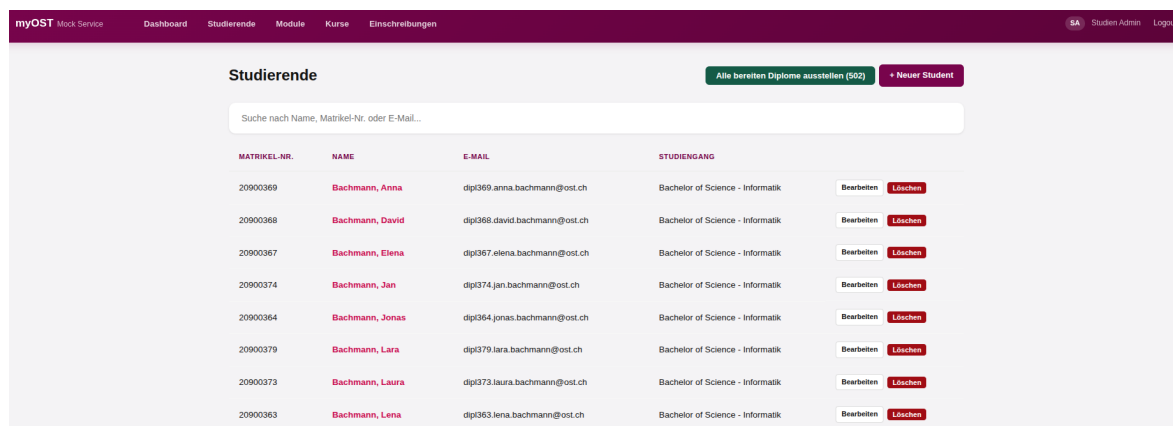


Figure C.2.: Landing page of the credential system with the edu-ID login.

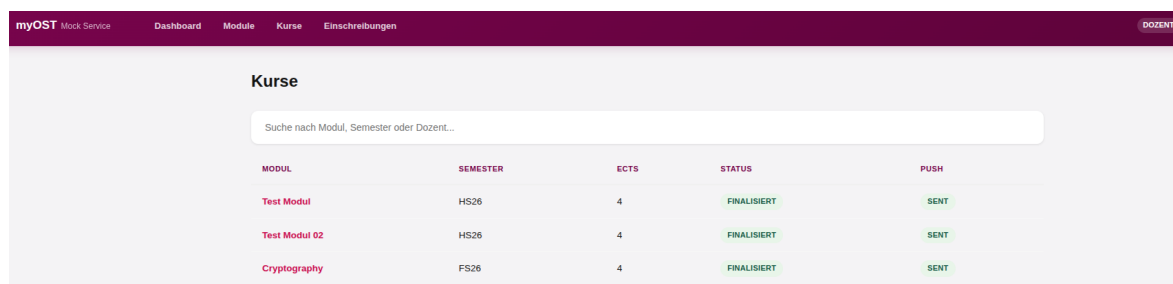
Figure C.3 shows the myOST mock service, the stand-in for the study-administration system. The study-administration view lists the student master data; its bulk action pushes the ready diplomas to the credential system as a single diploma-ready event, where the issuance itself only starts after the second four-eyes approval (section 8.5.1). The lecturer view finalises courses, which pushes the corresponding completion events to the credential system. The student self-service page lists the credential offers of the signed-in student and carries the wallet-loss button that starts the re-issuance flow of section 8.5.3.

Figure C.4 shows the verifier web application: the public request form, on which a verifying party chooses the credential type and enters its contact data, and the administrative view, on which the stored verification record with the disclosed claims is reviewed.

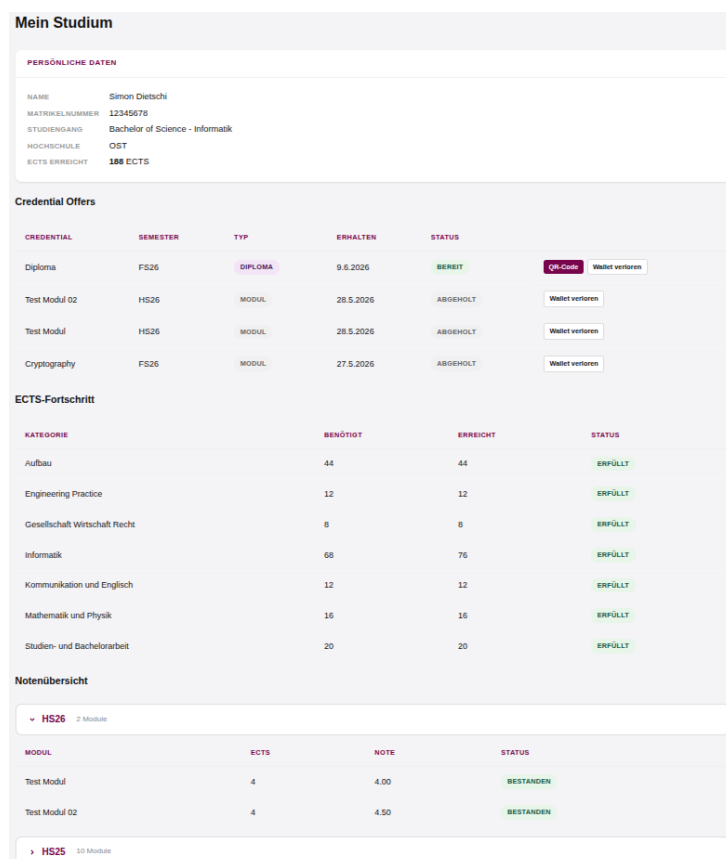
Figure C.5 completes the credential-type coverage: the module credential of section 8.4 as it appears in the swiyu wallet.



(a) Study-administration view with the diploma-ready push action.

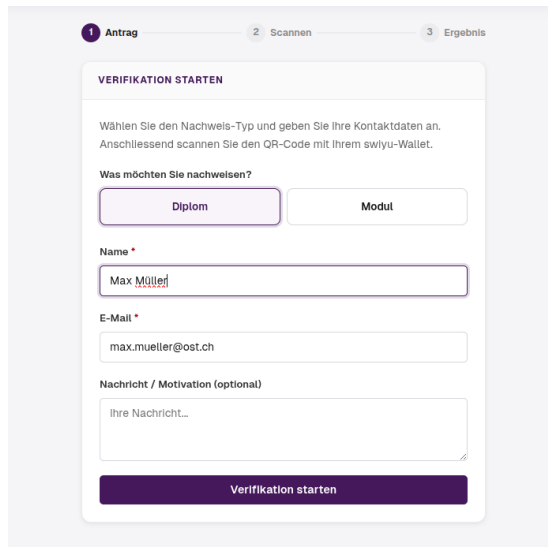


(b) Lecturer view with finalised courses.



(c) Student self-service page with credential offers.

Figure C.3.: The myOST mock service in its three role-specific views.



1 Antrag 2 Scannen 3 Ergebnis

VERIFIKATION STARTEN

Wählen Sie den Nachweis-Typ und geben Sie Ihre Kontaktdaten an. Anschließend scannen Sie den QR-Code mit Ihrem swiyu-Wallet.

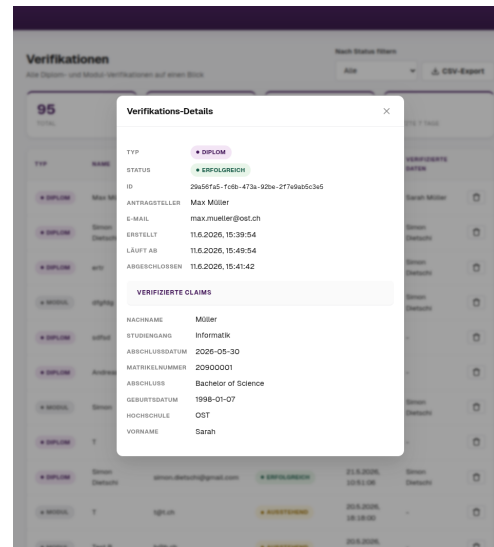
Was möchten Sie nachweisen?

Name *

E-Mail *

Nachricht / Motivation (optional)

(a) Public request form.



Verifikationen

Alle Diplom- und Modul-verifikationen auf einen Blick

95

Verifikations-Details

TYP: **DIPLOM**

STATUS: **ERFOLGREICH**

ID: 29a56f45-10b-473a-92b4-2f76a05c365

ANTRAGSTELLER: Max Müller

E-MAIL: max.mueller@ost.ch

ERSTELLT: 11.6.2026, 15:39:54

LÄUFT AB: 11.6.2026, 15:49:54

ABGESCHLOSSEN: 11.6.2026, 15:41:42

VERIFIZIERTE CLAIMS

NACHNAME	Müller
STUDIENGANG	Informatik
ABSCHLUSSDATUM	2026-05-30
MATRIKELNUMMER	20900001
ABSCHLUSS	Bachelor of Science
GEBURTSDATUM	1998-01-07
HOCHSCHULE	OST
VORNAME	Sarah

(b) Administrative verification record.

Figure C.4.: Public and administrative views of the verifier web application.

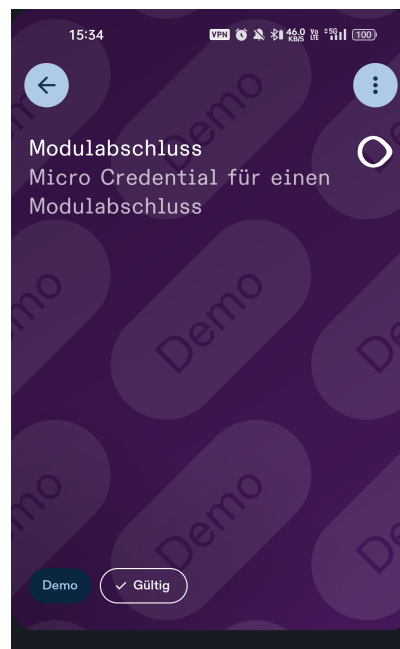


Figure C.5.: Module credential in the swiyu wallet.