

UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Ingegneria dell'Informazione, Elettrica e Matematica Applicata



Project work APS 2024-2025

**Progettazione di un Sistema Decentralizzato per la Gestione di
Credenziali Accademiche Digitali nel Contesto Erasmus**

The Binary Team



A cura di:

Gerardo Catapano: 0622702576

Francesco Pio Gargiulo: 0622702546

Indice

Introduzione	5
Work Package 1	6
1.1 Attori Onesti e Funzionalità del Sistema	6
1.2 Interazione Tra Attori Onesti	7
1.2.1 Definizione Formale di Correttezza	8
1.2.2 Definizione Formale di Completezza	8
1.3 Threat Model	9
1.3.1 Malicious Student	9
1.3.2 Compromised University	10
1.3.3 Network-Based Attacker (MITM)	12
1.3.4 Protocol-Based Attacker	12
1.3.5 Student Software-Hardware Compromise	13
1.4 Proprietà del Sistema	14
1.4.1 Confidenzialità (C)	14
1.4.2 Integrità (I)	15
1.4.3 Disponibilità (D)	15
1.4.4 Non ripudio (NP)	15
1.4.5 Autenticazione	15
1.4.6 Autenticità (A)	16
1.4.7 Efficienza (E)	16
1.4.8 Trasparenza (T)	16
1.5 Struttura della Credenziale	16
Work Package 2	18
2.1 Panoramica Generale del Sistema	18
2.1.1 Introduzione alla Blockchain Permissioned	18
2.2 Componenti del Sistema	19
2.2.1 Decentralized Identifiers (DID)	19
2.2.2 Scelta del Metodo did:web	21
2.2.3 Tipologie di Verifiable Credentials	21
2.2.4 Struttura della EligibilityErasmusCredential	22
2.2.5 Struttura della AcademicCredential	26
2.3 Pre-Game	28

2.3.1	Studente	29
2.3.2	Università di origine	30
2.4	Scambio delle credenziali accademiche	32
2.4.1	Interazione con l'università ospitante	32
2.4.2	Emissione e Gestione dell'Academic Credential	32
2.4.3	Verifica Decentralizzata tramite Blockchain	33
2.4.4	Presentazione selettiva e verifica	34
2.4.5	Revoca basata su ConsortiumRevocationRegistry2024	38
2.4.6	Architettura del Registry	38
2.4.7	Vantaggi rispetto a Revocation List	39
2.4.8	Inserimento della Credenziale	39
2.4.9	Verifica da parte di un'università	40
2.4.10	Revoca di una credenziale	40
Work Package 3		42
3.1	Confidenzialità	42
3.2	Integrità	42
3.3	Trasparenza	43
3.4	Efficienza	43
3.5	Non ripudio (NP)	44
3.6	Autenticazione	45
3.7	Autenticità (A)	45
3.8	Compromissione Hardware-Software	46
Work Package 4		47
4.1	University	47
4.1.1	BaseUniversity	47
4.1.2	UniversitySalerno	47
4.1.3	UniversityRennes	48
4.2	Student	48
4.3	Gestione Sicura degli Utenti	49
4.3.1	DatabaseManager	49
4.3.2	UserManager	50
4.3.3	PasswordManager	50
4.4	Blockchain Simulata	51
4.4.1	Classe Block	51
4.4.2	Classe Blockchain	51
4.5	Registro di Revoca	52
4.5.1	Struttura del Registro	52
4.5.2	Funzionalità principali	52

4.6	Merkle Tree	53
4.6.1	Calcolo della Merkle Root	53
4.6.2	Hash dei nodi foglia	53
4.6.3	Costruzione delle Merkle Proof	53
4.6.4	Verifica delle Merkle Proof	53
4.6.5	Ricostruzione della Merkle Root	54
4.7	Main	54
4.8	Analisi delle Prestazioni	57
4.9	Aree di Miglioramento	61
4.10	Versione 1.1	61

Introduzione

Nel contesto della trasformazione digitale, cresce l'esigenza di rilasciare, detenere e verificare attestazioni digitali relative a fatti, diritti, qualifiche o stati personali. Queste attestazioni, note come *credenziali digitali*, svolgono un ruolo chiave nel garantire l'integrità, l'affidabilità e la trasparenza nei processi di scambio di informazioni tra individui ed enti. Questa necessità si manifesta in modo particolarmente evidente nel settore dell'istruzione, soprattutto in contesti internazionali come il programma *Erasmus+*, dove studenti, università e organismi accademici (es. enti di accreditamento o la Commissione Europea) richiedono strumenti sicuri per condividere, verificare e, se necessario, revocare attestazioni accademiche (esami superati, titoli conseguiti, frequenze certificate, ecc.).

Le soluzioni attualmente impiegate si basano spesso su architetture centralizzate, con limitata interoperabilità e un controllo ridotto da parte dell'utente sulla propria privacy. Inoltre, questi sistemi raramente offrono una gestione efficace della revoca delle credenziali o un controllo granulare delle informazioni da condividere. Per affrontare queste criticità, si sta diffondendo un nuovo approccio alla gestione dell'identità digitale ispirato ai principi della *Self-Sovereign Identity* (SSI), che pone l'individuo al centro del controllo delle proprie informazioni.

In questo lavoro, ci si rifà a questo paradigma adottando una versione semplificata, focalizzata sui concetti di *Verifiable Credentials* (VC), *Verifiable Presentations* (VP) e Decentralized Identity (DID). L'obiettivo del presente *Project Work* è la progettazione di un sistema decentralizzato per la gestione delle credenziali accademiche digitali, con particolare attenzione alla selettività nella presentazione (*selective disclosure*) e alla revoca controllata, garantendo sicurezza, autonomia e trasparenza tra i diversi attori coinvolti.

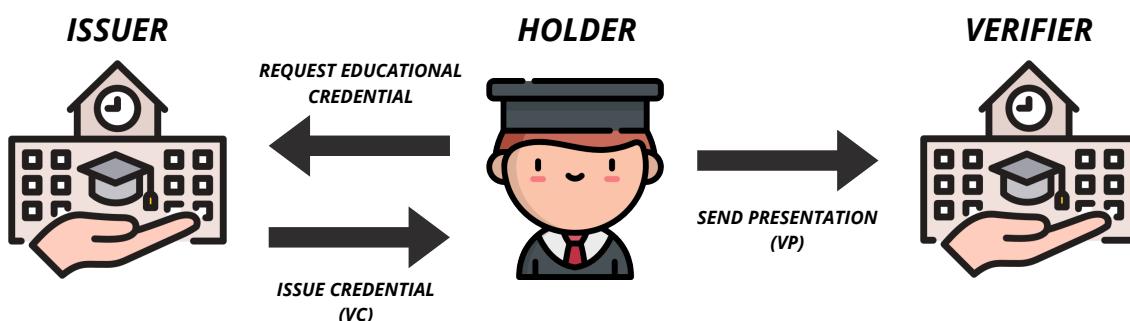


Figura 1: Schema introduttivo del sistema analizzato

Work Package 1

1.1 Attori Onesti e Funzionalità del Sistema

Il sistema proposto si fonda su quattro attori¹ principali:



Exchange Student (Holder)

È l'entità che riceve, possiede e controlla le proprie credenziali accademiche, potendole presentare a terze parti quando necessario. È autosovrano della propria identità, gestendo autonomamente le proprie informazioni senza dipendere da intermediari.



Università ospitante (Issuer)

È l'entità che genera e rilascia la credenziale digitale accademica. L'emittente è considerato affidabile, se accreditato, poiché attesta formalmente delle informazioni.



Università d'origine (Verifier)

È l'entità che riceve da parte degli studenti le credenziali e ne valuta la validità ai fini del riconoscimento delle attività svolte durante il periodo all'estero.



Consorzio Universitario Europeo (Trust Network)

È garante del corretto funzionamento del sistema attraverso un consorzio di università europee che stabilisce le regole comuni. Supervisiona l'ammissione di nuove università e ne verifica l'idoneità all'emissione di credenziali. Inoltre, definisce e mantiene lo standard condiviso delle credenziali accademiche, garantendo interoperabilità, coerenza semantica e verificabilità tra le università aderenti.

¹ Nel contesto dell'implementazione, si assume che i dispositivi utilizzati dagli attori onesti siano sicuri, ovvero privi di compromissioni hardware o software. Tuttavia, tali casistiche sono pienamente riconosciute e analizzate all'interno del threat model, in quanto rappresentano scenari rilevanti per la valutazione complessiva della sicurezza del sistema.

Oltre ai quattro attori principali che costituiscono il cuore operativo del sistema, è opportuno considerare due ulteriori entità che, pur avendo ruoli non centrali nella routine quotidiana del processo di verifica e rilascio delle credenziali, possono intervenire in momenti critici o fornire supporto tecnico.

1. Autorità Giudiziaria (Justice)

In scenari eccezionali, come nel caso di falsificazione di credenziali accademiche, violazioni gravi della privacy, o comportamenti fraudolenti, può rendersi necessario il coinvolgimento delle autorità giudiziarie competenti. Tale attore non partecipa direttamente al flusso standard delle credenziali, ma può disporre la revoca forzata di chiavi o credenziali oppure ordinare verifiche forensi sul comportamento degli attori coinvolti.

2. Team Di Sviluppo (The Binary Team)

Nel proof of concept, The Binary Team rappresenta l'entità tecnica responsabile della progettazione e implementazione del sistema. Pur non avendo alcun ruolo nelle decisioni accademiche o regolatorie, il team svolge una funzione essenziale nel garantire l'affidabilità e la sicurezza dell'infrastruttura. È considerato un attore neutrale e fidato, in quanto adotta le best practices per assicurare un alto livello di sicurezza. Tuttavia, una cattiva implementazione o la presenza involontaria di vulnerabilità nel codice potrebbe esporre il sistema a superfici d'attacco non previste a livello teorico.

1.2 Interazione Tra Attori Onesti

Assumendo che tutti gli attori del sistema si comportino in modo corretto e rispettino il protocollo previsto, e che le proprietà di correttezza e completezza siano garantite, è possibile descrivere l'interazione ideale tra le componenti del sistema. L'interazione si articola nei seguenti passaggi fondamentali:

0. Pre-Game: Uno studente manifesta il proprio interesse a partecipare a un'iniziativa accademica, come il programma Erasmus, candidandosi secondo le modalità previste dal bando dell'università. L'università di origine esamina la candidatura, verifica il possesso dei requisiti previsti e, se ritenuto idoneo, autentica lo studente e ne convalida l'ammissibilità al programma, fornendo evidenze verificabili a supporto.
1. Richiesta di attestazione: Al termine dell'esperienza formativa, lo studente (Holder), desideroso di dimostrare il possesso di determinati requisiti o esperienze, si rivolge all'università ospitante per ottenere un'attestazione formale (credenziale) relativa a uno o più fatti verificabili.
2. Emissione della credenziale: L'ente (Issuer), dopo aver verificato la richiesta e i dati forniti, emette una credenziale che attesta le informazioni rilevanti e la associa in modo

do inequivocabile al richiedente. La credenziale è strutturata in modo da poter essere successivamente verificata da terze parti.

3. Conservazione e gestione della credenziale: Il soggetto attestato (Holder) conserva la credenziale e ne mantiene il controllo. Può decidere quando e con chi condividerla, eventualmente presentandone solo una parte o un sottoinsieme coerente con la richiesta di un servizio.
4. Presentazione e verifica: Quando un altro attore del sistema (Verifier) richiede una prova di possesso delle informazioni attestabili, il soggetto (Holder) costruisce una presentazione basata sulla credenziale ricevuta. Il Verifier applica i controlli previsti dal protocollo e sotto ipotesi di completezza e correttezza, tali controlli hanno sempre esito positivo, e la presentazione viene accettata.
5. Accesso al servizio: Il Verifier accetta la presentazione e consente l'accesso al servizio o beneficio previsto, sotto le medesime ipotesi.

1.2.1 Definizione Formale di Correttezza

Un sistema di gestione delle Verifiable Credentials è corretto se nessun attore malevolo (Issuer, Holder o Verifier) può ottenere l'accettazione di una presentazione fraudolenta, non conforme al protocollo, o costruita con credenziali false, modificate, revocate o non autorizzate. In termini formali, sia:

I: Issuer accreditato,

H: Holder,

V: Verifier,

VC: Verifiable credential emessa da *I* e detenuta da *H*,

VP: Verifiable presentation costruita da *H* a partire da *VC*,

P: Protocollo standard del sistema.

Allora, il sistema è corretto se:

$$\forall(I, H, V) \text{ disonesto}, \quad \forall VC, VP \quad \text{t.c.} \quad \neg(Valid(VC, P) \wedge Conform(VP, P)) \implies V \models \text{Reject}(VP)$$

Dove il simbolo \models indica una tautologia, ossia qualcosa di sempre vero.

1.2.2 Definizione Formale di Completezza

Un sistema di gestione delle Verifiable Credentials è completo se, assumendo che tutti gli attori (Issuer, Holder, Verifier) si comportino onestamente e i canali di comunicazione siano sicuri, ogni presentazione correttamente generata da un Holder, con credenziali valide emesse da un Issuer accreditato, è accettata dal Verifier e il servizio richiesto viene erogato correttamente. In termini formali, sia:

I: Issuer accreditato,

H: Holder,

V: Verifier,

VC: Verifiable credential valida emessa da *I* e detenuta da *H*,

VP: Verifiable presentation costruita correttamente da *H* a partire da *VC*,

P: Protocollo standard del sistema.

Allora, il sistema è completo se:

$$\forall(I, H, V) \text{ onesti}, \quad \forall VC, VP \quad \text{t.c.} \quad Valid(VC, P) \wedge Conform(VP, P) \implies V \models \text{Accept}(VP)$$

Dove il simbolo \models indica una tautologia, ossia qualcosa di sempre vero.

1.3 Threat Model

Un sistema decentralizzato di gestione delle credenziali accademiche, come quello proposto, deve essere progettato tenendo conto di una molteplicità di minacce provenienti da attori interni ed esterni con risorse e motivazioni diverse. Le minacce vengono qui classificate per attore coinvolto e tipo di attacco (attivo o passivo). Per ciascuna minaccia sono specificate le proprietà che il sistema deve garantire anche in presenza dell'attacco (proprietà di resilienza).

1.3.1 Malicious Student

Gli studenti sono i principali utilizzatori delle credenziali accademiche digitali, ma rappresentano anche una potenziale fonte di minacce per l'integrità, la validità e la riservatezza del sistema.

Active Threats

- *The Forger* è lo studente stesso che tenta deliberatamente di alterare le proprie credenziali accademiche per ottenere vantaggi indebiti. Questo attore dispone di accesso legittimo al proprio profilo nei sistemi universitari e possiede competenze informatiche sufficienti per comprendere e manipolare la struttura delle credenziali. Cerca di presentare risultati accademici mai conseguiti, voti alterati, corsi non frequentati o credenziali interamente false, dunque deve disporre di risorse sufficienti per manipolare i dati accademici. Il sistema deve resistere a questo tipo di attacco garantendo che ogni credenziale sia verificabile come autentica, che non possa essere modificata in modo non rilevabile e che sia sempre riconducibile a un'emissione legittima.
- *RevocationBypasser* è lo studente che tenta di utilizzare una credenziale accademica non più valida, ad esempio perché revocata o scaduta. Questo tipo di attacco è possibile soprattutto quando il sistema non impone la verifica dello stato della credenziale come parte obbligatoria del processo di validazione. Poiché la validità delle Verifiable Credentials

può cambiare nel tempo (ad esempio in seguito alla perdita di privilegi o alla scadenza), è fondamentale che il sistema garantisca sempre l’accessibilità e l’integrità delle informazioni sullo stato della credenziale. Ogni eventuale modifica ai metadati deve essere rilevabile e l’uso di una credenziale revocata non deve mai poter passare inosservato.

- *CredentialSwapper* sono due studenti che possono colludere per condividere o trasferirsi reciprocamente Verifiable Presentations con l’intento di aggirare controlli d’identità, limiti temporali o condizioni di accesso. Questo tipo di attacco sfrutta l’assenza di un legame forte tra la presentazione e l’identità del titolare (credential binding), permettendo che una VP firmata venga riutilizzata da un altro individuo senza che il sistema ne rilevi l’anomalia. Il trasferimento avviene attraverso la condivisione di file o l’accesso al wallet compromesso. Le risorse richieste sono minime, essendo sufficiente la cooperazione tra due utenti e l’accesso al mezzo di trasmissione. Per contrastare questa minaccia, il sistema deve garantire autenticità e non trasferibilità.
- *OverCuriousHolder* è uno studente legittimo che tenta di accedere a porzioni della propria Verifiable Credential non destinate alla sua visione, come ad esempio note interne dell’istituzione. Sebbene questo tipo di attacco non richieda infrastrutture esterne, presuppone un buon livello di competenza tecnica, soprattutto per sfruttare funzionalità di ispezione del wallet. In questo modo, il principio di separazione tra dati pubblici e privati viene violato, compromettendo la riservatezza end-to-end tra Issuer e Verifier. Per prevenire tali scenari, è essenziale che le credenziali adottino meccanismi crittografici adeguati, garantendo che i contenuti riservati rimangano opachi e inaccessibili all’Holder, pur mantenendo la verificabilità da parte dei Verifier autorizzati.

Passive Threats

- *The Big Brother* è uno studente o un altro attore passivo che tenta di acquisire informazioni sensibili intercettando le comunicazioni tra l’università e lo studente o tra lo studente e il verificatore. Pur non modificando i messaggi, sfrutta la possibilità di accedere a reti condivise o poco sicure (ad esempio reti Wi-Fi aperte), senza necessità di elevate capacità computazionali ma con l’ausilio di strumenti di intercettazione del traffico. L’obiettivo è ottenere accesso non autorizzato a contenuti delle credenziali o ad altri dati accademici. Il sistema deve garantire la riservatezza delle comunicazioni, proteggendo sia i contenuti delle credenziali che i metadati eventualmente trasmessi, e assicurare che nessuna informazione sensibile possa essere dedotta o riutilizzata da un osservatore passivo.

1.3.2 Compromised University

La compromissione dell’università come entità emittente o verificatore può avvenire in modo ”silente”, senza che l’università stessa ne abbia consapevolezza. In questi casi, l’attacco è invisibile

e non può essere mitigato a priori, sia dal lato del verificatore che da quello dell'emittitore delle credenziali. Questo rende particolarmente difficile rilevare e prevenire tale compromissione, poiché non sono visibili segni di alterazione o manipolazione del sistema.

Active Threats

- *GhostIssuer* è un membro del personale universitario con accesso ai sistemi di emissione delle credenziali utilizza le proprie autorizzazioni per rilasciare una Verifiable Credential a favore di uno studente che non ne ha diritto. Anche senza competenze tecniche avanzate, l'insider agisce utilizzando gli strumenti ufficiali di emissione messi a disposizione dall'università, firmando digitalmente e registrando una credenziale fraudolenta che risulta indistinguibile da quelle valide. Questo attacco mina la fiducia nel sistema, compromettendo la proprietà di autenticità: il verificatore non ha modo di distinguere tra una VC genuina e una emessa abusivamente da un'autorità compromessa.
- *MaliciousVerifier* è un'università nel ruolo di Verifier che abusa della sua posizione per raccogliere informazioni eccedenti riguardo uno studente. Ad esempio, un ufficio Erasmus che richiede l'intera cronologia accademica dello studente anziché solo i dati relativi a un determinato corso, violando il principio di selective disclosure. In questo scenario, l'università sfrutta le proprie capacità computazionali e infrastrutturali, come l'accesso privilegiato ai database accademici e agli strumenti di verifica automatica, per profilare gli studenti, raccogliendo informazioni non strettamente necessarie. L'uso delle credenziali in modo indiscriminato mette a rischio la riservatezza e la privacy degli studenti, poiché i dati raccolti potrebbero essere usati per scopi non autorizzati o inadeguati.

Passive Threats

- *CrossPresentationLinker* è un attaccante passivo, tipicamente un verificatore o un insieme di verificatori collusi, tenta di correlare presentazioni effettuate dallo stesso studente in contesti separati, sfruttando la presenza di identificatori persistenti o metadati comuni all'interno delle Verifiable Presentations. Anche in assenza di dati identificativi esplicativi, la ripetizione di strutture, timestamp, nonce non randomizzati o correlatori impliciti può permettere la costruzione di profili comportamentali o l'identificazione indiretta del soggetto. Questo attacco è reso possibile dalla mancanza di misure di unlinkability nel wallet o nella piattaforma. L'attacco richiede risorse computazionali per l'analisi e il tracciamento di pattern su larga scala, specialmente in presenza di molteplici verificatori che condividono dati. Il sistema, per essere resiliente, deve garantire l'unlinkability tra presentazioni distinte, proteggendo l'identità dello studente anche in presenza di tentativi di correlazione basati su metadati impliciti o strutturali.

1.3.3 Network-Based Attacker (MITM)

- *MITMAttacker - (Issuer, Holder)*: Un attaccante potrebbe intercettare e manomettere la comunicazione tra un Issuer legittimo e un Holder, ottenendo potenzialmente accesso al contenuto delle Verifiable Credentials trasmesse o alterando le informazioni durante la trasmissione. L’attaccante può iniettare credenziali false o modificate, sostituendo una VC autentica con una manipolata. Questo attacco minaccia le proprietà di autenticità (credenziali false potrebbero essere accettate come valide), integrità (le credenziali ricevute potrebbero essere alterate rispetto a quelle originali) e confidenzialità (informazioni sensibili potrebbero essere esposte durante l’intercettazione). L’attaccante richiede capacità computazionali moderate e competenze tecniche per manipolare il traffico di rete e sfruttare vulnerabilità nei protocolli di comunicazione.
- *MITMAttacker - (Holder, Verifier)*: Un avversario potrebbe intercettare una Verifiable Presentation e riutilizzarla, impersonando il legittimo Holder e ottenendo accesso non autorizzato (es. accedere ai fondi Erasmus di un altro studente). Alternativamente, l’attaccante potrebbe intercettare e alterare la VP durante la trasmissione tra Holder e Verifier. Questo attacco compromette l’autenticità (il Verifier non può essere certo dell’identità del presentatore), l’integrità (la presentazione potrebbe essere alterata) e la confidenzialità (dati sensibili potrebbero essere intercettati). L’attaccante necessita di capacità computazionali basse-moderate ma richiede conoscenze avanzate di protocolli di rete e strumenti di intercettazione del traffico.

1.3.4 Protocol-Based Attacker

Una Verifiable Credential può includere numerosi claim relativi all’utente, tuttavia, se la selective disclosure non è implementata correttamente o viene ignorata nella fase di presentazione, l’utente si trova nella condizione di dover condividere l’intero contenuto della credenziale anche quando il servizio richiedente necessita solo di un’informazione specifica, come la conferma dell’iscrizione.

Questa mancanza di granularità espone l’utente a un’eccessiva condivisione di dati personali. Un fornitore di servizi malizioso può approfittarne per raccogliere informazioni aggiuntive non necessarie, con l’obiettivo di trarne vantaggio attraverso la profilazione o la rivendita a terze parti. Anche in assenza di intento malevolo, un verificatore onesto ma poco attento può comunque contribuire a un incremento del rischio di violazioni della privacy, semplicemente trattando e conservando più dati del necessario.

La minaccia non richiede capacità computazionali avanzate da parte dell’attaccante, ma si manifesta attraverso le debolezze del protocollo o un’interfaccia utente poco trasparente. In particolare, i wallet che non mostrano chiaramente all’utente quali claim verranno effettivamente

condivisi durante una presentazione aggravano ulteriormente il problema. Le proprietà di sicurezza compromesse includono la minimizzazione dei dati, la privacy dell’utente e la sua capacità di esercitare controllo consapevole sulle proprie informazioni.

1.3.5 Student Software-Hardware Compromise

La sicurezza di un sistema basato su credenziali digitali dipende in larga parte dall’integrità del *wallet* e dalla protezione della chiave privata associata all’identità dello studente. Tuttavia, questo presupposto può venire meno in contesti reali, dove gli studenti operano su dispositivi potenzialmente compromessi a livello software o hardware. La compromissione del *client* introduce nuove superfici d’attacco che si collocano al di fuori del perimetro di sicurezza tradizionale.

Un dispositivo infetto da *malware* può comportarsi come un *client fantoccio*, eseguendo operazioni sensibili a insaputa del titolare. Ad esempio, software malevolo integrato nel sistema operativo, oppure *wallet* contraffatti che replicano l’interfaccia di quelli legittimi, possono intercettare, archiviare e trasmettere *Verifiable Presentations* firmate. Tali presentazioni possono quindi essere riutilizzate da un attaccante in contesti diversi da quello originale, senza il consenso dell’utente. Questo tipo di attacco è noto come *replay attack*, e si verifica quando una presentazione legittima viene ripresentata ad un verificatore in un secondo momento o in un contesto diverso da quello previsto, al fine di ottenere un accesso non autorizzato. Sebbene qui venga descritto all’interno di uno scenario di compromissione del *client*, è importante sottolineare che il *replay attack* può avvenire anche in assenza di *malware* o furti informatici.

In scenari ancora più gravi, l’attaccante può riuscire a ottenere direttamente la chiave privata dello studente. Ciò può avvenire tramite esfiltrazione da dispositivi compromessi, ma anche a seguito della perdita fisica di un supporto contenente la chiave (ad esempio una *smartcard*, una chiavetta USB), o della duplicazione non autorizzata durante operazioni apparentemente sicure. In mancanza di un ambiente hardware sicuro che protegga la chiave dall’accesso diretto, un attaccante può utilizzarla per impersonare completamente l’identità dello studente, generando presentazioni arbitrarie e accedendo a servizi in modo fraudolento. Un’ulteriore minaccia è rappresentata dai *wallet* apparentemente affidabili ma in realtà malevoli. Questi possono registrare ogni attività dell’utente, incluse le presentazioni emesse e i contesti di verifica, consentendo non solo la raccolta di dati personali, ma anche la costruzione di profili comportamentali e la riproduzione automatizzata delle interazioni, senza che l’utente ne sia consapevole.

1.4 Proprietà del Sistema

Per garantire il corretto funzionamento del sistema di gestione delle credenziali digitali, è necessario che l'architettura e i meccanismi adottati assicurino alcune proprietà fondamentali di sicurezza. Queste proprietà rappresentano obiettivi di alto livello, che devono essere mantenuti indipendentemente dalle tecnologie, dagli standard o dalle soluzioni implementative adottate.

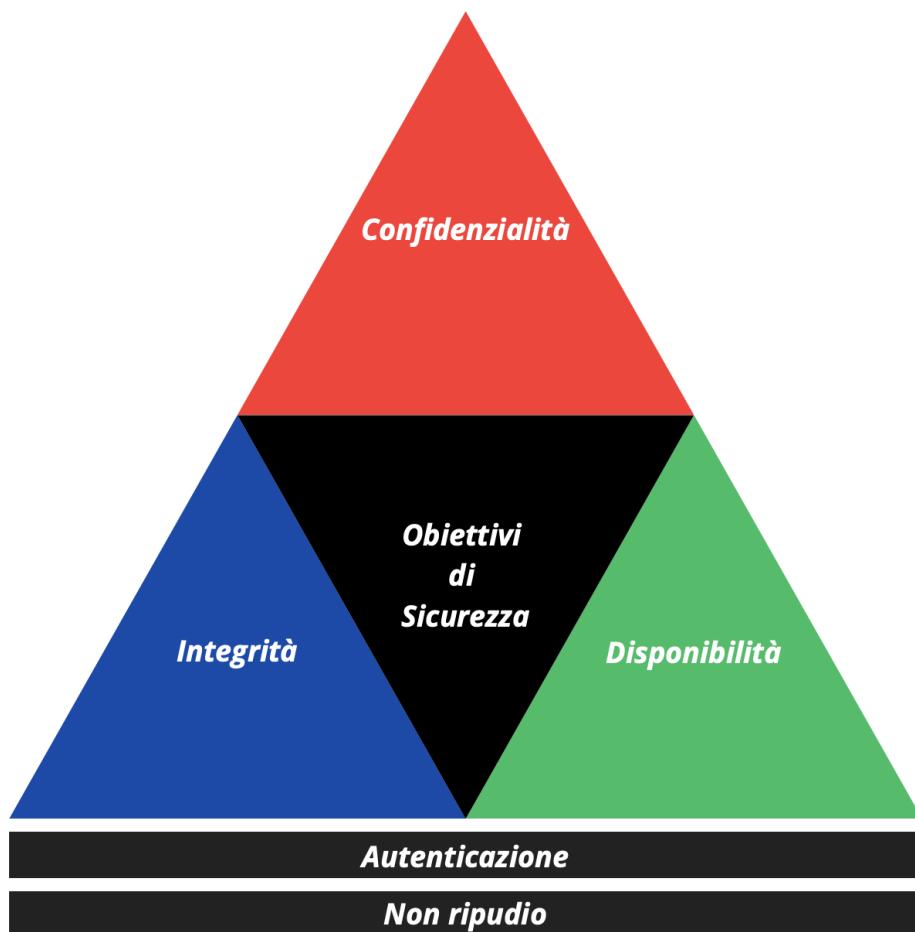


Figura 1.2: Triade CIA estesa

1.4.1 Confidenzialità (C)

La confidenzialità garantisce che solo gli attori autorizzati possano accedere alle informazioni contenute nelle credenziali digitali. Nel nostro contesto, ciò significa che:

- **C.1** I dati contenuti nelle credenziali devono essere accessibili solo a soggetti autorizzati, e limitatamente alle informazioni strettamente necessarie per il ruolo di ciascun attore;
- **C.2** Le comunicazioni tra Holder e Issuer devono avvenire tramite canali sicuri, che impediscono l'intercettazione o inferenza di informazioni sensibili, comprese quelle usate per la generazione delle credenziali;

- **C.3** Anche le comunicazioni tra Holder e università di origine devono avvenire in modo sicuro, impedendo la divulgazione non autorizzata dei dati dell’utente;

1.4.2 Integrità (I)

L’integrità assicura che le credenziali non vengano modificate in maniera non autorizzata.

- **I.1** Ogni credenziale deve restare integra, non alterata né danneggiata dopo l’emissione, sia a riposo che in trasito;
- **I.2** I dati esibiti in una presentazione devono essere parte coerente e verificabile della credenziale dello studente;
- **I.3** Il meccanismo di revoca deve essere resistente a manipolazioni, garantire tracciabilità e la sua validità deve essere verificabile in ogni momento.

1.4.3 Disponibilità (D)

La disponibilità garantisce che le funzionalità essenziali del sistema siano accessibili in modo affidabile e continuativo agli attori autorizzati.

- **D.1** Gli attori devono poter accedere alle proprie credenziali e utilizzarle anche in scenari offline o a bassa connettività;
- **D.2** I servizi di verifica e revoca devono essere resistenti a guasti e attacchi DoS (Denial of Service);
- **D.3** Il sistema deve prevedere meccanismi di backup o recupero delle credenziali in caso di perdita del wallet.

1.4.4 Non ripudio (NP)

Il non ripudio garantisce che un attore non possa negare di aver eseguito una determinata azione. In questo caso:

- **NP.1** Un’università che emette una credenziale non può negare in seguito di averla emessa;
- **NP.2** Un holder non può negare di aver presentato una certa credenziale;
- **NP.3** Qualsiasi attore in gioco non può negare di aver presentato la sua identità.

1.4.5 Autenticazione

L’autorizzazione garantisce che ogni attore possa eseguire solo le operazioni per cui è autorizzato, nel rispetto dei ruoli e dei permessi assegnati.

- **AU.1** Solo le istituzioni accademiche autorizzate possono emettere credenziali valide nel sistema
- **AU.2** L'accesso ai registri di revoca deve essere controllato e limitato agli attori autorizzati

1.4.6 Autenticità (A)

L'autenticità permette di accertare l'identità dell'attore che ha emesso una determinata informazione. Nello specifico:

- **A.1** Deve essere sempre possibile verificare che una credenziale sia stata emessa da un'istituzione accademica legittima;
- **A.2** La firma dell'issuer sulla credenziale deve essere crittograficamente verificabile e legata a un'identità certificata e pubblicamente riconosciuta nel sistema.

1.4.7 Efficienza (E)

L'efficienza assicura che il sistema possa essere utilizzato in modo pratico, con costi computazionali e temporali sostenibili per tutti gli attori coinvolti, senza compromettere la sicurezza.

- **E.1** Le operazioni crittografiche devono essere eseguibili in tempi compatibili con l'uso quotidiano e l'adozione su larga scala;
- **E.2** L'emissione, il trasferimento e la verifica delle credenziali devono essere ottimizzati per l'utilizzo su dispositivi con risorse limitate;

1.4.8 Trasparenza (T)

La trasparenza assicura che i meccanismi di funzionamento del sistema siano comprensibili, verificabili e auditabili da parte degli attori legittimi, in modo da rafforzare la fiducia nel sistema stesso.

- **T.1** Gli algoritmi crittografici, i protocolli di comunicazione e i criteri di validazione devono essere documentati e accessibili pubblicamente.

1.5 Struttura della Credenziale

Nel contesto della mobilità studentesca internazionale, come il programma Erasmus+, è fondamentale adottare soluzioni che garantiscano un'elevata **interoperabilità** tra i diversi sistemi delle istituzioni accademiche coinvolte. Per questo motivo, ci siamo ispirati alla struttura concettuale proposta dallo **standard W3C** per le Verifiable Credentials².

² Lo standard W3C definisce un modello per rappresentare attestazioni digitali in modo verificabile, sicuro e decentralizzato, supportando scenari di interoperabilità tra attori eterogenei.

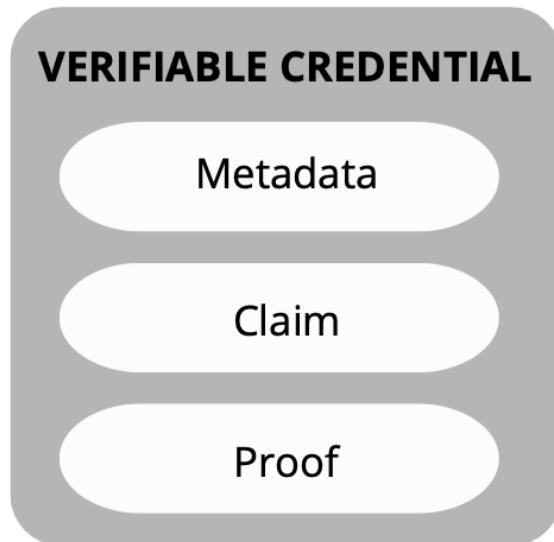


Figura 1.3: Macro struttura della Credenziale

Abbiamo quindi progettato una struttura personalizzata di credenziale accademica che estende il vocabolario definito dallo standard W3C, adottando una nostra versione del contesto semantico per supportare in modo più mirato le esigenze legate alle attestazioni accademiche. Pur non aderendo pienamente allo standard, la nostra credenziale mantiene i principi fondamentali di sicurezza, modularità e decentralizzazione. La credenziale è organizzata in tre sezioni principali:

- **Metadati generali**, contenenti informazioni strutturali essenziali per l'identificazione e l'interoperabilità della credenziale, tra cui: `id` (identificativo univoco), `type` (tipologia della credenziale), `issuer` (emittente), `issuanceDate` (data di emissione), e `@context` (contesto semantico per l'interpretazione condivisa dei dati).
- **Claim**, ovvero l'insieme delle informazioni attestanti, racchiuse in `credentialSubject`. In questo caso, tali informazioni descrivono il titolare (studente) e i suoi risultati accademici in modo strutturato, ad esempio: nome, cognome, identificativo universitario, corso frequentato, attività svolte e risultati ottenuti.
- **Proof**, che costituisce la prova crittografica dell'autenticità e integrità della credenziale. Comprende elementi quali il metodo di firma (`type`), la data di creazione della firma (`created`), lo scopo della prova (`proofPurpose`) e la firma digitale in formato JWS (`jws`).

La scelta di una macrostruttura, piuttosto che di un design dettagliato della credenziale, è motivata dalla natura ancora astratta del progetto in questa fase (WP1), in cui non sono stati ancora definiti i dettagli implementativi. Tale approccio consente di mantenere una visione generale e flessibile, individuando comunque i campi essenziali e universalmente riconosciuti, funzionali a garantire l'interoperabilità futura del sistema. Gli esempi di campi inclusi riflettono pertanto la necessità di predisporre un impianto concettualmente solido, ma tecnicamente estendibile.

Work Package 2

2.1 Panoramica Generale del Sistema

Il sistema proposto implementa un’architettura decentralizzata per la gestione delle credenziali accademiche digitali nel contesto della mobilità studentesca internazionale, con particolare focus sui programmi Erasmus+. In questo capitolo ci concentreremo sulla presentazione di una soluzione che risponde al modello identificato nel Work Package 1. L’obiettivo è quello di proporre un sistema che riesca a raggiungere un ragionevole compromesso tra efficienza, trasparenza, confidenzialità e sicurezza.

2.1.1 Introduzione alla Blockchain Permissioned

Pur essendo possibile concepire un sistema completamente decentralizzato basato su una rete federata di università, in cui ogni ateneo gestisce autonomamente il proprio database locale per credenziali e dati studenti e partecipa alla rete blockchain mantenendo un proprio nodo per il consenso distribuito, tale approccio presenta criticità fondamentali legate al problema della fiducia inter-istituzionale.

In un sistema completamente aperto, qualsiasi università potrebbe unirsi alla rete blockchain e partecipare al consenso senza controlli preliminari sulla qualità o legittimità dell’istituzione. Questo crea il paradosso per cui, per garantire la massima decentralizzazione, si rinuncia ai meccanismi che potrebbero assicurare la credibilità delle credenziali ancorate sulla blockchain. Senza criteri condivisi di accreditamento, il sistema rischierebbe una ”corsa verso il basso” nella qualità, dove credenziali di valore si mescolerebbero con titoli privi di significato accademico registrati da istituzioni non verificate.

Per questa ragione, il protocollo adotta una blockchain permissioned gestita da un consorzio di università accreditate, che costituisce un livello di fiducia condiviso tra tutte le istituzioni partecipanti. Solo le università che soddisfano specifici criteri di qualità e sono ammesse nel consorzio possono gestire nodi della rete blockchain e partecipare al consenso distribuito. Questo approccio permette di garantire immutabilità, tracciabilità e verificabilità delle credenziali ancorate, mantenendo un equilibrio ottimale tra decentralizzazione operativa e controllo qualitativo. Il modello consortile evita sia la complessità di una rete completamente aperta che la centralizzazione di un’autorità unica, risolvendo a monte il problema della fiducia attraverso l’accreditamento preventivo dei partecipanti. In questo contesto, la blockchain permissioned viene utilizzata esclusivamente come meccanismo di ancoraggio delle credenziali, garantendo l’integrità, la tracciabilità temporale e la verificabilità decentralizzata dei titoli rilasciati.

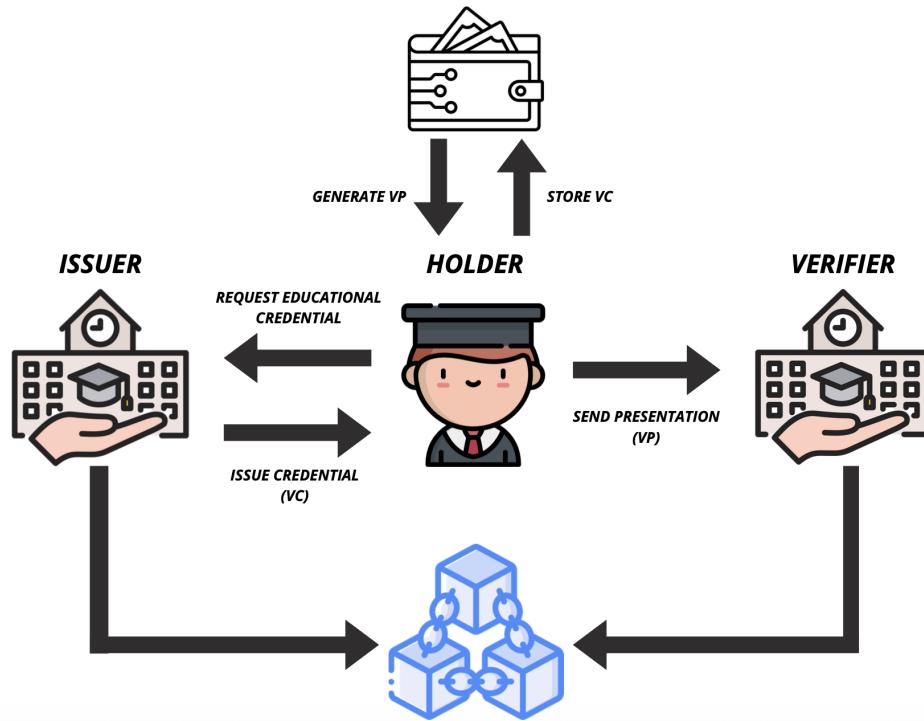


Figura 2.4: Architettura del sistema di gestione delle credenziali accademiche

2.2 Componenti del Sistema

2.2.1 Decentralized Identifiers (DID)

I Decentralized Identifiers rappresentano il fondamento dell'identità digitale nel sistema proposto. Ogni attore del sistema (studente, università di origine, università ospitante) possiede un identificatore univoco e verificabile in modo indipendente, che garantisce l'autenticità e l'integrità delle operazioni svolte.

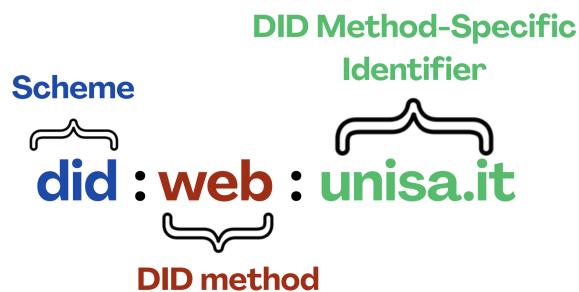


Figura 2.5: Struttura di un Decentralized Identifier

Un DID, es. **did:web:unisa.it**, è un URI strutturato secondo la sintassi generale dove:

- **did** è lo schema standard per tutti i Decentralized Identifier;

- web è il *DID method* utilizzato, che specifica come creare, risolvere e gestire DID basati su nomi di dominio HTTPS;
- unisa.it è l'identificativo specifico del metodo, corrispondente al dominio dell'organizzazione che controlla il DID.

Il soggetto del DID `did:web:unisa.it` è l'Università degli Studi di Salerno. In questo caso, l'università è anche il *controller* del DID, ovvero l'entità autorizzata a modificare il relativo documento (DID Document) grazie al possesso delle chiavi crittografiche specificate al suo interno. Sebbene in questo modello il soggetto e il controller coincidano, la specifica DID consente di separarli, ad esempio per affidare la gestione del DID a un servizio terzo o a una struttura delegata. Nel caso specifico del metodo `did:web`, l'operazione di *DID resolution* viene effettuata attraverso la dereferenziazione HTTPS del file JSON disponibile all'indirizzo `https://unisa.it/.well-known/did.json`, che contiene la rappresentazione del documento DID. Tale documento può essere ulteriormente "navigato" utilizzando un *DID URL*, che estende la sintassi del DID includendo componenti come `path`, `query` e, più comunemente, il `fragment`. Quest'ultimo consente di riferirsi a risorse specifiche all'interno del documento, come ad esempio una **chiave pubblica** (es. `did:web:unisa.it#keys-1`).

In particolare, l'uso del `fragment identifier` all'interno del DID URL consente di identificare in modo univoco la chiave crittografica usata per la firma o l'autenticazione, facilitando processi di verifica automatica e interazione sicura tra entità.

Tutti i DID si basano su un *verifiable data registry*, ovvero un sistema affidabile in grado di conservare e rendere disponibili i documenti associati agli identificatori. Nel nostro caso, tale registry è rappresentato da una struttura web gerarchica, in linea con la scelta di `did:web`.

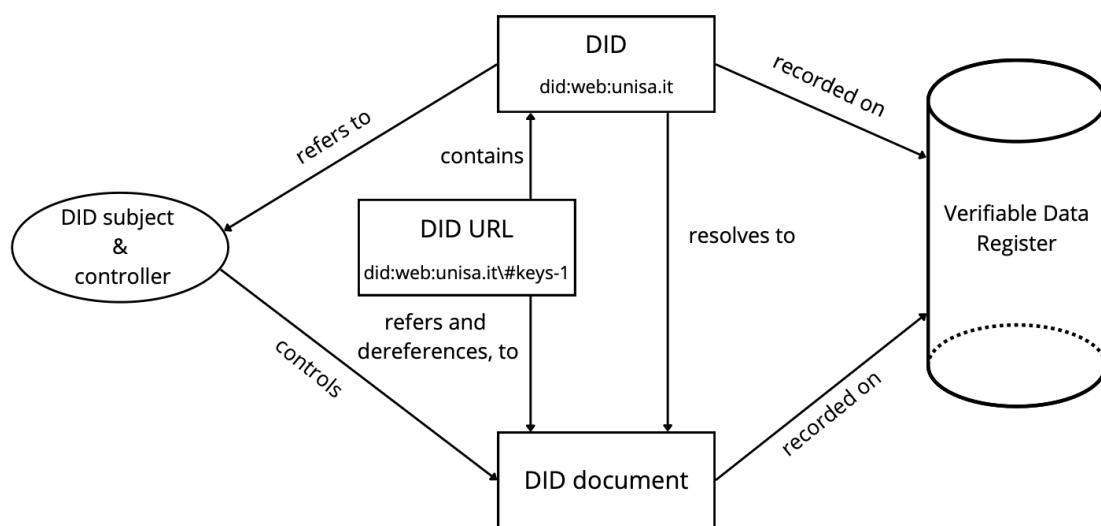


Figura 2.6: Architettura del sistema DID

2.2.2 Scelta del Metodo did:web

Abbiamo preferito il metodo `did:web` rispetto ad alternative come `did:eth`. La scelta è motivata da considerazioni tecniche, operative ed economiche. Sul piano tecnico-organizzativo, adottare una blockchain permissioned condivisa tra le università, come nel caso di `did:eth`, avrebbe richiesto che il consorzio si occupasse anche dell'accreditamento preventivo degli studenti. Questa soluzione sarebbe risultata incoerente con il principio di affidabilità istituzionale perseguito: gli studenti non devono essere soggetti accreditati né avere visibilità diretta sulla governance o sull'infrastruttura sottostante, proprio per garantire che il controllo rimanga nelle mani dell'istituzione. Una possibile alternativa sarebbe stata l'utilizzo di una blockchain pubblica eliminando così la necessità di accreditamento, ma questa scelta avrebbe compromesso il controllo locale e l'affidabilità istituzionale del sistema. Dal punto di vista economico, inoltre, l'uso di `did:eth` comporta costi intrinseci legati all'infrastruttura Ethereum, in particolare il pagamento di commissioni (gas fee) per ogni operazione di scrittura o aggiornamento. Mentre le università potrebbero eventualmente farsi carico di tali spese nell'ambito di iniziative europee per la digitalizzazione, per lo studente tale modello risulterebbe poco sostenibile, in quanto introdurrebbe oneri economici individuali per accedere e utilizzare credenziali che dovrebbero invece essere garantite come servizio pubblico.

In questo senso, `did:web` rappresenta una scelta più coerente con i principi di inclusività, controllo istituzionale e sostenibilità economica del sistema.

2.2.3 Tipologie di Verifiable Credentials

Il sistema proposto per la gestione della mobilità studentesca Erasmus si basa sull'utilizzo di Verifiable Credentials (VC) per certificare e attestare le informazioni accademiche e amministrative degli studenti. Il framework gestisce due categorie principali di credenziali:

- **Eligibility Erasmus Credential**, che attesta l'identità universitaria e il diritto dello studente a partecipare al programma Erasmus;
- **Academic Credential**, che certifica le attività formative svolte durante il periodo di mobilità internazionale.

L'analisi del comportamento del sistema ha evidenziato una soluzione progettuale che rappresenta un equilibrio ottimale tra efficienza e flessibilità. Sebbene inizialmente si fosse considerata la possibilità di separare l'identità universitaria (`UniversityIdentityCredential`) dall'attestazione di eligibilità (`EligibilityErasmusCredential`), la significativa sovrapposizione di attributi comuni (matricola, corso di studi, anno di iscrizione) ha suggerito un approccio unificato. L'università di origine emette quindi un'unica **EligibilityErasmusCredential** che integra gli elementi identificativi dell'identità accademica con i dati necessari per determinare l'idoneità Erasmus. Questa credenziale adotta una struttura standard, senza particolari meccanismi di organizzazione

interna dei *claims*, essendo destinata a un uso diretto e poco frammentato delle informazioni. Tale approccio offre vantaggi sostanziali in termini di riduzione della complessità gestionale ed efficienza operativa, consentendo ai verificatori di accedere a tutte le informazioni rilevanti attraverso una singola presentazione. Vale comunque sottolineare che l'approccio modulare, basato su più *Verifiable Credentials* distinte, rimane una valida alternativa, potenzialmente più aderente ai principi di *data minimization*, qualora si richiedesse un controllo granulare sulla separazione semantica e amministrativa delle attestazioni. La scelta definitiva dipende dal contesto applicativo e dai requisiti di interoperabilità tra le istituzioni coinvolte.

L'*Academic Credential* richiede invece un approccio differente, più strutturato e flessibile, per supportare scenari complessi di condivisione selettiva. In questo contesto, è la *Verifiable Presentation (VP)* a essere costruita con una struttura *Merkle-friendly*, che consente la verifica selettiva di singoli attributi senza rivelare l'intero contenuto della credenziale originaria. Questo modello risulta particolarmente vantaggioso quando lo studente deve dimostrare, ad esempio, l'acquisizione di un determinato numero di crediti o il completamento di specifici insegnamenti, mantenendo la riservatezza sugli altri dati contenuti nella VC. A differenza dell'approccio *naïve* basato su *monoclaim credentials* — in cui ogni attributo è attestato da una VC distinta — il modello adottato prevede l'emissione di un'unica *Verifiable Credential* contenente un insieme di claim. Questo consente una gestione molto più efficiente, evitando la ridondanza di metadati, firme e informazioni ausiliarie che caratterizza le soluzioni frammentate. La selettività nella condivisione non è quindi delegata alla frammentazione dell'informazione in più credenziali, ma è garantita dalla struttura stessa della *Verifiable Presentation*, progettata per supportare l'estrazione e la verifica di sottoinsiemi di dati in modo sicuro e verificabile.

2.2.4 Struttura della EligibilityErasmusCredential

La *EligibilityErasmusCredential* segue il formato standard definito dal W3C per le *Verifiable Credentials* con un contesto personalizzato. La credenziale è serializzata in JSON-LD e organizzata secondo il seguente schema:

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://consorzio-universita.example/credentials/v1"
  ],
  "type": ["VerifiableCredential", "EligibilityCredential"],
  "issuer": "did:web:unisa.it",
  "issuanceDate": "2025-04-10T10:15:00Z",
  "expirationDate": "2026-04-10T10:15:00Z",
  "credentialSubject": {
    "id": "did:web:luca-verdi.dev",
    "name": "Luca Verdi",
    "degree": "Bachelor of Science in Computer Science",
    "institution": "Università degli Studi di Salerno",
    "startYear": "2021-09-01",
    "endYear": "2025-06-30"
  }
}
```

```

    "givenName": "Luca",
    "familyName": "Verdi",
    "homeUniversity": "Università di Salerno",
    "erasmusStatus": "Eligible",
    "destinationUniversity": "Université de Rennes",
    "academicYear": "2025/2026"
  },
  "credentialStatus": {
    "id": "https://consorzio-univ.it/creds/unisa-erasmus-94567",
    "type": "ConsortiumRevocationRegistry2024",
    "registry": "0x534b89b798e45929A24a217d7324EAd0EAF9413E",
    "namespace": "0x1234567890123456789012345678901234567890",
    "revocationList": "0x7891011121314151617181920212223242526272829",
    "revocationKey": "0xa1b2c3d4e5f67890123456789abcdef0123456789abcdef"
  },
  "evidence": {
    "type": "BlockchainRecord",
    "description": "Hash della credenziale ancorato su blockchain",
    "transactionHash": "0xdeadbeefcafebabefeed1234...",
    "network": "ConsorzioReteUniversitaria"
  },
  "proof": {
    "type": "RsaSignature2023",
    "created": "2025-04-10T10:15:10Z",
    "proofPurpose": "assertionMethod",
    "verificationMethod": "did:web:unisa.it#key-1",
    "jws": "<firma-jws-generata>"
  }
}

```

Le principali componenti della credenziale sono:

- `@context`: specifica il contesto semantico del documento, permettendo l'interoperabilità attraverso l'utilizzo di vocabolari standard. In questo caso, viene utilizzato il contesto ufficiale delle Verifiable Credentials del W3C.
- `type`: indica le classi semantiche della credenziale. Oltre a `VerifiableCredential`, è incluso il tipo specifico `EligibilityCredential`, che distingue semanticamente il contenuto.
- `issuer`: rappresenta l'identificativo decentralizzato (DID) dell'ente emittente, in questo

caso l'università di origine. È modellato come un DID di tipo `did:web`.

- `issuanceDate` – `expirationDate`: data e ora di emissione e di scadenza della credenziale, entrambe espresse nel formato standard ISO 8601.
- `credentialSubject`: sezione che contiene i dati del soggetto titolare della credenziale. Include:
 - `id`: DID dello studente destinatario
 - `givenName`, `familyName`: nome e cognome
 - `homeUniversity`: ateneo di appartenenza
 - `erasmusStatus`: stato di eleggibilità al programma Erasmus
 - `destinationUniversity`: università straniera ospitante
 - `academicYear`: anno accademico di riferimento
- `credentialStatus`: specifica il meccanismo utilizzato per la revoca della credenziale. In questo caso si adotta un modello custom, `ConsortiumRevocationRegistry2024`, che si basa su un registro condiviso tra le università consorziate per la gestione decentralizzata dello stato di validità delle credenziali. Sebbene si tratti di un approccio custom, esso è coerente con quanto previsto dallo standard W3C, il quale lascia esplicitamente aperta la possibilità di definire modelli specifici per la revoca. Il modello implementato prende ispirazione da soluzioni già documentate nelle estensioni allo standard, come l'`Ethr Revocation 2022`, un meccanismo basato su registro per la revoca delle Verifiable Credentials. I campi inclusi nel nostro schema sono definiti in modo da garantire compatibilità con gli attuali strumenti di verifica e interoperabilità tra le istituzioni partecipanti al consorzio.
 - `id`: URL pubblico univoco che identifica l'elemento relativo allo stato della credenziale nel sistema di revoca.
 - `type`: indica il sistema di revoca adottato, `ConsortiumRevocationRegistry2024` rappresenta uno standard interno al consorzio universitario per la gestione delle revoche su una blockchain permissioned.
 - `registry`: indirizzo dello smart contract incaricato della gestione delle revoche all'interno della blockchain del consorzio.
 - `namespace`: identificatore logico che rappresenta un sottoinsieme di credenziali gestite e controllate direttamente dall'ente emittente, come una specifica università.
 - `revocationList`: riferimento alla struttura dati che contiene l'elenco effettivo delle credenziali revocate all'interno del namespace.

- `revocationKey`: chiave univoca che consente di individuare e verificare lo stato di revoca della credenziale all'interno della `revocationList`, secondo il protocollo di revoca adottato.
 - `evidence`: contiene le informazioni che attestano la prova dell'ancoraggio della credenziale su una blockchain.
 - `type`: specifica il tipo di evidenza fornita, in questo caso `BlockchainRecord`, indicando che l'evidenza è rappresentata da una registrazione blockchain.
 - `description`: breve descrizione del contenuto dell'evidenza, qui indica che è stato calcolato e memorizzato l'hash della credenziale sulla blockchain.
 - `transactionHash`: l'hash della transazione blockchain specifica in cui è stato registrato l'hash della credenziale, utilizzabile per la verifica diretta sull'infrastruttura blockchain.
 - `network`: nome della rete blockchain o consortium che ospita la transazione, utile per indirizzare correttamente la verifica nel contesto giusto (ad esempio, una rete permissioned universitaria).
 - `proof`: contiene la firma digitale dell'issuer, necessaria a garantire l'integrità e l'autenticità del documento. Include:
 - `type`: tipo di firma digitale utilizzata
 - `created`: timestamp della firma
 - `proofPurpose`: scopo della prova, tipicamente `assertionMethod`
 - `verificationMethod`: chiave pubblica utilizzata per la verifica, indicata tramite fragment DID
 - `jws`: la firma vera e propria in formato JWS (JSON Web Signature)

L'ancoraggio su blockchain avviene tramite l'hash dell'intera Verifiable Credential, senza supportare la selective disclosure. Il verificatore può quindi usare il `transactionHash` per controllare l'integrità della credenziale e, interrogando la blockchain del consorzio, verificarne la validità o eventuale revoca. Questo processo garantisce una verifica trasparente, indipendente e non falsificabile.

Il campo `network` è inserito per fornire contesto sulla rete blockchain utilizzata; nel caso di presentazione a terze parti esterne al consorzio, l'ente verificatore identifica la rete di riferimento e può interrogare il consorzio per confermare validità e integrità.

L'inserimento della proprietà `expirationDate`, con validità di un anno dalla data di emissione, riflette la natura temporanea dell'idoneità, evitando la conservazione indefinita di attestazioni

non più rilevanti. Tale scelta assicura una gestione sicura e conforme al ciclo di vita previsto per le informazioni di mobilità internazionale.

2.2.5 Struttura della AcademicCredential

Passiamo dunque ad analizzare nel dettaglio la struttura della *Academic Credential*, il cui obiettivo è certificare i risultati accademici ottenuti dallo studente durante il periodo di mobilità. Nel seguente esempio JSON è riportata una credenziale conforme allo standard delle *Verifiable Credentials*, emessa dall'università ospitante (Université de Rennes), che certifica il superamento di tre esami sostenuti dallo studente Luca Verdi. Si noti come il contesto (@context) rimanga invariato, mentre lo *schema* adottato per la credenziale differisca, riflettendo la diversa tipologia e struttura informativa di questa specifica VC.

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://consorzio-universita.example/credentials/v1"
  ],
  "type": ["VerifiableCredential", "AcademicCredential"],
  "issuer": "did:web:rennes.it",
  "issuanceDate": "2025-06-07T12:00:00Z",
  "expirationDate": "2027-06-07T12:00:00Z",
  "credentialSubject": {
    "id": "did:web:luca-verdi.dev",
    "givenName": "Luca",
    "familyName": "Verdi",
    "homeUniversity": "Università di Salerno",
    "exams": [
      {"examId": "EX101",
       "name": "Matematica",
       "credits": 6,
       "grade": 28,
       "date": "2024-10-01"},
      {"examId": "EX102",
       "name": "Fisica",
       "credits": 9,
       "grade": 25,
       "date": "2024-11-15"},
      {"examId": "EX103",
       "name": "Informatica",
       "credits": 10,
       "grade": 30,
       "date": "2024-12-15"}
    ]
  }
}
```

```

    "credits": 12,
    "grade": 30, "date":
    "2024-12-10"},

    {"examId": "EX104",
    "name": "Chimica",
    "credits": 6,
    "grade": 22,
    "date": "2025-01-20" }

],
},
"credentialStatus": {
    "id": "https://consorzio-univ.it/creds/rennes-academic-2025",
    "type": "ConsortiumRevocationRegistry2024",
    "registry": "0xABCDEF1234567890ABCDEF1234567890ABCDEF12",
    "namespace": "0x9876543210FEDCBA09876543210FEDCBA0987654",
    "revocationList": "0x456789ABCDEF456789ABCDEF456789ABCDEF1234",
    "revocationKey": "0xfeedfacecafebabe1234567890abcdefabcdef12"
},
"evidence": {
    "type": "BlockchainRecord",
    "description": "Merkle Root firmata e registrata su blockchain",
    "transactionHash": "0xdeadbeefcafebabefeed1234...",
    "network": "ConsorzioReteUniversitaria"
}
"proof": {
    "type": "RsaSignature2023",
    "created": "2025-04-10T10:15:10Z",
    "proofPurpose": "assertionMethod",
    "verificationMethod": "did:web:unisa.it#key-1",
    "jws": "<firma-jws-generata sulla merkle root>"
}
}

```

La struttura della credenziale comprende i seguenti elementi principali (sono omessi i campi ridondanti):

- `expirationDate`: a differenza della Eligibility, l'Academic ha validità 2 anni.
- `credentialSubject`: contiene i dati accademici dello studente, tra cui:
 - `id`: DID dello studente.

- `givenName, familyName`: nome e cognome.
- `homeUniversity`: università di provenienza.
- `exams`: elenco degli esami superati con i seguenti dettagli: identificativo (`examId`), nome, CFU, voto e data.
- `evidence`: sezione che collega la credenziale a una transazione blockchain, all'interno della quale è stata registrata la Merkle Root firmata.

Durante la fase di emissione, per abilitare la *selective disclosure*, viene costruito un *Merkle Tree* a partire dagli attributi degli esami. La credenziale sopra riportata non contiene né la Merkle Root né le relative Merkle Proofs all'interno del documento. Tale scelta è intenzionale e segue le best practices stabilite a livello internazionale, secondo i seguenti principi:

- **Dimensione e Generalità:** la Verifiable Credential originale deve essere il più possibile leggera e generica. Includere tutte le possibili combinazioni di *Merkle Proofs* renderebbe la VC inutilmente pesante e difficilmente gestibile, poiché ogni possibile subset dei dati rivelabili necessiterebbe una propria prova.
- **Principio di Minimizzazione:** la VC contiene esclusivamente i dati attestati dall'emittente. Le prove per la divulgazione selettiva (*Selective Disclosure*) sono responsabilità esclusiva dell'*holder* (in questo caso, lo studente).
- **Coerenza con gli Standard:** secondo il *W3C Verifiable Credentials Data Model*, le Merkle Proofs non sono parte integrante della VC originale, ma vanno generate al momento della costruzione della Verifiable Presentation (VP). In questo contesto, è sufficiente che la Merkle Root firmata venga ancorata su blockchain (come mostrato nel campo `evidence`), fungendo da riferimento per tutte le future presentazioni parziali.

2.3 Pre-Game

La fase di Pre-Game comprende tutti i prerequisiti tecnologici e organizzativi che devono essere soddisfatti prima che uno studente possa richiedere una credenziale accademica attestante le attività svolte durante un periodo di mobilità.

Si presuppone che lo studente abbia già completato la procedura di immatricolazione. Durante questa fase preliminare, lo studente ha provveduto a verificare che il DID dell'università fosse presente nell'elenco degli enti accreditati dal consorzio, procedendo successivamente al salvataggio di tale identificativo nel proprio wallet digitale. Una volta confermata l'accreditazione, lo studente ha comunicato il proprio DID all'università, la quale ha provveduto a registrare nel proprio database interno l'insieme delle informazioni relative al profilo dello studente. Per garantire l'aggiornamento delle informazioni e ottimizzare le comunicazioni, il consorzio si assume la

responsabilità di notificare direttamente allo studente l’eventuale revoca dell’accreditamento dell’università. Questo meccanismo di notifica proattiva elimina la necessità di continue verifiche presso l’ente di accreditamento. Lo studente, riconoscendo il consorzio come autorità di fiducia nel sistema, può fare affidamento sulla validità del DID universitario memorizzato, avendo la certezza che tale identificativo rimanga aggiornato e valido fino a eventuale comunicazione contraria.

Nel momento in cui lo studente si prepara a candidarsi per un programma di mobilità internazionale, come un’esperienza Erasmus, inizia una fase di preparazione e verifica delle identità digitali. Questa fase è fondamentale per garantire che lo studente sia realmente chi dichiara di essere e che possa interagire con l’università in modo sicuro e affidabile, senza ambiguità o possibilità di frode.

2.3.1 Studente

Lo studente, che rappresenta l’attore principale del sistema, è dotato di un’identità digitale formalizzata tramite un **DID web**, conforme allo standard W3C. Tale DID assume la forma `did:web:localhost:8443`, ovvero un identificatore URI che punta a un documento DID in formato JSON pubblicato su un server accessibile via protocollo HTTPS. Questo documento contiene la chiave pubblica e metadati utili alla verifica delle firme digitali, ed è consultabile da qualsiasi entità che conosca il DID per eseguire controlli di autenticità.

L’identità digitale non è un semplice identificativo statico, ma è legata a una coppia di chiavi crittografiche asimmetriche: una chiave privata sk (secret key), mantenuta segreta dallo studente, e una chiave pubblica pk , pubblicata nel documento DID. La generazione della coppia avviene mediante un algoritmo GenKey , che accetta in input il parametro di sicurezza n , producendo:

$$(sk, pk) \leftarrow \text{GenKey}(1^n)$$

La chiave privata sk è utilizzata per firmare messaggi tramite una funzione di firma Sign , mentre chiunque può verificare tali firme usando pk con una funzione di verifica Verify , secondo la relazione:

$$\sigma \leftarrow \text{Sign}_{sk}(m), \quad \text{Verify}_{pk}(m, \sigma) \rightarrow \{\text{true}, \text{false}\}$$

Lo studente, intenzionato a candidarsi al programma Erasmus, accede al sistema universitario mediante un meccanismo di autenticazione basato su credenziali tradizionali. In un primo momento verifica la corrispondenza tra il DID pubblicato dall’università e quello presente nel proprio wallet. Solo in caso di esito positivo procede inserendo le proprie credenziali (username e password) nel portale. Il server calcola l’hash della password, lo concatena al salt e confronta il risultato con il valore memorizzato nel database per verificare l’identità dell’utente. Sebbene sia stato progettato un sistema di autenticazione a due fattori con codice OTP (One-

Time Password) generato tramite un generatore pseudocasuale deterministico per garantire una maggiore sicurezza, tale meccanismo non è stato implementato nella versione attuale del sistema per ragioni di semplificazione implementativa. Il sistema progettato prevedeva la condivisione di un seme crittografico tra università e studente durante la fase di immatricolazione, necessario per la sincronizzazione dei codici OTP generati.

Durante il processo di candidatura Erasmus, lo studente, già autenticato al sistema tramite le credenziali istituzionali, comunica il proprio identificatore DID, precedentemente registrato al momento dell’immatricolazione. Il server effettua innanzitutto un controllo formale, verificando che il DID fornito corrisponda a quello associato al profilo dello studente nel database. Se la corrispondenza è confermata, viene avviato un protocollo di tipo challenge-response per garantire che lo studente sia effettivamente in possesso della chiave privata associata a quel DID

Il server genera una sfida casuale $c \leftarrow \text{Random}(1^n)$, che lo studente firma utilizzando la propria chiave privata:

$$r = \text{Sign}_{\text{sk}}(c)$$

Successivamente, il server recupera il documento DID pubblicato all’URL corrispondente, estrae la chiave pubblica pk e verifica la firma ricevuta:

$$\text{Verify}_{\text{pk}}(c, r) \stackrel{?}{=} \text{true}$$

Se la verifica ha esito positivo, il sistema conclude che lo studente è effettivamente in possesso della chiave privata associata al DID dichiarato, e l’associazione tra profilo utente e identità decentralizzata è completata. Questo passaggio è fondamentale per poter poi emettere Verifiable Credentials firmate digitalmente o per presentare prove crittografiche selettive in modo sicuro.

Sebbene il documento DID sia pubblicato su HTTPS, quindi trasportato in modo sicuro, non è possibile inferire automaticamente che l’autore del documento coincida con chi ne reclama il controllo. Solo una prova attiva di possesso della chiave privata, tramite challenge-response, può dimostrare che lo studente non si sta spacciando per qualcun altro semplicemente copiando un URL. Per questo motivo, nonostante la comunicazione HTTPS, la prova crittografica resta necessaria per stabilire il legame tra identità off-chain (profilo utente) e DID controllato.

Una volta emessa, la VC viene trasferita nel portafoglio digitale (*wallet*) dello studente, dove può essere custodita in sicurezza. Attraverso il wallet, lo studente potrà successivamente presentare questa credenziale alle università ospitanti o ad altri enti coinvolti nel processo Erasmus, in modo verificabile e immutabile.

2.3.2 Università di origine

L’Università di origine svolge un ruolo cruciale di *Verificatore Affidabile* e *Issuer* nel processo Erasmus, certificando in modo ufficiale l’idoneità degli studenti attraverso l’emissione di Veri-

fiable Credentials conformi agli standard W3C. In questa fase preliminare, l'università funge da *Issuer* e attesta ufficialmente l'idoneità Erasmus dello studente tramite il rilascio di una VC firmata digitalmente, conforme al modello selettivo e crittograficamente verificabile.

Parallelamente al sistema di identità decentralizzata, l'università mantiene anche un sistema di account interno per ogni studente, basato su un meccanismo di autenticazione sicura tramite hash con salt. Per ogni password pw , viene generato un valore casuale $salt \leftarrow \text{Random}(1^n)$ memorizzato in modo sicuro nel database dell'università, e si calcola l'hash protetto:

$$h = \text{Hash}(pw \parallel salt)$$

Al momento dell'autenticazione, lo studente reinserisce la password e il server replica localmente il calcolo dell'hash secondo la formula:

$$h' = \text{Hash}(pw_{\text{inserita}} \parallel salt)$$

confrontando infine h' con il valore h memorizzato nel database per determinare la validità delle credenziali.

L'emissione della VC è accompagnata dall'ancoraggio crittografico su blockchain: idealmente, ogni credenziale corrisponde a una singola transazione che regista l'hash della VC stessa, garantendo integrità e non ripudio. Tuttavia, per ragioni di efficienza operativa e riduzione dei costi di gas, l'università può aggregare più emissioni in un'unica transazione batch.

In questo caso, per consentire la verifica indipendente di ogni singola credenziale all'interno della transazione aggregata, sarebbe necessario costruire una struttura di tipo Merkle Tree, con le relative proof di appartenenza da fornire al verificatore. Queste proof permetterebbero di dimostrare l'inclusione dell'hash della VC nella transazione complessiva ancorata su blockchain. Per semplicità e chiarezza, nel nostro scenario non implementiamo questo meccanismo: assumiamo pertanto che ogni credenziale venga ancorata con una singola transazione, fornendo direttamente il relativo `transactionHash` per la verifica.

Questa credenziale, firmata digitalmente mediante la chiave privata dell'università e conforme agli standard W3C, viene emessa dall'università in qualità di *Issuer* e inviata allo studente, che ne diventa a tutti gli effetti il *Subject*. La firma, associata al *verificationMethod* del DID dell'università, consente a terze parti di verificarne l'autenticità utilizzando la corrispondente chiave pubblica.

2.4 Scambio delle credenziali accademiche

2.4.1 Interazione con l'università ospitante

Dopo aver ottenuto la Verifiable Credential (VC) di idoneità Erasmus dall'università di origine, lo studente si autentica presso l'università ospitante con un metodo analogo a quello utilizzato dall'università di origine. Restano valide le stesse assunzioni già formulate nella fase preliminare (pre-game). Una volta verificata l'identità dello studente, questi presenta la VC di idoneità Erasmus, firmata digitalmente dall'università emittente.

L'università ospitante procede quindi a verificare la validità crittografica della credenziale esaminando il campo `proof`, che contiene la firma digitale, il metodo di verifica e il riferimento al Decentralized Identifier (DID) dell'emittente.

Per ottenere la chiave pubblica necessaria alla verifica, l'università ospitante risolve il DID dell'emittente tramite il consueto meccanismo standard di risoluzione:

$$\text{pk}_{\text{issuer}} \leftarrow \text{Resolve}(\text{DID}_{\text{issuer}})$$

Prima di procedere alla risoluzione, si verifica che il DID dell'emittente sia presente nella lista interna dei DID accreditati dal consorzio universitario, a garanzia della fiducia nell'emittente.

Dal DID Document ottenuto si estrae il campo `verificationMethod`, contenente la chiave pubblica utilizzata per la firma digitale. La verifica della firma JWS avviene quindi secondo la seguente equazione:

$$\text{Verify}(\text{jws}, \text{pk}_{\text{issuer}}) = \text{true}$$

Se la firma risulta valida e la struttura della VC è conforme agli standard W3C, l'università ospitante procede a calcolare l'hash sull'intera credenziale e lo confronta con l'hash ancorato sulla blockchain del consorzio. Solo in caso di corrispondenza l'attestazione viene accettata come valida e autentica, certificando ufficialmente l'idoneità Erasmus dello studente.

2.4.2 Emissione e Gestione dell'Academic Credential

Durante il periodo di mobilità, l'università ospitante registra i risultati accademici dello studente. Al termine del soggiorno, essa emette una **Verifiable Credential** che include l'elenco completo degli esami sostenuti. Ogni esame viene trattato come un singolo *claim*, e per ciascuno di essi l'università calcola localmente l'hash del relativo contenuto, considerandolo come una foglia del Merkle Tree. Questa struttura consente, in fase di verifica, di dimostrare l'integrità della credenziale (o di una sua parte) attraverso la presentazione delle corrispondenti *Merkle Proofs*,

eventualmente verificabili anche on-chain. Ogni foglia l_i del Merkle Tree rappresenta un esame codificato come un JSON serializzato, ad esempio:

```
{
    "examId": "EX104",
    "name": "Chimica",
    "credits": 6,
    "grade": 22,
    "date": "2025-01-20"
}
```

Ogni foglia viene hashata:

$$h_i = H(l_i)$$

dove H è una funzione hash crittografica sicura. Le foglie vengono poi combinate ricorsivamente per ottenere la Merkle Root root secondo:

$$\text{root} = \text{MerkleRoot}(h_1, h_2, \dots, h_n)$$

La radice root così ottenuta viene:

- Firmata con la chiave privata dell'università ospitante (sk_{host}) tramite una firma JWS ed aggiunta nella proof della credenziale;
- Salvata sulla blockchain, generando una prova di inclusione immutabile. Il valore registrato è:

$$\text{Tx}_{\text{blockchain}} = \text{Publish}(\text{root})$$

Infine, la credenziale dei risultati accademici viene trasferita allo studente, che la conserva nel proprio wallet digitale. Al momento della presentazione, lo studente potrà selezionare uno o più esami da rivelare e generare le relative *Merkle Proofs*, dimostrando l'autenticità e l'integrità dei dati selezionati rispetto alla radice Merkle firmata. Il dettaglio del procedimento viene descritto nelle sezioni successive.

2.4.3 Verifica Decentralizzata tramite Blockchain

Dopo la generazione della *Academic Credential*, è fondamentale garantirne l'integrità, la tracciabilità e la verificabilità pubblica attraverso una registrazione su blockchain. L'obiettivo non è archiviare il contenuto completo della credenziale, ma fornire una prova crittograficamente verificabile della sua esistenza, senza compromettere la riservatezza dei dati personali.

Il processo si articola come segue:

- Viene calcolata la **Merkle Root** dell’insieme dei *claims* presenti nella credenziale;
- La **Merkle Root** e un timestamp vengono infine pubblicati su blockchain.

Nessuna informazione sensibile o direttamente riconducibile allo studente viene mai registrata on-chain: la sola Merkle Root, crittograficamente derivata, è sufficiente a rappresentare in modo univoco e non reversibile l’intero contenuto attestato.

Nel nostro sistema, ogni *Academic Credential* corrisponde a una transazione autonoma registrata sulla blockchain. Tale transazione non solo funge da prova di esistenza e autenticità della credenziale, ma diventa anche un punto di riferimento verificabile.

Più precisamente, per ogni credenziale vengono registrati i seguenti elementi:

- la **Merkle Root** calcolata sui dati della credenziale;
- un **timestamp** che ne certifica la data di emissione;
- il **DID dell’issuer**, utile per recuperare la chiave pubblica di verifica.

Il risultato di questa operazione è una transazione blockchain identificata da un `transactionHash`, che viene poi incluso nella credenziale stessa (nel campo `evidence`). In fase di verifica, tale hash consente di risalire direttamente alla transazione corrispondente, permettendo di confrontare la Merkle Root presentata con quella effettivamente registrata su blockchain.

Questo approccio garantisce un elevato livello di fiducia distribuita, mantenendo al contempo il pieno controllo e la privacy dei dati in capo allo studente.

2.4.4 Presentazione selettiva e verifica

Quando lo studente fa ritorno presso l’università di origine, può decidere di presentare una selezione degli esami sostenuti all’estero utilizzando un meccanismo di **presentazione selettiva** (*Selective Disclosure*).

Per effettuare la presentazione, il wallet digitale dello studente costruisce una *Verifiable Presentation* (VP). Il wallet, essendo un’applicazione standardizzata a livello europeo e obbligatoria per tutti gli studenti Erasmus, implementa protocolli uniformi. Questa standardizzazione garantisce che tutti gli studenti, indipendentemente dall’università di origine o destinazione, utilizzino le stesse procedure crittografiche e gli stessi formati di presentazione, assicurando interoperabilità completa nel sistema accademico europeo. La *Verifiable Presentation* (VP) sarà costituita da:

- L’**Academic Credential** originale emessa dall’università ospitante;
- Le foglie del Merkle Tree corrispondenti agli esami selezionati $\{l_{i_1}, l_{i_2}, \dots, l_{i_k}\}$;
- Le rispettive **Merkle Proofs** $\{\pi_{i_1}, \pi_{i_2}, \dots, \pi_{i_k}\}$, composte da:

- Per i campi *divulgati*, i valori in chiaro accompagnati dai percorsi di hash (*left* / *right*) necessari per verificarne l'inclusione;
- Per i campi *non divulgati*, solo il percorso di hash, senza rivelare il valore associato.
- La firma digitale dello studente sulla VP.

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://consorzio-universita-europee.eu/presentations/v1"
  ],
  "type": ["VerifiablePresentation"],
  "holder": "did:web:luca.verdi.[UUID].localhost",
  "verifiableCredential": {
    "@context": [
      "https://www.w3.org/2018/credentials/v1",
      "https://consorzio-universita.example/credentials/v1"
    ],
    "type": ["VerifiableCredential", "AcademicCredential"],
    "issuer": "did:web:rennes.it",
    "issuanceDate": "2025-07-21T08:55:18Z",
    "credentialSubject": {
      "id": "did:web:luca.verdi.[UUID].localhost",
      "disclosedClaims": {
        "EX001": {
          "name": "Machine Learning",
          "grade": 28,
          "credits": 9,
          "date": "2025-01-20"
        },
        "EX002": {
          "name": "Sistemi Embedded"
          // grade, credits e date NON divulgati
        }
      },
      "proofs": {
        "EX001": {
          "name": {
            "proof": [ ["right", "hash1"], [ "left", "hash2" ] ]
          },
        }
      }
    }
  }
}
```

```
"grade": {
    "proof": [ ["left", "hash3"], [ "right", "hash4"] ]
},
"credits": {
    "proof": [ [ "right", "hash5"], [ "left", "hash6"] ]
},
"date": {
    "proof": [ [ "left", "hash7"], [ "right", "hash8"] ]
},
"EX002": {
    "name": {
        "proof": [ [ "left", "hashA"], [ "right", "hashB"] ]
    },
    "grade": {
        "proof": [ [ "right", "hashC"], [ "right", "hashD"] ]
    },
    "credits": {
        "proof": [ [ "left", "hashE"], [ "left", "hashF"] ]
    },
    "date": {
        "proof": [ [ "right", "hashG"], [ "left", "hashH"] ]
    }
}
},
"credentialStatus": {
    "id": "https://consorzio-univ.it/creds/luca.verdi-academic-status",
    "type": "ConsortiumRevocationRegistry2024",
    "registry": "0xRegistryAddress",
    "namespace": "rennes",
    "revocationList": "<revocation_list_hash>",
    "revocationKey": "<revocation_key>"
},
"evidence": {
    "type": "BlockchainRecord",
    "description": "Merkle Root registrata su blockchain",
    "transactionHash": "<tx_hash>",
    "network": "ConsorzioReteUniversitaria"
```

```

    },
    "proof": {
        "type": "RsaSignature2023",
        "created": "2025-07-21T08:55:18Z",
        "verificationMethod": "did:web:rennes.it#key-1",
        "jws": "<firma_ente>"
    }
},
"proof": {
    "type": "RsaSignature2023",
    "created": "2025-07-21T08:55:18Z",
    "verificationMethod": "did:web:luca.verdi.[UUID].localhost#key-1",
    "nonce": "<nonce_randomico>",
    "jws": "<firma_studente>"
}
}
}

```

Il processo di verifica da parte dell'università di origine si articola nei seguenti passaggi:

1. **Verifica della firma digitale** sulla *Verifiable Presentation*, per accertarsi che la presentazione sia stata effettivamente costruita dallo studente:

$$\text{Verify}(\text{signature}_{\text{student}}, \text{pk}_{\text{student}}) = \text{true}$$

dove $\text{pk}_{\text{student}}$ è ottenuta dal DID dello studente.

2. **Recupero della Merkle Root dalla blockchain**, utilizzando il `transactionHash` incluso nella presentazione, e verifica della firma digitale dell'università ospitante su tale valore:

$$\text{Verify}(\text{signature}_{\text{issuer}}, \text{pk}_{\text{issuer}}) = \text{true}$$

dove $\text{pk}_{\text{issuer}}$ è ottenuta dal DID dell'università ospitante.

3. **Verifica dello stato di revoca della credenziale**, interrogando lo smart contract tramite la funzione

$$\text{isRevoked}(\text{namespace}, \text{list_id}, \text{revocation_key}) = \text{false}$$

dove i parametri sono estratti dalla credenziale o dalla Verifiable Presentation e identificano univocamente la credenziale nel registro di revoca. Se il risultato è `true`, la credenziale è considerata revocata e quindi non valida.

4. **Verifica delle Merkle Proofs**, accertandosi che ogni foglia presentata sia correttamente collegata a una medesima radice:

$$\text{VerifyMerkleProof}(l_{i_j}, \pi_{i_j}) = \text{true} \quad \forall j = 1, \dots, k$$

per ciascun esame presentato nella Verifiable Presentation.

5. **Ricostruzione della Merkle Root** a partire dalle foglie e dalle prove fornite, e confronto con quella recuperata on-chain:

$$\text{ReconstructedRoot} \stackrel{?}{=} \text{OnChainRoot}$$

assicurando che i dati rivelati provengano realmente dalla credenziale registrata e firmata.

6. **Valutazione accademica** finale da parte dell'università di origine, basata sugli esami verificati e sulle proprie regole di riconoscimento.

Questo meccanismo consente allo studente di condividere solo gli esami rilevanti per il riconoscimento accademico, mantenendo la riservatezza sugli altri dati. Allo stesso tempo, l'università di origine può effettuare una verifica completa e decentralizzata, senza dover confidare ciecamente sull'università emittente.

2.4.5 Revoca basata su ConsortiumRevocationRegistry2024

Il sistema di revoca delle credenziali implementato nel nostro prototipo si basa su un meccanismo denominato **ConsortiumRevocationRegistry2024**. Questo modello adotta un approccio on-chain e decentralizzato, costruito su un'unica blockchain consortile condivisa da tutte le università partecipanti. Al centro dell'architettura si trova un *Registry Contract* comune, che funge da punto di riferimento per la pubblicazione e la consultazione dello stato di revoca delle credenziali.

2.4.6 Architettura del Registry

La struttura del ConsortiumRevocationRegistry2024 è organizzata in namespace distinti, ciascuno assegnato a una specifica università membro del consorzio. Ogni namespace identifica univocamente l'ambito di gestione di un ente, il quale ha il pieno controllo delle revoche delle proprie credenziali. Alcuni esempi:

- `unibo.consortium` – Università di Bologna
- `uniroma1.consortium` – Sapienza Università di Roma
- `polimi.consortium` – Politecnico di Milano

Ogni namespace può registrare e gestire una o più liste di revoca indicizzate on-chain (revocation bitmaps), ciascuna rappresentata da una sequenza di bit. Ogni bit corrisponde allo stato di una credenziale specifica: se impostato a `true`, indica che la credenziale è stata revocata; se `false`, la credenziale è considerata valida.

2.4.7 Vantaggi rispetto a Revocation List

A differenza dei modelli tradizionali, come quello basato su *Revocation List*, in cui ogni ente mantiene una lista separata pubblicata su URL esterni e firmata digitalmente, questo modello sfrutta la blockchain per garantire trasparenza, immutabilità e verificabilità immediata. Questo approccio elimina la necessità di infrastrutture off-chain, semplificando il processo di verifica. Dunque i principali vantaggi sono:

- Decentralizzazione reale: non esistono URL centralizzati o server esterni, tutte le informazioni sono on-chain e accessibili a chiunque.
- Verificabilità immediata: la validità di una credenziale può essere controllata in tempo reale, interrogando direttamente il contratto sulla blockchain.
- Sicurezza e immutabilità: una volta pubblicata una revoca, questa non può essere modificata o rimossa arbitrariamente.

2.4.8 Inserimento della Credenziale

Quando un'università emette una credenziale, essa include nel documento firmato un campo `credentialStatus`, che fornisce tutte le informazioni necessarie per verificarne lo stato di validità anche in futuro. In particolare, questo campo contiene:

- **L'indirizzo del *Registry Contract*** consortile, condiviso da tutte le università partecipanti alla rete. È l'unico smart contract autorizzato a gestire le liste di revoca.
- **Il namespace dell'ente emittente** (es. `unibo.consortium`), che identifica l'autorità responsabile della gestione della revocation list e ne delimita i diritti di scrittura.
- **La revocation list** (campo `revocationList`), ovvero una stringa in formato `bytes32` che rappresenta l'identificatore univoco della lista di revoca indicizzata in cui sarà registrato lo stato della credenziale. Tale valore può essere generato applicando un algoritmo di hashing (ad esempio SHA-256) all'unione tra il namespace dell'ente emittente e l'identificatore della categoria della credenziale (ad esempio, un programma di studi o un

tipo di certificazione). Questo garantisce l'unicità e la tracciabilità della lista all'interno del namespace. .

- **La chiave di revoca** (campo `revocationKey`), una stringa in formato bytes32 che rappresenta la posizione specifica (indice) all'interno della lista di revoca. Il valore può anch'esso essere generato tramite hash (ad esempio SHA-256) partendo da un identificatore univoco della credenziale (come un UUID o un hash del contenuto).

Non è però richiesta alcuna transazione iniziale verso il *Registry Contract*: grazie all'architettura del registro, tutti i possibili namespace, liste di revoca e chiavi di revoca esistono implicitamente con valore `false` (bit impostato a zero). Questo significa che al momento dell'emissione non viene effettuata alcuna scrittura sulla catena e la credenziale è considerata non revocata per impostazione predefinita. Solo nel momento in cui l'ente decide di revocare la credenziale, viene inviata una transazione al *Registry Contract* per aggiornare il bit associato a `true`, indicando la revoca effettiva.

2.4.9 Verifica da parte di un'università

Quando un'università riceve una credenziale da uno studente, il processo di verifica della validità della credenziale avviene secondo i seguenti passaggi:

1. L'ente verificatore estrae dalla credenziale il campo `credentialStatus`, che contiene tutte le informazioni necessarie per determinare lo stato della credenziale.
2. Viene identificato il namespace dell'emittente, l'identificativo della lista di revoca pubblicata on-chain e l'indice che indica la posizione della credenziale all'interno della lista.
3. Il verificatore consulta il contratto di registro sulla blockchain consortile, interrogando la lista di revoca specificata all'interno del namespace indicato.
4. Il bit associato a quell'indice viene letto: se il bit è impostato a `true`, significa che la credenziale è stata revocata e dunque non è più valida; se invece è `false`, la credenziale è ancora valida e può essere considerata affidabile.

L'intero processo è decentralizzato e non richiede contatti diretti con l'università emittente, in quanto tutte le informazioni necessarie sono pubblicamente disponibili sulla blockchain consortile e verificabili crittograficamente.

2.4.10 Revoca di una credenziale

La revoca di una credenziale può essere effettuata esclusivamente dall'ente che l'ha emessa, all'interno del proprio namespace. Il processo avviene attraverso una transazione sulla blockchain,

con la quale viene aggiornato lo stato della lista di revoca.

In particolare:

- L'università esegue una transazione on-chain che imposta a `true` il bit corrispondente alla posizione della credenziale all'interno della lista di revoca.
- Tale operazione è permanente, immutabile e pubblicamente verificabile da qualsiasi altro partecipante del consorzio.
- Nessun altro ente è autorizzato a modificare o sovrascrivere i dati relativi alle liste di revoca di un namespace che non gestisce, garantendo così l'integrità e la responsabilità distribuita del sistema.

Questa architettura, pur operando su un'infrastruttura condivisa, mantiene la piena sovranità di ciascuna università sulle proprie credenziali, offrendo un equilibrio tra decentralizzazione, trasparenza e sicurezza.

Work Package 3

Lo scopo del seguente capitolo è analizzare la soluzione evidenziata all'interno del WP2 alla luce delle proprietà di confidenzialità, integrità, trasparenza ed efficienza, presentate nel WP1.

3.1 Confidenzialità

Di seguito vengono discussi, uno per uno, gli attaccanti che potrebbero compromettere le proprietà di confidenzialità definite nel WP1:

- **C.1** I dati contenuti nelle credenziali devono essere accessibili solo a soggetti autorizzati, e limitatamente alle informazioni strettamente necessarie per il ruolo di ciascun attore. I possibili avversari interessati a compromettere la proprietà *C.1* sono *MaliciousVerifier* e *OverCuriousHolder*. Il primo è mitigato dal protocollo di selective disclosure, mentre il secondo dalla cifratura (encryption) dei claims tramite chiave privata dell'Issuer.
- **C.2** Le comunicazioni tra *Holder* e *Issuer* devono avvenire tramite canali sicuri, che impediscono l'intercettazione o l'inferenza di informazioni sensibili, comprese quelle utilizzate per la generazione delle credenziali. I possibili avversari interessati a compromettere la proprietà *C.2* sono *MITMAttacker* (Issuer, Holder) e *The Big Brother*. Entrambi sono mitigati utilizzando protocolli di cifratura end-to-end, che proteggono il contenuto da accessi non autorizzati.
- **C.3** Anche le comunicazioni tra *Holder* e università di origine devono avvenire in modo sicuro, prevenendo la divulgazione non autorizzata dei dati dell'utente. I possibili avversari interessati a compromettere la proprietà *C.3* sono *MITMAttacker* (Holder, Verifier) e *The Big Brother*, mitigati come descritto per la proprietà *C.2*.

3.2 Integrità

- **I.1** Ogni credenziale deve restare integra, non alterata né danneggiata dopo l'emissione, sia a riposo che in transito. I possibili avversari interessati a compromettere la proprietà *I.1* sono *The Forger*, *MITMAttacker* (Issuer, Holder), *MITMAttacker* (Holder, Verifier). Ogni Verifiable Credential (VC) è sigillata con la chiave privata dell'emittente (l'università); di conseguenza, qualsiasi modifica al suo contenuto, anche la più piccola, verrebbe immediatamente rilevata, rendendo la credenziale non valida e smascherando il tentativo di frode.

- **I.2** I dati esibiti in una presentazione devono essere parte coerente e verificabile della credenziale dello studente; Un'altra sfida all'integrità emerge con il "RevocationBypasser", uno studente che tenta di utilizzare credenziali non più valide. Per contrastarlo, il sistema integra un robusto meccanismo di revoca, basato su una Bitstring Revocation List (BRL) on-chain. Questo registro è la fonte autorevole sullo stato di validità di ogni credenziale. Quando un'università revoca una credenziale, lo stato nella BRL viene aggiornato e, durante il processo di verifica, il Verifier consulta tale lista, bloccando l'uso di credenziali obsolete o revocate.

3.3 Trasparenza

Il sistema proposto gode di un alto grado di trasparenza grazie alla blockchain permissioned, in quanto i registri delle emissioni e degli stati di revoca sono immutabili e consultabili dai partecipanti autorizzati, fornendo una fonte di verità comune e verificabile.

La proprietà T.1: "Gli algoritmi crittografici, i protocolli di comunicazione e i criteri di validazione devono essere documentati e accessibili pubblicamente" si traduce in diversi aspetti chiave:

- Algoritmi Crittografici Pubblici: Il sistema dovrebbe adottare e utilizzare algoritmi crittografici standard e pubblicamente noti. La documentazione deve specificare esattamente quali algoritmi vengono impiegati e come vengono utilizzati per garantire l'integrità e la riservatezza. ;
- Protocolli di Comunicazione Aperti: Tutti i protocolli utilizzati per lo scambio di informazioni (tra Holder, Issuer e Verifier), come quelli per l'emissione, la presentazione e la verifica delle credenziali, devono essere specificati in dettaglio e resi accessibili;
- Criteri di Validazione Dettagliati: I meccanismi e le regole che un Verifier deve seguire per determinare la validità di una credenziale devono essere chiaramente documentati. Questo include come verificare le firme digitali, come consultare e interpretare lo stato di revoca, e quali condizioni devono essere soddisfatte perché una credenziale sia considerata valida.

3.4 Efficienza

- **E.1** Le operazioni crittografiche devono essere eseguibili in tempi compatibili con l'uso quotidiano e l'adozione su larga scala. Il sistema è progettato per ottimizzare le prestazioni attraverso diverse strategie. L'utilizzo di algoritmi crittografici standard come RSA-2048 e SHA-256 garantisce operazioni efficienti e ampiamente supportate dall'hardware moderno. La costruzione del Merkle Tree avviene una sola volta durante l'emissione della

credenziale, mentre le verifiche successive richiedono solo il calcolo di hash logaritmici rispetto al numero totale di esami.

- **E.2** L’emissione, il trasferimento e la verifica delle credenziali devono essere ottimizzati per l’utilizzo su dispositivi con risorse limitate. L’architettura del sistema considera le limitazioni dei dispositivi mobili degli studenti. Le Verifiable Presentations contengono solo i dati strettamente necessari per la verifica selettiva, riducendo la dimensione dei payload trasmessi. L’utilizzo di `did:web` elimina la necessità di complesse operazioni blockchain sui dispositivi degli studenti, delegando le computazioni più intensive ai server universitari. Le operazioni di verifica delle Merkle Proofs sono computazionalmente leggere e possono essere eseguite efficientemente anche su dispositivi con risorse limitate.

3.5 Non ripudio (NP)

- **NP.1** Un’università che emette una credenziale non può negare in seguito di averla emessa. Il possibile avversario interessato a compromettere la proprietà *NP.1* è GhostIssuer. Ogni Verifiable Credential è firmata digitalmente con la chiave privata dell’università emittente, creando una prova crittografica dell’origine della credenziale. La firma digitale garantisce che l’università non possa successivamente negare di aver emesso la credenziale, poiché solo essa possiede la chiave privata corrispondente alla chiave pubblica verificabile.
- **NP.2** Un holder non può negare di aver presentato una certa credenziale. I possibili avversari interessati a compromettere la proprietà *NP.2* sono The Forger, CredentialSwapper e RevocationBypasser. L’holder non può modificare i propri esami perché l’università ha pubblicato il merkle root firmato su blockchain, rendendo le alterazioni rilevabili. Gli esami sono correlati al DID dello studente, impedendo trasferimenti non autorizzati. Durante la presentazione, l’holder firma la VP con la propria chiave privata, creando una prova irrefutabile dell’azione.
- **NP.3** Qualsiasi attore in gioco non può negare di aver presentato la sua identità. I possibili avversari interessati a compromettere la proprietà *NP.3* sono MITMAttacker (Issuer, Holder), MITMAttacker (Holder, Verifier), CrossPresentationLinker e MaliciousVerifier. Ogni interazione nel sistema richiede l’autenticazione reciproca degli attori coinvolti tramite protocolli di challenge-response basati sui rispettivi DID. Durante ogni scambio, entrambi gli attori si autenticano mutuamente: l’università prova la propria identità firmando una sfida con la chiave privata associata al proprio DID, mentre lo studente o il verificatore fanno altrettanto.

3.6 Autenticazione

- **AU.1** Solo le istituzioni accademiche autorizzate possono emettere credenziali valide nel sistema. I possibili avversari interessati a compromettere la proprietà *AU.1* sono GhostIssuer e attori esterni non autorizzati che tentano di impersonare università legittime. Il sistema utilizza un registro distribuito di istituzioni accademiche autorizzate, dove ogni università legittima deve registrare il proprio DID e la corrispondente chiave pubblica attraverso un processo di verifica consortile. Solo le università presenti in questo registro possono emettere Verifiable Credentials valide, poiché i verificatori controllano sempre che l'emittente sia incluso nella lista delle autorità riconosciute. Ogni credenziale è firmata con la chiave privata dell'università, permettendo la verifica dell'autenticità dell'emittente tramite la corrispondente chiave pubblica registrata.
- **AU.2** L'accesso ai registri di revoca deve essere controllato e limitato agli attori autorizzati. I possibili avversari interessati a compromettere la proprietà *AU.2* sono MaliciousVerifier, attori esterni malintenzionati e potenziali insider threat. Il sistema implementa controlli di accesso granulari sui registri di revoca (Bitstring Revocation List) utilizzando meccanismi di autenticazione basati su DID e autorizzazioni specifiche. Solo le università emittenti possono modificare lo stato di revoca delle proprie credenziali, mentre i verificatori autorizzati possono solamente consultare le informazioni necessarie per la validazione.

3.7 Autenticità (A)

- **A.1** Deve essere sempre possibile verificare che una credenziale sia stata emessa da un'istituzione accademica legittima. I possibili avversari interessati a compromettere la proprietà *A.1* sono The Forger, GhostIssuer e attori esterni che tentano di impersonare università legittime. Il sistema mantiene un registro distribuito e immutabile delle istituzioni accademiche autorizzate, accessibile pubblicamente per la verifica. Ogni università legittima ha il proprio DID registrato insieme alla chiave pubblica corrispondente, verificata attraverso processi di accreditamento istituzionale. Durante la validazione di una credenziale, il verificatore consulta automaticamente questo registro per confermare che l'emittente sia effettivamente un'istituzione riconosciuta. La combinazione di firma digitale e presenza nel registro garantisce che solo le università legittime possano emettere credenziali accettate dal sistema.
- **A.2** La firma dell'issuer sulla credenziale deve essere crittograficamente verificabile e legata a un'identità certificata e pubblicamente riconosciuta nel sistema. I possibili avversari interessati a compromettere la proprietà *A.2* sono MITMAttacker (Issuer, Holder), The Forger e attori che tentano di falsificare firme digitali. Ogni Verifiable Credential è firmata digitalmente dall'università emittente utilizzando algoritmi crittografici robusti con

la chiave privata associata al proprio DID. La chiave pubblica corrispondente è registrata nel sistema distribuito di identità e può essere recuperata tramite la risoluzione del DID dell'emittente. Durante la verifica, il sistema valida crittograficamente la firma utilizzando la chiave pubblica dell'università, garantendo che la credenziale non sia stata alterata e provenga effettivamente dall'emittente dichiarato. L'integrità crittografica della firma e il legame con l'identità certificata rendono impossibile la falsificazione delle credenziali.

3.8 Compromissione Hardware-Software

La sicurezza di un sistema basato su credenziali digitali è intrinsecamente legata all'integrità del wallet e alla protezione della chiave privata dello studente. Per mitigare queste minacce, è fondamentale adottare un approccio multi-livello:

- Per contrastare i client fantoccio e i wallet malevoli che intercettano o esfiltrano dati, l'Holder dovrebbe impiegare sistemi di rilevamento delle intrusioni. Un Host-based Intrusion Detection System (HIDS) monitora l'attività interna del dispositivo, rilevando comportamenti anomali o sospetti indicativi di malware o tentativi di accesso non autorizzato al wallet o alla chiave. allo stesso modo, un Network-based Intrusion Detection System (NIDS), in particolare un Anomaly-based IDS, può monitorare il traffico di rete in entrata e in uscita per rilevare anomalie comportamentali legate all'infezione o all'esfiltrazione di dati.
- La minaccia più grave è l'esfiltrazione diretta della chiave privata. Per prevenire questo, è essenziale utilizzare soluzioni che proteggano la chiave in un ambiente hardware sicuro. Gli Hardware Security Modules (HSMs) sono progettati per custodire le chiavi private ed eseguire le operazioni crittografiche al loro interno, senza mai esporre la chiave al sistema operativo. Questo impedisce ai malware di intercettare o copiare la chiave, anche in caso di compromissione software del dispositivo.
- Per mitigare il rischio di perdita di accesso alle credenziali a causa della perdita fisica del dispositivo o della chiave, è possibile implementare un sistema di recupero chiavi tramite terze parti fiduciarie. Utilizzando servizi di escrow sicuri, le chiavi possono essere sottoposte a backup e recuperate in modo controllato, salvaguardando l'utente dalla perdita permanente dei suoi dati accademici digitali.
- Per contrastare specificamente i replay attack, il protocollo di presentazione delle credenziali deve includere meccanismi anti-replay robusti. Questo si ottiene tramite l'uso di nonce nelle Verifiable Presentations.

Work Package 4

In questo capitolo viene presentata l'implementazione dell'architettura proposta, descrivendo le principali componenti software sviluppate. L'intero sistema è stato realizzato in linguaggio Python, e si compone di diverse classi che modellano le funzionalità delle università partecipanti, la gestione delle credenziali, la simulazione di una blockchain e altri elementi necessari per garantire l'integrità, la verifica e l'autenticazione delle informazioni.

4.1 University

4.1.1 BaseUniversity

Alla base della struttura ad oggetti vi è la classe astratta `BaseUniversity`, che rappresenta un'università generica. Essa incapsula tutte le funzionalità comuni che ogni ateneo deve possedere, tra cui:

- **Gestione del DID (Decentralized Identifier):** ogni università dispone di una coppia di chiavi crittografiche e di un documento DID associato, generato o caricato da file, e memorizzato in una struttura JSON conforme allo standard W3C.
- **Autenticazione degli studenti:** attraverso un sistema di *challenge-response* firmato con chiavi private e verificato con chiavi pubbliche registrate.
- **Registrazione e aggiornamento degli utenti:** ogni studente viene registrato in un database locale e può essere successivamente associato ad un DID e a una chiave pubblica.
- **Revoca delle credenziali:** la classe gestisce la comunicazione con il registro di revoca per aggiornare lo stato delle credenziali rilasciate.
- **Gestione dei DID trusted:** per poter validare le credenziali di altri atenei, viene mantenuta una cartella contenente i DID ‘‘fidati’’.

Questa classe funge da *template* per le due università specifiche, ereditata poi da `UniversitySalerno` e `UniversityRennes`.

4.1.2 UniversitySalerno

L'Università di Salerno ha il ruolo di università d'origine nel contesto Erasmus. Le sue funzionalità principali includono:

- **Generazione della Erasmus Credential:** una credenziale firmata che attesta l'idoneità dello studente a partecipare al programma Erasmus.
- **Registrazione sulla blockchain simulata:** un hash della credenziale viene ancorato in un blocco della blockchain interna.
- **Firma crittografica della credenziale:** mediante la chiave privata dell'ateneo, garantendo l'integrità dei dati.
- **Verifica delle presentazioni selettive:** quando uno studente presenta la credenziale accademica rilasciata al termine del percorso, viene verificata sia la firma della presentazione, sia la validità temporale, lo stato di revoca, e le Merkle Proof associate ai campi rivelati.

4.1.3 UniversityRennes

L'Università di Rennes ha il ruolo di università ospitante. Le sue principali funzionalità sono:

- **Verifica della Erasmus Credential:** effettua controlli approfonditi su firma, integrità, validità temporale e stato di revoca.
- **Generazione della Academic Credential:** al termine del soggiorno Erasmus, viene emessa una credenziale accademica che include una Merkle Root calcolata sugli esami sostenuti.
- **Registrazione della Merkle Root sulla blockchain:** la root firmata viene immessa nella blockchain simulata, fungendo da riferimento per future verifiche selettive.

4.2 Student

La classe Student rappresenta uno studente Erasmus ed è responsabile della gestione delle credenziali, della firma di messaggi e della generazione di presentazioni selettive.

Essa incapsula le seguenti funzionalità:

- **Generazione delle chiavi RSA:** al primo avvio, ogni istanza dello studente genera una coppia di chiavi crittografiche (pubblica e privata), salvate in formato PEM.
- **Creazione del documento DID:** viene generato un documento DID secondo lo standard W3C, includendo la chiave pubblica dello studente. Il documento è salvato in locale in formato JSON.
- **Firma dei messaggi:** lo studente è in grado di firmare messaggi arbitrari utilizzando la propria chiave privata. Le firme sono prodotte in formato base64, tramite l'algoritmo RSA.

- **Gestione delle credenziali:** il wallet dello studente mantiene localmente le credenziali ricevute (es. *Erasmus Credential* e *Academic Credential*) in file JSON dedicati.
- **Presentazione selettiva:** lo studente può generare una Verifiable Presentation basata sulla *Academic Credential*, includendo solamente un sottoinsieme dei dati (es. solo nome esame e voto, ma non la data), mantenendo tuttavia la verificabilità dei campi nascosti tramite Merkle Proof. Il processo è gestito automaticamente con una configurazione predefinita.
- **Gestione dei DID fidati:** viene mantenuta una lista predefinita di DIDs ritenuti affidabili (es. `did:web:unisa.it`, `did:web:rennes.it`), utilizzata durante la verifica delle presentazioni.
- **Organizzazione su file system:** per ogni studente viene creato un wallet locale con struttura a cartelle, contenente le chiavi (`keys/`), il documento DID (`did/`) e le credenziali (`credentials/`).

La classe supporta quindi un’interazione autonoma dello studente con il sistema decentralizzato, rendendolo capace di ricevere credenziali, firmare dati e presentare in modo selettivo informazioni ad enti terzi.

4.3 Gestione Sicura degli Utenti

La componente di gestione degli utenti si fonda su tre classi principali: `DatabaseManager`, `UserManager` e `PasswordManager`. Queste classi garantiscono la persistenza sicura dei dati, la protezione delle credenziali e la gestione del ciclo di vita degli utenti Erasmus.

4.3.1 DatabaseManager

La classe `DatabaseManager` è responsabile del caricamento, salvataggio e cifratura di un database JSON locale. Ogni istanza gestisce un database dedicato e una chiave di cifratura univoca.

- **Inizializzazione personalizzata:** ogni università (es. UNISA o Rennes) può avere un proprio file database e una propria chiave.
- **Cifratura con Fernet:** i dati vengono cifrati usando Fernet (AES-128 in modalità CBC con HMAC) garantendo confidenzialità e integrità.
- **Recupero automatico:** se il database non esiste o è vuoto, viene creato automaticamente con una struttura base (`"users" : []`).
- **Rigenerazione sicura della chiave:** se la chiave di cifratura è mancante o corrotta, viene generata una nuova chiave.

4.3.2 UserManager

Estendendo DatabaseManager, la classe UserManager implementa tutte le operazioni di registrazione e autenticazione degli utenti Erasmus:

- **Registrazione sicura** (first_login): ogni utente è salvato con ID, nome, cognome, username, hash della password, salt, DID e chiave pubblica PEM.
- **Autenticazione** (authenticate_user): verifica le credenziali confrontando l'hash della password calcolato con quello salvato.
- **Ricerca flessibile**: sono disponibili metodi per recuperare un utente per ID, username o DID.
- **Aggiornamento DID e chiave pubblica**: è possibile aggiornare dinamicamente l'identificatore decentralizzato e la chiave pubblica.

4.3.3 PasswordManager

La classe PasswordManager fornisce operazioni per l'hashing e la verifica delle password degli utenti.

- **Hashing con PBKDF2-HMAC-SHA256**: ogni password è derivata usando 100.000 iterazioni e un salt casuale da 32 byte.
- **Salt base64**: il salt è generato in maniera crittograficamente sicura e codificato in base64.
- **Verifica dell'hash**: la funzione verify_password ricostruisce l'hash e lo confronta con quello salvato.

Questo meccanismo rende le password resistenti e permette un'autenticazione sicura senza mai salvare la password in chiaro.

Struttura dei Dati

Ogni utente è rappresentato da un dizionario JSON criptato, con la seguente struttura:

```
{
  "id": "user123",
  "username": "mario.rossi",
  "hash_password": "...",
  "salt": "...",
  "first_name": "Mario",
  "last_name": "Rossi",
  "did": "did:web:unisa.it:users:user123",
  "public_key_pem": "..."}
```

```

    "created_at": "2025-07-01T10:25:00"
}

```

Tutti i file vengono salvati in directory predefinite, `/database/` per i DB cifrati e `/keys/` per le chiavi di cifratura.

4.4 Blockchain Simulata

La componente `Blockchain` fornisce un'infrastruttura decentralizzata simulata per l'ancoraggio e la verifica di dati crittografici, come hash di credenziali e Merkle Root. Essa non si basa su una rete distribuita reale, ma emula la struttura e le proprietà fondamentali di una blockchain classica, mantenendo i dati in un file JSON condiviso.

4.4.1 Classe Block

La classe `Block` rappresenta un singolo blocco della catena. Ogni blocco contiene:

- **Index:** posizione del blocco nella catena.
- **Timestamp:** data e ora di creazione in formato ISO 8601 UTC.
- **Dati:** contenuto arbitrario, ad esempio una Merkle Root o un identificatore di credenziale.
- **Hash del blocco precedente:** per garantire la continuità e l'integrità della catena.
- **Hash corrente:** calcolato come SHA-256 del contenuto del blocco.

Il metodo `calculate_hash` calcola un hash univoco del blocco, concatenando tutti i suoi campi principali. Ogni blocco è serializzabile in JSON tramite il metodo `to_dict`.

4.4.2 Classe Blockchain

La classe `Blockchain` gestisce l'intera catena di blocchi, salvata nel file `shared_blockchain.json`.

Le funzionalità principali includono:

- **Creazione del blocco genesi:** il primo blocco della catena contiene un messaggio statico e un hash nullo come precedente.
- **Caricamento e salvataggio persistente:** ogni modifica alla catena è riflessa in modo persistente nel file di stato.
- **Validazione della catena:** la funzione `is_chain_valid` verifica che ogni blocco punti correttamente al precedente e che i suoi hash siano coerenti.
- **Aggiunta di blocchi:** nuovi blocchi possono essere aggiunti solo se la catena è valida.

- **Recupero della Merkle Root:** dato un hash di blocco, il metodo `get_merkle_root` restituisce il campo `merkleRoot` se presente all'interno del blocco.

Questa simulazione fornisce un meccanismo affidabile per verificare l'immutabilità e la presenza storica di dati sensibili come credenziali accademiche o revoche.

4.5 Registro di Revoca

La classe `RevocationRegistry` implementa un meccanismo di revoca decentralizzato basato su spazio dei nomi e chiavi hashate. Il registro è persistito in un file JSON chiamato `revocation_registry.json`.

4.5.1 Struttura del Registro

Il registro è strutturato secondo i seguenti livelli:

- **Namespace:** ad esempio `unisa` o `rennes`.
- **List ID:** identificatore hashato (SHA-256) di una lista di revoca, generato da namespace e categoria.
- **Revocation Key:** hash SHA-256 di una credenziale da poter revocare.

Ogni chiave è associata ad un valore booleano che indica se la credenziale è stata revocata (`True`) o meno (`False`).

4.5.2 Funzionalità principali

- **Generazione degli identificatori:** metodi dedicati creano identificatori univoci per liste e chiavi di revoca.
- **Creazione di nuove voci:** il metodo `create_revocation_entry` inizializza una nuova chiave con valore `False`.
- **Revoca:** con il metodo `revoke`, una credenziale viene marcata come revocata.
- **Verifica:** il metodo `is_revoked` consente di interrogare lo stato di una credenziale specifica.
- **Persistenza automatica:** ogni modifica è automaticamente salvata su file.

Il pattern Singleton assicura che esista una sola istanza del registro in esecuzione, condivisa tra tutti i moduli che lo importano.

4.6 Merkle Tree

Per garantire l'integrità e la verificabilità selettiva dei dati contenuti nelle credenziali (es. esami, voti, ecc.), il sistema utilizza una struttura di tipo *Merkle Tree*. Questo approccio consente di fornire prove crittografiche per ogni campo rivelato, senza esporre l'intera credenziale.

4.6.1 Calcolo della Merkle Root

La funzione `merkle_root` prende in input una lista di hash (fogli) e calcola ricorsivamente la **Merkle Root** concatenando i nodi due a due e applicando la funzione hash SHA-256:

- Se il numero di foglie è dispari, l'ultimo elemento viene duplicato per mantenere la struttura binaria completa.
- L'elaborazione continua fino ad ottenere un singolo hash, che rappresenta la root dell'albero.

4.6.2 Hash dei nodi foglia

La funzione `hash_leaf` converte un oggetto `dict` in una stringa JSON ordinata e calcola il suo hash SHA-256. Questo hash rappresenta la foglia dell'albero Merkle.

4.6.3 Costruzione delle Merkle Proof

La funzione `build_merkle_proof` genera una **proof** per una foglia target rispetto all'elenco completo delle foglie. Il risultato è una lista di tuple del tipo `(direction, sibling_hash)` che rappresentano il percorso da una foglia alla radice.

- Ogni passo della proof indica se il nodo fratello si trova a sinistra o a destra rispetto al nodo corrente.
- È possibile ricostruire la Merkle Root conoscendo solo la foglia, la proof e l'algoritmo di hashing.

4.6.4 Verifica delle Merkle Proof

La funzione `verify_merkle_proof` prende una foglia hashata, una proof, e una Merkle Root, e verifica se la proof conduce correttamente alla root:

- Per ogni livello della proof, si concatena l'hash del fratello secondo la direzione indicata.
- Alla fine, il risultato viene confrontato con la root attesa.

4.6.5 Ricostruzione della Merkle Root

La funzione `reconstruct_merkle_root` riceve in input una struttura a dizionario che rappresenta una presentazione parziale di credenziali (es. solo alcuni campi di alcuni esami). Essa calcola gli hash foglia per ogni campo rivelato e successivamente genera la Merkle Root corrispondente.

- Se viene fornito il valore originale, l'hash viene calcolato al momento.
- Se è disponibile un hash pre-calcolato (`leafHash`), questo viene usato direttamente.

Questa funzionalità è fondamentale per l'integrazione con sistemi di verifica delle università, che possono così autenticare una presentazione parziale confrontando la root generata con quella registrata sulla blockchain.

4.7 Main

Il file `main.py` orchestra l'intero ciclo di vita delle Verifiable Credentials nel sistema, simulando l'interazione tra studenti, università e blockchain in conformità con il paradigma issuer–holder–verifier. Le fasi che compongono questo processo coprono l'intero flusso, dall'inizializzazione fino alla revoca e al tentativo di riutilizzo delle credenziali.

L'esecuzione del programma si apre con una fase preliminare dedicata alla pulizia dei dati, durante la quale vengono eliminate le directory contenenti dati residui delle esecuzioni precedenti. Questa operazione prepara un ambiente pulito per la successiva inizializzazione. Segue quindi la fase 0, in cui vengono generate e salvate nuove chiavi di crittografia per le università coinvolte, UniSA e Rennes, e creati i relativi database di utenti. Viene verificata la validità della blockchain, e completata la configurazione iniziale delle due università.

FASE 0: INIZIALIZZAZIONE

Nuova chiave di crittografia generata e salvata in

`\keys\unisa_users_encryption.key`

Database salvato correttamente in

`\database\unisa_users.json`

La blockchain è valida.

Chiavi per did:web:unisa.it generate e salvate.

Nuova chiave di crittografia generata e salvata in

`\keys\rennes_users_encryption.key`

Database salvato correttamente in

`\database\rennes_users.json`

La blockchain è valida.

Università inizializzate correttamente.

La fase 1 è dedicata alla creazione e registrazione degli studenti nel sistema. Per ciascuno di essi, vengono generate nuove chiavi crittografiche, creati i documenti DID, e registrati sia nel database di UniSA che in quello di Rennes. Questa fase termina solo dopo che tutti gli studenti sono stati registrati correttamente. Durante la fase 2 gli studenti si autenticano con UniSA, utilizzando un meccanismo di challenge-response che conferma la loro identità. Tutti gli studenti ottengono esito positivo nelle procedure di autenticazione.

FASE 1: CREAZIONE E REGISTRAZIONE STUDENTI

Registro lo studente 1: mario.rossi

Nuove chiavi generate per lo studente mario.rossi.

Documento DID salvato in c:\...\wallet-mario.rossi\did\...

...

Tutti gli studenti registrati.

FASE 2: AUTENTICAZIONE CON UNISA

Autenticazione di mario.rossi...

Autenticazione riuscita per mario.rossi

...

La fase 3 riguarda la richiesta e l'emissione delle credenziali Erasmus da parte di UniSA. Ogni studente richiede la propria credenziale, che viene generata, firmata, salvata nel wallet dello studente e ancorata alla blockchain, dopo averne verificato la validità.

FASE 3: CREDENZIALE ERASMUS

mario.rossi richiede credenziale Erasmus

La blockchain è valida.

Credenziale Erasmus salvata

Credenziale Erasmus generata

...

Nella fase 4 gli studenti si autenticano con Rennes per richiedere l'emissione della credenziale accademica. L'università verifica la validità della credenziale Erasmus ricevuta, confermandone la firma, la validità temporale e lo stato di non revoca. Successivamente, Rennes genera la credenziale accademica per ciascuno studente, la firma, la salva nel rispettivo wallet e ne registra la root del Merkle Tree sulla blockchain.

FASE 4: RICHIESTA VOTI A RENNES

Esami caricati con successo da exams.json

mario.rossi si autentica con Rennes

Firma credenziale Erasmus valida

La credenziale è ancora valida temporalmente

Credenziale non revocata

La blockchain è valida

Credenziale accademica emessa e salvata in wallet-mario.rossi

Credenziale accademica generata

...

La fase 5 consiste nella generazione e presentazione selettiva delle credenziali accademiche da parte degli studenti a UniSA. Ogni studente crea una Verifiable Presentation che include solo i dati selezionati, accompagnata dalle Merkle Proof necessarie. UniSA verifica la validità della presentazione controllando firma, validità temporale, Merkle Proof, corrispondenza della Merkle Root con quella on-chain e lo stato di revoca, e convalida con successo ogni presentazione.

FASE 5: PRESENTAZIONE SELETTIVA

Generazione presentazione per mario.rossi

Verifiable Presentation salvata in wallet-mario.rossi

Presentazione generata

UniSA verifica la presentazione di mario.rossi

Firma valida

La credenziale è ancora valida temporalmente

Merkle Root recuperata

Credenziale non revocata

Tutte le Merkle proof sono valide

Merkle Root ricostruita corrisponde alla root on-chain

Presentazione valida

...

La fase 6 riguarda la revoca delle credenziali da parte di entrambe le università. UniSA revoca le credenziali Erasmus, mentre Rennes revoca le credenziali accademiche. Le informazioni sulla revoca vengono registrate sulla blockchain in modo immutabile.

FASE 6: REVOCA CREDENZIALI

Credenziale revocata correttamente per UniSA

Erasmus revocata

Credenziale revocata correttamente per Rennes

Credenziale accademica revocata

Infine, nella fase 7 gli studenti tentano di riutilizzare le credenziali che sono state revocate. Sebbene le firme e la struttura formale siano ancora valide, il sistema riconosce correttamente lo stato di revoca e nega l'accettazione delle presentazioni, dimostrando il corretto funzionamento del meccanismo di revoca implementato.

FASE 7: RIUTILIZZO DOPO REVOCA

mario.rossi riprova a usare la credenziale Erasmus

Firma valida

La credenziale è ancora valida temporalmente

La credenziale è stata revocata

Revoca Erasmus funzionante (non è più valida)

anna.bianchi riprova a usare la presentazione accademica

Firma valida

La credenziale è ancora valida temporalmente

Merkle Root recuperata

La credenziale è stata revocata

Revoca accademica funzionante (non è più valida)

Questa sequenza completa delle fasi mostra un flusso robusto e conforme agli standard W3C, in cui vengono assicurati autenticità, integrità, privacy, e verificabilità decentralizzata delle Verifiable Credentials.

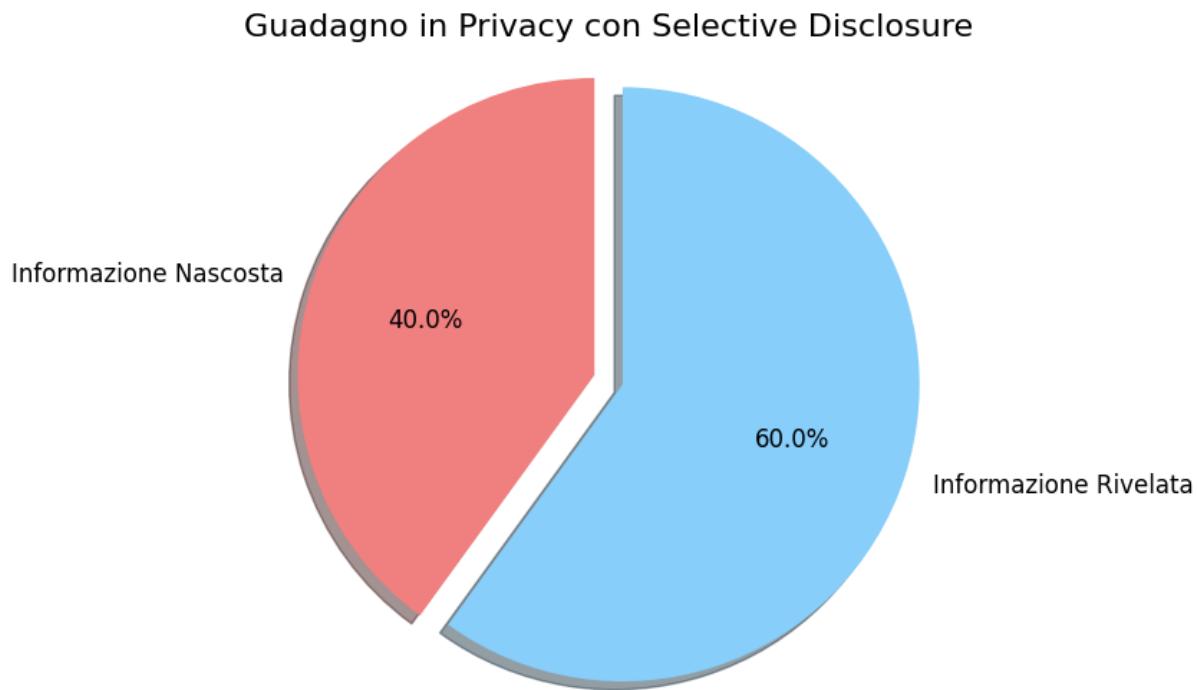
4.8 Analisi delle Prestazioni

Fase di Analisi 1: Guadagno in Privacy vs Full Disclosure

Il grafico a torta mostra la percentuale di attributi rivelati rispetto al totale degli attributi presenti in una credenziale accademica completa. In questo scenario, la credenziale contiene 50 attributi complessivi, di cui 30 sono stati rivelati nella presentazione selettiva. Ciò si traduce in un **guadagno in privacy del 40%** (ovvero, il 40% degli attributi è rimasto nascosto) e in una **percentuale di informazione rivelata pari al 60%**.

Tuttavia, va sottolineato che questo non rappresenta il caso ottimale dal punto di vista della privacy: i soli attributi necessari per la verifica del superamento dell'esame sono l'identificativo dell'esame e il voto conseguito. Altri campi come il nome dell'esame o il numero di crediti possono essere considerati ridondanti, in quanto già presenti nel *Learning Agreement*. Se si rivelassero solo questi due attributi minimi per ciascun esame, la percentuale di informazione rivelata scenderebbe ulteriormente, facendo salire il guadagno in privacy fino al **60%**.

Questo esempio evidenzia come l'uso della *selective disclosure* consenta un controllo fine sulla quantità di dati condivisi, permettendo una maggiore tutela della privacy dello studente pur garantendo la validità delle informazioni rivelate.



Fase di Analisi 2: Efficienza Dimensionale

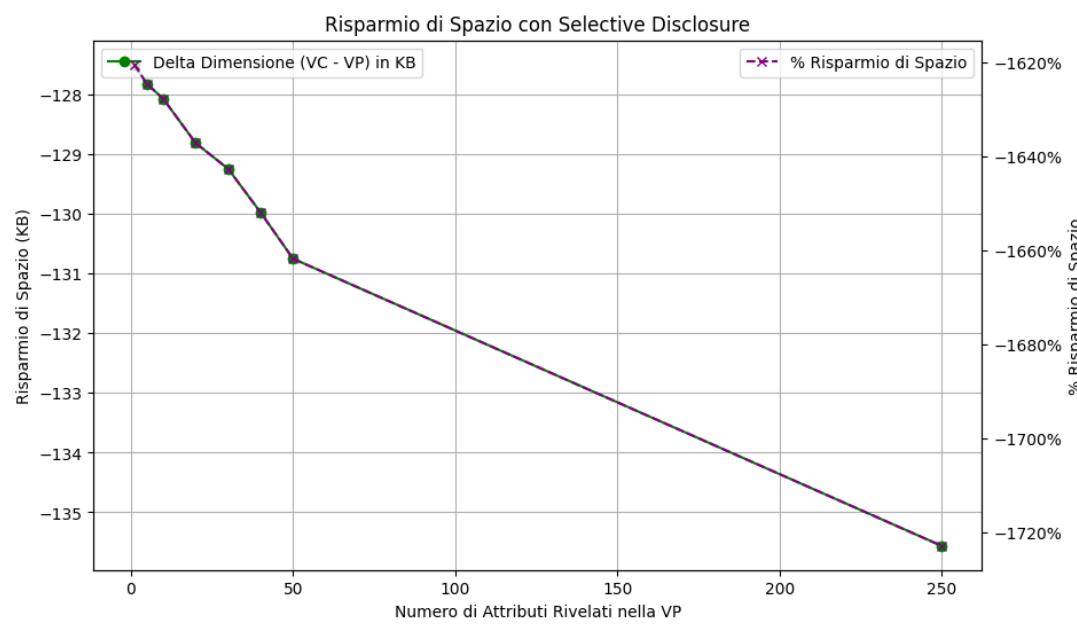
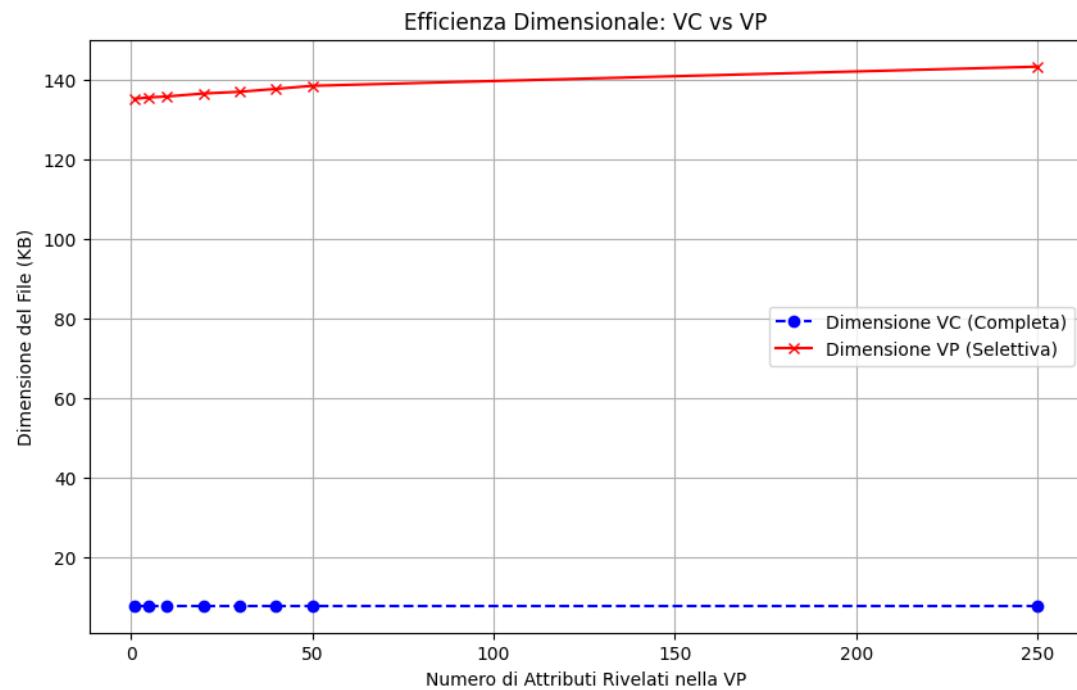
L’obiettivo di questa analisi è valutare l’efficienza in termini di spazio della presentazione selettiva (*Verifiable Presentation*, VP) rispetto alla credenziale completa (*Verifiable Credential*, VC), al variare del numero di attributi divulgati.

A partire da una VC accademica contenente 50 esami finti, sono state generate diverse configurazioni di presentazione selettiva con un numero crescente di attributi rivelati. Per ogni configurazione, è stata prodotta la relativa prova di validità e calcolata la sua dimensione in KB, quindi confrontata con quella della VC originale.

I risultati mostrano un comportamento controllato: all’aumentare degli attributi rivelati, la dimensione della VP cresce significativamente, arrivando addirittura a superare di molto quella della VC originale. Con 50 esami, la VP ha raggiunto una dimensione di 143.02 KB, a fronte di soli 7.88 KB della VC, con un “risparmio” negativo del –1689.13%.

Questo comportamento si spiega osservando che l’overhead introdotto dal formato della presentazione selettiva (in particolare i metadati, la struttura di disclosure) può diventare sproporzionato rispetto alla dimensione reale delle informazioni rivelate. Il nostro approccio, pur offrendo un guadagno in termini di privacy e controllo, mostra quindi un costo in termini di efficienza dimensionale quando viene utilizzato per divulgare un numero elevato di attributi.

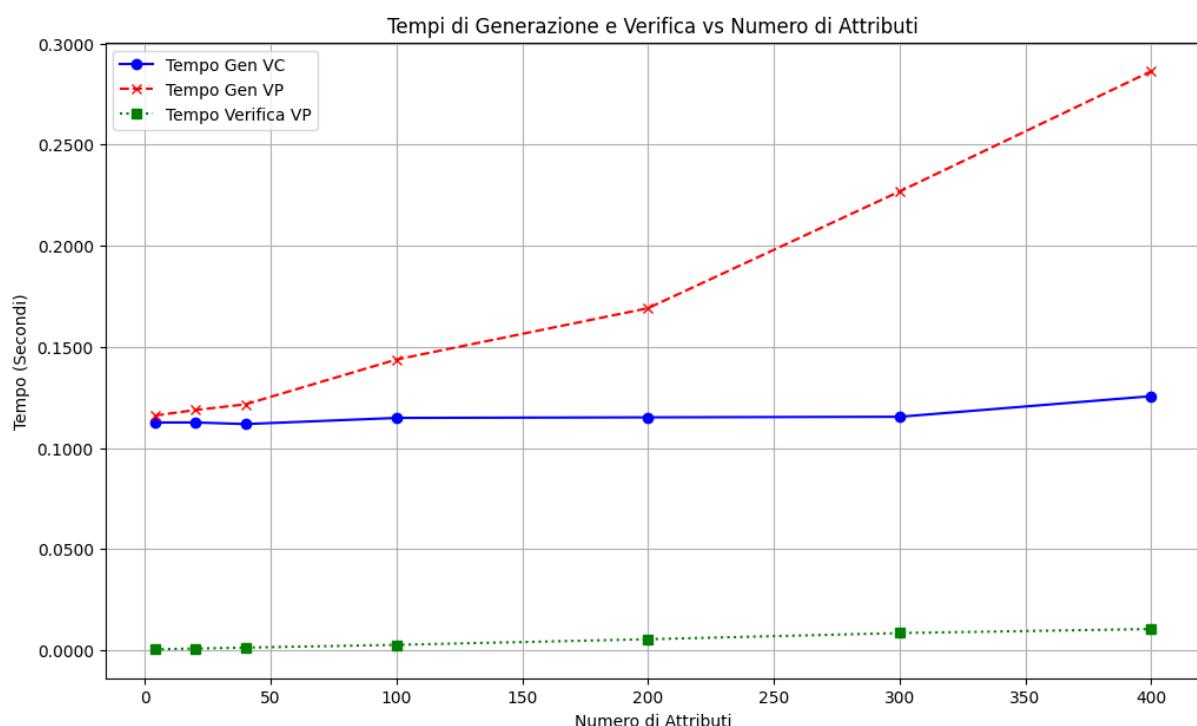
In contesti in cui la minimizzazione della dimensione dei messaggi è critica, sarà dunque fondamentale limitare il numero di attributi rivelati o ottimizzare ulteriormente la serializzazione e la struttura della VP.



Fase di Analisi 3: Tempi di Generazione e Verifica

Questa fase del main ha lo scopo di valutare l'impatto del numero di attributi contenuti nella credenziale accademica sui tempi di generazione della credenziale stessa, sui tempi di generazione della presentazione selettiva e sui tempi di verifica da parte dell'università ricevente. L'analisi viene effettuata tramite la generazione artificiale di un insieme di esami fintizi, simulando così diversi scenari di complessità crescente.

I risultati delle tre misurazioni vengono infine tracciati su un grafico, con il numero totale di attributi sull'asse delle ascisse e i tempi (in secondi) sull'asse delle ordinate. Le curve ottenute rappresentano, rispettivamente, il tempo di generazione della credenziale (Gen VC), il tempo di generazione della presentazione selettiva (Gen VP) e il tempo di verifica da parte dell'ateneo. Mostrando come la verifica beneficia dell'utilizzo del Merkle Tree.



4.9 Aree di Miglioramento

Un’importante area di miglioramento per il sistema di divulgazione selettiva implementato nel progetto riguarda l’introduzione di un salt per ciascuna foglia del Merkle Tree utilizzato nella costruzione delle Verifiable Credentials. Attualmente, l’hashing diretto dei valori dei claim può esporre il sistema a rischi per la privacy. Se un attaccante o un Verifier malintenzionato conoscesse o riuscisse a indovinare il contenuto di determinati attributi (come un voto o il nome di un corso), potrebbe calcolare gli hash corrispondenti e confrontarli con quelli presenti nella credenziale. In tal modo, sarebbe possibile inferire informazioni che lo studente ha scelto di non rivelare o correlare tra loro diverse credenziali, compromettendo la riservatezza dei dati accademici.

Per mitigare questo rischio, si propone l’aggiunta di un valore casuale e univoco (salt) ad ogni singolo claim prima dell’applicazione dell’hash e della costruzione del Merkle Tree. Il salt verrebbe generato dall’Issuer (nel progetto, l’Università di Rennes) al momento dell’emissione della Verifiable Credential, e consegnato allo Holder (lo studente) insieme al valore in chiaro del claim. Quando lo studente desidera rivelare un attributo specifico, includerà nella Verifiable Presentation non solo il valore dell’attributo, ma anche il relativo salt. Il Verifier (es. l’Università di Salerno), ricevendo entrambi, potrà ricalcolare l’hash corretto e verificarne l’inclusione nell’albero di Merkle tramite la prova crittografica. Per gli attributi non rivelati, il Verifier riceverà esclusivamente l’hash “saltato” e la Merkle proof, senza accesso al valore originale né al salt, impedendo così ogni tentativo di ricostruzione.

4.10 Versione 1.1

La versione aggiornata del codice utilizza un salt deterministico, generato a livello di protocollo e diverso per ogni foglia, concatenando le informazioni:

{exam_id}:{field}:{value}:{student.did}.

Rispetto alla versione 1.0, non retrocompatibile, è stata modificata anche la modalità di costruzione del Merkle Tree: oltre ai dati relativi agli esami, ora viene incluso anche il campo credentialStatus. Questa scelta è stata introdotta per preservare l’integrità di tale informazione durante la presentazione selettiva: poiché la presentazione non riporta direttamente l’intera credenziale firmata, il campo di stato viene inserito come ulteriore claim e protetto tramite la struttura Merkle, costruendo così un albero composto dall’insieme delle foglie degli esami e dei metadati di stato della credenziale.