

# UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE ED ELETTRICA E  
MATEMATICA APPLICATA



Corso di Laurea Magistrale in Ingegneria Informatica

## Decentralized E-commerce Review System

reverseEkans

Nome	Cognome	Matricola	E-Mail	Responsabile
Giuseppe Alfonso	Mangiola	0622702372	g.mangiola1@studenti.unisa.it	WP1
Emanuele	Relmi	0622702368	e.relmi@studenti.unisa.it	WP2
Francesco	Quagliuolo	0622702412	f.quagliuolo@studenti.unisa.it	WP3

ANNO ACCADEMICO 2024/2025

---

# INDICE

<b>1</b>	<b>WORK PACKAGE 1</b>	<b>5</b>
1.1	Descrizione del modello . . . . .	5
1.2	Attori del Sistema . . . . .	6
1.2.1	Utente . . . . .	6
1.2.2	Venditori . . . . .	6
1.2.3	La piattaforma di e-commerce . . . . .	6
1.2.4	Validator della blockchain . . . . .	6
1.3	Attaccanti del Sistema . . . . .	7
1.4	Proprietà . . . . .	11
1.4.1	Confidenzialità . . . . .	11
1.4.2	Privacy . . . . .	11
1.4.3	Integrità . . . . .	12
1.4.4	Trasparenza . . . . .	13
1.4.5	Efficienza . . . . .	13
1.5	Completeness . . . . .	13
<b>2</b>	<b>WORK PACKAGE 2</b>	<b>15</b>
2.1	Architettura generale del sistema . . . . .	16
2.1.1	Componenti principali del sistema . . . . .	16
2.1.2	Modalità di gestione della reputazione . . . . .	17
2.2	Funzionamento delle parti oneste . . . . .	18
2.2.1	Gestione utente e creazione del DID . . . . .	18
2.2.2	Venditore . . . . .	19
2.2.3	Interazione dell'utente con il sistema tramite DID . . . . .	19
2.3	Utilizzo di NFT come prova di acquisto . . . . .	21
2.3.1	Emissione del token . . . . .	22

2.3.2	Verifica e pubblicazione della recensione . . . . .	22
2.3.3	NFT scaduti . . . . .	23
2.3.4	Vantaggi della soluzione NFT . . . . .	23
2.4	Uso delle Verifiable Credentials nel sistema . . . . .	24
2.4.1	Emissione della Verifiable Credential . . . . .	24
2.4.2	Presentazione e verifica . . . . .	24
2.4.3	Gestione della Revoca delle VC . . . . .	25
2.4.4	Privacy e Selective Disclosure . . . . .	26
2.4.5	Compromesso tra sicurezza e costo . . . . .	26
2.5	Gestione dei casi limite . . . . .	26
2.5.1	Utenti inattivi . . . . .	26
2.5.2	VC scadute o revocate . . . . .	27
2.5.3	DID revocati o compromessi . . . . .	27
2.5.4	Utente Bannato . . . . .	27
2.6	Gestione del compromesso tra pseudo-anonimato e unicità dell'utente . . . . .	28
2.6.1	Verifiable Credential univoca . . . . .	28
2.6.2	DID multipli, ma dimostrabilmente unificati . . . . .	28
2.6.3	Struttura della ZKP anti-Sybil . . . . .	28
2.6.4	Risultato del compromesso . . . . .	29
2.7	Smart Contract . . . . .	29
2.7.1	Funzionalità implementate . . . . .	29
2.8	Validator . . . . .	30
2.8.1	Compiti principali . . . . .	30
2.8.2	Assunzione di sicurezza . . . . .	31
2.9	Modulo di Audit . . . . .	31
2.9.1	Funzionalità . . . . .	31
2.10	Scelta della blockchain permissionless . . . . .	32
2.10.1	Ethereum vs Hyperledger Fabric . . . . .	33
2.10.2	Innovazioni future . . . . .	34
2.11	Utilizzo per l'utente (dApp) . . . . .	34
2.12	Formalismi matematici crittografici . . . . .	35
2.12.1	Generazione DID . . . . .	35
2.12.2	Firma digitale . . . . .	35
2.12.3	Verifiable Credential (VC) . . . . .	35
2.12.4	Verifiable Presentation (VP) . . . . .	36
2.12.5	Revoca con bitstring . . . . .	36
2.12.6	Zero-Knowledge Proof (ZKP) . . . . .	36
2.12.7	Selective Disclosure con BBS+ . . . . .	36
2.13	Conclusione . . . . .	37

---

<b>3</b>	<b>WORK PACKAGE 3</b>	<b>38</b>
3.1	Confidenzialità . . . . .	38
3.1.1	C1 – Protezione dell’identità reale nelle recensioni . . . . .	38
3.1.2	C2 – Comunicazioni cifrate tra client e piattaforma . . . . .	39
3.1.3	C3 – Minimizzazione dei metadati esposti . . . . .	40
3.1.4	C4 – Isolamento tra attività dell’utente . . . . .	41
3.2	Privacy . . . . .	42
3.2.1	P1 – Minimizzazione dei dati . . . . .	42
3.2.2	P2 – Verificabilità condizionata (Predicate Disclosure) . . . . .	42
3.2.3	P3 – Non tracciabilità (Unlinkability) . . . . .	43
3.2.4	P4 – Non riutilizzabilità (Non-transferability) . . . . .	44
3.2.5	P5 – Non falsificabilità (Unforgeability) . . . . .	44
3.2.6	P6 – Protezione contro correlazioni (Untraceability) . . . . .	45
3.2.7	P7 – Resistenza alla sorveglianza passiva (Unobservability) . . . . .	45
3.3	Integrità . . . . .	46
3.3.1	I1 – Immutabilità delle recensioni . . . . .	46
3.3.2	I2 – Verifica dell’acquisto . . . . .	47
3.3.3	I3 – Firma e non ripudio . . . . .	47
3.3.4	I4 – Prevenzione delle aggregazioni fraudolente . . . . .	48
3.3.5	I5 – Ordine cronologico garantito . . . . .	48
3.4	Trasparenza . . . . .	49
3.4.1	T1 – Algoritmi di visibilità pubblici e immutabili . . . . .	49
3.4.2	T2 – Accesso pubblico alle transazioni . . . . .	50
3.4.3	T3 – Regole di modifica e revoca predefinite . . . . .	50
3.4.4	T4 – Fiducia tecnica nelle componenti terze . . . . .	51
3.5	Efficienza . . . . .	51
3.5.1	E1 – Verifica dell’acquisto rapida . . . . .	51
3.5.2	E2 – Pubblicazione fluida e a basso costo . . . . .	52
3.5.3	E3 – Ottimizzazione delle operazioni crittografiche . . . . .	53
3.5.4	E4 – Scalabilità e prestazioni stabili . . . . .	53
3.6	Completezza . . . . .	54
3.6.1	Comportamento corretto in assenza di attacchi . . . . .	54
3.7	Limiti strutturali del sistema . . . . .	55
<b>4</b>	<b>WORK PACKAGE 4</b>	<b>58</b>
4.1	Smart contracts . . . . .	58
4.2	Deploy scripts . . . . .	59
4.3	Interact scripts (CLI) . . . . .	60
4.4	Interazione via GUI . . . . .	61

## INDICE

---

4.4.1	Server Backend Express . . . . .	62
4.4.2	Identity Provider Mock . . . . .	63
4.4.3	Frontend React (dApp) . . . . .	64
4.5	Come testare . . . . .	66
4.6	Prestazioni del Sistema . . . . .	67
4.6.1	Benchmark di Consumo di Gas e Tempo di Esecuzione . . . . .	67
4.6.2	Efficienza del Consumo di Gas e del Tempo di Esecuzione . . . . .	67
4.6.3	Valutazione delle Prestazioni . . . . .	68
4.6.4	Analisi del Gas e del Tempo: Confronto con Sistemi Tradizionali . . . . .	68
4.7	Limitazioni e sviluppi futuri . . . . .	69
4.7.1	Limitazioni attuali . . . . .	69
4.7.2	Evoluzioni possibili . . . . .	69
4.8	Conclusione . . . . .	70

---

---

# CAPITOLO 1

---

## WORK PACKAGE 1

In questo primo Work Package definiamo il modello di riferimento per il sistema decentralizzato di recensioni. Identifichiamo innanzitutto gli attori onesti coinvolti nel sistema, analizzando i loro obiettivi e le funzionalità che si intendono realizzare.

Gli attori onesti sono quegli individui o entità che agiscono in conformità con le regole e le politiche stabilite, cercando di raggiungere i loro obiettivi senza compromettere l'integrità del sistema. Successivamente, esamineremo i possibili avversari (o Threat Models) che potrebbero essere interessati a compromettere il sistema, esaminando le loro risorse e le motivazioni che li spingono ad agire. Questa analisi ci permetterà di identificare gli attacchi che il sistema potrebbe subire e di comprendere quali misure di sicurezza dovranno essere adottate per contrastarli. Una volta compreso il contesto in cui il sistema opera, identificheremo le proprietà di sicurezza che si vorrebbero preservare in presenza di attacchi. Queste proprietà sono fondamentali per garantire il corretto funzionamento del sistema e la protezione delle informazioni sensibili.

### 1.1 Descrizione del modello

Il sistema che intendiamo progettare mira a decentralizzare la gestione delle recensioni all'interno delle piattaforme di e-commerce, sfruttando una blockchain per garantire:

- autenticità delle recensioni
- trasparenza nell'ordinamento e nella visibilità delle recensioni
- immutabilità delle valutazioni una volta registrate (se non nelle condizioni di modifica/revoca definite)
- incentivazione alla partecipazione onesta

Gli utenti devono essere verificati come effettivi acquirenti di un determinato prodotto prima di poter rilasciare una recensione, le quali saranno poi memorizzate sulla blockchain. Il sistema prevede anche meccanismi anti-frode, come la prevenzione da account falsi e la penalizzazione dell'inattività o dei comportamenti manipolativi.

## 1.2 Attori del Sistema

### 1.2.1 Utente

Gli utenti acquirenti sono soggetti che effettuano acquisti reali tramite la piattaforma. Dopo aver completato una transazione, essi hanno il diritto di rilasciare una recensione sul prodotto acquistato. L'obiettivo dell'utente è condividere la propria esperienza di acquisto in modo onesto, contribuendo così a costruire una base di recensioni affidabili per la comunità. L'utente deve poter contare su un sistema che protegga la sua identità tramite forme di pseudo-anonimato, pur garantendo la verificabilità della sua legittimità come recensore effettivo. Inoltre, deve avere la possibilità di modificare o revocare la propria recensione secondo condizioni predefinite e di contribuire alla qualità del sistema tramite la pubblicazione di recensioni oneste e tempestive.

### 1.2.2 Venditori

I venditori sono attori che propongono prodotti all'interno del marketplace. Il loro interesse principale è ottenere feedback autentici sui propri articoli, che riflettano realmente la qualità dei beni offerti. Attraverso il sistema di recensioni decentralizzate, il venditore può migliorare la propria reputazione in modo trasparente e meritocratico. I venditori onesti si impegnano a non tentare di manipolare il sistema, né incentivando recensioni false, né ostacolando recensioni negative legittime.

### 1.2.3 La piattaforma di e-commerce

La piattaforma di e-commerce rappresenta l'interfaccia principale tramite cui avviene l'interazione tra utenti e prodotti. Essa si occupa di verificare localmente l'avvenuto acquisto e di emettere un attestato crittografico (NFT non trasferibile) che costituisce la Proof-of-Purchase (PoP), poi utilizzata per autorizzare la pubblicazione di recensioni. Inoltre, la piattaforma fornisce accesso agli smart contract e alla rete blockchain tramite la dApp, ma non controlla direttamente né l'ordinamento né la registrazione delle recensioni, che sono gestiti in modo autonomo e trasparente dalla logica on-chain. La piattaforma deve essere progettata per non poter alterare la visibilità delle recensioni, garantendo l'imparzialità e la non manipolabilità del sistema.

### 1.2.4 Validator della blockchain

I validators sono attori fondamentali per il corretto funzionamento dell'infrastruttura blockchain su cui si basa il sistema di recensioni per l'e-commerce. Il loro compito consiste

nel validare e inserire in blocchi le transazioni effettuate dagli utenti, tra cui l'inserimento delle recensioni e le eventuali revoche o modifiche autorizzate. Nel modello proposto, si assume che i validators si comportino in modo corretto, ovvero eseguano le operazioni previste dal protocollo di consenso in modo imparziale e senza manipolazioni. In particolare, si presume che i blocchi prodotti rispettino l'ordine temporale delle transazioni, che non vi siano censure arbitrarie e che le operazioni valide non vengano rifiutate. I validators onesti rappresentano la garanzia che le regole del sistema vengano applicate in modo coerente, assicurando proprietà fondamentali come l'immutabilità, la tracciabilità e la trasparenza dei dati memorizzati sulla blockchain.

### 1.3 Attaccanti del Sistema

Nel contesto del sistema descritto, è fondamentale analizzare e comprendere i potenziali attaccanti che potrebbero cercare di comprometterlo, considerando le motivazioni e le risorse a loro disposizione. Questi attori e gruppi di attaccanti rappresentano diverse minacce al sistema e richiedono misure di sicurezza specifiche per essere contrastati efficacemente.

- **Utente fraudolento**

- **Descrizione:** Si tratta di un avversario che cerca deliberatamente di compromettere l'affidabilità del sistema utilizzando una singola identità per inviare recensioni false o fuorvianti. Può tentare di ottenere un NFT senza aver realmente effettuato un acquisto (eg. tramite exploit della piattaforma o uso non autorizzato di NFT altrui), oppure utilizzare una Verifiable Credential compromessa o falsificata. Il suo obiettivo è alterare la reputazione dei prodotti senza possedere un diritto legittimo alla recensione.
- **Risorse**
  - \* Capacità tecniche medie per aggirare i meccanismi di verifica
  - \* Uso di NFT ottenuti fraudolentemente o riutilizzati
  - \* Possesso di VC rubate o mal rilasciate

- **Venditore malevolo**

- **Descrizione:** è un attore che, pur facendo parte legittimamente del sistema, tenta di manipolarne il funzionamento per aumentare artificialmente la propria reputazione. Può commissionare recensioni false a utenti fraudolenti o tentare di incentivare solo recensioni positive offrendo sconti o regali. Talvolta agisce anche in modo sleale contro concorrenti, cercando di sabotarne la reputazione con feedback negativi fittizi.
- **Risorse**
  - \* Budget economico per incentivare utenti malintenzionati
  - \* Reti di profili social o identità multiple
  - \* Accesso a piattaforme esterne per organizzare campagne coordinate



- **Attaccante Sybil**

- **Descrizione:** Un attaccante Sybil genera un gran numero di identità pseudo-anonime (DID) ciascuna associata a una Verifiable Credential apparentemente valida, con l'obiettivo di manipolare il sistema scrivendo più recensioni false, creando così un'impressione artificiale di consenso o popolarità.
- **Risorse**
  - \* Automazione per la creazione di molteplici DID e wallet
  - \* Accesso a più VC emesse fraudolentemente o a issuer compromessi
  - \* Capacità di orchestrare pubblicazioni multiple coordinate

- **Identity Thief**

- **Descrizione:** avversario che possiede o mira a rubare identità e/o credenziali per impersonare gli utenti legittimi e ottenere accesso non autorizzato ai servizi. Potrebbe utilizzare tecniche di phishing, furto di credenziali o exploit delle vulnerabilità per ottenere informazioni sensibili e assumere l'identità di altri utenti.
- **Risorse**
  - \* Disponibilità di risorse computazionali sufficienti per eseguire attacchi brute-force, decrittazione o altri metodi per ottenere informazioni sensibili o compromettere la sicurezza del sistema
  - \* Possesso di un vasto database di informazioni personali ottenute illegalmente, che consente la creazione di profili dettagliati delle vittime e di utilizzare le informazioni per scopi malevoli
  - \* Competenze nel phishing e nell'ingegneria sociale, capacità di creare siti web e messaggi convincenti

- **Insider malevolo**

- **Descrizione:** si tratta di un utente reale che ha accesso legittimo al sistema (acquirente o venditore), ma che sfrutta tale accesso per aggirare le regole, ad esempio vendendo le proprie credenziali. Può anche collaborare con un venditore per manipolare le recensioni in cambio di vantaggi.
- **Risorse**
  - \* Accesso autentico al sistema
  - \* Conoscenza delle regole di funzionamento
  - \* Comunicazioni private con altri attori coinvolti

- **Reviewer a pagamento**

- **Descrizione:** è un attore esterno che fornisce recensioni dietro compenso, fingendo esperienze d’acquisto mai avvenute. Opera in coordinamento con venditori maliziosi o agenzie di marketing scorrette. L’attività si basa sull’acquisizione o l’aggiramento delle prove d’acquisto (PoP), sfruttando identità multiple o strumenti per eludere i controlli anti-frode del sistema.
- **Risorse**
  - \* Identità multiple o prestanome
  - \* Esperienza nel mascherare pattern ripetitivi
  - \* Account acquistati o affittati già “invecchiati” per aggirare controlli
- **Validator corrotto**
  - **Descrizione:** in una blockchain, i validators hanno il compito di includere o rifiutare transazioni. Un validator corrotto può censurare recensioni sgradite o includere solo quelle sponsorizzate, sabotando la neutralità del sistema.
  - **Risorse**
    - \* Accesso diretto al consenso del sistema
    - \* Collusione con altri validatori o attori
    - \* Capacità di censura selettiva e invisibile
- **Venditore Collusivo Fuori Sistema**
  - **Descrizione:** si tratta di un venditore che, pur operando apparentemente in modo regolare sulla piattaforma, contatta l’acquirente attraverso canali esterni (eg. email, social network, app di messaggistica) dopo l’acquisto, proponendo un rimborso parziale o totale in cambio di una recensione positiva. L’obiettivo di questo comportamento è quello di ottenere feedback favorevoli per migliorare la propria reputazione pubblica, senza che l’incentivo illecito venga rilevato dal sistema di recensioni. Dal punto di vista del protocollo, la recensione appare autentica (l’utente ha realmente acquistato il prodotto), ma è in realtà distorta da un accordo economico privato non verificabile, che mina la trasparenza e l’affidabilità del sistema.
  - **Risorse**
    - \* Accesso ai dati di contatto del cliente post-acquisto
    - \* Canali alternativi (email, social, chat) per negoziare incentivi non tracciabili
    - \* Capacità di effettuare rimborsi discreti tramite metodi esterni al sistema (PayPal, crypto, buoni regalo, ecc.)
- **Attaccante di tipo phishing/malware**
  - **Descrizione:** un avversario che tenta di ottenere l’accesso alle chiavi private di un utente tramite phishing o malware. Compromette il controllo su wallet o dispositivi,

può firmare transazioni arbitrarie, inviare recensioni fraudolente o riscattare incentivi legittimi altrui.

– **Risorse**

- \* Capacità tecniche medie per creare pagine clone, app dannose o estensioni malevole.
- \* Accesso a database di email/identità per campagne phishing mirate.
- \* Malware per keylogging, intercettazione clipboard o inject di firma.

• **Issuer compromesso**

- **Descrizione:** un'entità formalmente affidabile (Issuer) che, a seguito di compromissione o comportamento malevolo, emette Verifiable Credentials (VC) false o duplicate. Queste credenziali possono essere utilizzate per creare account multipli o alterare l'equilibrio reputazionale del sistema.

– **Risorse**

- \* Accesso al proprio sistema di emissione VC e alla chiave privata di firma.
- \* Collaborazione con attori esterni che utilizzano le VC emesse.
- \* Capacità di firmare VC formalmente valide ma semanticamente scorrette.

• **Attaccante passivo di rete**

- **Descrizione:** un osservatore che intercetta comunicazioni, come l'invio di una Verifiable Presentation, tra utente e smart contract. Se non viene usata una challenge o un nonce dinamico, può riutilizzare una presentazione firmata per replicare un'azione (replay attack), anche senza accedere alla chiave privata dell'utente.

– **Risorse**

- \* Capacità di intercettare pacchetti o monitorare traffico su canali insicuri.
- \* Tool di replay automatico e scripting.
- \* Accesso alla rete o a browser compromessi.

• **Analista comportamentale (Data Correlator)**

- **Descrizione:** attore passivo che, senza alterare il sistema, analizza in modo massivo metadati pubblici (timestamp, hash, CID IPFS, ecc.) per tentare di correlare le azioni di uno stesso utente su DID diversi, violando l'unlinkability. Può impiegare tecniche di fingerprinting, clustering temporale o analisi predittiva, agendo come "profilatore" degli utenti.

– **Risorse**

- \* Accesso completo ai dati pubblici della blockchain e dei CID IPFS.
- \* Competenze statistiche e di data mining.
- \* Capacità di analisi AI per identificare pattern temporali e linguistici.

## **1.4 Proprietà**

L'obiettivo di questo progetto è sviluppare un sistema di recensioni decentralizzato per l'e-commerce, capace di offrire un ambiente affidabile, trasparente e resistente a manipolazioni. Il sistema permette agli utenti di recensire prodotti soltanto dopo averne dimostrato l'acquisto tramite meccanismi di verifica e garantisce che tali recensioni siano pubblicate in modo immutabile e consultabile da tutti. Allo stesso tempo, è previsto il supporto per la modifica o revoca delle recensioni secondo condizioni predefinite con logiche di incentivazione/disincentivazione. Per garantire la robustezza del sistema in un ambiente basato su blockchain è necessario definire con precisione alcune proprietà fondamentali: confidenzialità, privacy (selective disclosure), integrità, trasparenza ed efficienza. Queste proprietà sono essenziali non solo per il corretto funzionamento del sistema in condizioni normali, ma anche per assicurare la resilienza rispetto a comportamenti malevoli, come la creazione di identità false, la compravendita di feedback o la manipolazione della visibilità delle recensioni. La loro implementazione costituisce la base per un modello credibile, meritocratico e decentralizzato di reputazione online.

### **1.4.1 Confidenzialità**

- **C1:** L'identità reale dell'utente non deve essere associata pubblicamente alle recensioni espresse. Ogni interazione deve avvenire tramite identificatori pseudo-anonimi, eventualmente riconducibili all'utente solo in presenza di specifiche condizioni legali.
- **C2:** Tutte le comunicazioni tra client e piattaforma, comprese le operazioni di verifica d'acquisto (Proof-of-Purchase) e scrittura recensioni, devono avvenire tramite canali sicuri e cifrati (eg. TLS o crittografia applicativa end-to-end).
- **C3:** Il sistema deve ridurre al minimo i metadati esposti (eg. timestamp precisi, IP, pattern comportamentali) per evitare rischi di deanonimizzazione tramite tecniche di correlazione.
- **C4:** Il sistema deve evitare che le attività di uno stesso utente possano essere correlate tra loro nel tempo, a meno che l'utente stesso non lo autorizzi esplicitamente. Questo implica l'uso di tecniche come l'impiego di chiavi crittografiche diverse per ogni recensione. In questo modo si riduce il rischio di tracciamento comportamentale a lungo termine e si protegge ulteriormente il diritto alla riservatezza dell'utente.

### **1.4.2 Privacy**

Il sistema impiega identificatori pseudo-anonimi (DID) e credenziali verificabili (VC) per proteggere l'identità dell'utente, garantendo che ogni interazione sia verificabile, ma non tracciabile. A tal fine, vengono rispettate le seguenti proprietà:

- **Minimizzazione dei dati (Minimization):** ogni transazione include solo le informazioni strettamente necessarie (eg. hash del contenuto, identificativo del token), evitando esposizione di dati personali o metadati sensibili.

- **Verificabilità condizionata (Predicate Disclosure):** l'utente può dimostrare di aver diritto a pubblicare una recensione (eg. possesso di NFT e VC valida), senza rivelare la propria identità reale né altri attributi non necessari.
- **Non tracciabilità (Unlinkability):** recensioni diverse inviate dallo stesso utente non sono collegabili tra loro, a meno che l'utente non decida volontariamente di accumulare reputazione aggregata (via prove ZKP). In tal caso, la verifica avviene in modo crittograficamente sicuro, senza comprometterne l'anonimato.
- **Non riutilizzabilità (Non-transferability):** ogni NFT può essere impiegato una sola volta per scrivere una recensione, impedendo la duplicazione o la cessione dei diritti di pubblicazione.
- **Non falsificabilità (Unforgeability):** ogni credenziale e ogni transazione è firmata digitalmente. Nessun utente può pubblicare una recensione senza possedere sia una VC valida che un NFT non scaduto o revocato.
- **Protezione contro correlazioni (Untraceability):** il sistema impiega DID diversi e rotabili per ogni interazione e minimizza i timestamp visibili, impedendo correlazioni tra recensioni inviate dallo stesso utente.
- **Resistenza alla sorveglianza passiva (Unobservability):** anche osservando il traffico blockchain o IPFS, un attore esterno non può inferire legami tra utenti, recensioni e reputazione, grazie all'uso di hash, CID e metadati minimizzati.

Queste proprietà derivano da una progettazione orientata alla privacy e all'autonomia dell'utente, ispirata ai principi del paradigma Self-Sovereign Identity e alle tecniche di presentazione selettiva delle credenziali (*Selective Disclosure*).

### 1.4.3 Integrità

- **I1:** Ogni recensione deve essere immutabile una volta pubblicata, salvo condizioni specifiche di modifica o revoca definite a livello di smart contract.
- **I2:** Solo utenti che hanno effettivamente acquistato un prodotto devono poter lasciare una recensione; la verifica dell'acquisto deve essere crittograficamente solida e legata all'identità pseudo-anonima dell'utente.
- **I3:** Le recensioni devono essere firmate digitalmente, in modo da impedire manipolazioni o negazioni ex post (non ripudio).
- **I4:** Il sistema deve impedire che due utenti possano aggregare le proprie credenziali o attività per simulare una recensione congiunta o aumentare artificiosamente la reputazione di un contenuto.

- **I5:** Il sistema deve garantire che le recensioni valide vengano registrate secondo l'ordine temporale con cui sono state emesse, senza che attori intermedi (eg. piattaforma, validator) possano ritardare o censurare selettivamente transazioni per influenzare la visibilità o la reputazione dei contenuti.

### 1.4.4 Trasparenza

- **T1:** Gli algoritmi di ordinamento e visibilità delle recensioni devono essere pubblici e non modificabili, così da evitare favoritismi o manipolazioni da parte della piattaforma.
- **T2:** Tutte le transazioni rilevanti (recensioni, modifiche) devono essere consultabili pubblicamente sulla blockchain o in un registro parallelo verificabile, secondo il principio dell'immutabilità.
- **T3:** Le regole per la revoca o la modifica di una recensione devono essere chiare, predefinite e non arbitrate dalla piattaforma, ma applicate automaticamente dal sistema.
- **T4:** La fiducia in eventuali componenti terze (eg. piattaforma, validator) deve essere giustificata da meccanismi tecnici o economici che ne scoraggino comportamenti sleali (eg. penalità, perdita di reputazione, verifica pubblica).

### 1.4.5 Efficienza

- **E1:** Il sistema consente una rapida verifica dell'acquisto, senza richiedere operazioni pesanti lato utente o validator.
- **E2:** Il processo di pubblicazione della recensione deve essere fluido e a basso costo computazionale, compatibile con dispositivi consumer (eg. smartphone).
- **E3:** Le operazioni crittografiche coinvolte (firma, verifica, Proof-of-Purchase) devono essere ottimizzate per l'uso quotidiano e non introdurre frizioni inutili nell'esperienza utente.
- **E4:** Il sistema deve essere progettato per gestire un elevato numero di recensioni e accessi contemporanei, mantenendo prestazioni stabili anche in caso di picchi di attività.

## 1.5 Completeness

La proprietà di *completeness* descrive il comportamento del sistema quando tutti gli attori coinvolti agiscono in modo onesto, rispettando le regole previste dal protocollo. In tale scenario, non si verificano attacchi né violazioni, e il sistema è in grado di offrire tutte le funzionalità desiderate in maniera sicura, trasparente ed efficiente.

In particolare:

- un utente effettua un acquisto legittimo attraverso una piattaforma e-commerce integrata con il sistema di recensioni;

- la piattaforma verifica e registra la transazione, generando una prova di acquisto (*Proof-of-Purchase*) che attesta, in modo crittograficamente sicuro, che l'utente ha realmente usufruito del servizio;
- tramite un'identità pseudo-anonima, l'utente accede alla funzionalità di recensione e, dopo il superamento della verifica, redige un feedback firmato digitalmente, che viene inoltrato alla rete;
- i validator onesti includono la recensione nella blockchain secondo l'ordine cronologico, garantendone l'immutabilità, la tracciabilità e la pubblica consultabilità, senza censure né ritardi;
- la piattaforma mostra la recensione applicando criteri di ordinamento e visibilità pubblici, verificabili e non modificabili in modo arbitrario;
- l'utente ha facoltà, qualora previsto, di revocare o aggiornare la recensione entro i limiti e secondo le regole definite nel sistema (eg. tempo massimo, consenso firmato);
- l'utente riceve reputazione positiva se pubblica la recensione entro il tempo previsto, o negativa se non lo fa;
- il sistema assegna eventuali incentivi e reputazione sulla base del rispetto delle scadenze e della pubblicazione effettiva di recensioni;
- durante l'intero processo, il sistema protegge la privacy dell'utente, evitando l'esposizione di metadati sensibili e garantendo l'integrità e il corretto tracciamento di tutte le operazioni.

---

---

# CAPITOLO 2

---

## WORK PACKAGE 2

In questo capitolo viene proposta una soluzione progettuale che risponde al modello delineato nel *Work Package 1 (WP1)*. L'obiettivo è progettare un sistema decentralizzato per la gestione di recensioni online affidabili nel contesto dell'e-commerce, che rispetti i vincoli funzionali e di sicurezza individuati, raggiungendo un ragionevole compromesso tra efficienza, trasparenza, confidenzialità, privacy e integrità.

La soluzione proposta si basa sull'utilizzo di una blockchain *permissionless*, che garantisce l'immutabilità delle recensioni, la verificabilità pubblica delle interazioni e la resistenza a manipolazioni arbitrarie da parte di malintenzionati. Gli utenti interagiscono tramite identità pseudo-anonime e possono pubblicare recensioni solo a seguito della verifica crittografica dell'avvenuto acquisto. Tutte le interazioni significative (inserimento, modifica, revoca) sono firmate digitalmente e registrate on-chain, assicurando tracciabilità e non ripudio.

La progettazione presentata in questo WP descrive dettagliatamente il comportamento delle parti oneste coinvolte e i componenti principali del sistema, con particolare attenzione ai seguenti aspetti:

- Verifica dell'acquisto tramite meccanismi crittografici (*Proof-of-Purchase*) legati all'identità pseudo-anonima dell'utente;
- Meccanismo di scrittura e pubblicazione delle recensioni, con struttura dei dati immutabile e regole di visibilità e ordinamento trasparenti e non modificabili;
- Sistema di reputazione basato esclusivamente sul comportamento verificabile (recensione pubblicata o meno), senza giudizi soggettivi da parte di altri utenti;
- Meccanismi di incentivazione che premiano automaticamente l'utente dopo l'acquisto, subordinatamente alla pubblicazione della recensione entro un tempo massimo stabilito;
- Possibilità di revoca o modifica delle recensioni secondo regole predefinite e automatizzate tramite smart contract;




- Protezione della privacy dell'utente e prevenzione della tracciabilità incrociata, nel rispetto dello pseudo-anonimato.
- Utilizzo del protocollo IPFS (InterPlanetary File System) per la conservazione off-chain di contenuti testuali delle recensioni, metadati sensibili, e registri di revoca. Ogni contenuto archiviato è referenziato tramite il suo CID (Content Identifier), il cui hash è pubblicato on-chain.

Tutte le scelte architetturali saranno giustificate in relazione alle proprietà analizzate nel WP1 e alle minacce potenziali, per garantire la solidità e l'affidabilità del sistema.

### 2.1 Architettura generale del sistema


Il sistema proposto per la gestione decentralizzata delle recensioni nel contesto dell'e-commerce si basa su una rete blockchain *permissionless*, supportata da smart contract e da una piattaforma applicativa decentralizzata (dApp) che funge da interfaccia utente. L'architettura è progettata per garantire un comportamento verificabile, trasparente e resistente alla manipolazione da parte di attori malevoli. Ogni interazione critica viene registrata on-chain, mentre i dati sensibili sono gestiti off-chain in modo sicuro.

#### 2.1.1 Componenti principali del sistema

- **Utente pseudo-anonimo** (

---

16

- **Piattaforma e-commerce** (

### 2.1.2 Modalità di gestione della reputazione

La reputazione di ciascun utente è calcolata automaticamente sulla base del numero di NFT ricevuti e del numero di recensioni pubblicate nei tempi previsti.

- Ogni NFT rappresenta un'opportunità per pubblicare una recensione.
- Se l'utente recensisce entro 60 giorni, l'NFT è marcato come "utilizzato" e si assegna reputazione positiva.
- Se l'utente non recensisce entro il termine, l'NFT è marcato come "scaduto" e si assegna reputazione negativa.

L'utente può dimostrare, tramite prove a conoscenza zero, di aver maturato una reputazione positiva, senza rivelare identità o contenuti. Questo approccio consente di preservare la proprietà dell'anonimato e al contempo mitigare i rischi legati agli attacchi Sybil o alla creazione massiva di recensioni scollegate.

## 2.2 Funzionamento delle parti oneste

Questa sezione descrive il comportamento previsto delle componenti oneste del sistema, ovvero le entità che agiscono nel rispetto delle regole e delle proprietà progettuali identificate nel WP1. Ogni attore onesto segue un protocollo deterministico e verificabile, che garantisce il corretto funzionamento del sistema e la coerenza dei dati registrati on-chain. Le azioni compiute dai partecipanti sono regolate da procedure crittografiche sicure e dalla logica codificata nei contratti intelligenti, allo scopo di prevenire errori, frodi e ambiguità.

### 2.2.1 Gestione utente e creazione del DID

Nel sistema proposto, la gestione dell'identità dell'utente si basa sull'impiego di *Decentralized Identifiers (DID)*, ovvero identificatori univoci e pseudo-anonimi che permettono a ciascun utente di operare sulla piattaforma senza esporre la propria identità reale. I DID sono auto-generati dagli utenti in locale e sono legati a una coppia di chiavi crittografiche. Questa scelta garantisce un buon compromesso tra autenticità delle azioni e tutela della privacy.

#### Registrazione iniziale

Prima dell'avvio della procedura, l'utente si autentica tramite un sistema di identità digitale certificata, come **SPID** o **CIE**. Questa autenticazione consente all'Issuer di verificare l'identità reale dell'utente, mantenendo separati i dati personali dal successivo identificatore pseudo-anonimo (DID). La prima interazione con la piattaforma è connotata dai seguenti passaggi:

1. L'utente genera localmente una **coppia di chiavi** ( $sk_{DID}$ ,  $pk_{DID}$ ) e il corrispondente DID (eg. `did:ethr:0xABC123...`), secondo lo standard *W3C*.
2. L'utente richiede una **Verifiable Credential (VC)** da un Issuer autorizzato, che attesta la validità della sua registrazione e firma la VC.
3. L'utente memorizza la VC in un wallet compatibile (MetaMask).
4. L'utente invia una **Verifiable Presentation (VP)**, un pacchetto firmato che dimostra il possesso della VC, e una firma al contratto di registrazione.
5. Lo smart contract verifica:
  - L'autenticità della VC e la firma del *trusted Issuer*;
  - L'unicità della registrazione (la VC non è stata già usata);
  - Il legame tra DID e chiave usata per firmare la richiesta.
6. Se tutte le verifiche hanno esito positivo, il DID viene registrato come identità pseudo-anonima abilitata all'uso della piattaforma, mediante il mapping `registered[DID] = hash(VC)`, che viene aggiornato on-chain.

**Vantaggi** L'adozione dei DID offre i seguenti benefici:

- **Privacy:** l'identità reale dell'utente non viene mai esposta on-chain.
- **Sybil resistance:** l'emissione della VC è vincolata a un'autorità che rilascia un solo attestato per identità reale, difatti l'uso di SPID o CIE come prerequisito per l'emissione impedisce la creazione di identità multiple, pur mantenendo il DID completamente pseudo-anonimo on-chain.
- **Autenticazione decentralizzata:** l'utente dimostra di essere titolare del DID tramite challenge-response, firmando ogni richiesta con la propria chiave privata.
- **Interoperabilità:** i DID possono essere riutilizzati in altri contesti Web3 compatibili, migliorando l'usabilità e la coerenza dell'identità decentralizzata.

**Mitigazioni di sicurezza** Il sistema include alcune contromisure critiche:

- **Check di non riutilizzo:** ogni VC può essere usata una sola volta per registrare un DID.
- **Protezione contro identity theft:** solo il possessore della chiave privata può firmare la richiesta di accesso, riducendo il rischio di furti.
- **Registrazione dell'hash on-chain:** i contenuti delle VC, comprensivi di eventuali dati sensibili, sono conservati off-chain nel wallet dell'utente. Solo l'hash del documento viene registrato on-chain, al fine di garantire integrità e verificabilità, senza esporre il contenuto originale.

In questo modo, l'identificazione è solida e rispettosa della privacy, e il sistema impedisce la creazione di account multipli o il furto d'identità, senza la necessità di un'autorità centrale che conservi dati personali.

### 2.2.2 Venditore

Il venditore propone un prodotto ed è soggetto alla ricezione di recensioni pubbliche. Non ha la possibilità di modificare, rimuovere o influenzare l'ordine, la visibilità o il contenuto delle recensioni associate ai propri prodotti. La reputazione del venditore è determinata in modo trasparente e automatico sulla base delle recensioni effettivamente pubblicate dai clienti che hanno completato un acquisto. La piattaforma fornisce al venditore solo l'accesso alla consultazione pubblica dei dati registrati on-chain, senza controllo diretto sulle logiche di gestione.

### 2.2.3 Interazione dell'utente con il sistema tramite DID

Una volta completata la fase di registrazione e ottenuto un identificatore decentralizzato (DID), l'utente è in grado di partecipare attivamente al sistema, mantenendo un livello elevato di riservatezza. Tutte le operazioni compiute vengono autorizzate attraverso la firma digitale generata

con la propria chiave privata associata al DID, garantendo così l'autenticità delle azioni, senza mai esporre informazioni personali. Tutte le interazioni sono registrate tramite smart contract sulla blockchain.

### Pubblicazione di una recensione

Quando l'utente finalizza un acquisto sulla piattaforma e-commerce, riceve un attestato crittografico sotto forma di **NFT non trasferibile** (*soulbound token*) che ne conferma la transazione. Questo token funge da credenziale spendibile per l'invio di una recensione entro un termine massimo di 60 giorni.

Il processo si articola nei seguenti passaggi:

- L'utente redige la recensione e firma la richiesta con la chiave privata (EdDSA) associata al proprio DID;
- Invia la recensione e l'ID dell'NFT al contratto di gestione delle recensioni;
- Lo smart contract verifica che il token sia valido e appartenga all'utente, che non sia stato già utilizzato (`reviewUsed == false`) e che non sia scaduto;
- In caso di esito positivo, la recensione viene registrata on-chain e referenziata via IPFS, divenendo pubblicamente consultabile, e l'NFT viene marcato come usato. Inoltre, all'utente viene assegnata reputazione positiva e l'incentivo previsto viene immediatamente assegnato.

### NFT non utilizzati

Se l'utente non pubblica alcuna recensione entro il termine massimo di 60 giorni:

- lo smart contract marca l'NFT come "scaduto";
- viene assegnata **reputazione negativa** all'utente.

### Reputazione e prove crittografiche

Per contribuire alla reputazione senza violare l'anonimato, l'utente può generare una **Zero-Knowledge Proof** che attesti proprietà aggregate, come:

- Numero minimo di recensioni pubblicate entro il termine previsto;
- Assenza di NFT scaduti o penalità reputazionali;
- Partecipazione continuativa e coerente nel tempo.

La prova viene verificata da uno smart contract compatibile con circuiti ZK (Semaphore), senza che sia necessario rivelare esplicitamente il contenuto delle recensioni firmate né il DID dell'utente.

### Modifica e Revoca della Recensione

Ogni recensione registrata sulla blockchain può essere successivamente **modificata** o **revocata** dall'autore legittimo, rispettando condizioni di integrità e trasparenza.

**Revoca** L'utente può revocare la propria recensione soltanto se:

- è ancora il proprietario del NFT associato (verifica ownership);
- non ha già modificato o revocato in precedenza la stessa recensione;
- la richiesta è firmata digitalmente con la propria chiave privata associata al DID.

La revoca è implementata come un evento immutabile **ReviewRevoked** che mantiene traccia della decisione e il contenuto della recensione viene sostituito con un placeholder nullo. Il token di prova d'acquisto viene comunque marcato come “consumato” e non potrà essere riutilizzato, impedendo strategie abusive.

**Modifica** L'autore può modificare il contenuto testuale di una recensione, purché:

- ne detenga ancora la proprietà (verifica NFT);
- il contenuto precedente non sia stato revocato;
- venga fornita una nuova versione firmata, referenziata tramite un identificatore crittografico univoco (hash) e memorizzata off-chain in modo verificabile.

Le modifiche:

1. invalidano la versione precedente della recensione, mantenendo la tracciabilità storica;
2. vengono registrate come eventi di aggiornamento on-chain;
3. sono visibili attraverso lo storico pubblico delle versioni.

**Cooldown** Per prevenire abusi da parte di recensori che aggiornano ripetutamente il contenuto, viene introdotto un periodo di cooldown minimo di 24 ore tra due modifiche successive.

## 2.3 Utilizzo di NFT come prova di acquisto

Per garantire che solo gli utenti che abbiano effettivamente acquistato un prodotto possano pubblicarne una recensione, il sistema impiega un meccanismo di **Proof-of-Purchase** basato su *Non-Fungible Tokens (NFT)*. Ogni acquisto su una piattaforma e-commerce integrata genera un NFT. Il token rappresenta un attestato crittografico unico e verificabile, associato in modo univoco all'utente (tramite DID) e all'acquisto effettuato.

### 2.3.1 Emissione del token

Al momento della conferma di un acquisto su una piattaforma e-commerce integrata, lo smart contract corrispondente esegue il *minting* di un NFT contenente:

- `productId`: identificativo del prodotto acquistato;
- `ownerDID`: DID dell'utente;
- `purchaseDate`: data e ora dell'acquisto;
- `NFTstatus == valid`;
- `reviewStatus`: inizialmente impostato su `pending`;
- Eventuali metadati (eg. nome del prodotto, codice ordine hashato).

Il token viene assegnato all'utente ed è **non trasferibile** (soulbound), inoltre viene marcato come utilizzato una volta impiegato per pubblicare una recensione. Questo previene il riutilizzo fraudolento e la possibilità di vendere il diritto di recensire. Oltre all'NFT di acquisto, il sistema assegna automaticamente all'utente, dopo l'invio di una recensione, un **badge soulbound** (NFT non trasferibile) che attesta la partecipazione al sistema di recensioni. Questo badge può essere usato per sbloccare funzionalità o status futuri.

### 2.3.2 Verifica e pubblicazione della recensione

L'utente può utilizzare l'NFT ricevuto per pubblicare una recensione entro 60 giorni dalla data di acquisto. Per fare ciò, deve presentare:

- Il contenuto della recensione, firmato con la propria chiave privata;
- L'identificativo dell'NFT associato all'acquisto.

Lo smart contract verifica che:

- Il token sia ancora valido e appartenga all'utente;
- Che lo stato (`reviewStatus`) sia ancora `pending`;
- Che non siano trascorsi più di 60 giorni dalla data di acquisto.

Se tutti i controlli sono soddisfatti:

- La recensione viene registrata sulla blockchain (hash su IPFS);
- Il token viene marcato come usato (`NFTstatus == used`);
- viene assegnata reputazione positiva all'utente.

Se la recensione viene pubblicata entro il termine, oltre all'incremento reputazionale, può essere aggiornato anche il livello del badge NFT posseduto, per riflettere il grado di partecipazione, secondo il seguente schema:

- **Bronze Reviewer** – 1 recensione pubblicata;
- **Silver Reviewer** – almeno 10 recensioni puntuali;
- **Gold Reviewer** – almeno 50 recensioni e zero penalità.

### 2.3.3 NFT scaduti

Se l'utente non recensisce entro 60 giorni:

- lo smart contract marca l'NFT come scaduto (`NFTstatus == expired`);
- viene automaticamente assegnata una penalità reputazionale (reputazione negativa).

Questa logica incoraggia gli utenti a contribuire attivamente al sistema di recensioni e scoraggia comportamenti inattivi o opportunistici.

### 2.3.4 Vantaggi della soluzione NFT

L'adozione di NFT soulbound per rappresentare la prova d'acquisto offre numerosi vantaggi, tra cui:

- **Immutabilità e trasparenza:** la prova dell'acquisto è pubblica e verificabile da chiunque;
- **Autenticità verificabile on-chain (Non ripudio):** ogni recensione è legata in modo inequivocabile a un acquisto valido tramite un token pubblico;
- **Resistenza allo spam:** un solo acquisto equivale ad un solo diritto di recensire;
- **Resistenza alla manipolazione:** gli NFT non sono trasferibili né falsificabili;
- **Compatibilità con reputazione verificabile:** la presenza o l'uso degli NFT può contribuire a meccanismi di reputazione, validabili via ZKP;
- **Automazione reputazionale:** il sistema calcola la reputazione esclusivamente sulla base del comportamento verificabile;
- **Incentivazione tempestiva:** l'utente riceve l'incentivo subito dopo l'acquisto, ma la reputazione dipende dal rispetto del termine di pubblicazione.

L'intero processo di generazione e verifica del token avviene on-chain, per assicurare massima affidabilità, trasparenza e auditabilità. Tuttavia, i metadati estesi o sensibili (eg. descrizione completa dell'ordine) sono conservati off-chain, tramite IPFS, e referenziati via hash, per ridurre costi e tutelare la privacy.



## 2.4 Uso delle Verifiable Credentials nel sistema

Per rafforzare il controllo sull'unicità e sull'autenticità delle identità senza rinunciare al principio di pseudo-anonimato, il sistema adotta il modello delle **Verifiable Credentials (VC)**, standardizzato dal W3C. Le VC permettono di attestare in modo sicuro e crittograficamente verificabile che un determinato DID appartenga a un utente registrato, senza dover rivelare direttamente l'identità reale.

### 2.4.1 Emissione della Verifiable Credential

Durante la fase di registrazione, l'utente interagisce con un'entità di fiducia, che agisce come *Issuer*, la quale rilascia una credenziale firmata (VC) contenente informazioni minime che attestano la registrazione dell'utente.

La VC include:

- L'identificatore del soggetto (`did:ethr:0xABC123...`);
- Lo stato di utente verificato (eg. `verified-user: true`);
- L'identificativo dell'Issuer;
- La firma crittografica della VC (EdDSA);
- Data di scadenza e condizioni di revoca.

La credenziale viene conservata nel wallet dell'utente (MetaMask con plugin VC) e non è pubblicata on-chain. Solo il suo hash può essere referenziato per finalità di verifica, riducendo l'esposizione di metadati sensibili.

### 2.4.2 Presentazione e verifica

Quando un utente desidera compiere un'operazione vincolata all'identità (registrarsi, scrivere una recensione), genera una **Verifiable Presentation (VP)**.

La VP è una struttura firmata che contiene:

- La VC originale o selezionata parzialmente (selective disclosure);
- Una **challenge** generata dal verificatore, usata per impedire replay-attack;
- Una firma che prova il possesso legittimo della VC.

Lo smart contract agisce da **Verifier** e controlla:

- La validità della firma dell'Issuer (*trusted Issuer*);
- Che la VC non sia scaduta o revocata (tramite `revocation registry`);
- Che la presentazione non sia stata già riutilizzata (*non replayable*);
- Che il DID sia legittimo e coerente con la registrazione.

Solo se tutte queste condizioni risultano soddisfatte, l'azione richiesta dall'utente viene autorizzata.

### 2.4.3 Gestione della Revoca delle VC

La revoca delle Verifiable Credentials (VC) è fondamentale per garantire la validità e unicità dell'identità nel tempo, in particolare in caso di compromissione, riemissione o declassamento dell'identità stessa.

#### Modello adottato

Il sistema adotta il modello **W3C Revocation List 2020**, in cui ogni VC include un **revocationIndex** associato a una bitstring pubblica. La lista di revoca è mantenuta *off-chain* e distribuita tramite *IPFS*. Ogni versione è identificata da un **CID (Content Identifier)**, pubblicato dallo smart contract dell'Issuer. In questo modo, ogni utente o smart contract può accedere al file JSON firmato, scaricarlo da IPFS e verificare lo stato di revoca senza dipendere da API centralizzate.

#### Verifica della validità

Al momento della presentazione di una VC, lo smart contract recupera l'hash della stessa e l'indice di revoca e verifica che il bit corrispondente nella bitstring non sia impostato a 1, altrimenti nega l'autorizzazione.

In pseudo-codice:

```
if RevocationList[vcIndex] == 1 → reject
```

Il contratto accede al CID della lista e confronta l'hash SHA-256 della VC con la posizione indicata.

#### Sicurezza e Privacy

Il meccanismo garantisce:

- **Integrità:** la Revocation List è firmata e referenziata tramite hash.
- **Verificabilità pubblica:** chiunque può scaricarla via IPFS.
- **Privacy:** nessun dato personale è esposto, solo l'hash della VC è usato per la verifica.

#### Alternative future: EVOKE

In scenari ad alta scalabilità, il sistema potrà adottare accumulatori crittografici (eg. Merkle tree o accumulatori RSA) come nel modello **EVOKE**, per ridurre lo spazio di verifica locale e migliorare le performance su dispositivi IoT.

### 2.4.4 Privacy e Selective Disclosure

Il sistema adotta tecniche di *selective disclosure* per minimizzare i dati esposti durante la presentazione delle credenziali. In particolare, viene impiegata la tecnica **BBS+ Signature**, che consente di derivare sottoprove firmate a partire da una VC completa, mantenendo validità crittografica. L'utente può così presentare selettivamente solo i campi strettamente necessari, proteggendo le altre informazioni contenute nella VC originale, senza bisogno di emettere più credenziali.

Questo approccio migliora:

- La **confidenzialità**, evitando l'esposizione di dati personali on-chain;
- La **non linkabilità**, poiché transazioni distinte non sono correlabili tra loro;
- La **flessibilità**, supportando interazioni multi-identità e selettive in ambienti decentralizzati.

### 2.4.5 Compromesso tra sicurezza e costo

L'intero ciclo  $VC \rightarrow VP \rightarrow \text{verifica}$  è progettato per essere scalabile:

- Le **VC** sono emesse e conservate **off-chain** nei wallet, con hash referenziati on-chain solo quando necessario.
- Le **Revocation List** sono distribuite tramite IPFS, identificabili attraverso un *CID* firmato e referenziato negli smart contract, evitando congestione e dipendenza da API centrali.
- Le **VP** sono firmate localmente e verificate solo nei punti critici (registrazione, pubblicazione recensione), minimizzando il costo computazionale on-chain.

In questo modo, si ottiene un equilibrio ottimale tra sicurezza crittografica, protezione dell'identità e sostenibilità tecnica.

## 2.5 Gestione dei casi limite

Il sistema è progettato per mantenere coerenza, correttezza e sicurezza anche in presenza di condizioni anomale, errori operativi o utenti inattivi. Questa sezione descrive i principali scenari limite e le strategie previste per gestirli senza compromettere l'integrità del sistema reputazionale.

### 2.5.1 Utenti inattivi

Gli utenti che non interagiscono con il sistema per lunghi periodi non perdono l'accesso, ma:

- le loro **VC possono scadere** naturalmente, impedendo operazioni critiche finché non ne viene emessa una nuova;

- gli **NFT non utilizzati** per scrivere recensioni vengono marcati automaticamente come **expired** dopo 60 giorni;
- a ogni NFT scaduto corrisponde una penalità reputazionale automatica (reputazione negativa).

### 2.5.2 VC scadute o revoked

Ogni VC include un campo di scadenza e un indice nella Revocation List. L'azione viene bloccata dal contratto se si verifica almeno una delle seguenti condizioni:

- la data di validità è scaduta;
- il bit associato nella Revocation List è impostato a 1.

In entrambi i casi, l'utente dovrà ottenere una nuova VC valida per proseguire.

### 2.5.3 DID revocati o compromessi

Nel caso in cui un utente smarrisca la propria chiave privata o sospetti un furto, può:

1. generare un nuovo DID,
2. ottenere una nuova VC dal medesimo Issuer,
3. invalidare quella precedente tramite aggiornamento della Revocation List.

Il sistema consente di migrare identità senza esporre dati personali e mantiene coerenza nei circuiti ZKP tramite aggiornamento dei nullifier.

### 2.5.4 Utente Bannato

Un utente può essere bannato dal sistema in seguito alla rilevazione automatica o manuale di comportamenti fraudolenti, come la pubblicazione di recensioni false, la violazione delle policy di registrazione o l'abuso dei meccanismi reputazionali. Il DID dell'utente viene inserito in una *ban list* consultabile on-chain, impedendogli di:

- pubblicare recensioni;
- ricevere incentivi;
- modificare o revocare recensioni precedenti.

Il ban può essere revocato solo tramite una procedura di verifica off-chain condotta da un comitato di validatori.

## 2.6 Gestione del compromesso tra pseudo-anonimato e unicità dell'utente

Nel progettare un sistema decentralizzato che tuteli la riservatezza dell'utente ma impedisca abusi come la creazione di identità multiple, è necessario bilanciare due esigenze apparentemente opposte: la **non tracciabilità delle azioni** e l'**unicità verificabile dell'identità**. Infatti, l'adozione di identificatori pseudo-anonimi (DID) per separare tra loro le recensioni, se da un lato protegge la privacy dell'utente, dall'altro introduce il rischio di attacchi di tipo *Sybil*, in cui un singolo individuo agisce come molteplici entità per ottenere vantaggi indebiti (incentivi ripetuti, manipolazione reputazionale). Per affrontare questo compromesso, il sistema impiega un'architettura fondata su Verifiable Credentials e Zero-Knowledge Proof, in grado di separare logicamente identità pubbliche e azioni on-chain, mantenendo al contempo la garanzia che ogni utente rappresenti una sola entità logica.

Per mitigare questo rischio, il sistema prevede l'impiego della seguente strategia:

### 2.6.1 Verifiable Credential univoca

Durante la fase di registrazione, ogni utente deve ottenere una **Verifiable Credential** (VC) firmata da un Issuer affidabile, attestante che ha superato una procedura di identificazione controllata. Questa VC non contiene dati personali, ma è legata crittograficamente a un segreto univoco controllato dall'utente (*nullifier*). Tale VC rappresenta la “radice identitaria” che dimostra che l'utente è registrato una sola volta. A partire da essa, l'utente può generare qualsiasi numero di DID operativi senza perdere la proprietà di unicità logica.

### 2.6.2 DID multipli, ma dimostrabilmente unificati

Per preservare l'**unlinkability** tra le interazioni, l'utente può scegliere di utilizzare un DID diverso per ogni azione. Tuttavia, quando necessario (eg. per ricevere premi o partecipare a meccanismi di reputazione), l'utente è tenuto a generare una **Zero-Knowledge Proof** (ZKP) che attesti il possesso di una VC valida. Questo meccanismo consente di:

- Mantenere DID disaccoppiati nel dominio pubblico;
- Garantire che tutte le azioni provengano da un'unica identità logica (anti-Sybil);
- Prevenire il rilascio multiplo di premi;
- Proteggere la reputazione senza dover rivelare il contenuto delle recensioni.

### 2.6.3 Struttura della ZKP anti-Sybil

Il sistema utilizza un circuito ZKP predefinito (*Semaphore*) per generare una prova non rivelabile contenente:

- **hash(VC)**: identificativo crittografico della credenziale;
- **nullifier**: segreto univoco per prevenire riutilizzi;
- **MerkleRoot**: radice di un albero contenente tutti gli utenti registrati.

La verifica viene effettuata on-chain tramite un contratto compatibile con ZK-verification.

### 2.6.4 Risultato del compromesso

Questa strategia garantisce:

- **Non linkabilità**: ogni recensione o voto può essere firmato con DID diversi, senza possibilità di tracciamento esterno;
- **Unicità garantita**: ogni utente può dimostrare, quando richiesto, di essere un'entità registrata unica;
- **Resilienza agli attacchi Sybil**: nessun attore può ottenere vantaggi moltiplicando le proprie identità operative.

## 2.7 Smart Contract

Gli smart contract costituiscono il nucleo logico e immutabile del sistema. Tutte le regole di registrazione, validazione, reputazione e pubblicazione delle recensioni sono codificate in modo trasparente e deployate sulla blockchain. Nessuna entità può modificarne il comportamento dopo la pubblicazione.

### 2.7.1 Funzionalità implementate

I principali contratti intelligenti implementano le seguenti funzionalità:

- **Registrazione dell'utente**: verifica della VC e del DID tramite challenge crittografica e registrazione nel sistema.
- **Minting dell'NFT**: generazione automatica di NFT come prova d'acquisto alla conferma dell'ordine.
- **Pubblicazione della recensione**: verifica dell'NFT e registrazione della recensione firmata. Se entro 60 giorni, viene marcata come “valida” e assegna reputazione positiva.
- **Scadenza automatica NFT**: gestione periodica dei token inutilizzati entro il termine. Se scaduti, l'utente riceve reputazione negativa.
- **Revoca e modifica recensione**: permette all'utente di revocare o aggiornare il contenuto della recensione, nel rispetto dei vincoli di integrità e storico.

- **Calcolo reputazione:** calcola la reputazione di un DID come rapporto tra NFT ricevuti e recensioni pubblicate entro i termini.
- **Gestione utenti bannati:** controlla l'esistenza del DID in una ban list consultabile, impedendo azioni non autorizzate.
- **Verifica ZKP:** accetta prove a conoscenza zero per dimostrare l'identità o la partecipazione, senza rivelare dati sensibili.

Funzione	Descrizione
registerVC	Verifica autenticità della VC e registra il DID nel sistema
mintNFT	Genera NFT alla conferma di un acquisto valido
submitReview	Registra una recensione se l'NFT è valido; assegna reputazione positiva
expireNFT	Marca automaticamente come "scaduti" gli NFT oltre i 60 giorni; assegna reputazione negativa
editReview	Permette la modifica della recensione da parte dell'autore
revokeReview	Revoca la recensione pubblicata e disabilita l'NFT corrispondente
getReputationScore	Restituisce il punteggio reputazionale di un DID
isBanned	Verifica se il DID è presente nella ban list on-chain
verifyZKP	Verifica una prova a conoscenza zero presentata dall'utente

Table 2.1: Funzionalità implementate negli smart contract

## 2.8 Validator

I validator sono i nodi della rete blockchain che eseguono la validazione e l'inclusione delle transazioni nei blocchi. Operano secondo il meccanismo di consenso del network (eg. Proof-of-Stake), assicurando che tutte le azioni rilevanti vengano registrate in modo ordinato e immutabile.

### 2.8.1 Compiti principali

I validator onesti assicurano che ogni transazione conforme al protocollo venga elaborata correttamente, contribuendo alla trasparenza e all'affidabilità del sistema.

In particolare, essi si occupano di:

- **Inclusione delle recensioni:** garantiscono che ogni recensione validamente firmata e verificata venga scritta sulla blockchain secondo l'ordine temporale previsto.
- **Registrazione delle modifiche:** ogni operazione di revoca o aggiornamento viene tracciata come evento separato, e i validator assicurano che sia correttamente inclusa in un blocco.

- **Gestione delle scadenze:** le transazioni automatiche che marciano NFT come **expired** e aggiornano la reputazione devono essere processate senza ritardi o omissioni.
- **Verifica di prove ZKP:** i validator processano le transazioni contenenti Zero-Knowledge Proof, garantendo che la validazione delle identità e delle interazioni anonime avvenga correttamente.
- **Conservazione della coerenza temporale:** assicurano che tutte le interazioni (eg. pubblicazione recensione, modifica, scadenza NFT) avvengano in ordine cronologico, evitando manipolazioni strategiche.

### 2.8.2 Assunzione di sicurezza

Il modello prevede che una maggioranza onesta di validator partecipi al protocollo. Anche in presenza di minoranze malevoli, la blockchain garantisce:

- Ordine cronologico delle recensioni e delle modifiche;
- Disponibilità e persistenza dei dati pubblicati;
- Resistenza alla censura, alla sostituzione di contenuti o all'omissione di eventi validi.

## 2.9 Modulo di Audit

Il sistema prevede un modulo di audit pubblico che consente a utenti, revisori indipendenti o enti di fiducia di verificare la correttezza e la trasparenza delle operazioni registrate sulla piattaforma. Questo modulo agisce da interfaccia di consultazione per i dati on-chain e off-chain referenziati, rendendo possibile un controllo distribuito sull'intero sistema, favorendo un ecosistema verificabile e meritocratico.

### 2.9.1 Funzionalità

- **Consultazione recensioni:** ogni recensione è associata a un identificatore crittografico (hash) e a un NFT. Il modulo permette la consultazione dei contenuti, dei metadati essenziali (data di emissione, autore), dello stato (attiva, modificata, revocata) e dell'esito delle verifiche effettuate in fase di pubblicazione.
- **Tracciamento delle modifiche e revoche:** il sistema mantiene uno storico completo delle versioni di ciascuna recensione. Eventuali modifiche o revoche sono visualizzabili sotto forma di eventi pubblici registrati e l'utente può verificare la validità di ciascuna versione tramite l'hash originale.
- **Verifica degli NFT:** è possibile ispezionare lo stato di ogni NFT (usato, scaduto) e correlarlo con la pubblicazione della recensione.



- **Esposizione reputazionale:** la reputazione degli utenti è calcolata automaticamente dal sistema come rapporto tra NFT ricevuti e recensioni pubblicate entro i termini.
- **Verifica ZKP:** l'utente può dimostrare la propria reputazione in modo anonimo tramite Zero-Knowledge Proof. Il modulo consente di visualizzare la presenza e validità delle prove registrate, senza rivelare l'identità dell'utente.
- **Monitoraggio anomalie:** il sistema può evidenziare schemi sospetti, come attività automatizzate, voti incrociati o comportamenti strategici non conformi, offrendo strumenti per il monitoraggio comunitario e per il mantenimento dell'equilibrio reputazionale.
- **Accesso strutturato ai dati:** il modulo espone i dati registrati in formato machine-readable, facilitando analisi statistiche, audit esterni o esportazione verso strumenti di visualizzazione interattiva.

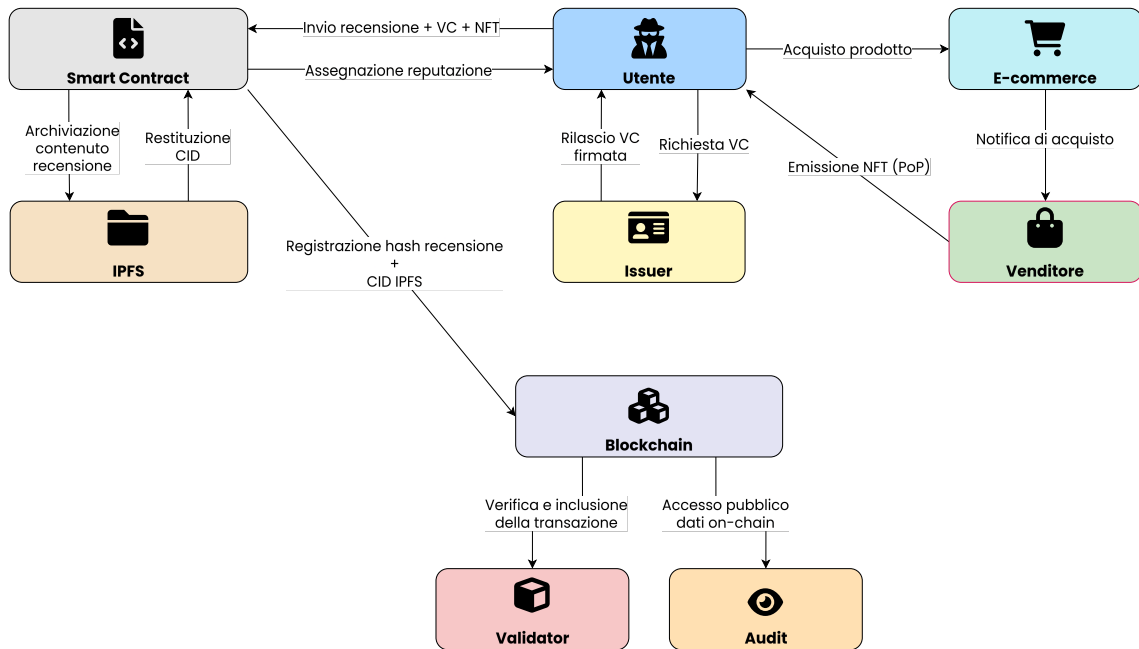


Figure 2.1: Diagramma dell'architettura del sistema

## 2.10 Scelta della blockchain permissionless

Il sistema proposto adotta una blockchain di tipo permissionless per garantire trasparenza, immutabilità e accessibilità pubblica dei dati. Questa scelta architetturale permette di garantire che nessun attore possa alterare, censurare o manipolare i dati registrati, favorendo un ecosistema aperto e meritocratico. Tuttavia, tale scelta implica anche alcune sfide in termini di costi e prestazioni. Per tale motivo, si propone un confronto sintetico tra le due principali opzioni architetturali:

Criterio	Blockchain permissionless	Blockchain permissioned
Accesso	Aperto a chiunque, nessuna autorizzazione richiesta	Limitato a nodi autorizzati
Decentralizzazione	Elevata (assenza di trust centralizzato)	Parziale, dipende da chi controlla l'accesso
Resistenza alla censura	Garantita (nessuno può impedire la scrittura di dati validi)	Debole (il consorzio può filtrare o rifiutare)
Trasparenza	Totale, i dati sono pubblici e verificabili da chiunque	Limitata, dipende dal livello di accesso concesso
Prestazioni	Più lente (consenso pubblico)	Più rapide (consenso ottimizzato)
Costi	Costi di transazione più elevati (gas)	Costi più bassi e controllabili
Governance	Distribuita e trasparente	Centralizzata o consortile

Table 2.2: Confronto tra blockchain permissionless e permissioned

Alla luce dei requisiti funzionali e di sicurezza analizzati, si ritiene che l'adozione di una rete *permissionless* sia la scelta più coerente per:

- Garantire la massima verificabilità pubblica delle recensioni;
- Evitare l'introduzione di autorità centrali di controllo;
- Supportare un sistema di reputazione distribuito e sottoponibile ad audit;
- Ridurre al minimo i presupposti di fiducia.

Eventuali esigenze prestazionali future possono essere soddisfatte tramite layer off-chain o soluzioni di scaling, mantenendo inalterata la natura pubblica e immutabile del registro principale.

### 2.10.1 Ethereum vs Hyperledger Fabric

Viene di seguito rilasciata una tabella comparativa tra Ethereum e Hyperledger Fabric, al fine di mettere in luce gli aspetti che hanno condotto alla scelta di una blockchain permissionless per questo progetto.

Parametro	Ethereum (permissionless)	Hyperledger Fabric (permissioned)
<b>Modello di trust</b>	Trustless, basato su consenso distribuito (PoS)	Basato su identità note e fidate
<b>Accessibilità</b>	Aperto a chiunque, adatto a scenari pubblici e community-driven	Limitato a entità registrate (consorzi privati)
<b>Scalabilità</b>	Scalabilità media	Alta scalabilità grazie al controllo su accessi e consenso
<b>Governance</b>	Decentralizzata, gestita dalla community e dagli stakeholder di Ethereum	Centralizzata o consorziata, con meccanismi privati di aggiornamento
<b>Censura</b>	Censura-resistente, ogni transazione è registrata pubblicamente	Possibile censura o controllo delle transazioni da parte degli admin
<b>Auditabilità</b>	Pubblica, trasparente, verificabile da chiunque	Audit interna, limitata ai membri autorizzati
<b>Costo transazioni</b>	Gas fee variabile	Costi trascurabili, ottimizzato per ambienti enterprise
<b>Adatto per</b>	Sistemi pubblici, e-commerce, reputazione e incentivazione aperta	Applicazioni interaziendali, supply chain, settori regolamentati

Table 2.3: Confronto tra Ethereum e Hyperledger Fabric

### 2.10.2 Innovazioni future

In uno sviluppo futuro, l'adozione di soluzioni Layer 2 per il rollup (come zkSync o Arbitrum) potrebbe ridurre i costi di transazione e migliorare la scalabilità, mantenendo l'integrità e la trasparenza del sistema Ethereum.

## 2.11 Utilizzo per l'utente (dApp)

Per rendere il sistema utilizzabile anche da utenti non tecnici, è prevista una piattaforma web decentralizzata (dApp) che funge da interfaccia intuitiva per tutte le operazioni previste, le

cui caratteristiche principali sono:

- **Integrazione con wallet:** supporto per MetaMask con estensioni per la gestione di VC e firme ZKP.
- **Interfacce grafiche guidate (GUI):** la dApp fornisce interazioni semplificate per:
  - generazione del DID;
  - richiesta e caricamento VC;
  - caricamento della recensione con upload automatico su IPFS;
  - gestione e consultazione della reputazione.
- **Gestione errori:** feedback chiari in caso di VC scaduta, DID non valido o NFT scaduto/usato.
- **Notifiche off-chain:** aggiornamenti via email o notifiche browser su scadenze NFT, stato della VC, ricezione incentivi.

## 2.12 Formalismi matematici crittografici

### 2.12.1 Generazione DID

Un DID assume la forma: `did:ethr:0xABC123....`

È creato localmente tramite generazione di una coppia chiave pubblica/privata  $(pk, sk)$  e registrazione su un resolver Ethereum compatibile.

### 2.12.2 Firma digitale

Ogni messaggio  $m$  è firmato localmente tramite:

$$\sigma = \text{Sign}_{sk}(m)$$

La verifica avviene tramite:

$$\text{Verify}_{pk}(m, \sigma) = \text{true}$$

### 2.12.3 Verifiable Credential (VC)

Una VC è un JSON strutturato contenente:

- `credentialSubject.id` = `did:ethr:...`
- `Issuer` = `did:ethr:0xBEEF1234...`
- `proof.signature` =  $\text{Sign}_{sk_{\text{Issuer}}}(VC)$

### 2.12.4 Verifiable Presentation (VP)

Contiene una VC più una firma:

$$\sigma_{VP} = \text{Sign}_{sk_{user}}(\text{challenge} \parallel \text{hash}(VC))$$

Verificabile via:

$$\text{Verify}_{pk_{user}}(\text{challenge} \parallel \text{hash}(VC), \sigma_{VP}) = \text{true}$$

### 2.12.5 Revoca con bitstring

Ogni VC contiene un **revocationIndex**. L'Issuer mantiene una bitstring pubblica:

$$\text{RevocationList}[\text{revocationIndex}] = 1 \Rightarrow VC \text{ revocata}$$

Verificabile pubblicamente da chiunque.

### 2.12.6 Zero-Knowledge Proof (ZKP)

L'utente può generare una prova non rivelabile di possesso di una VC valida, senza esporre il contenuto o il proprio DID. Il circuito predefinito (*Semaphore*) utilizza i seguenti elementi:

- **hash(VC)**: hash della credenziale usata;
- **nullifier**: segreto univoco per impedire il riutilizzo della prova;
- **MerkleRoot**: radice dell'albero delle identità registrate.

La prova generata è:

$$\pi = \text{ZK-Proof}(\text{hash}(VC), \text{nullifier}, \text{Merkle root})$$

Verificabile via:

$$\text{Verify}_{ZK}(\pi) = \text{true}$$

### 2.12.7 Selective Disclosure con BBS+

Per garantire la minimizzazione dei dati esposti, il sistema adotta le firme **BBS+**, che permettono all'utente di selezionare un sottoinsieme dei dati contenuti nella VC e firmarli mantenendo validità crittografica.

Data una credenziale firmata:

$$VC = \{\text{claim}_1, \text{claim}_2, \dots, \text{claim}_n\}$$

L'utente può presentare:

$$\text{Proof}_{BBS+} = \text{Sign}_{sk}(\text{subset}(VC))$$

Tale sottoprova è verificabile senza accesso all'intera VC:

$$\text{Verify}_{pk}(\text{subset}(VC), \text{Proof}_{BBS+}) = \text{true}$$


---

## **2.13 Conclusione**

Il sistema progettato nel presente WP2 rappresenta una soluzione completa e coerente rispetto al modello individuato nel WP1, capace di affrontare le criticità più rilevanti dei sistemi di recensioni centralizzati nel contesto dell'e-commerce. La progettazione tiene conto della necessità di garantire l'autenticità delle recensioni, la verifica dell'effettivo utilizzo del servizio, l'integrità dei dati, la confidenzialità degli utenti e la trasparenza delle regole applicate.

L'adozione di una blockchain permissionless consente di registrare in modo immutabile e pubblico le recensioni e tutte le operazioni critiche correlate, eliminando la necessità di una fiducia centralizzata. L'uso di NFT soulbound come prova di acquisto e Verifiable Credentials come attestati d'identità verificata garantisce la legittimità delle recensioni, prevenendo spam e falsi utenti. Inoltre, l'implementazione di tecniche a conoscenza zero (ZKP) permette di costruire una reputazione verificabile, senza compromettere l'anonimato degli utenti.

Ogni componente del sistema (smart contract, verificatori, moduli di audit) è pensato per essere trasparente, verificabile e resistente alla manipolazione. Le scelte architetturali (on-chain per le prove e i diritti, off-chain per i dati sensibili) permettono un'elevata sicurezza senza sacrificare prestazioni e scalabilità.

---

---

# CAPITOLO 3

---

## WORK PACKAGE 3

In questo capitolo viene analizzata la soluzione progettuale descritta nel WP2, verificando in che misura soddisfa le proprietà desiderate individuate nel WP1. L'analisi è articolata in sei sezioni, ognuna corrispondente a una proprietà fondamentale: confidenzialità, privacy, integrità, trasparenza, efficienza e completezza. Per ciascuna proprietà, saranno evidenziate le scelte architetturelle e le contromisure adottate, valutandone l'efficacia in riferimento al modello di minaccia definito nel WP1.

### 3.1 Confidenzialità

#### 3.1.1 C1 – Protezione dell'identità reale nelle recensioni

**Descrizione** Il sistema deve assicurare che l'identità reale dell'utente non venga mai associata, in modo diretto o indiretto, alle recensioni che pubblica. Ogni partecipazione avviene tramite un'identità digitale pseudo-anonima (DID), priva di riferimenti espliciti alla persona fisica. Solo in casi eccezionali e ben regolati, ad esempio su richiesta legale, può essere possibile risalire al soggetto che ha prodotto una recensione. Questo livello di riservatezza è fondamentale per tutelare la libertà di espressione e prevenire ritorsioni o condizionamenti esterni.

#### Minacce

- **Identity Thief:** tenta di violare wallet o sottrarre Verifiable Credentials per scoprire l'identità reale associata a un DID, oppure per agire a nome dell'utente e produrre recensioni non autorizzate.
- **Analista comportamentale (Data Correlator):** usa tecniche di fingerprinting o analisi temporale su metadati pubblici per ipotizzare correlazioni tra DID diversi e ricostruire

l'identità dell'utente.

- **Issuer compromesso:** un ente certificatore corrotto può rilasciare VC con caratteristiche non univoche e tracciabili, violando il principio di anonimato.
- **Attaccante malware o phishing:** ottiene accesso diretto al dispositivo dell'utente o alla sua chiave privata, permettendo il tracciamento completo delle attività firmate.

#### Contromisure adottate

- **DID generati localmente:** ogni utente genera autonomamente il proprio identificatore e la chiave privata associata, senza dipendere da registri nominativi.
- **Separazione tra identità reale e on-chain:** l'identità reale è verificata solo off-chain tramite SPID o CIE, ma non viene mai esposta sulla blockchain.
- **Selective Disclosure con BBS+:** l'utente presenta solo i dati minimi indispensabili, firmati crittograficamente, evitando esposizione inutile.
- **VC conservata off-chain:** la credenziale rimane nel wallet dell'utente, on-chain è registrato solo il suo hash.
- **Meccanismo di revoca:** in caso di compromissione, l'utente può revocare la VC e sostituirla con una nuova, mantenendo il controllo sul proprio profilo.
- **ZKP per la reputazione:** l'utente può dimostrare la propria identità logica o la reputazione senza rivelare alcun legame tra le recensioni pubblicate.

#### 3.1.2 C2 – Comunicazioni cifrate tra client e piattaforma

**Descrizione** Tutte le comunicazioni tra l'utente e il sistema, in particolare quelle legate alla verifica dell'acquisto, al rilascio di credenziali e alla pubblicazione di recensioni, devono avvenire su canali sicuri e cifrati. È essenziale impedire che informazioni sensibili (come token, firme digitali o contenuti delle recensioni) vengano intercettate, alterate o riutilizzate da attaccanti, anche quando viaggiano su reti non affidabili.

#### Minacce

- **Attaccante passivo di rete:** intercetta pacchetti in transito su canali insicuri (eg. HTTP), nel tentativo di ottenere credenziali, firme o altri dati sensibili. Può riutilizzare una Verifiable Presentation per attuare attacchi di replay.
- **Attaccante phishing/malware:** può intercettare dati tramite malware installato localmente (eg. keylogger, clipboard hijack), ottenendo accesso a contenuti prima che vengano cifrati.



- **Issuer compromesso:** un issuer malevolo potrebbe indurre l'utente a comunicare in chiaro o utilizzare protocolli non sicuri per la trasmissione della VC.

#### Contromisure adottate

- **Crittografia TLS:** ogni interazione client-server avviene tramite HTTPS con certificati validi, impedendo l'intercettazione dei dati in transito.
- **Crittografia end-to-end:** in scenari critici (eg. invio della VP), le informazioni vengono firmate e cifrate anche a livello applicativo, proteggendo il contenuto a prescindere dal canale usato.
- **Uso di challenge e nonce:** per prevenire replay attack, ogni Verifiable Presentation include una challenge univoca firmata, che rende inutilizzabile ogni replay.
- **Hash e CID invece di contenuto diretto:** il contenuto delle recensioni e delle credenziali è archiviato off-chain e identificato solo tramite hash, limitando l'impatto di eventuali intercettazioni.
- **Verifica VC lato smart contract:** ogni credenziale presentata viene verificata sulla chain solo se firmata da issuer affidabili, minimizzando il rischio derivante da issuer corrotti.

#### 3.1.3 C3 – Minimizzazione dei metadati esposti

**Descrizione** Il sistema deve ridurre al minimo la quantità di metadati visibili pubblicamente (eg. timestamp precisi, indirizzi IP, pattern di comportamento), per limitare la possibilità che un osservatore esterno possa trarre conclusioni sull'identità o sulle abitudini di un utente. La visibilità di tali dati rappresenta una potenziale minaccia alla privacy, anche in assenza di dati esplicitamente personali.

#### Minacce

- **Analista comportamentale (Data Correlator):** osserva l'attività pubblica sulla blockchain e su IPFS per identificare schemi temporali, occorrenze o fingerprint comportamentali, con l'obiettivo di associare azioni a identità.
- **Attaccante passivo di rete:** può monitorare i pacchetti trasmessi e raccogliere metadati (frequenza, orari, contenuti ricorrenti) da cui derivare profili utente.
- **Attaccante malware:** se installato localmente, può osservare i pattern di utilizzo (eg. orari di firma, frequenza delle transazioni) anche senza accedere direttamente alle chiavi.

#### Contromisure adottate

- **Uso di DID rotabili:** ogni interazione può essere firmata da un DID diverso, rendendo difficile correlare più azioni a uno stesso utente.

- **Timestamp offuscati:** i timestamp visibili sulla blockchain sono limitati alla granularità necessaria (eg. giorno, non ora), riducendo il valore delle analisi temporali.
- **Contenuti su IPFS con hash:** le recensioni e le VC sono archiviate off-chain e referenziate tramite hash, rendendo impossibile l'ispezione dei dati senza accesso diretto.
- **Selective Disclosure:** solo i dati strettamente necessari vengono presentati nei messaggi firmati, evitando l'inclusione di informazioni aggiuntive non richieste.
- **No log su piattaforma:** la piattaforma e-commerce non registra né archivia metadati sensibili localmente, tutte le transazioni sensibili avvengono lato blockchain o wallet.

#### 3.1.4 C4 – Isolamento tra attività dell'utente

**Descrizione** Le attività svolte dallo stesso utente nel tempo (eg. pubblicazione di più recensioni) non devono poter essere facilmente correlate tra loro, a meno che l'utente non scelga esplicitamente di farlo. Questa proprietà protegge l'utente da ricostruzioni cronologiche o aggregazioni di attività che potrebbero compromettere la sua privacy.

##### Minacce

- **Analista comportamentale (Data Correlator):** tenta di associare più recensioni tra loro in base a pattern temporali o contenutistici, per dedurre identità univoche.
- **Attaccante passivo di rete:** osserva la sequenza e la periodicità delle transazioni per ricostruire l'identità comportamentale di un utente.
- **Identity Thief o Issuer malevolo:** potrebbe creare VC con tratti identificativi costanti, facilitando l'identificazione delle attività connesse.

##### Contromisure adottate

- **Impiego di DID disaccoppiati:** l'utente può firmare ogni recensione con un DID diverso, non riconducibile agli altri, riducendo la linkabilità.
- **Zero-Knowledge Proof:** l'utente può dimostrare di essere l'autore di più recensioni (eg. per ottenere reputazione aggregata) senza svelare i singoli legami.
- **Revisione crittografica delle VC:** le VC vengono firmate senza includere elementi fissi tracciabili, e il loro contenuto è referenziato via hash.
- **Cooldown e anti-spam:** l'intervallo minimo tra due recensioni successive (24h) riduce la possibilità che un pattern temporale diventi un identificatore implicito.
- **Nessun tracciamento server-side:** la piattaforma non registra sessioni utente o pattern di navigazione, impedendo correlazioni fuori dalla blockchain.

## 3.2 Privacy

### 3.2.1 P1 – Minimizzazione dei dati

**Descrizione** Ogni transazione deve includere solo le informazioni strettamente necessarie all'azione richiesta. Il sistema deve evitare la trasmissione o pubblicazione di dati personali, dettagli d'acquisto non essenziali o altri elementi che potrebbero compromettere l'anonimato o facilitare l'identificazione indiretta dell'utente.

#### Minacce

- **Issuer compromesso:** può emettere Verifiable Credentials contenenti attributi inutili o eccessivi, che facilitano il tracciamento o l'identificazione.
- **Analista comportamentale (Data Correlator):** tenta di sfruttare qualsiasi informazione extra nei dati on-chain o nei metadati IPFS per identificare pattern o correlare identità.
- **Attaccante passivo di rete:** può osservare il contenuto delle transazioni e raccogliere dettagli ridondanti per costruire profili, anche se non direttamente identificativi.

#### Contromisure adottate

- **Uso di hash e CID:** le recensioni e altri contenuti sono conservati off-chain su IPFS, mentre on-chain viene registrato solo il loro identificatore (hash), evitando esposizione di testo o dati inutili.
- **Selective Disclosure:** grazie alle firme BBS+, ogni utente può presentare solo i campi rilevanti della propria VC, omettendo qualsiasi informazione non necessaria per l'azione corrente.
- **Design minimale dei contratti:** gli smart contract non richiedono dati personali né dettagli d'ordine, operano solo su hash, token ID e firme digitali.
- **Conservazione locale della VC:** la credenziale è mantenuta nel wallet dell'utente, non viene mai trasmessa interamente sulla rete o pubblicata in forma leggibile.

### 3.2.2 P2 – Verificabilità condizionata (Predicate Disclosure)

**Descrizione** Il sistema deve permettere all'utente di dimostrare il possesso dei requisiti necessari per un'azione (eg. diritto a recensire), senza dover rivelare l'intera identità o altri attributi personali non richiesti. Questo approccio preserva la privacy e limita l'esposizione informativa.

#### Minacce

- **Issuer compromesso:** potrebbe forzare l'utente a presentare VC complete, imponendo la condivisione di attributi non necessari.

- **Attaccante passivo di rete:** osserva le credenziali presentate e tenta di ricavare più informazioni di quelle strettamente necessarie.

#### Contromisure adottate

- **Supporto nativo a Predicate Disclosure:** l'utente può dimostrare di avere una VC valida o un diritto specifico senza rivelare l'intero contenuto della credenziale.
- **Selective disclosure tramite BBS+:** le firme permettono di derivare sottoprove contenenti solo le informazioni minime, senza invalidare la validità crittografica.
- **Verifica lato smart contract:** il contratto verifica solo la validità della prova parziale, senza accesso ad altri attributi.
- **Challenge firmata:** ogni presentazione avviene in risposta a una challenge univoca, evitando il riutilizzo della prova o l'accesso non autorizzato ad altri attributi.

#### 3.2.3 P3 – Non tracciabilità (Unlinkability)

**Descrizione** Le recensioni pubblicate da uno stesso utente non devono poter essere collegate tra loro, a meno che egli stesso non decida di renderle aggregabili. Questo impedisce a osservatori esterni di dedurre relazioni o ricostruire una storia delle azioni.

#### Minacce

- **Analista comportamentale (Data Correlator):** esamina il contenuto delle recensioni, orari o CID per identificare correlazioni che suggeriscano un'origine comune.
- **Issuer compromesso:** rilascia VC con elementi statici ricorrenti che permettono di legare presentazioni diverse tra loro.

#### Contromisure adottate

- **DID diversi per ogni recensione:** ogni pubblicazione avviene con un'identità crittografica distinta, priva di legami visibili.
- **ZKP per reputazione aggregata:** l'utente può dimostrare di avere pubblicato un certo numero di recensioni senza rivelare quali o quando.
- **Contenuti su IPFS:** le recensioni sono visibili ma non contengono metadati identificativi o riutilizzabili.
- **VC senza elementi ricorrenti:** le credenziali usano chiavi dinamiche e non includono identificatori persistenti.

#### 3.2.4 P4 – Non riutilizzabilità (Non-transferability)

**Descrizione** Ogni NFT che consente la pubblicazione di una recensione deve essere utilizzabile una sola volta ed esclusivamente dal suo legittimo proprietario. Questo impedisce che il diritto a recensire venga venduto o ceduto a terzi.

##### Minacce

- **Reviewer a pagamento:** acquista o riceve NFT da altri utenti per scrivere recensioni false su prodotti non acquistati.
- **Attaccante Sybil:** tenta di usare più identità per riutilizzare NFT apparentemente validi ma già spesi.
- **Utente fraudolento:** prova a riutilizzare lo stesso NFT su più DID per aggirare i controlli.

##### Contromisure adottate

- **NFT soulbound:** ogni token è non trasferibile e legato al DID dell'utente che ha effettuato l'acquisto.
- **Verifica ownership lato smart contract:** al momento della recensione, il contratto controlla che il token appartenga al mittente.
- **Marcatore reviewUsed:** impedisce che un NFT venga usato più di una volta per pubblicare o modificare una recensione.
- **Tracciamento stato NFT:** ogni stato (pending, used, expired) è aggiornato on-chain in modo immutabile.

#### 3.2.5 P5 – Non falsificabilità (Unforgeability)

**Descrizione** Ogni azione nel sistema, dalla registrazione dell'utente alla pubblicazione di una recensione, deve essere firmata digitalmente e verificabile crittograficamente. Nessuno può presentare una recensione o una credenziale falsa senza possedere effettivamente i diritti.

##### Minacce

- **Utente fraudolento:** tenta di scrivere una recensione senza aver effettuato l'acquisto o falsificando una VC.
- **Issuer compromesso:** emette VC non legittime che sembrano formalmente valide.
- **Validator corrotto:** accetta transazioni non firmate correttamente o contenenti credenziali false.

##### Contromisure adottate

- **Firma digitale obbligatoria:** ogni messaggio (VC, VP, recensione) deve essere firmato con la chiave privata.
- **Verifica crittografica on-chain:** gli smart contract controllano la validità di ogni firma e l'autenticità dell'emittente (Issuer affidabile).
- **Registro di revoca:** se un Issuer viene compromesso, tutte le VC rilasciate possono essere invalidate tramite bitstring pubblica.
- **Verifiche multiple incrociate:** l'azione è autorizzata solo se anche l'NFT e la VC corrispondono logicamente.

#### 3.2.6 P6 – Protezione contro correlazioni (Untraceability)

**Descrizione** Il sistema deve impedire che l'osservazione incrociata di più transazioni (eg. registrazione, invio recensione, verifica VC) possa rivelare legami tra identità o azioni dell'utente.

##### Minacce

- **Attaccante passivo di rete:** monitora le interazioni tra wallet e smart contract per correlare recensioni a una stessa identità.
- **Analista comportamentale:** osserva il traffico blockchain e tenta di legare DID a VC sulla base di pattern o metadati comuni.
- **Issuer malevolo:** inserisce riferimenti statici nelle VC, rendendo possibili correlazioni.

##### Contromisure adottate

- **DID rotabili e disaccoppiati:** ogni interazione utilizza una nuova coppia chiave/DID, priva di riferimenti al precedente.
- **Use of ZKP:** l'utente può dimostrare la validità delle sue azioni in modo anonimo, senza legarle tra loro.
- **VC con selective disclosure:** i dati non necessari non vengono mai rivelati, impedendo confronti incrociati.
- **Hash non riutilizzabili:** ogni interazione on-chain usa identificatori univoci e non riciclabili.

#### 3.2.7 P7 – Resistenza alla sorveglianza passiva (Unobservability)

**Descrizione** Anche osservando costantemente il traffico sulla rete blockchain o IPFS, un attore esterno non deve poter inferire legami tra identità, azioni, contenuti e reputazione dell'utente.

##### Minacce

- **Attaccante passivo di rete:** monitora il traffico e cerca di inferire relazioni tra DID, timestamp, CID o contenuti hashati.
- **Data Correlator:** raccoglie tutti i dati pubblici per creare mappe di reputazione o identità.

#### Contromisure adottate

- **Crittografia end-to-end:** ogni comunicazione tra wallet, smart contract e IPFS è cifrata o firmata.
- **Uso di CID non correlabili:** ogni contenuto pubblicato su IPFS ha un hash unico, senza informazioni leggibili.
- **Metadati minimizzati:** le transazioni e i documenti contengono solo l'indispensabile, evitando esposti inutili.
- **Assenza di logging sulla dApp:** la piattaforma non conserva cronologie utente, IP o fingerprint locali.

## 3.3 Integrità

### 3.3.1 I1 – Immutabilità delle recensioni

**Descrizione** Una recensione, una volta pubblicata, non deve poter essere modificata o cancellata arbitrariamente. Eventuali aggiornamenti o revoche devono seguire regole prestabilite, e l'intera cronologia deve restare consultabile.

#### Minacce

- **Utente fraudolento:** pubblica una recensione positiva, riceve un incentivo, poi la modifica in modo scorretto o rimuove contenuti.
- **Validator corrotto:** tenta di alterare lo storico delle modifiche o impedire la tracciabilità delle versioni.

#### Contromisure adottate

- **Hash IPFS immutabile:** ogni recensione è referenziata tramite hash univoco, rendendo ogni versione verificabile.
- **Registro eventi on-chain:** ogni modifica o revoca genera un nuovo evento tracciabile e immutabile sulla blockchain.
- **Regole codificate negli smart contract:** solo l'autore può modificare/revocare entro limiti specifici (eg. cooldown, una sola modifica).
- **Audit pubblico:** il sistema mantiene accesso allo storico delle versioni e degli hash precedenti.

#### 3.3.2 I2 – Verifica dell'acquisto

**Descrizione** Solo utenti che hanno realmente acquistato un prodotto possono lasciare una recensione. Tale condizione deve essere verificata crittograficamente, legando la possibilità di recensire a una prova non falsificabile.

##### Minacce

- **Reviewer a pagamento:** tenta di recensire prodotti mai acquistati sfruttando NFT rubati o prestati.
- **Utente fraudolento:** forza il sistema per ottenere NFT senza aver eseguito un acquisto valido.
- **Issuer malevolo:** rilascia VC o prove di acquisto fasulle.

##### Contromisure adottate

- **NFT non trasferibili:** emessi solo a seguito di un acquisto e associati univocamente al DID dell'utente.
- **Verifica del token:** gli smart contract controllano che l'NFT sia valido, non scaduto e non usato.
- **Prove crittografiche dell'acquisto:** ogni NFT contiene hash del prodotto, timestamp e identificativo on-chain.
- **VC da Issuer affidabile:** l'identità dell'utente è attestata da un'entità verificata che rilascia una sola credenziale per persona.

#### 3.3.3 I3 – Firma e non ripudio

**Descrizione** Ogni recensione deve essere firmata digitalmente, in modo che l'autore non possa negare in seguito di averla pubblicata. Le firme devono garantire autenticità e tracciabilità dell'origine del contenuto.

##### Minacce

- **Utente fraudolento:** tenta di negare di aver scritto una recensione negativa o sospetta.
- **Attaccante malware:** agisce per conto dell'utente cercando di non lasciare tracce verificabili.
- **Validator corrotto:** pubblica recensioni non firmate, falsificando l'autenticità.

##### Contromisure adottate

- **Firma digitale obbligatoria:** ogni recensione è firmata con la chiave privata del DID che possiede l'NFT.



- **Verifica integrata nello smart contract:** nessuna transazione è accettata senza firma valida e legata al token corretto.
- **Accesso autenticato:** ogni operazione richiede challenge firmata, prevenendo spoofing.
- **Tracciabilità completa degli eventi:** le firme e i riferimenti IPFS permettono verifica e non ripudio a posteriori.

#### 3.3.4 I4 – Prevenzione delle aggregazioni fraudolente

**Descrizione** Il sistema deve impedire che più utenti (o identità fasulle) si coordinino per manipolare artificialmente la reputazione di un contenuto. Nessuno deve poter unire più credenziali o NFT per pubblicare una recensione amplificata.

##### Minacce

- **Attaccante Sybil:** crea molte identità per generare recensioni duplicate o moltiplicare l'impatto reputazionale.
- **Utenti collusi:** tentano di aggregare NFT e VC per far risultare una recensione multipla o sovradimensionata.
- **Issuer corrotto:** distribuisce credenziali multiple alla stessa persona.

##### Contromisure adottate

- **VC univoca per identità reale:** ogni utente riceve una sola credenziale da un Issuer fidato.
- **Verifica della singola ownership NFT:** ogni recensione è associata a un solo NFT legato a un solo DID.
- **ZKP anti-Sybil:** l'utente può dimostrare in forma anonima di essere unico, senza rivelare l'identità.
- **Mappatura VC  $\rightarrow$  DID one-time:** impedisce di usare la stessa credenziale con più identità operative.

#### 3.3.5 I5 – Ordine cronologico garantito

**Descrizione** Le recensioni devono essere registrate seguendo l'ordine reale di pubblicazione. Nessun attore deve poter alterare, ritardare o censurare le transazioni per manipolare visibilità o reputazione.

##### Minacce

- **Validator corrotto:** ritarda o censura recensioni sgradite, inserendole in blocchi successivi o evitando la registrazione.

- **Utente malevolo:** tenta di forzare l'ordine con timestamp falsi o transazioni simultanee.

#### Contromisure adottate

- **Blockchain permissionless:** ogni transazione è registrata secondo l'ordine di propagazione e consenso pubblico.
- **Inclusione automatica:** i validator onesti inseriscono le recensioni senza preferenze, seguendo logiche FIFO.
- **Timestamp derivati dal blocco:** la data di pubblicazione è quella del mining, non del client.
- **Audit pubblico:** il modulo di verifica consente il controllo della sequenza delle recensioni in modo trasparente.

## 3.4 Trasparenza

### 3.4.1 T1 – Algoritmi di visibilità pubblici e immutabili

**Descrizione** Gli algoritmi che regolano l'ordinamento e la visibilità delle recensioni devono essere trasparenti, accessibili a tutti e non modificabili in modo arbitrario. Questo garantisce l'equità del sistema e impedisce favoritismi da parte della piattaforma.

#### Minacce

- **Piattaforma centralizzata:** può manipolare l'ordine di visualizzazione delle recensioni, oscurando quelle negative o sponsorizzando contenuti specifici.
- **Validator corrotto:** collabora con la piattaforma per includere selettivamente recensioni favorevoli nei blocchi prioritari.
- **Attore esterno collusivo:** tenta di influenzare il sistema off-chain per alterare la visibilità apparente.

#### Contromisure adottate

- **Logica di ordinamento on-chain:** gli smart contract gestiscono direttamente la visibilità in base a regole codificate e pubbliche.
- **Algoritmi non modificabili:** il codice di ordinamento è deployato su blockchain, quindi immutabile dopo la pubblicazione.
- **Accesso pubblico alla logica:** ogni utente può consultare il funzionamento e verificare che l'ordine non sia arbitrario.
- **Audit della DApp:** l'interfaccia utente applica i criteri di ordinamento definiti on-chain senza alterazioni locali.

#### 3.4.2 T2 – Accesso pubblico alle transazioni

**Descrizione** Tutte le transazioni significative (pubblicazione, modifica, revoca di recensioni) devono essere pubblicamente consultabili. Questo garantisce tracciabilità, verifica diffusa e previene manipolazioni non autorizzate.

##### Minacce

- **Validator corrotto:** esclude eventi di modifica o revoca, creando una visione distorta della cronologia.
- **Attaccante con accesso privilegiato:** nasconde o filtra recensioni già pubblicate tramite modifiche UI.

##### Contromisure adottate

- **Storage on-chain degli eventi:** ogni azione è registrata pubblicamente come evento immutabile sulla blockchain.
- **Link IPFS trasparente:** le recensioni e le loro versioni sono sempre accessibili tramite CID verificabili.
- **Modulo di audit pubblico:** consente a chiunque di consultare, verificare e tracciare ogni recensione e stato.
- **Smart contract open-source:** il codice che regola la gestione delle recensioni è ispezionabile da tutti.

#### 3.4.3 T3 – Regole di modifica e revoca predefinite

**Descrizione** Le condizioni per modificare o revocare una recensione devono essere stabilite in anticipo e applicate in modo automatico e imparziale. Nessuna entità deve poter decidere arbitrariamente quali recensioni rimuovere o alterare.

##### Minacce

- **Utente fraudolento:** modifica o revoca la recensione più volte per trarne vantaggio.
- **Validator malevolo:** ignora revoche lecite o permette modifiche non autorizzate.

##### Contromisure adottate

- **Regole codificate nello smart contract:** l'utente può modificare o revocare solo entro i limiti stabiliti.
- **Verifica della proprietà del token:** ogni operazione richiede che l'utente sia ancora in possesso del relativo NFT.

- **Eventi on-chain tracciabili:** ogni modifica o revoca è registrata come evento pubblico, consultabile da tutti.
- **Cooldown tra modifiche:** impedisce aggiornamenti ripetuti e manipolazioni strategiche.

#### 3.4.4 T4 – Fiducia tecnica nelle componenti terze

**Descrizione** Se il sistema si affida a componenti esterne (eg. piattaforma o validator), la fiducia in esse deve essere motivata da meccanismi tecnici o economici che ne scoraggino comportamenti scorretti, senza richiedere un'autorità centralizzata.

##### Minacce

- **Validator corrotto:** ignora transazioni valide o censura contenuti per interesse personale o pressione esterna.
- **Piattaforma non trasparente:** modifica localmente i dati, filtra recensioni o altera la reputazione visibile.
- **Entità collusa:** più attori si accordano per manipolare selettivamente il sistema senza essere rilevati.

##### Contromisure adottate

- **Rete permissionless:** nessun attore ha autorità esclusiva e ogni nodo può partecipare alla validazione.
- **Meccanismi di incentivo/penalità:** comportamenti sleali portano a perdita di reputazione o ban.
- **Audit decentralizzato:** ogni azione è verificabile pubblicamente tramite smart contract e moduli di audit.
- **Logica client stateless:** la piattaforma funge solo da interfaccia e non ha controllo diretto sulla visibilità o sulle regole applicate.

## 3.5 Efficienza

### 3.5.1 E1 – Verifica dell'acquisto rapida

**Descrizione** Il sistema deve permettere la verifica dell'effettivo acquisto in modo rapido e fluido, senza introdurre complessità inutili per l'utente o rallentamenti lato validator.

##### Rischi prestazionali

- **Utente disincentivato:** se il processo è lento o complesso, l'utente può rinunciare a recensire.

- **Validator sovraccarico:** se le operazioni sono computazionalmente pesanti, possono accumularsi ritardi nelle validazioni.
- **Piattaforma inefficiente:** genera NFT o gestisce transazioni con lentezza, rallentando il flusso d'uso.

#### Contromisure adottate

- **NFT generato automaticamente:** viene creato al momento dell'acquisto tramite minting on-chain immediato.
- **Verifica NFT minimale:** il contratto controlla solo che il token sia valido, non scaduto e non usato.
- **Operazioni asincrone:** la generazione e la validazione possono avvenire in passaggi separati, minimizzando la latenza percepita.
- **Requisiti minimi per review:** l'utente deve fornire solo firma, token ID e testo, rendendo il processo fluido.

#### 3.5.2 E2 – Pubblicazione fluida e a basso costo

**Descrizione** Il processo di pubblicazione di una recensione deve essere semplice per l'utente e poco oneroso in termini di risorse computazionali e costi (gas).

#### Rischi prestazionali

- **Utente disincentivato:** non pubblica la recensione per via dei costi di gas o della complessità.
- **Overhead computazionale:** genera congestione nella rete se le operazioni richiedono troppe risorse.
- **Validator inefficiente:** impiega troppo tempo a processare la transazione, causando delay nella pubblicazione.

#### Contromisure adottate

- **Dati sensibili off-chain:** la recensione è caricata su IPFS, mentre on-chain si registra solo l'hash.
- **Transazioni ottimizzate:** ogni azione è ridotta a operazioni basilari su token ID, hash e firma.
- **Interfaccia guidata (dApp):** semplifica l'interazione e minimizza gli errori utente.

#### 3.5.3 E3 – Ottimizzazione delle operazioni crittografiche

**Descrizione** Le firme, verifiche e presentazioni crittografiche devono essere efficienti, supportabili anche su dispositivi consumer come smartphone, senza rallentare l'esperienza utente.

##### Rischi prestazionali

- **Utente disincentivato:** non completa la procedura se la verifica crittografica è lenta o complessa.
- **DApp non ottimizzata:** rende l'esperienza frustrante su dispositivi mobili.
- **Malware:** approfitta di latenze per interferire con la firma o con i dati in memoria.

##### Contromisure adottate

- **Uso di EdDSA e BBS+:** algoritmi di firma efficienti, compatibili con dispositivi leggeri.
- **ZKP pre-computati localmente:** l'utente genera la prova a conoscenza zero prima dell'invio, evitando costi computazionali on-chain.
- **VP con selective disclosure:** l'utente presenta solo ciò che serve, riducendo il carico crittografico.
- **Plugin wallet:** compatibilità con MetaMask e plugin VC/ZKP.

#### 3.5.4 E4 – Scalabilità e prestazioni stabili

**Descrizione** Il sistema deve mantenere prestazioni adeguate anche in presenza di picchi di traffico o un numero elevato di utenti attivi, evitando colli di bottiglia o comportamenti degradanti.

##### Rischi prestazionali

- **Validator sovraccarico:** rallenta il processo di validazione e pubblicazione, penalizzando l'esperienza.
- **Rete congestionata:** le transazioni diventano troppo costose o lente in caso di alto utilizzo.
- **DoS reputazionale:** un attore malizioso genera troppe interazioni per saturare il sistema.

##### Contromisure adottate

- **Architettura distribuita:** sfrutta la natura permissionless per bilanciare il carico tra validator.
- **Soluzioni di scaling compatibili:** il sistema può essere migrato o supportato da Layer 2 e sistemi di rollup.
- **Dati off-chain su IPFS:** riduce la pressione sulla blockchain principale spostando contenuti esterni.

- **Cooldown e rate-limiting:** limita la frequenza di invio recensioni per utente, prevenendo abusi.

## 3.6 Completezza

### 3.6.1 Comportamento corretto in assenza di attacchi

**Descrizione** La proprietà di completezza garantisce che, in uno scenario privo di attacchi e con attori onesti, il sistema operi come previsto. Tutte le funzionalità devono essere disponibili, verificabili e accessibili, e le regole applicate devono portare a risultati coerenti con gli obiettivi del progetto.

#### Scenario

- **Utente onesto:** effettua un acquisto legittimo tramite la piattaforma integrata e riceve un NFT soulbound che rappresenta la prova d'acquisto.
- **Pubblicazione corretta:** l'utente, tramite DID pseudo-anonimo, accede alla funzione di recensione, redige un feedback firmato e lo invia tramite la dApp.
- **Validazione:** lo smart contract verifica l'NFT, la firma digitale e il possesso di una VC valida. Se tutto è corretto, la recensione è pubblicata su IPFS e referenziata on-chain.
- **Visibilità e tracciabilità:** la recensione è consultabile pubblicamente, ordinata secondo le regole codificate e può essere modificata o revocata solo secondo le condizioni predefinite.
- **Reputazione:** all'utente viene assegnata reputazione positiva per la pubblicazione tempestiva della recensione. In caso contrario, l'NFT scade e viene applicata una penalità.
- **Privacy:** in tutto il processo, l'identità dell'utente non viene mai esposta. Le transazioni sono firmate con DID e la VC non è pubblicata, ma solo verificata.

#### Garanzie

- Ogni funzionalità rispetta le proprietà di sicurezza definite in fase di progettazione (C, P, I, T, E).
- Le componenti on-chain e off-chain cooperano in modo verificabile, senza necessità di fiducia in entità centrali.
- Il comportamento degli utenti onesti viene premiato, mentre quello inadempiente comporta una penalizzazione automatica.
- La trasparenza e la tracciabilità sono garantite, ma la privacy dell'utente è preservata tramite ZKP e selective disclosure.

## 3.7 Limiti strutturali del sistema

### Descrizione

Nonostante la progettazione abbia previsto meccanismi crittografici, reputazionali e procedurali per affrontare una vasta gamma di minacce, alcune vulnerabilità strutturali restano parzialmente o completamente non risolvibili all'interno del modello decentralizzato adottato. In questa sezione si analizzano i principali limiti riconosciuti del sistema.

### Reviewer retribuiti

Il sistema non può impedire che un utente reale (in possesso di NFT e VC legittimi) sia incentivato economicamente da un venditore esterno a lasciare una recensione positiva. Poiché l'utente ha effettivamente effettuato l'acquisto, e la recensione rispetta tutte le condizioni tecniche, non esistono indicatori crittografici per distinguere un comportamento genuino da uno corrotto.

*Mitigazione parziale:* il sistema limita la visibilità di pattern sospetti attraverso il modulo di audit, ma non può rilevare accordi economici off-chain.

### Venditore collusivo fuori sistema

Analogamente, un venditore può contattare un acquirente tramite canali esterni (eg. email, social) e offrirgli un rimborso in cambio di una recensione positiva. Poiché la transazione è reale e la recensione formalmente valida, non è possibile intervenire on-chain.

*Mitigazione parziale:* possibili indizi possono emergere da analisi esterne, ma il sistema non ha accesso ai canali di comunicazione off-chain.

### Malware sul dispositivo

Il sistema non può proteggere l'utente nel caso in cui il dispositivo usato sia compromesso (keylogger, clipboard hijack, browser modificati). In tal caso, la chiave privata può essere esfiltrata e usata per firmare azioni indesiderate.

*Mitigazione parziale:* l'uso di wallet hardware, autenticazioni multi-livello e notifiche off-chain può ridurre l'impatto, ma non risolve il problema alla radice.

### Attacchi comportamentali basati su analisi AI

Anche in presenza di DID rotabili, CID anonimi e metadati minimizzati, un attaccante dotato di capacità avanzate (eg. AI per l'analisi linguistica o temporale) può cercare correlazioni stilistiche tra recensioni.

*Mitigazione minima:* l'anonimato linguistico è fuori dallo scope della progettazione tecnica.

### Compromissione degli Issuer

Sebbene il sistema preveda il controllo dell'identità tramite Issuer fidati, la fiducia in essi è ancora un punto centrale. Un Issuer compromesso può rilasciare VC a identità fittizie.



*Mitigazione tecnica:* l'uso di revocation list, audit e circuiti ZKP limita l'impatto, ma non può eliminarlo del tutto, in quanto la fiducia nell'Issuer è un requisito inevitabile.

#### **Conclusione**

I limiti sopra esposti non derivano da errori progettuali, ma da vincoli intrinseci al paradigma decentralizzato. Il sistema riesce a coprire in modo robusto tutte le minacce identificabili sul piano tecnico-crypto-comportamentale, ma resta aperto a dinamiche esterne non tracciabili on-chain, la cui rilevazione richiede strumenti supplementari (eg. governance, analisi esterne, auditing umano).

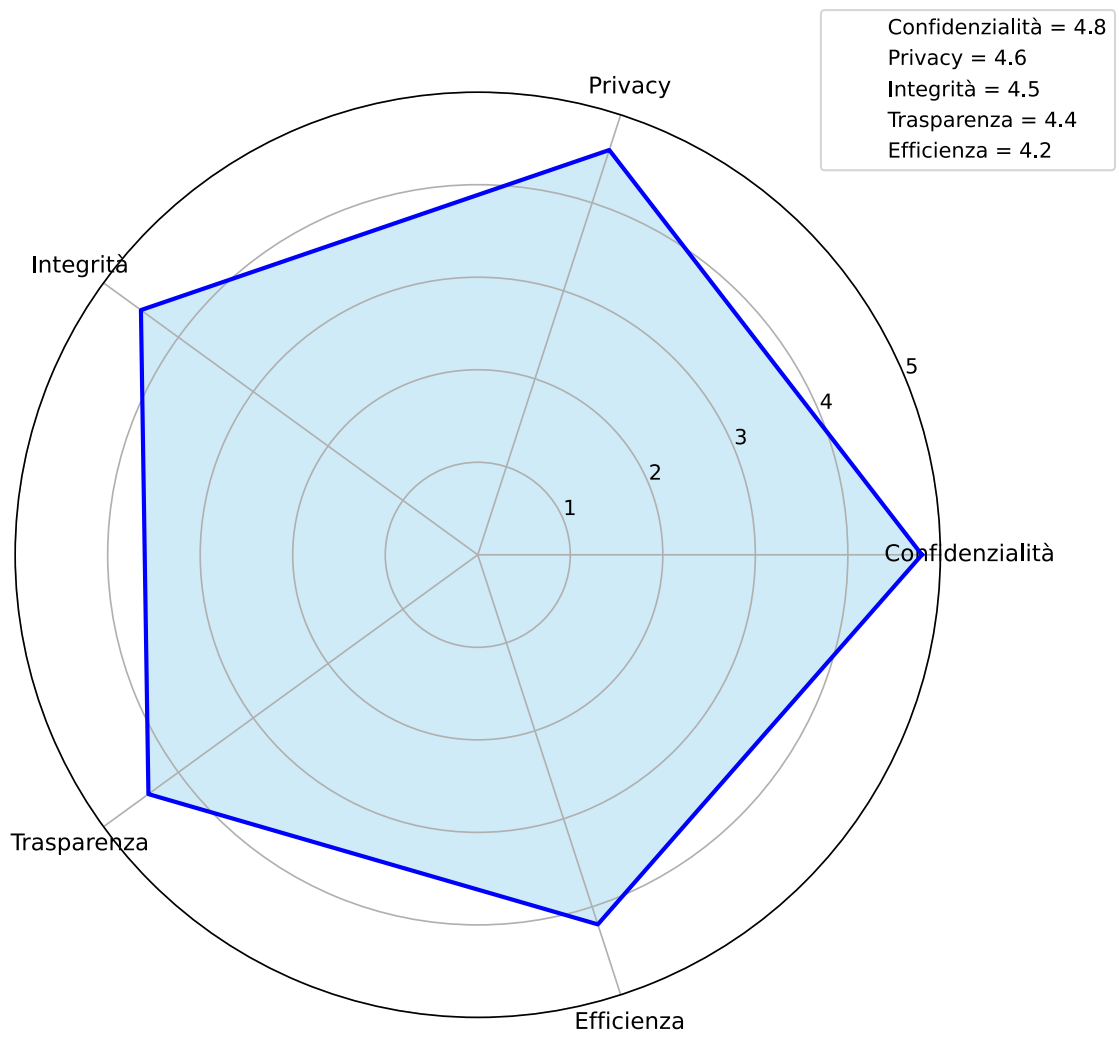


Figure 3.1: Grado di soddisfazione delle proprietà principali di sicurezza

---

---

# CAPITOLO 4

---

## WORK PACKAGE 4

In questo capitolo viene presentata l'implementazione del sistema decentralizzato per la gestione delle recensioni online, come delineato nei Work Package precedenti. L'obiettivo principale è dimostrare la realizzabilità tecnica della soluzione proposta nel WP2 e verificarne il corretto comportamento in un contesto applicativo reale. A tal fine, si procede a illustrare l'organizzazione del codice sorgente, i moduli implementati, i meccanismi di interazione previsti sia in modalità programmatica (CLI) che grafica (GUI) e infine le metriche prestazionali rilevate in ambiente simulato. Il capitolo è strutturato in modo da distinguere le componenti che interagiscono direttamente con la blockchain tramite riga di comando (smart contract, script di deploy e interazione) da quelle accessibili tramite interfaccia utente (back-end, mock di identità digitale, dApp). A seguire, sarà effettuata un'analisi delle limitazioni attualmente presenti nel sistema e una discussione sulle possibili evoluzioni future, concludendo il tutto con una valutazione complessiva della robustezza dell'implementazione rispetto agli obiettivi progettuali.

### 4.1 Smart contracts

L'implementazione dei contratti intelligenti costituisce il nucleo logico del sistema, incaricato di garantire le proprietà crittografiche, reputazionali e comportamentali previste dal modello. Tutti i contratti sono sviluppati in linguaggio Solidity (versione 0.8.25) e sono organizzati all'interno della directory `/code/contracts/`. Ciascun contratto è stato progettato per essere modulare, riutilizzabile e ispezionabile, in coerenza con i principi di trasparenza e auditabilità introdotti nel WP2. Le funzioni critiche sono protette da controlli di proprietà, timestamp o firma, per prevenire usi impropri o double-spending/replay attack.

Di seguito si riporta una sintesi dei contratti sviluppati

- **ReviewNFT.sol** Gestisce l'emissione di NFT soulbound che fungono da prova d'acquisto

(Proof-of-Purchase). Ogni token è associato a un identificatore di prodotto, un DID, una data di acquisto e uno stato (valido, usato, scaduto). Il contratto include meccanismi di marcatura automatica dello stato e protezione contro il riutilizzo o la cessione fraudolenta.

- **ReviewStorage.sol** Rappresenta il registro crittograficamente verificabile delle recensioni. Conserva l'hash IPFS del contenuto off-chain, lo stato (attiva, modificata, revocata) e gli eventi storici di modifica. Integra inoltre vincoli temporali (cooldown) e restrizioni di modifica/revoca legate alla proprietà del token.
- **BadgeNFT.sol** Implementa la logica reputazionale mediante NFT soulbound che rappresentano livelli di partecipazione (Bronze, Silver, Gold), aggiornati dinamicamente in funzione del numero e della tempestività delle recensioni effettuate.
- **DIDRegistry.sol** Registra in maniera univoca un DID associato all'hash di una Verifiable Credential. Il contratto garantisce che ciascuna VC venga utilizzata una sola volta e consente la successiva verifica della legittimità del recensore.
- **VCRegistry.sol** Responsabile della verifica delle Verifiable Credential presentate tramite Verifiable Presentation. Include logiche di revoca basate su revocation index e hash su IPFS.
- **BanRegistry.sol** Tiene traccia dei DID bannati in seguito a comportamenti scorretti o fraudolenti. L'inserimento in ban list è irreversibile a livello on-chain e impedisce qualsiasi futura interazione del soggetto col sistema. Il contratto è consultabile pubblicamente ed è richiamato come vincolo di validazione in tutti gli altri moduli.
- **AttestationRegistry.sol** Memorizza attestazioni pubbliche o prove a conoscenza zero (ZKP) che consentono all'utente di dimostrare proprietà aggregate (eg. reputazione) senza compromettere l'anonimato, allegando on-chain l'esito hashato delle prove.
- **Semaphore.sol** / **SemaphoreVerifier.sol** Moduli ausiliari per la verifica di ZKP secondo lo standard **Semaphore**. Questi contratti garantiscono la non linkabilità delle interazioni, prevenendo attacchi Sybil e assicurando che ogni azione firmata sia riconducibile a una sola identità logica, senza renderla identificabile.

I contratti sono stati compilati, lanciando il comando: `npx hardhat compile`

## 4.2 Deploy scripts

La procedura di deployment dei contratti intelligenti è stata progettata per garantire riproducibilità, chiarezza operativa e modularità. Essa si articola mediante una serie di script in JavaScript localizzati nella directory `/code/scripts/deploy/`, ciascuno dei quali è responsabile del deploy di uno specifico contratto. L'ambiente di esecuzione adottato è Hardhat, configurato per operare in modalità locale con Ganache impostato su `localhost:8545`, con possibilità di estensione

a reti Ethereum compatibili.

Ogni script segue una struttura uniforme:

- inclusione delle librerie Hardhat (**ethers**) e delle interfacce ABI;
- compilazione e deploy del contratto (con parametri costruttori quando richiesti);
- attesa della conferma di inclusione in blockchain;
- stampa a console dell'indirizzo del contratto deployato e memorizzazione dello stesso in un file apposito (**contract-addresses.json**).

L'output del deploy include hash delle transazioni e indirizzi contrattuali, che possono essere verificati tramite **Hardhat console** o interfacce custom di auditing. Tutti gli script possono essere eseguiti individualmente oppure concatenati tramite uno script composito.

### 4.3 Interact scripts (CLI)

Gli script di interazione implementano le operazioni fondamentali che le parti oneste del sistema possono eseguire sulla blockchain, secondo i protocolli descritti nel WP2. Essi permettono di simulare scenari reali, testare le funzionalità dei contratti, e validare il corretto comportamento delle entità coinvolte. Tali script, collocati nella directory `/code/scripts/interact/`, sono sviluppati in JavaScript e utilizzano il framework **Hardhat** in combinazione con **ethers.js** per l'interazione diretta con gli smart contract. Ogni script è autonomo e richiamabile da riga di comando. I parametri (indirizzi, hash, identificativi NFT) sono forniti direttamente nel codice o tramite file di configurazione. Gli script producono log dettagliati per ogni operazione eseguita, inclusi gli esiti di verifica e gli hash delle transazioni generate.

Le operazioni principali supportate sono le seguenti:

- **CHANGE-THIS-FILE-fetchDID.js**

Deriva identità Ethereum reali (2 issuer e 2 holder) da un mnemonic Ganache, genera quattro DID, uno per ogni identità, e salva localmente i corrispondenti file **issuer/holder.json** per l'uso negli altri script.

NOTA: Il nome del file va modificato in **fetchDID.js** e il mnemonic va cambiato con il proprio di Ganache locale.

- **interactReviewNFT.js**

Simula il minting di un NFT soulbound associato a un prodotto e a un holder DID, e successivamente il suo utilizzo (marcandolo come usato) per autorizzare una recensione.

- **interactReviewStorage.js**

Carica un contenuto testuale su IPFS, associa l'hash generato a un NFT valido e ne registra l'hash nel contratto **ReviewStorage**. Inoltre, dimostra la possibilità di aggiornare una recensione con un nuovo CID, preservando lo storico.

- `interactVCRegistry.js`

Firma off-chain una Verifiable Credential (VC), ne calcola l'hash, la registra nel contratto `VCRegistry`, ne verifica la validità e ne simula la revoca. Prevede uso combinato con `DIDRegistry`.

- `interactDIDRegistry.js`

Esegue operazioni di consultazione e aggiornamento sul registro delle identità decentralizzate, tra cui owner lookup, aggiunta/rimozione delegati e cambio proprietario del DID.

- `interactBadgeNFT.js`

Interagisce con il contratto `BadgeNFT` per verificare la logica di assegnazione e aggiornamento dei badge reputazionali. Include la verifica della non trasferibilità del token.

- `interactAttestationRegistry.js`

Registra attestazioni crittografiche derivate da prove di tipo `Semaphore`, `BBS+` o `VP`. L'utente può scegliere il tipo di prova da ancorare specificando parametri via CLI.

- `interactBanRegistry.js`

Inserisce o rimuove un DID dalla lista dei soggetti bannati nel contratto `BanRegistry`, mostrando il relativo stato prima e dopo l'operazione.

- `verifyVP.js`

Genera e verifica una Verifiable Presentation (VP) standard e una sua variante Selective Disclosure (SD-JWT). Produce hash riutilizzabili in attestazioni successive.

- `verifyBBS.js`

Dimostra la generazione e la verifica di una prova `BBS+`, partendo da una VC firmata. Supporta selective disclosure e verifica di validità crittografica della sottoprova.

- `verifySemaphore.js`

Crea un gruppo `Semaphore`, genera una Zero-Knowledge Proof basata su Merkle tree e verifica on-chain la validità della prova tramite `validateProof()`.

La separazione tra fase di deploy e fase di interazione consente un testing modulare, utile sia per validare singoli contratti, sia per eseguire scenari end-to-end. Tutti gli script sono progettati per poter essere integrati in pipeline CI/CD o in ambienti di test automatico. Tutti gli script possono essere eseguiti individualmente oppure concatenati tramite uno script composito.

### 4.4 Interazione via GUI

A complemento degli script CLI, il sistema implementa una serie di componenti ad interfaccia grafica che consentono l'interazione utente tramite browser. Tali moduli sono concepiti per offrire un'esperienza d'uso completa, accessibile anche a utenti non tecnici, e riproducono integralmente

i flussi funzionali definiti nel WP2.

L'interazione si basa su 3 diverse componenti:

- **Back-end Express:** funge da nodo ausiliario per la gestione delle credenziali, la generazione delle Verifiable Credential (VC) e l'interazione REST con la rete blockchain.
- **Identity Provider Mock (IdP):** un modulo simulato conforme ai requisiti di SPID, responsabile dell'autenticazione iniziale e dell'emissione controllata delle credenziali in formato SAML2.
- **Frontend React (dApp):** una piattaforma decentralizzata lato client, attraverso cui l'utente può registrarsi, visualizzare NFT, inviare recensioni, consultare lo storico e dimostrare la propria reputazione tramite VP/BBS+/ZKP.

A differenza degli script CLI, che sono orientati alla validazione funzionale dei singoli contratti, l'interfaccia GUI realizza una simulazione integrata del sistema in condizioni realistiche, consentendo l'emulazione completa del ciclo di vita di una recensione (dall'autenticazione fino all'audit pubblico). Nei paragrafi seguenti, ciascun componente sarà descritto in termini di obiettivi funzionali, struttura, interfacce e meccanismi di comunicazione con la blockchain.

### 4.4.1 Server Backend Express

Il back-end dell'applicazione è implementato come server **Express** scritto in **TypeScript** (file `server.ts`), con l'obiettivo di gestire l'interazione off-chain tra l'utente, il sistema di identità e la rete blockchain. Esso funge da punto di intermediazione fidata per la creazione delle Verifiable Credential (VC) e l'invio di Verifiable Presentation (VP), in conformità con le specifiche tecniche del paradigma Self-Sovereign Identity (SSI).

#### Funzionalità principali

Il backend offre un insieme di endpoint REST che implementano le seguenti funzionalità:

- **Autenticazione dell'utente** tramite SPID (simulata) o sistema di login decentralizzato.
- **Validazione e sanificazione** dei dati ricevuti dal frontend, con controllo formale di correttezza (es. campi alfabetici, indirizzi wallet validi).
- **Generazione di VC** firmate con chiave privata dell'Issuer, in formato JWT conforme a `did-jwt-vc`.
- **Revoca e verifica** di VC tramite accesso alla Revocation List distribuita su IPFS.
- **Accesso ai moduli di attuazione on-chain**, come:
  - `appBanRegistry.js` – per ban/unban di DID;
  - `appReviewNFT.js` – per esecuzione automatica della funzione `expireNFTs()`;

- `appAttestationRegistry.js` – per registrare attestazioni crittografiche;
- `appVP.js`, `appBBS.js` – per generazione e archiviazione locale di prove ZKP (SD-JWT, BBS+).

### Struttura del codice

All'avvio, il server carica le chiavi dell'*Issuer* da file `issuer-did.json` e configura un signer Ethereum (via `ethers.Wallet`) per autorizzare operazioni su contratti. La rete attesa è un endpoint JSON-RPC locale, compatibile con Ganache o Hardhat.

Tra le funzioni implementate nel back-end si evidenziano:

- `getUserHash()` – derivazione SHA-256 di un'identità utente simulata, concatenando i campi del form apposito per il rilascio della VC;
- supporto per *Selective Disclosure*;

### Interfacciamento on-chain

Il back-end funge da *Issuer* autorizzato e pubblica le hash delle VC nel contratto `VCRegistry`, fornendo il riferimento per la verifica crittografica durante la registrazione dell'utente nel `DIDRegistry`. Esso opera come ponte tra identità off-chain (verificate) e entità on-chain, garantendo che ogni DID sia collegato a una sola identità reale, ma senza rivelarne l'identificativo.

### Ruolo nel sistema

Il back-end agisce come nodo di controllo fidato per operazioni amministrative e simulazioni di comportamento realistico (eg. utenti inattivi, revoca NFT, ban manuali). Inoltre, fornisce utili astrazioni per il popolamento iniziale del sistema e per l'integrazione con prove generate off-chain.

#### 4.4.2 Identity Provider Mock

L'emulazione dell'Identity Provider è realizzata mediante un server indipendente localizzato nella directory `/code/idp-mock/`, implementato con `Express-TypeScript` e contenuto nel file `server.ts`. Questo componente ha lo scopo di simulare, in ambiente controllato, il comportamento essenziale di un Identity Provider conforme al profilo SPID, limitandosi alla generazione di una *SAML2 Response* non firmata contenente un'asserzione di autenticazione. È stato implementato per scopi puramente dimostrativi delle potenzialità e per offrire un'alternativa di login federata, così da ottenere compatibilità anche con sistemi quali eIDAS-2.0 (EUDIW).

### Flusso di autenticazione

Il server espone un endpoint POST su `/sso`, che emula la fase di login dell'utente. Alla ricezione della richiesta, il server costruisce:



- una **SAML2 Assertion** contenente i campi **Subject**, **Issuer**, **Conditions** e **AuthnStatement**;
- una **SAML2 Response** che incapsula tale assertion, formattata secondo lo standard `urn:oasis:names:tc:SAML:2.0`.

Tutti i dati sono generati dinamicamente con riferimenti temporali coerenti, ma la risposta non è firmata digitalmente, in quanto l'obiettivo è esclusivamente dimostrativo.

### Comunicazione con il Service Provider

Il server restituisce una pagina HTML contenente un form POST automatico verso l'endpoint `/assert` del backend applicativo (in ascolto su porta 8082). Il campo **SAMLResponse** include la risposta codificata in Base64, simulando il comportamento effettivo di un browser dopo login SPID.

### Caratteristiche tecniche

- Porta di ascolto: 8443;
- Protocollo: HTTP semplice, senza TLS;
- Linguaggio: **TypeScript**, con dipendenze minime;

### Limitazioni

Il mock non implementa alcuna firma XML, validazione degli attributi, gestione sessione o Discovery Service, e non rispetta pienamente lo standard SPID. Tuttavia, la struttura della risposta è conforme e può essere facilmente estesa per includere una firma XML, supporto a certificati X.509 e controllo degli ID.

### Funzione nel sistema

Il mock IdP rappresenta un componente fondamentale per simulare un'alternativa al rilascio controllato dell'identità iniziale e per testare localmente il flusso completo di autenticazione SPID, senza dipendere da ambienti esterni o provider ufficiali. Il collegamento tra questo modulo e il backend consente di generare un'identità logica che verrà successivamente associata ad una VC.

### 4.4.3 Frontend React (dApp)

L'interfaccia grafica per l'utente finale è realizzata tramite una *decentralized application* (dApp) sviluppata in **React** e scritta interamente in **TypeScript**, con utilizzo di **Vite** per il bundling e **TailwindCSS** per la gestione dello stile. L'obiettivo del frontend è rendere accessibile l'interazione con i contratti e la logica reputazionale in modalità completamente client-side, mantenendo al contempo compatibilità con le librerie crittografiche utilizzate off-chain. L'interfaccia utente del sistema è organizzata in forma modulare, con componenti React raggruppati in **Panel**, ciascuno responsabile di una sezione logica della dApp. Ogni pannello è

implementato come componente `.tsx` indipendente e importato nella struttura principale della pagina tramite il file `ReviewDApp.tsx`. Di seguito si analizzano i tre pannelli principali presenti nel sistema. La dApp dispone di script interattivi appositamente creati per l'utilizzo in questo campo, separandoli dunque dall'interazione effettuata tramite CLI.

### Struttura generale

Il file principale `App.tsx` definisce l'interfaccia di alto livello e carica il componente `ReviewDApp`, che gestisce la logica principale della dApp. Quest'ultimo si occupa di orchestrare l'interazione con i wallet (via `MetaMask`), il caricamento e la visualizzazione delle componenti chiave: identità decentralizzate, credenziali, NFT di acquisto, recensioni e badge reputazionali.

### WalletAccessPanel

Il pannello `WalletAccessPanel` gestisce l'interazione iniziale con il wallet dell'utente e la visualizzazione delle sue credenziali. Il pannello incorpora internamente i componenti `DIDTable.tsx` e `VCCard.tsx`, consentendo un'interfaccia coesa tra identità e credenziali.

Le funzionalità includono:

- connessione al wallet Ethereum tramite `MetaMask`;
- lettura e visualizzazione della VC dell'utente;
- creazione (tramite funzione `EdDSA`) e visualizzazione dei DID e del loro stato (registrato, bannato), con possibilità di eliminarli;

### SignedReviewsPanel

Questo pannello è incaricato della gestione e consultazione delle recensioni pubblicate. Il caricamento dei contenuti testuali delle recensioni è effettuato lato client mediante `IPFS`, utilizzando `ipfs-http-client`. Le operazioni gestite includono:

- minting di un NFT (inserito per testing);
- selezione del DID e del NFT che si vogliono utilizzare per recensire;
- campo testuale per la scrittura del contenuto della recensione;
- bottone per l'invio della recensione firmata al contratto `ReviewStorage`, con validazione del corrispondente NFT;
- visualizzazione delle recensioni associate all'utente corrente;
- tracciamento delle modifiche e delle revoche, con `CID` e date coerenti.

### ZKPreputationPanel

Tale pannello consente all'utente di generare e gestire prove circa l'identità e la propria reputazione aggregata.

Esso offre:

- interfaccia per selezionare il tipo di prova desiderata (SD-JWT, BBS+, Semaphore);
- generazione della prova in locale (per VP SD-JWT e BBS+);
- ancoraggio dell'esito hashato on-chain (per VP SD-JWT e BBS+);
- generazione, verifica e ancoraggio on-chain per Semaphore.

## 4.5 Come testare

Per testare il comportamento di ogni componente finora descritta, è suggerito eseguire i seguenti passaggi:

1. `git clone https://github.com/Kirito-Emo/Decentralized-Ecommerce.git`
2. `cd code/ && npm install`
3. `npx hardhat compile`
4. `npx hardhat run scripts/deploy/deployBanRegistry.js --network localhost`  
`npx hardhat run scripts/deploy/deployReviewNFT.js --network localhost`  
`npx hardhat run scripts/deploy/deployBadgeNFT.js --network localhost`  
`npx hardhat run scripts/deploy/deployReviewStorage.js --network localhost`  
`npx hardhat run scripts/deploy/deployVCRegistry.js --network localhost`  
`npx hardhat run scripts/deploy/deployAttestationRegistry.js --network localhost`  
`npx hardhat run scripts/deploy/deploySemaphoreVerifier.js --network localhost`  
`npx hardhat run scripts/deploy/deploySemaphore.js --network localhost`  
`npx hardhat run scripts/deploy/deployDIDRegistry.js --network localhost`
5. `npx hardhat run scripts/interact/fetchDID.js --network localhost`  
`npx hardhat run scripts/interact/interactBanRegistry.js --network localhost`  
`npx hardhat run scripts/interact/interactReviewStorage.js --network localhost`  
`npx hardhat run scripts/interact/interactReviewNFT.js --network localhost`  
`npx hardhat run scripts/interact/interactBadgeNFT.js --network localhost`  
`npx hardhat run scripts/interact/interactDIDRegistry.js --network localhost`  
`npx hardhat run scripts/interact/interactVCRegistry.js --network localhost`  
`npx hardhat run scripts/interact/verifyVP.js --network localhost`  
`npx hardhat run scripts/interact/interactAttestationRegistry.js --network localhost`

```
npx hardhat run scripts/interact/verifyBBS.js --network localhost
npx hardhat run scripts/interact/verifySemaphore.js --network localhost
```

6. `cd code/backend && npm install && npm start`
7. `cd code/idp-mock && npm install && npm start`
8. `cd code/frontend && npm install && npm run dev`

Le componenti necessitano di differenti pacchetti npm, per tale motivo è necessario effettuare diversi `npm install` a seconda della cartella in cui si naviga. Tale scelta è stata presa con l'idea di modularizzare il sistema e renderlo flessibile e lightweight per coloro che intendessero testare esclusivamente via CLI, lasciando la decisione di installare ulteriori pacchetti relativi alla GUI, solo a coloro che intendessero testare il sistema nella sua interezza.

## 4.6 Prestazioni del Sistema

In questa sezione, analizziamo le prestazioni dei contratti intelligenti utilizzati nel sistema decentralizzato di recensioni per l'e-commerce. I seguenti benchmark sono stati condotti per misurare il consumo di gas e i tempi di esecuzione delle funzioni più critiche: `ReviewNFT`, `ReviewStorage`, `BadgeNFT`, `VCRegistry`, `DIDRegistry`, `BanRegistry` e `AttestationRegistry`.

### 4.6.1 Benchmark di Consumo di Gas e Tempo di Esecuzione

Funzione	Consumo Gas (Avg)	Tempo di Esecuzione (Avg)	Tempo di Esecuzione (Std Dev)
<code>ReviewNFT (mint + use)</code>	240,589	75.88 ms	5.21 ms
<code>ReviewStorage (store + update)</code>	297,816	75.10 ms	5.40 ms
<code>BadgeNFT (updateReputation)</code>	44,692	95.60 ms	7.98 ms
<code>VCRegistry (register + revoke)</code>	213,157	52.13 ms	3.88 ms
<code>DIDRegistry (changeOwner)</code>	32,465	25.10 ms	4.11 ms
<code>BanRegistry (ban + unban)</code>	49,406	20.50 ms	2.52 ms
<code>AttestationRegistry (anchoring attestation)</code>	115,678	142.68 ms	0.00 ms

Table 4.1: Consumo di Gas e Tempo di Esecuzione dei Contratti

### 4.6.2 Efficienza del Consumo di Gas e del Tempo di Esecuzione

I dati indicano che le funzioni `minting` e `storing reviews` sono le operazioni più costose in termini di gas, con `ReviewNFT` e `ReviewStorage` che richiedono più di 240.000 gas (240,589 e 297,816 gas rispettivamente) e tempi di esecuzione simili (circa 75 ms). Questi sono costi attesi per le interazioni blockchain, considerando la complessità della gestione di NFT e della memorizzazione delle recensioni sulla blockchain.

- `**ReviewNFT (mint + use)**` e `**ReviewStorage (store + update)**` sono entrambe molto costose in termini di gas, ma i tempi di esecuzione sono rapidi (circa 75 ms), il che è accettabile per operazioni interattive con l'utente.

- **\*\*BadgeNFT (updateReputation)\*\*** è la funzione meno costosa in termini di consumo di gas (44,692 gas), ma il tempo di esecuzione è più lungo (95.60 ms), suggerendo che, sebbene il consumo di gas sia basso, la logica interna della funzione potrebbe comportare passaggi aggiuntivi che aumentano il tempo di esecuzione.

- Le funzioni **VCRegistry** e **DIDRegistry**, come la registrazione e il cambio del proprietario, sono molto efficienti sia in termini di gas che di tempo, con costi di gas inferiori a 220.000 e tempi di esecuzione di circa 50 ms.

- La funzione **BanRegistry** (ban/unban) è estremamente efficiente, con il consumo di gas più basso, pari a 49,406 e tempi di esecuzione rapidi (20 ms).

- La funzione **AttestationRegistry**, responsabile dell'ancoraggio delle attestazioni, è relativamente più lenta (142.68 ms), probabilmente a causa del processamento aggiuntivo necessario per validare e ancorare le attestazioni sulla blockchain.

### 4.6.3 Valutazione delle Prestazioni

Funzione	Valutazione delle Prestazioni
ReviewNFT (mint + use)	Alto consumo di gas, ma tempi di esecuzione ragionevoli. Adatto per operazioni occasionali di minting.
ReviewStorage (store + update)	Costoso in termini di gas, ma ottimizzato nei tempi di esecuzione, adatto per la gestione periodica delle recensioni.
BadgeNFT (updateReputation)	Efficiente in termini di gas, ma i tempi di esecuzione potrebbero essere migliorati per una migliore esperienza utente.
VCRegistry (register + revoke)	Molto efficiente per la registrazione e la revoca, garantendo operazioni fluide nel sistema.
DIDRegistry (changeOwner)	Ottimizzato, efficiente e veloce. Adatto per interazioni rapide da parte dell'utente.
BanRegistry (ban + unban)	Estremamente efficiente, adatto per funzioni frequentemente invocate.
AttestationRegistry (anchoring attestation)	Richiede più tempo di esecuzione, ma i costi di gas sono nei limiti accettabili per operazioni infrequenti.

Table 4.2: Valutazione delle Prestazioni delle Funzioni

### 4.6.4 Analisi del Gas e del Tempo: Confronto con Sistemi Tradizionali

Nei sistemi tradizionali, operazioni come la memorizzazione delle recensioni degli utenti o l'elaborazione dei cambiamenti di reputazione fanno affidamento su database centralizzati, con costi di gas trascurabili. Tuttavia, sulla blockchain, ogni transazione richiede gas, rendendo queste operazioni più costose. Nonostante ciò, la natura decentralizzata del sistema offre vantaggi significativi, come:

- **Immutabilità:** La blockchain garantisce che le recensioni e le transazioni non possano essere modificate, assicurando l'autenticità.
- **Trasparenza:** Tutte le azioni sono verificabili pubblicamente sulla blockchain, il che aumenta la fiducia
- **Resistenza alla Censura:** Nessuna autorità centrale può alterare o rimuovere i dati, proteggendo l'integrità delle recensioni

Considerando questi fattori, i maggiori costi di gas sono giustificati dall'aumentata sicurezza e trasparenza fornita dalla blockchain.

## 4.7 Limitazioni e sviluppi futuri

Nonostante l'implementazione rispetti fedelmente le specifiche delineate nei Work Package precedenti, alcune funzionalità rimangono parzialmente sviluppate o soggette a margini di miglioramento, sia sotto il profilo tecnico che crittografico. In questa sezione si analizzano le principali limitazioni identificate durante la fase di sviluppo e test, proponendo al contempo possibili evoluzioni future.

### 4.7.1 Limitazioni attuali

- **Revoca delle VC parziale**

Il sistema prevede un meccanismo di revoca delle VC tramite una lista off-chain (bitstring su IPFS), ma tale lista non è aggiornata dinamicamente dal backend né gestita tramite interfacce dedicate. L'aggiornamento della revocation list avviene manualmente.

- **Assenza di firma XML**

Il mock IdP emette risposte SAML2 non firmate. Sebbene utile per test didattici, ciò limita l'aderenza allo standard SPID e impedisce l'integrazione con reali Service Provider compatibili.

- **Interazione IPFS locale**

Il caricamento dei contenuti su IPFS richiede un nodo locale attivo sulla porta 5001. Non è prevista integrazione con gateway pubblici o fallback automatici in caso di errore.

### 4.7.2 Evoluzioni possibili

- **Supporto per SPID reale**

L'integrazione di una firma XML (con librerie come `xml-crypto` e certificati auto-firmati) consentirebbe il collegamento con simulatori SPID conformi.

- **Gestione automatica revoche**

Un modulo backend può essere implementato per aggiornare dinamicamente la revocation list su IPFS, generando e firmando la nuova versione in modo compatibile con i contratti.

- **Transizione a Layer 2**

Per ridurre i costi del gas, è previsto il porting della piattaforma su rete Layer 2 compatibile con Ethereum (eg. Arbitrum, Optimism), mantenendo le garanzie di trasparenza e auditabilità.

- **Dashboard di audit pubblica**

È possibile sviluppare una dashboard esterna per consultare statistiche aggregate sul sistema, anomalie comportamentali e reputazione distribuita.

## **4.8 Conclusione**

L'implementazione presentata in questo Work Package dimostra la fattibilità tecnica della soluzione progettata nel WP2, confermando la coerenza rispetto al modello concettuale delineato nel WP1 e alla valutazione di sicurezza svolta nel WP3. Il sistema risultante realizza un ambiente decentralizzato, verificabile e pseudo-anonimo per la pubblicazione di recensioni, integrando smart contract Solidity, componenti crittografiche avanzate e un'interfaccia grafica intuitiva.

Dal punto di vista architetturale, il sistema si compone di una dApp React, un backend minimalista per la gestione delle credenziali e dei flussi ausiliari, un Identity Provider simulato conforme al profilo SPID, e un insieme modulare di contratti deployati su blockchain Ethereum compatibile. Le interazioni critiche (registrazione, recensione, reputazione) sono completamente tracciabili e verificabili on-chain, mentre i contenuti sensibili rimangono off-chain tramite IPFS.

Nonostante alcune funzionalità, come la firma SAML2, siano attualmente simulate o delegate a script esterni, il sistema fornisce già una base solida per esperimenti futuri nel settore della decentralizzazione delle identità. Le performance osservate in ambiente locale confermano la sostenibilità della soluzione anche in scenari più ampi, con possibilità di ottimizzazione tramite soluzioni Layer 2.

Nel complesso, l'implementazione non solo soddisfa i requisiti funzionali e di sicurezza posti all'inizio del progetto, ma offre anche un'architettura estensibile e ben documentata, che si presta a essere adattata a contesti applicativi differenti o integrata con componenti avanzati di governance e auditing.