

UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE ED ELETTRICA E
MATEMATICA APPLICATA



Project work - Blockchains

Progettazione di un sistema decentralizzato di recensioni online

Docente

Carlo Mazzocca - cmazzocca@unisa.it

Membri del gruppo

Nome e Cognome	Matricola	E-mail
De Caro Antonio	622702316	a.decaro39@studenti.unisa.it
Conato Christian	0622702273	c.conato@studenti.unisa.it
Garofalo Mariachiara	0622702173	m.garofalo38@studenti.unisa.it

ANNO ACCADEMICO 2024/2025

1 WP1 - Modello del sistema	3
1.1 Funzionalità da realizzare	3
1.1.1 Il problema	3
1.1.2 L'obiettivo	3
1.1.3 Attori onesti del sistema	4
1.1.4 Strumenti del sistema	7
1.2 Threat model del sistema	9
1.2.1 Attori disonesti del sistema	9
1.2.2 Descrizione avversari	11
1.3 Proprietà di resilienza	19
2 WP2 - Soluzione	21
2.1 Panoramica	21
2.2 Tecnologie adottate	22
2.2.1 W3C:	22
2.2.2 Verifiable Credential	22
2.2.3 Verifiable Presentation	23
2.2.4 Selective Disclosure tramite BBS+	23
2.2.5 Utilizzo della blockchain	24
2.3 Fase di registrazione	25
2.4 Fase di acquisto	26
2.5 Fase di recensione	27
2.5.1 Gestione delle recensioni, reward e penalizzazioni	28
2.6 Credenziali di accesso e policy di acquisto	29
3 WP3 - Analisi	31
3.1 Analisi degli avversari	31
3.1.1 Attacchi alla qualità delle recensioni (inserire o manipolare feedback) . .	31
3.1.2 Forzare l'accesso o falsificare le credenziali	35
3.1.3 Man in the middle (MITM) e replay attack: furto, reinvio di VC/VP .	36
3.1.4 Erosione della privacy/de-anonimizzazione	39
3.2 Trasparenza ed efficienza	40

4	WP4 - Implementazione e Prestazioni	42
4.1	Setup iniziale	42
4.2	Fase preliminare	43
4.3	Fase di registrazione	43
4.4	Fase di acquisto	44
4.5	Fase di recensione	45
4.6	Fase di reward	47

WP1 - Modello del sistema

Lo scopo del seguente capitolo è quello di effettuare un’analisi degli attori onesti all’interno del sistema, esaminando i loro scopi, le funzionalità che intendono sfruttare e i loro limiti.

Gli attori onesti sono quei soggetti/entità che operano nel rispetto delle regole e delle politiche stabilite dal sistema che utilizzano, cercando di raggiungere i loro obiettivi senza compromettere l’integrità del sistema stesso. In questo stesso capitolo, si esaminano i potenziali avversari (o **threat model**) che potrebbero tentare di compromettere il sistema. Di tali figure si valutano le risorse e le motivazioni che li spingono ad agire. Al termine del capitolo si identificano poi le proprietà di sicurezza che si desidera mantenere in presenza di possibili attacchi (le **proprietà di resilienza** del sistema). Queste ultime sono essenziali per assicurare il corretto funzionamento del sistema e la protezione delle informazioni sensibili.

1.1 Funzionalità da realizzare

Di seguito vi è una descrizione dettagliata del contesto di lavoro, il problema affrontato, gli attori onesti coinvolti nel sistema e gli elementi che utilizzano, mettendo in evidenza le interazioni tra i vari componenti del sistema.

1.1.1 Il problema

I sistemi di recensioni online, oggi fondamentali per le decisioni di consumatori e aziende, soffrono di un grave problema di affidabilità. La facilità con cui si generano identità fasulle, l’opacità degli algoritmi che determinano la visibilità dei commenti, la mancanza di incentivi economici per chi lascia feedback autentici e le pratiche collusive tra piattaforme e operatori commerciali convergono nel minare la credibilità complessiva del sistema.

1.1.2 L’obiettivo

L’obiettivo del progetto è realizzare un sistema decentralizzato per la gestione di recensioni online affidabili, nel contesto di una piattaforma e-commerce.

Il sistema vuole garantire che le recensioni siano consultabili da chiunque in modo pubblico e trasparente, ma che la loro pubblicazione, modifica, revoca e valutazione sia riservata esclusivamente a utenti autenticati e che abbiano effettivamente acquistato il prodotto oggetto della recensione, fornendo meccanismi di reward e penalizzazioni per aumentare la loro affidabilità. L’idea alla base è quella di coniugare le caratteristiche offerte dalle tecnologie decentralizzate, come l’immutabilità e la pubblica verificabilità delle informazioni, con la necessità di autenticare in modo sicuro e pseudo-anonimo gli utenti che interagiscono con il sistema.

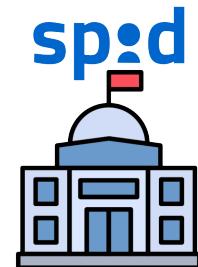
1.1.3 Attori onesti del sistema

Esistono diversi attori coinvolti nel sistema, ognuno con specifici obiettivi e funzionalità.

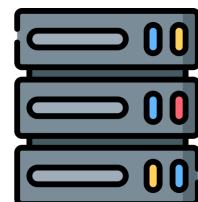
Nota!

Si considerano utenti tutti gli attori/entità che utilizzano il sistema sviluppato, sia in maniera onesta che non. Gli utenti sono quindi sia i **Client** (legittimi fruitori del servizio) che eventuali **Avversari** (tentano di forzare il sistema per i loro scopi).

- **Issuer (SPID):** Autorità affidabile incaricata al rilascio delle **Verifiable Credentials (VC)** contenenti le credenziali utili per gli utenti che intendono utilizzare un servizio, attestando la loro identità in modo verificabile, ma mantenendo la possibilità di pseudo-anonimato tramite l'uso di **Decentralized Identifiers (DIDs)**.
- **Strumenti:**
 - Richiesta credenziali degli utenti;
 - DID (*Decentralized Identifier*) issuer.



- **Verifier V:**
 - **Descrizione:** Sito e-commerce su cui è possibile acquistare dei prodotti. Il servizio è rivolto ai soli utenti in possesso di credenziali che ne certifichino i requisiti di accesso.
 - **Obiettivi:**
 - * Fornire il proprio servizio ai soli utenti meritevoli;
 - * Riconoscere le sole credenziali fornite da issuer affidabili;
 - **Strumenti:**
 - * Credenziali (*Verifiable Presentation*) di accesso degli utenti;
 - * DID (*Decentralized Identifier*) verifier.
 - **Esempio:** L'accesso alla risorsa richiede che il luogo di residenza o il luogo di nascita del client sia Roma. Il client è nato a Roma e vuole accedere al verifier V; egli può farlo solo e soltanto se V riconosce come affidabile l'issuer che ha rilasciato le credenziali del client C.



- Client C:

- **Descrizione:** I client sono utenti che utilizzano onestamente l'infrastruttura realizzata.

Si possono identificare due tipologie di client:

- * **Ospite:** colui che intende solo acquistare o visualizzare un prodotto dall'e-commerce senza effettuare registrazione presso il servizio;

- * **Holder:** colui che intende registrarsi presso il servizio per poter usufruire di tutte le funzionalità previste.

A tal fine è necessario ottenere le credenziali dall'issuer.



- **Obiettivi:**

- * Ottenere credenziali di accesso contenute nelle VC rilasciate dall'issuer;
 - * Utilizzare le credenziali per accedere al servizio di e-commerce;
 - * Condividere solo le informazioni personali essenziali per mantenere la privacy.

- **Strumenti:**

- * VC (*Verifiable Credential*);
 - * VP (*Verifiable Presentation*);
 - * DID (*Decentralized Identifier*) holder.

- **Blockchain Ethereum:**

– **Descrizione:** Infrastruttura decentralizzata che conserva informazioni pubblicamente verificabili e trasparenti.

Si possono identificare due utilizzi della blockchain:

- * **Verifiable data registry (VDR):** Si conservano i DID, DID document degli attori che sono registrati all'e-commerce per effettuare acquisti o lasciare recensioni;

- * **Smart contract logico:** Gestisce la logica delle recensioni e storing dei prodotti.

– **Obiettivi:**

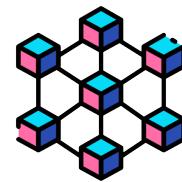
- * Conservare identificativi DID e DID document per verificare identità in modo pseudoanonimo degli utenti registrati;

- * Conservare le recensioni dei prodotti acquistati degli utenti registrati all'e-commerce con i relativi **Content Id (CID)** generati da IPFS.

– **Strumenti:**

- * user DID;

- * DID Document;



- **IPFS:**

– **Descrizione:** File system pubblico e trasparente che contiene l'insieme delle recensioni inserite dagli utenti registrati all'e-commerce per le quali viene generato un Content Id ossia un identificatore univoco a cui poter fare riferimento per ottenere informazioni sulle recensioni.

– **Obiettivi:**

- * Conservare le recensioni dei prodotti acquistati dagli utenti registrati ad e-commerce.

– **Strumenti:**

- * Recensioni;

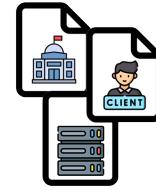


1.1.4 Strumenti del sistema

Al fine di chiarire meglio lo scenario mettiamo in evidenza gli strumenti fondamentali per la realizzazione degli obiettivi assegnati:

- DID:

- **Rilasciato da:** attore (issuer, holder, verifier);
- **Consegnato a:** Client registrato, Issuer, Verifier;
- **Descrizione:** DID è un tipo di identificatore digitale autonomo, verificabile e decentralizzato: consente a persone, organizzazioni o dispositivi di avere un'identità digitale senza dipendere da un'autorità centrale (come Google, Facebook o lo Stato).



- **DID Document:** Un DID Document è una struttura dati che descrive un'identità digitale decentralizzata, specificando come verificare la proprietà e interagire con essa. È definito dallo standard W3C "Decentralized Identifiers (DIDs) v1.1. Nello specifico i campi del DID Document sono:

```
did:example:123456789abcdefghi
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ],
  "id": "did:example:123456789abcdefghi",
  "authentication": [
    {
      "id": "did:example:123456789abcdefghi#keys-1",
      "type": "Ed25519VerificationKey2020",
      "controller": "did:example:123456789abcdefghi",
      "publicKeyMultibase": "QpubLcxyg9U11m3gE--39zX20V1n0g6PMm3uVA5ZP... "
    }
  ]
}
```

- **context:** È la sezione JSON-LD che indica quali vocabolari semantici si stanno usando:
 - * **https://www.w3.org/ns/did/v1:** termini standard del DID Core (obbligatorio);
 - * **https://w3id.org/security/suites/ed25519-2020/v1:** definisce la sintassi e gli algoritmi della suite crittografica Ed25519 Verification Key 2020.
- **id:** Il DID vero e proprio del soggetto del documento;
- **authentication:** Array dei verification methods che il controller può usare per dimostrare di essere il proprietario del DID:
 - * id: etichetta univoca della chiave all'interno del documento;
 - * type: specifica l'algoritmo e il formato;
 - * controller: uno o più DIDs che controllano il documento;
 - * publicKeyMultibase: la chiave pubblica vera e propria.

- **Verifiable Credential (VC) :**

- **Rilasciato da:** Issuer;
- **Consegnato a:** Holder;
- **Descrizione:** Una Verifiable Credential è una credenziale digitale che consente di attestare in modo sicuro e verificabile un'informazione su una persona, un'organizzazione o un oggetto, grazie a firme crittografiche che ne garantiscono l'autenticità e l'integrità. Permette al titolare di dimostrare determinati claim (come un titolo di studio o un'autorizzazione) in modo decentralizzato, rispettando la privacy e senza la necessità di contattare direttamente l'ente emittente.



- **Verifiable Presentation (VP):**

- **Rilasciato da:** Holder;
- **Consegnato a:** Verifier;
- **Descrizione:** Una Verifiable Presentation è una rappresentazione selettiva e verificabile di una o più Verifiable Credentials, creata dal titolare per condividerne solo le informazioni necessarie con un verificatore. Serve a dimostrare il possesso di credenziali in modo sicuro e controllato, permettendo al soggetto di proteggere la propria privacy e di scegliere quali dati rivelare e a chi;



1.2 Threat model del sistema

La definizione del threat model è cruciale per progettare un sistema incentrato sulla security. Il threat model aiuta a identificare e analizzare le potenziali minacce che il sistema potrebbe affrontare, consentendo di implementare misure di sicurezza adeguate.

1.2.1 Attori disonesti del sistema

Nell'ottica del threat model è possibile suddividere gli attori in sotto-categorie. Questa distinzione è utile per comprendere con quali pattern un avversario potrebbe tentare di fare leva e per mettere in evidenza la realtà analizzata.

Nota!

Si assume che:

- l'issuer sia SPID ossia un'entità affidabile e non corrotta;
- le comunicazioni tra holder e issuer avvengano in maniera sicura senza la presenza di un avversario che osserva questo canale di comunicazione per ottenere informazioni sensibili dell'holder e che l'issuer non le utilizzi per secondi fini (vendita di dati a terzi);
- nessun utente esterno allo schema possa impersonare l'issuer; l'holder è quindi sempre sicuro di star contattando l'issuer fidato.

Non si considereranno invece:

- **Client C:**
 - **Client ignaro:** si comporta come il client onesto ma contattando inconsapevolmente issuer e/o server disonesti;
 - **Client hackerati:** client che si comportano in maniera onesta ma che usufruiscono di un dispositivo malevolo (*che ad esempio ruba i dati di accesso per conto di un avversario*).

- **Avversari:** gli avversari sono utenti non contemplati tra gli attori previsti nello svolgimento delle normali funzionalità da sviluppare. Essi sono per antonomasia disonesti ed ognuno con un proprio obiettivo, tipologia e risorse:

– **Tipologia:**

- * **Passivi:** Gli avversari passivi si limitano a osservare le comunicazioni senza alterarle o interferire attivamente. Il loro obiettivo principale è raccogliere informazioni.
- * **Attivi:** Gli avversari attivi sono entità malevole che non si limitano a osservare passivamente il sistema, ma intervengono direttamente per alterare, manipolare o interrompere le comunicazioni all'interno del sistema stesso.

– **Scopi:**

- * **Rubare VC e VP:** Gli avversari possono essere interessati a sfruttare le funzionalità implementate per rubare le VC e VP di un holder;
- * **Ottenimento credenziali dall'issuer:** Gli avversari potrebbero tentare di ottenere delle credenziali che non gli spettano dall'issuer;
- * **Accesso al verifier:** Gli avversari potrebbero essere interessati ad accedere a un servizio al quale non avrebbero il diritto di accesso (*rientrano in questa categoria gli avversari che tentano di falsificare delle credenziali*);
- * **Far perdere credibilità alle recensioni:** Gli avversari potrebbero essere utenti contro il sistema di recensione che intendono volontariamente rendere il servizio inaffidabile con recensioni non attendibili, anche a scopo di lucro;
- * **Aggirare policy interne dell'e-commerce:** Gli avversari potrebbero essere interessati a determinati servizi e quindi potrebbero aggirare le policy interne richieste (età maggiore di una certa soglia);
- * **Ottenere maggiori informazioni:** Gli avversari potrebbero essere interessati a ottenere maggiori informazioni di altri utenti onesti che sono registrati al servizio di e-commerce.

– **Risorse:**

* **Informazioni:**

- **Canale di comunicazione tra Issuer e Holder:** È facile che un avversario abbia accesso alle comunicazioni dell'issuer;
- **Canale di comunicazione tra Holder e Verifier:** È facile che un avversario abbia accesso alle comunicazioni dell'holder;
- **VC e VP di vari utenti:** Un avversario potrebbe aver ottenuto le VC e le VP di utenti registrati al sistema. Non è da escludere che un avversario possa aver ottenuto una gran quantità di Credenziali di accesso, magari tramite la collaborazione con altri avversari.
- * **Ruoli all'interno dell'issuer o del verifier:** L'avversario potrebbe anche essere un impiegato che interagisce dietro le quinte del sistema come un impiegato dell'issuer o di un verifier. Per semplicità issuer o verifier che si comportano in maniera impropria rientrano in questa categoria.

– Organizzazione:

- * **Utente singolo:** L'avversario più comune è il singolo utente che ha a propria disposizione uno o più calcolatori e che si interfaccia all'issuer e al verifier con una singola connessione;
- * **Enclave di utenti:** Un gruppo di avversari può organizzarsi per sferrare un attacco combinato. Si noti che per gli avversari è complicato organizzarsi in un vasto gruppo.

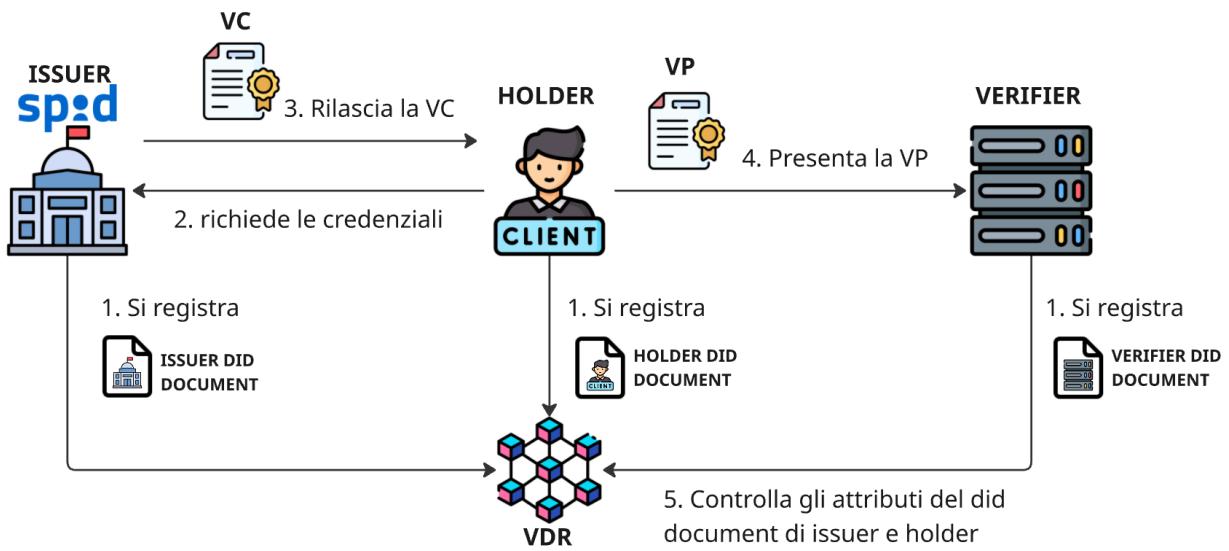


Figura 1.1: Modello completo del sistema

1.2.2 Descrizione avversari

Per una migliore comprensione del sistema, ogni possibile avversario interessato ad attaccare le funzionalità descritte, è identificato da un nome.

Si assume inoltre che i dispositivi di chi è onesto non siano corrotti: l'hardware non è compromesso e il software non è controllato da malware/virus/trojans.

È evitato, quando irrilevante per l'analisi, la definizione di avversari strettamente meno forti di altri (che hanno solo meno risorse).

Non sono presi in considerazione avversari impossibili date le ipotesi di progetto (holder che cercano di ottenere la chiave privata di altri holder per forgiare la loro firma utile per presentare la VP al verifier).

Beetle:

Figura 1.2: Beetle

- **Tipologia:** attivo;
- **Scopo:** far perdere credibilità alle recensioni;
- **Risorse:** canale di comunicazione tra Holder e Verifier;
 - Ospite: non possiede alcuna VC e VP;
 - Holder registrato: VC e VP;
- **Organizzazione:** utente singolo;
- **Descrizione:** Beetle è un avversario che può essere sia ospite che utente registrato al servizio di e-commerce il cui obiettivo è quello di inserire una recensione fraudolenta di un prodotto che non ha mai acquistato.

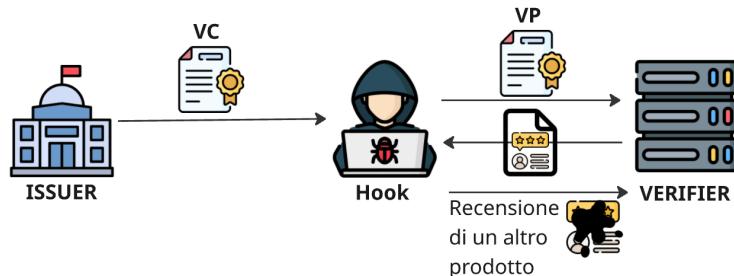
Hook:

Figura 1.3: Hook

- **Tipologia:** Attivo;
- **Scopo:** far perdere credibilità alle recensioni;
- **Risorse:** canale di comunicazione tra Holder e Verifier, prova di acquisto di un prodotto;
- **Organizzazione:** utente singolo;
- **Descrizione:** Hook è un avversario che consiste in un Holder che acquista un prodotto sull'e-commerce e riceve la relativa prova di acquisto, ma cerca di modificarla per recensire un altro prodotto che non ha acquistato.

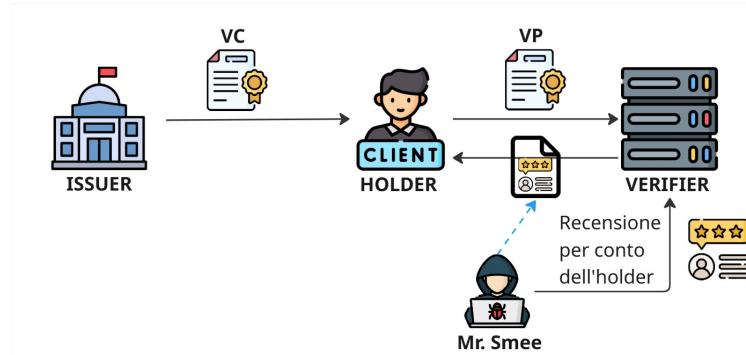
Mr.Smee:

Figura 1.4: Mr.Smee

- **Tipologia:** passivo e attivo;
- **Scopo:** far perdere credibilità alle recensioni;
- **Risorse:** canale di comunicazione tra Holder e Verifier, prova di acquisto di un prodotto;
- **Organizzazione:** utente singolo;
- **Descrizione:** Mr.Smee è un avversario inizialmente passivo in quanto ascolta sul canale di comunicazione tra verifier e holder per ottenere la prova di acquisto di un determinato prodotto effettivamente acquistato da un holder onesto. Una volta ottenuta la prova di acquisto, Mr.Smee diventa attivo, in quanto la utilizza per inserire una recensione al prodotto impersonando l'holder onesto.

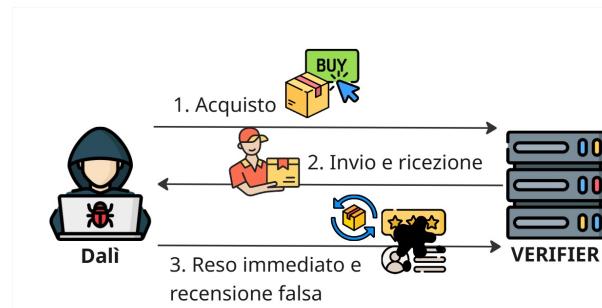
Dalì:

Figura 1.5: Dalì

- **Tipologia:** attivo;
- **Scopo:** far perdere credibilità alle recensioni;
- **Risorse:** canale di comunicazione tra Holder e Verifier, VC e VP;
- **Organizzazione:** utente singolo;

- **Descrizione:** Dalì è un utente registrato in possesso di VC e VP valide che acquista il prodotto dall'e-commerce dal quale riceve la conferma d'acquisto e lascia una recensione non veritiera del prodotto quando ne fa il reso e questo può avvenire o subito dopo aver ricevuto la proof of purchase o quando il prodotto gli arriva a casa.

Cosmo e Wanda:

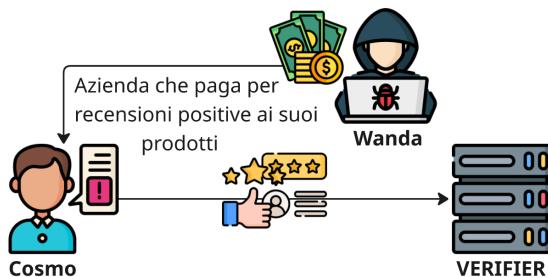


Figura 1.6: Cosmo e Wanda

- **Tipologia:** attivo;
- **Scopo:** far perdere credibilità alle recensioni;
- **Risorse:** canale di comunicazione tra Holder e Verifier, VC e VP di holders;
- **Organizzazione:** enclave di utenti;
- **Descrizione:** Cosmo e Wanda è un gruppo di avversari che consiste in un'azienda professionale (Wanda) che paga un holder (Cosmo) per acquistare il suo prodotto e inserire una recensione positiva per attirare altri utenti ad acquistare il prodotto stesso.

Beagle Boys:

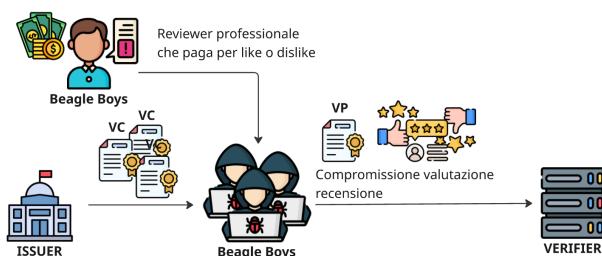


Figura 1.7: Beagle Boys

- **Tipologia:** attivo;
- **Scopo:** far perdere credibilità alle recensioni;
- **Risorse:** canale di comunicazione tra Holder e Verifier, VCs e VPs di holders;
- **Organizzazione:** enclave di utenti;
- **Descrizione:** Beagle boys è un gruppo di avversari che consiste in un insieme di holder ingaggiati da un reviewer professionale che valuta in maniera disonesta determinate recensioni per aumentare la reputazione di alcune e sminuire la reputazione di altre.

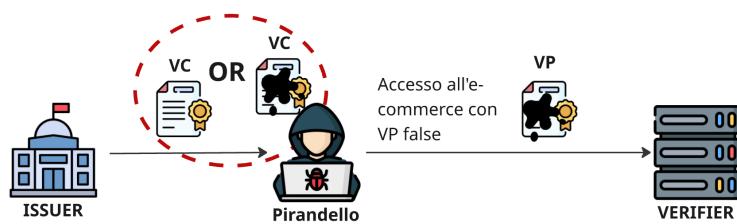
Pirandello:

Figura 1.8: Pirandello

- **Tipologia:** attivo;
- **Scopo:** accesso al Verifier;
- **Risorse:** canale di comunicazione tra Issuer e Holder, Holder e Verifier;
- **Organizzazione:** utente singolo;
- **Descrizione:** Pirandello è un avversario che cerca di creare delle Verifiable Presentation (VP) con claim falsi, o a partire da una VC valida ricevuta da un issuer fidato o auto-forgiandone una per poter accedere al servizio di e-commerce (Verifier) pur non essendo possessore di credenziali certificate.

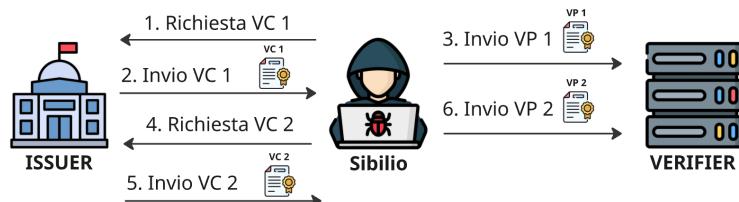
Sibilio:

Figura 1.9: Sibilio

- **Tipologia:** attivo;
- **Scopo:** accesso al verifier;
- **Risorse:** canale di comunicazione con issuer e verifier, VC e VP;
- **Organizzazione:** utente singolo;
- **Descrizione:** Sibilio vuole accedere al servizio di e-commerce tramite la creazione di più VC che contengono le medesime credenziali con l'obiettivo di creare più profili falsi.

Tom:

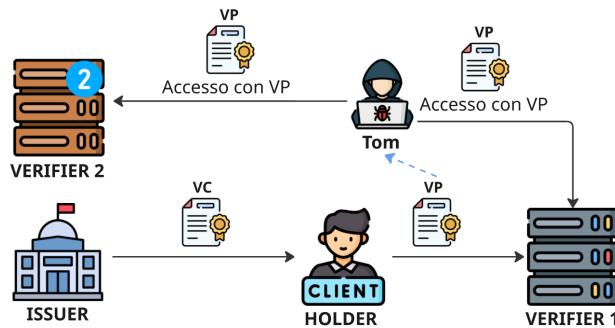


Figura 1.10: Tom

- **Tipologia:** passivo e attivo;
- **Scopo:** accesso al verifier;
- **Risorse:** canale di comunicazione tra Holder e Verifier, VP;
- **Organizzazione:** utente singolo;
- **Descrizione:** Tom è un utente che ruba la VP inviata dall'holder al verifier o perché ascolta sul canale di comunicazione tra Holder e Verifier o perché ne è venuto a conoscenza tramite un data breach. Il suo obiettivo è quindi quello di usare la VP appena rubata per poter accedere a diversi verifier impersonando l'holder.

Jerry:

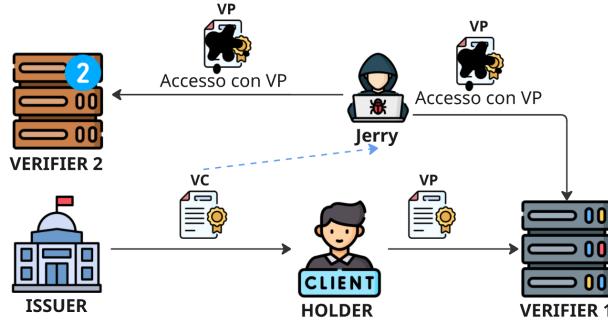


Figura 1.11: Jerry

- **Tipologia:** passivo e attivo;
- **Scopo:** accesso al verifier;
- **Risorse:** canale di comunicazione tra Issuer e Holder, VC;
- **Organizzazione:** utente singolo;
- **Descrizione:** Jerry è un utente che ruba la VC inviata dall'issuer all'holder o perché ascolta sul canale di comunicazione tra Issuer e Holder o perché ne è venuto a conoscenza tramite un data breach. Il suo obiettivo è quindi quello di forgiare una VP a partire dalla VC appena rubata per poter accedere a diversi verifier impersonando l'holder.

Pooh:

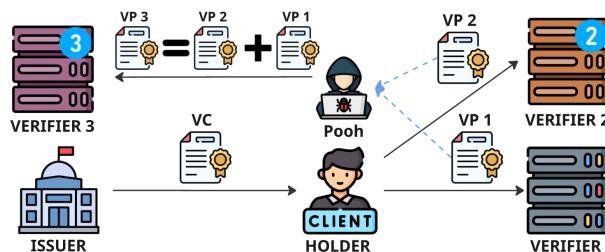


Figura 1.12: Pooh

- **Tipologia:** passivo e attivo;
- **Scopo:** accesso al verifier;
- **Risorse:** canale di comunicazione tra Holder e Verifier, VP;
- **Organizzazione:** utente singolo;
- **Descrizione:** Pooh è in ascolto sui canali di comunicazione tra holder e verifier 1 e 2. Il suo scopo è quello di rubare le VP dirette a questi ultimi per poter mettere assieme i claim e accedere al verifier 3.

Giuda & the Apostles:

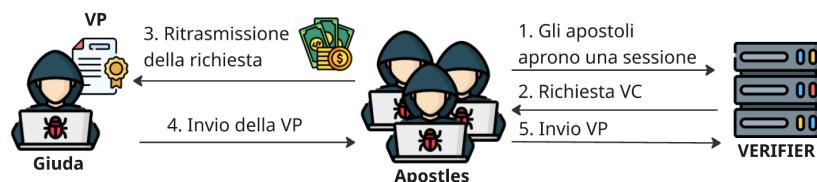


Figura 1.13: Giuda e gli apostoli

- **Tipologia:** attivo;
- **Scopo:** far perdere credibilità alle recensioni;
- **Risorse:** canale di comunicazione tra Holder e Verifier, VP di Giuda.
- **Organizzazione:** enclave di utenti;
- **Descrizione:** Giuda e gli apostoli sono un enclave di utenti in cui Giuda, ossia un holder in possesso di una VC e VP vende la sua VP per far accedere all'e-commerce gli apostoli per conto suo.

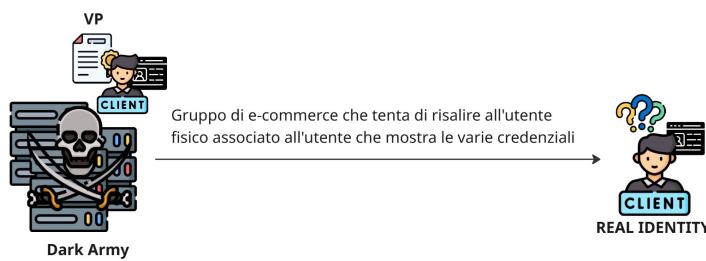
Dark Army:

Figura 1.14: Dark Army

- **Tipologia:** passivo;
- **Scopo:** ottenere maggiori informazioni;
- **Risorse:** canale di comunicazione tra Holder e Verifier, VPs di Holders.
- **Organizzazione:** enclave di utenti;
- **Descrizione:** Dark Army è un gruppo di avversari che consiste in un insieme di insider di verifier che, date le VP fornite da holder per accedere ai diversi siti, cercano di risalire alle identità degli specifici holder in modo da tracciare le loro attività o ottenere informazioni aggiuntive.

1.3 Proprietà di resilienza

In questo sottoparagrafo, verranno esaminate le principali proprietà che l'Issuer e il Verifier devono possedere per garantire un ambiente sicuro, affidabile e performante.

L'analisi verrà effettuata sulla base delle seguenti proprietà:

- **Confidenzialità:**
 - **C1:** Le informazioni che l'holder mostra per ottenere delle credenziali, sono leggibili solo da egli stesso e dall'issuer onesto;
 - **C2:** Le credenziali di accesso valide (VC) non devono poter essere carpite da attori disonesti;
 - **C3:** Il servizio fornito da parte di un Verifier a un Holder deve poter essere visibile e accessibile al solo Holder;
 - **C4:** Il servizio di un Verifier non deve essere accessibile a un Holder sprovvisto di credenziali valide rilasciate da un'issuer affidabile;
 - **C5:** La possibilità di recensire è esclusiva per gli holder in possesso della verifica di acquisto;
- **Minimization:** le informazioni presenti all'interno delle credenziali devono essere quelle strettamente necessarie per l'accesso all'e-commerce;
- **Predicate disclosure:** le Verifiable Credential (VC) e le relative Verifiable Presentation (VP) devono consentire all'holder di provare che un attributo soddisfa un determinato predicato (es. “età maggiore di 18 anni”) senza rivelarne il valore esatto, preservando quindi la privacy del dato sottostante;
- **Unlinkability:** il verifier, date due VP non deve essere in grado di determinare se provengono dalla stessa VC;
- **Unobservability:** l'issuer non deve essere in grado di riconoscere quando, dove e a quale verifier viene presentata la credenziale fornita all'holder;
- **Untraceability:** gli issuer e i verifier collusi non possono rintracciare le credenziali di un utente attraverso più interazioni effettuate nel tempo;
- **Non-transferability:** quando un soggetto riceve una credenziale verificabile (ad esempio, un certificato di laurea in formato digitale), essa è strettamente associata alla sua identità digitale, rappresentata dal relativo DID. Tale credenziale può essere utilizzata esclusivamente dal titolare per generare VP e dimostrare il possesso di determinati attributi o qualifiche. Non è possibile trasferire, delegare o condividere la credenziale con terzi, analogamente a quanto accade con un documento d'identità personale: il suo utilizzo è riservato unicamente al soggetto cui è stata rilasciata;
- **Unforgeability:** un holder non deve essere in grado di forgiare una VC legittima senza passare da un issuer fidato;

- **Reputation:**

- **R1:** un utente non deve essere in grado di creare più di un account per poter partecipare al sistema di recensioni;
- **R2:** il sistema di valutazione delle recensioni deve essere considerato affidabile;

- **Integrità:**

- **I1:** garantire che le informazioni che vengono inviate dal client all'issuer per la richiesta dell'ottenimento di una Verifiable Credential non vengano modificate da attori disonesti;
- **I2:** garantire che le credenziali (VC) che vengono inviate dall'issuer/verifier all'holder non vengano modificate da attori disonesti;
- **I3:** garantire che i claim presenti nella VP che viene inviata dall'holder al verifier siano coerenti con quelle presenti nella VC;
- **I4:** garantire che la verifica di acquisto di un prodotto non possa essere modificata dall'holder per poter recensire un altro prodotto;

- **Trasparenza:**

- **T1:** i protocolli utilizzati per una comunicazione sicura issuer-holder devono essere noti a tutti gli attori;
- **T2:** è possibile risalire all'issuer che ha generato le credenziali in modo da poter scoraggiare abusi;
- **T3:** gli algoritmi e i protocolli utilizzati dall'issuer devono essere noti agli utenti e ai verifier al fine di poter verificare la validità delle decisioni prese;
- **T4:** è possibile, anche tramite tecniche avanzate, verificare il corretto rilascio delle credenziali;
- **T5:** i protocolli utilizzati per una comunicazione sicura holder-verifier devono essere noti a tutti gli attori;
- **T6:** gli algoritmi e i protocolli utilizzati dal verifier per convalidare le credenziali generate dall'issuer, devono essere noti agli holder in modo tale da poter pienamente comprendere le scelte fatte dal verifier in caso di rifiuto di accesso;
- **T7:** il meccanismo con cui si conservano, modificano o rimuovono le recensioni deve essere pubblicamente noto, così come le recensioni stesse;
- **T8:** il meccanismo di incentivazione/disincentivazione per l'inserimento e la valutazione delle recensioni deve essere pubblicamente noto;

- **Efficienza:**

- **E1:** garantire che la generazione delle credenziali sia eseguita in modo rapido ed efficiente;
- **E2:** ridurre al minimo le comunicazioni issuer-holder;
- **E3:** garantire che l'identificazione presso un verifier tramite le credenziali ottenute in precedenza avvenga in modo semplice e rapido;
- **E4:** garantire che il numero di credenziali gestite dagli holder siano lo stretto necessario.

Il sistema da proporre si pone l'obiettivo di raggiungere un buon compromesso tra le proprietà di resilienza prima elencate, cercando quando possibile, di rispettarle tutte ma dando sempre priorità alla decentralizzazione e alla privacy dei client coinvolti.

2.1 Panoramica

Per costruire un ambiente decentralizzato basato su recensioni il più affidabile e trasparente possibile, in questo capitolo approfondiremo il modello presentato nel WP1 estendendolo alle sue funzionalità e alla soluzione proposta

Prima di entrare completamente nella soluzione, si supponga che l'utente, in una fase preliminare abbia ottenuto una propria carta d'identità per poter essere riconosciuto come egli dichiara di essere e tramite essa aver creato un account SPID per essere digitalmente riconosciuto. Nella carta d'identità sono presenti diverse informazioni come il suo nome, cognome, la sua data e il luogo di nascita, il suo sesso ecc... le stesse informazioni sono presenti anche nel suo profilo SPID.

Il suo obiettivo è quindi quello di poter accedere al servizio di e-commerce, che si comporta da verifier, per poter acquistare dei prodotti e recensirli oppure valutare recensioni fatte da altri utenti.

Per i possessori di credenziali verificabili (VC), che vengono rilasciate da SPID (issuer), il verifier permette l'accesso a tutte le funzionalità disponibili, ossia acquisto e recensione di un prodotto e valutazione delle recensioni, mentre per gli altri utenti (gli ospiti) è consentito solo l'acquisto di determinati prodotti che non hanno vincoli di policy interne (es. coltellini che possono essere venduti esclusivamente agli utenti maggiorenni). Per questo motivo, si assume che il Verifier permetta l'accesso ai soli utenti in possesso di credenziali firmate da SPID, ossia ritiene come veritiero le credenziali rilasciate da quest'ultimo.

Per gli utenti registrati le funzionalità aggiuntive del servizio di e-commerce, permettono anche di ricevere dei reward in modo tale da incentivare la registrazione e la partecipazione attiva al meccanismo di recensione.

2.2 Tecnologie adottate

2.2.1 W3C:

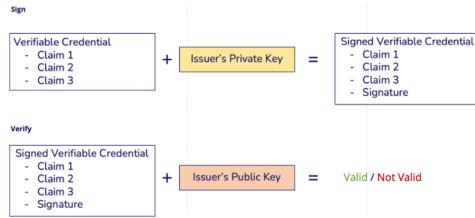
Il protocollo **World Wide Web Consortium (W3C) Verifiable Credentials & Verifiable Presentations** è un insieme di raccomandazioni W3C che definisce come rappresentare, firmare e verificare attestazioni digitali in modo interoperabile, decentralizzato e privacy-oriented. In particolare:

- **Interoperabilità:** tutti parlano lo stesso "dialetto" JSON-LD/JWT ossia qualunque wallet può usare la VC emessa da qualunque issuer;
- **Decentralizzazione:** vengono utilizzati i DID anziché Public Key Infrastructures (PKI) centralizzate;
- **Verificabilità online:** chiunque, anche senza contattare l'issuer può controllare firme e revoca della VC/VP;
- **Controllo dell'utente:** l'utente (holder) decide quando e quali claim rivelare al verifier. A questo scopo W3C supporta il *selective disclosure* o le *Zero Knowledge Proofs (ZKP)*.

2.2.2 Verifiable Credential

Payload Decodificato (Contenuto della VC):

```
{
  "vc": {
    "@context": [
      "https://www.w3.org/2018/credentials/v1",
      "http://localhost:3000/access-vc-context.jsonld"
    ],
    "type": [
      "VerifiableCredential",
      "AccessCredentials"
    ],
    "credentialSubject": {
      "id": "did:ethr:0x539:0x28eF1b572275207B77945fa6C90801527Bb0153C",
      "nation": "NED",
      "over18": true,
      "isStudent": true,
      "weaponPermit": true
    }
  },
  "sub": "did:ethr:0x539:0x28eF1b572275207B77945fa6C90801527Bb0153C",
  "nbf": 1749894237,
  "iss": "did:ethr:0x539:0x418411214bcfaef26741EE014177Ee222802aA1a6E"
}
```



Una *Verifiable Credential* (VC) è una struttura che contiene un insieme di attributi (*claims*) su una determinata identità.

È verificabile in maniera crittografica e decentralizzata e viene memorizzata nel *wallet* dell'*holder*, tipicamente sul suo dispositivo sotto controllo diretto.

Una VC è composta da:

- URI del *subject* (l'entità a cui si riferisce);
- URI dell'*issuer*;
- firma digitale dell'*issuer* (chiave privata);
- condizioni di scadenza.

La VC viene rilasciata dall'*issuer* all'*holder*, che ne verifica l'autenticità tramite la chiave pubblica dell'*issuer*, controllando che non sia stata manomessa durante il trasferimento.

2.2.3 Verifiable Presentation

Dopo che l'issuer ha firmato la Verifiable Credential (VC), l'holder la conserva nel proprio wallet. A questo punto egli decide di volersi registrare al sito di e-commerce che ha anche il ruolo di Verifier tramite l'utilizzo di uno smart contract apposito che permette l'accesso ad esso solo agli utenti in possesso delle giuste credenziali firmate da un'autorità certificata.

Quello che il verifier, nella fase di **Presentation Request** fa, è inviare al wallet dell'holder una **Presentation Definition** definendo gli attributi (claim) (es. e-mail, età maggiore di 18 anni) necessari per l'accesso all'e-commerce e un campo **nonce/challenge** casuale utile per prevenire Replay Attacks in quanto la VP può essere usata una sola volta in quel contesto/dominio.

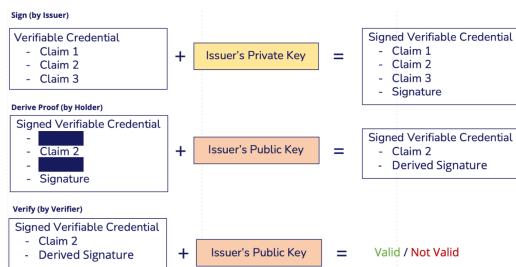
In risposta l'holder fornisce una **Verifiable Presentation** inviando allo smart contract solo i campi minimi tramite **BBS+** che a sua volta firma con la propria chiave privata in modo tale che il verifier possa verificarne l'autenticità.

Una VP non è nient'altro che un documento JSON-LD firmato dall'holder che "incarta" per intero una o più VC nel campo "verifiableCredential" e le rende verificabili da chi le riceve grazie alla proof che tra le tante cose contiene la firma digitale applicata dall'holder.

2.2.4 Selective Disclosure tramite BBS+

Non è detto che però la voce "verifiableCredential" della VP debba per forza includere l'intera VC originale.

Per questo motivo verrà preso in considerazione l'utilizzo del **Selective Disclosure** tramite **Boneh-Boyen-Shacham (BBS)**, diventata successivamente **BBS+** grazie alla variante perfezionata da Au, Susilo e Mu.



Di fianco si può osservare lo schema di funzionamento del BBS+.

In particolare l'holder, a partire dalla VC fornita dall'issuer e firmata con la sua chiave privata, è in grado di generare una **Derived Proof** in cui nasconde alcuni claim e usa la chiave pubblica dell'issuer come parametro dell'algoritmo di derivazione per "randomizzare" la firma originale e produrre una **derived signature** che coinvolge *solo* i claim scelti.

Questo processo non necessita di alcuna chiave segreta in quanto la sicurezza viene dalla casualità ossia dal **nonce**, un numero casuale crittografato privato, generato dal wallet dell'holder e inserito nel calcolo della firma derivata; questo garantisce che due prove derivate sugli stessi claim risultino sempre diverse (poiché il nonce sarà diverso), risolvendo così l'**unlinkability**. La VP viene quindi poi inviata al verifier il quale verifica contemporaneamente due aspetti:

- **L'autenticità e integrità dei dati rivelati (VC/derived proof)** tramite la chiave pubblica dell'issuer contenuta all'interno del suo DID document che il verifier recupererà, accedendo all'identificativo nel campo "issuer" della VP e interrogando il **Verifiable Data Registry VDR** (che verrà descritto approfonditamente nella prossima sezione). La verifica andrà a buon fine solo se la chiave pubblica è presente nella sua trust-list.
- **La proprietà della presentazione (VP)** tramite la chiave pubblica dell'holder contenuta all'interno del suo DID document che il verifier recupererà, accedendo all'identificativo nel campo "credentialSubject.id" della VP e interrogando il VDR.

Se la verifica va a buon fine il verifier è certo che i claim rivelati sono stati davvero firmati dall'issuer originario e che l'holder che glieli sta mostrando ne è davvero in possesso.

2.2.5 Utilizzo della blockchain

La soluzione proposta, sfrutta la blockchain che svolge il ruolo di **Verifiable Data Registry (VDR)**.

In questo registro distribuito vengono memorizzati:

- i DID e le relative chiavi pubbliche di issuer e verifier;
- una “trust list” di issuer attendibili;
- gli eventuali stati di revoca di credenziali;
- tutta la logica applicativa ossia la registrazione delle recensioni e gestione di like o dislike.

Tutte queste funzionalità verranno implementate tramite smart contract appositi localizzati on-chain.

IPFS

I contenuti voluminosi, cioè le recensioni testuali ed eventuali allegati, non vengono scritti on-chain ma vengono caricati su **IPFS**, che restituisce un **Content ID (CID)**.

Il contratto salva o emette in evento soltanto questo hash, così la blockchain conserva la prova di integrità e l'ordine temporale, mentre lo storage vero rimane off-chain, riducendo drasticamente il costo in gas e mantenendo i dati reperibili.

2.3 Fase di registrazione

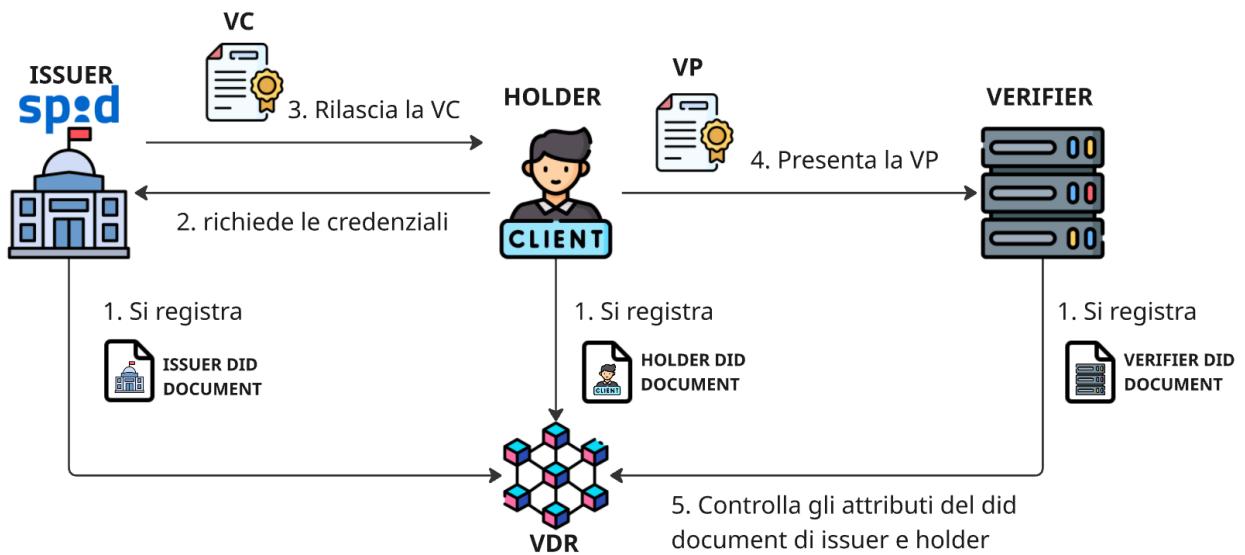


Figura 2.1: Fase di registrazione

1. Inizialmente tutti gli attori creano un proprio DID e annesso un DID document per essere riconosciuti su blockchain;
2. Il client (futuro holder), per poter usufruire delle funzionalità del servizio di e-commerce, deve registrarsi presso di esso. Per fare ciò, necessita di una credenziale basata sulla sua identità, quindi contatta SPID inviando il suo DID;
3. L'issuer effettua una verifica che consiste in:
 - controllo delle credenziali richieste rispetto a quelle che l'utente possiede;
 - controllo che l'utente non abbia già richiesto in precedenza quelle specifiche credenziali usando un did diverso. Questo perché l'utente, per accedere a diversi servizi può richiedere più VC associate a diversi did da lui creati.
4. Se la verifica va a buon fine, l'issuer gli rilascia la Verifiable Credential (VC), secondo gli standard del protocollo W3C, firmata con la sua chiave privata attestando in modo sicuro e verificabile una o più informazioni sull'utente. NB. Nella versione attuale SPID rilascia un token SAML 2.0, quindi si dovrebbe passare da un altro issuer per creare la VC vera e propria, però dato che i claim sono gli stessi, si assume in questa soluzione che SPID abbia la funzionalità di inviare la VC per il protocollo W3C;
5. Ora il client assume il ruolo di holder e, per registrarsi al sito di e-commerce, presenta una Verifiable Presentation (VP) firmata da lui stesso. Questa VP contiene un sottoinsieme delle informazioni presenti nella Verifiable Credential (VC), selezionato per soddisfare i requisiti del verifier. Per la generazione della VP viene utilizzato il sistema di firme BBS+, che consente la selezione di attributi specifici mantenendo la validità crittografica. In particolare, nella VP il client include una firma derivata a partire dalla firma originale dell'issuer, applicando la chiave pubblica dell'issuer alla firma contenuta nella VC;
6. Infine il verifier effettua un controllo sull'autenticità della VP e se questo ha esito positivo allora permette l'accesso al servizio di e-commerce.

2.4 Fase di acquisto

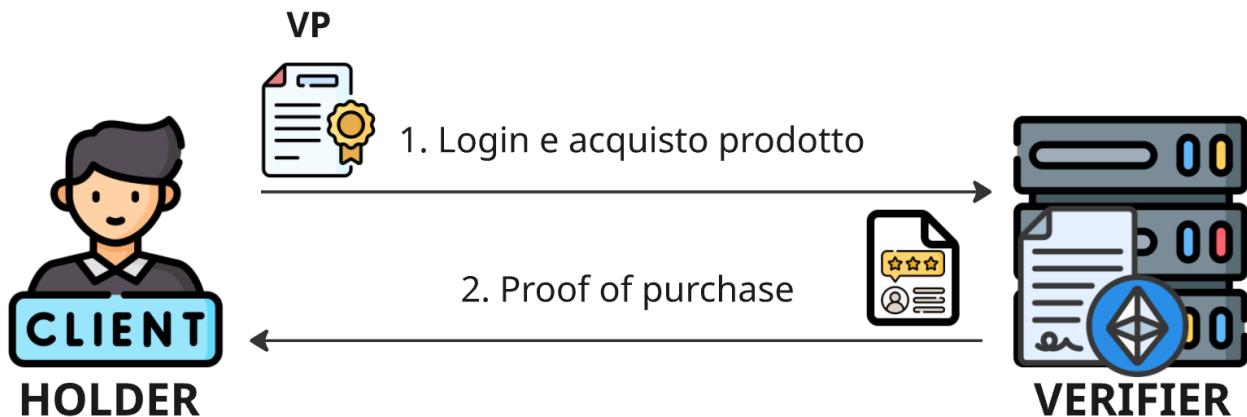


Figura 2.2: Fase di acquisto

- **Holder registrato:**

1. L'Holder effettua login all'e-commerce presentando la VP; successivamente decide di acquistare un prodotto e, se in possesso, anche di una VC (la proof of purchase rilasciata dal verifier) che contiene tutti i claim degli acquisti fatti precedentemente in modo tale che il verifier possa inserire il claim del nuovo prodotto acquistato;
2. Il verifier verifica con le informazioni contenute nella VP, se l'utente può effettuare l'acquisto in base alle policy interne definite (es. l'acquisto di un pacco di sigarette necessita un'età maggiore di 16 anni);
3. Se la verifica va a buon fine, concede l'acquisto e fornisce la proof of purchase del prodotto all'utente registrato, quando il prodotto gli arriva, che consiste di:
 - una nuova VC generata e firmata dal verifier che contiene come claim il prodotto acquistato dall'holder, se quest'ultimo non ha inviato in fase di login la VC degli acquisti fatti in precedenza;
 - una VC modificata, generata e firmata dal verifier che contiene il nuovo claim che attesta l'acquisto del prodotto se l'holder ha inviato in fase di login la VC degli acquisti fatti in precedenza
4. A questo punto l'utente, in possesso di questa prova potrà poi recensire il prodotto (fase descritta nella sezione successiva).

- **Ospite:** non essendo ancora registrato al sito e non avendo fornito alcuna informazione identificativa, può acquistare esclusivamente prodotti non soggetti a restrizioni previste dalle policy interne. Non gli viene fornita alcuna proof aggiuntiva, poiché la registrazione è necessaria per accedere alle funzionalità avanzate del sito. Di conseguenza, il client può solo acquistare un sottoinsieme limitato di prodotti e consultare le recensioni pubblicate dagli utenti registrati.

2.5 Fase di recensione

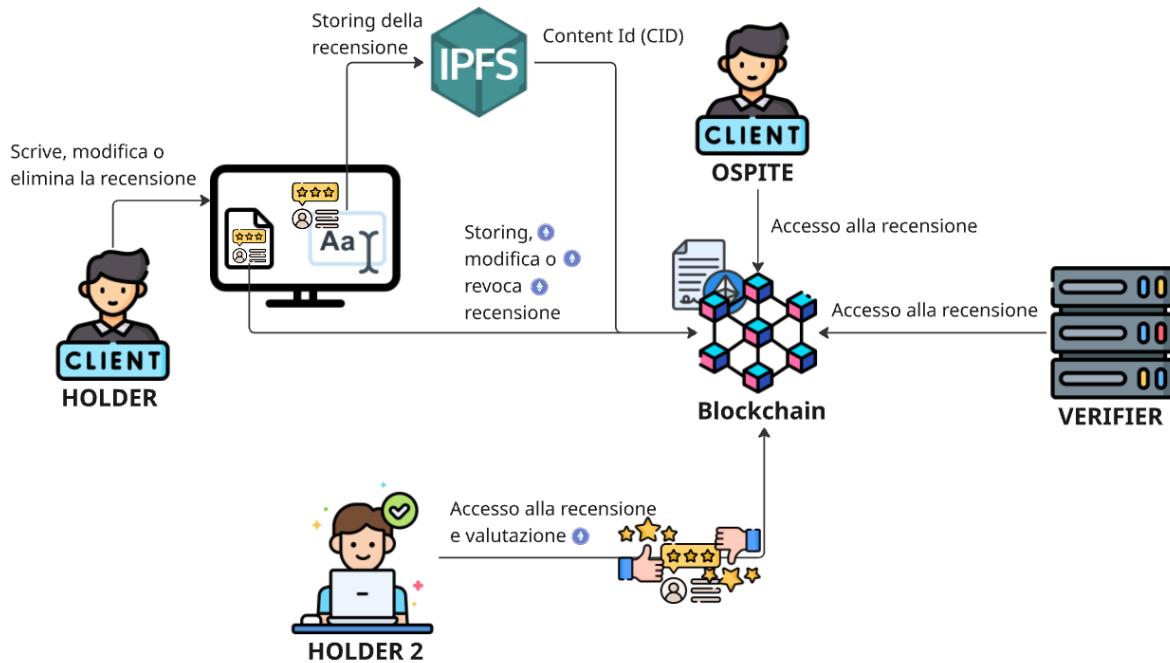


Figura 2.3: Fase di recensione

Descriviamo questa fase ad alto livello e nel successivo sottoparagrafo definiamo nello specifico i meccanismi di gestione delle recensioni, reward e penalizzazione proposti dalla soluzione.

- L'ospite ed e-commerce (verifier) possono contattare il contratto nella blockchain per visualizzare le recensioni ma non possono valutarle: il primo perché non è un utente registrato all'e-commerce e il secondo per costruzione del modello del sistema;
 - L'Holder 2 registrato può contattare il contratto per visualizzare le recensioni, in aggiunta può valutare le recensioni degli altri holder con un like o dislike;
 - L'Holder registrato con prova di acquisto può contattare il contratto per visualizzare le recensioni, valutare le recensioni degli altri, inserire una nuova recensione per il prodotto acquistato, modificarla o invalidarla se l'aveva già inserita precedentemente.
- N.B. Quando l'utente scriverà la recensione, questa verrà inserita su IPFS il quale rilascerà un CID che verrà caricato sulla blockchain.

2.5.1 Gestione delle recensioni, reward e penalizzazioni

Nella fase di gestione delle recensioni, il sistema stabilisce che la visualizzazione delle recensioni pubblicate sulla blockchain sia completamente libera e gratuita.

Tutti gli utenti, compresi gli ospiti non registrati, così come il sito web dell'e-commerce, possono accedere alle recensioni associate ai prodotti.

Questo garantisce un elevato livello di trasparenza e fiducia, in quanto le recensioni risultano consultabili da chiunque, senza restrizioni o costi aggiuntivi.

Gli holder registrati, cioè utenti che hanno completato il processo di registrazione al sito web, possono non solo visualizzare le recensioni, ma anche valutarle esprimendo un giudizio positivo o negativo.

Al fine di proteggere l'integrità del sistema, solo chi è registrato può esprimere voti.

Per poter valutare una recensione, l'holder deve autenticarsi attraverso la propria Verifiable Presentation che ha creato per poter accedere all'e-commerce.

La presenza del DID tra quelli autorizzati viene verificata direttamente dallo smart contract prelevandolo dalla VP e consentendo al sistema di assicurarsi che solo gli utenti legittimi possano influenzare la reputazione delle recensioni.

Le operazioni di votazione, come ogni modifica allo stato della blockchain, richiedono il pagamento di una piccola quota di gas sulla rete Ethereum.

Ciò comporta che ogni valutazione sia deliberata e responsabile, limitando gli abusi e i comportamenti opportunistici.

Il processo di creazione delle recensioni è regolato da requisiti ancora più stringenti: solamente gli holder registrati che hanno acquistato realmente un determinato prodotto possono pubblicare una recensione per quel prodotto.

A garanzia di questo requisito, l'utente, al momento dell'inserimento della recensione, deve presentare una Verifiable Presentation (VP) generata a partire da una Verifiable Credential (VC) che attesta l'effettivo acquisto (proof di acquisto rilasciata dal verifier). La VP permette di provare selettivamente l'acquisto specifico, senza esporre altri dati personali o la cronologia completa degli acquisti, rispettando così i principi di privacy propri delle tecnologie decentralizzate.

Questa scelta architetturale, basata sulla verifica dinamica tramite VP mira a garantire che solo gli utenti certificati dall'e-commerce possano inserire una recensione per un prodotto da loro acquistato.

Per incentivare comportamenti virtuosi, il sistema introduce un meccanismo di premi che si basa sulla reputazione dei diversi reviewer. Gli autori di recensioni che raggiungono il 51% di like dalla rete di reviewer, ricevono ricompense in Ethereum.

Inoltre, gli utenti che accumulano un'elevata reputazione sulle loro recensioni, quindi ogni 100 recensioni considerate di qualità, possono ricevere **Non-Fungible Token (NFT)** premio. Gli NFT, emessi tramite smart contract, conferiscono benefici reali sulla piattaforma e-commerce, come buoni sconto, accesso prioritario ai pre-ordini dei nuovi prodotti o biglietti per eventi esclusivi.

Una volta inserita, una recensione può essere modificata o revocata dal suo autore. In caso di revoca o modifica, il sistema provvede all'azzeramento di tutti i like e dislike precedentemente associati alla recensione. Se la recensione aveva accumulato una reputazione positiva significativa, l'autore subirà una decurtazione proporzionale della sua reputazione complessiva.

Quando l'utente fa il reso di un prodotto che aveva precedentemente recensito, il sistema inserisce la data di reso all'interno della recensione. Anche gli holder che si limitano a valutare le recensioni degli altri vengono incentivati: qualora il loro voto si riveli corretto (ad esempio un dislike su una recensione che viene poi riconosciuta come non affidabile dalla community), ricevono una ricompensa proporzionale in ethereum.

Il sistema premia pertanto non solo chi produce contenuti di qualità, ma anche chi contribuisce a mantenere alta la qualità delle recensioni attraverso il meccanismo di valutazione.

In contrapposizione ai premi, sono previsti anche meccanismi di penalizzazione. Se un autore riceve numerosi dislike sulle proprie recensioni, la sua reputazione scende, in particolare dopo **ogni 25 recensioni valutate negativamente**, se era in possesso di NFT gli viene revocato, altrimenti viene escluso dal diritto di recensire ulteriormente inserendolo in una blacklist.

Dal punto di vista tecnico, ogni modifica o revoca non comporta l'eliminazione fisica della recensione dalla blockchain, ma viene implementata attraverso un'apposita marcatura logica. In questo modo, la trasparenza e l'immutabilità della blockchain vengono rispettate, preservando una traccia storica verificabile di tutte le interazioni. L'intera architettura è disegnata per creare un ecosistema bilanciato, in cui le recensioni siano autentiche, le valutazioni siano attendibili, e ogni partecipante sia responsabilizzato, con strumenti di incentivo o penalizzazione calibrati per rafforzare progressivamente la qualità e la fiducia nella piattaforma. In questo modo, si punta ad avere una rete di reviewer onesti perché ricompensati, in grado di rilevare e segnalare comportamenti disonesti in quanto strettamente interessati nell'aumentare la fiducia nella piattaforma per guadagnare più ETH e NFT.

2.6 Credenziali di accesso e policy di acquisto

Come è stato detto in precedenza, per poter accedere al sito di e-commerce, è necessaria una Verifiable Credential (VC) contenente tutti i claim dell'holder. Egli presenterà alcuni o tutti i claim tramite la Verifiable Presentation (VP) all'e-commerce, il quale, tramite uno smart contract apposito che si comporta da verifier, verificherà se sono presenti i claim richiesti per l'accesso.

Proprio a questo scopo, in questa sezione saranno descritti i claim necessari che servono all'utente per potersi registrare e quelli opzionali che l'holder rivelerà se ha intenzione di acquistare un determinato prodotto regolamentato da determinate policy interne.

- **Credenziali minime per l'accesso:** utili per mostrare i prodotti, coerentemente alle caratteristiche degli utenti secondo analisi esterne:

- **Età:** utile per mostrare contenuti indirizzati al pubblico adulto o minorenne;
- **Residenza:** utile per filtrare i prodotti a seconda delle regolamentazioni nazionali;
- **Sesso:** utile per suggerire prodotti statisticamente inclini al determinato sesso;
- **DID;**

- Credenziali opzionali per abilitare acquisti regolamentati:** l'utente può, al momento della registrazione o in un secondo momento, presentare all'interno della VP ulteriori claim opzionali che abilitano l'accesso a categorie di prodotti regolamentati:

Categoria prodotto	Policy di accesso	Campi richiesti nella VC
Alcolici	Età 18 (21 in alcuni paesi)	"age": 21 oppure "over18": true
Prodotti per adulti	Età 18	"over18": true
Farmaci da banco	Età 18, residenza in paese autorizzato	"over18": true, "residenza": "IT"
Dispositivi medici professionali	Qualifica sanitaria	"healthProfessional": true, "professione": "farmacista"
Armi softair / coltelleria	Età 18, licenza softair (se richiesta)	"over18": true, "weaponPermit": {"tipo": "softair"}
Prodotti a uso fiscale agevolato	Codice fiscale abilitato o stato sanitario	"taxBenefitEligible": true
Contenuti digitali education	Status studente o docente	"vcStudent": true oppure "vcTeacher": true
Forniture B2B / all'ingrosso	Azienda registrata o P. IVA valida	"partitaIVA": "IT1234567890", "azienda": true
Servizi locali	Residenza in area geografica abilitata	"residenza": "Lombardia"
Acquisti minorili con limite	Autorizzazione parentale con soglia	"vcParentalConsent": {"guardian": "did:abc", "maxSpesa": 50}
Software tecnico / strumenti avanzati	Certificazione professionale	"certificazione": "EQF5", "ambito": "elettrotecnica"

Come è stato detto in precedenza si vuole garantire il rispetto dei quattro pilastri fondamentali della sicurezza informatica: **confidenzialità, integrità, trasparenza ed efficienza**. Segue quindi un'analisi delle funzionalità da sviluppare in funzione dei parametri prima citati.

3.1 Analisi degli avversari

In questa sezione si analizzerà ogni avversario nell'ottica delle proprietà di resilienza elencate nel capitolo 1 escludendo trasparenza ed efficienza per le quali saranno dedicate due sezioni a parte, per comprendere se la soluzione proposta nel capitolo 2 è consistente al threat model definito nel primo capitolo. In particolare si possono individuare quattro tipi di avversari.

3.1.1 Attacchi alla qualità delle recensioni (inserire o manipolare feedback)

L'obiettivo di questi avversari è quello di peggiorare la credibilità delle recensioni dell'e-commerce tramite il rilascio di recensioni false e valutazione non veritiera di queste ultime tramite like/dislike fittizi.

Beetle:



Figura 3.1: Beetle

Beetle viene battuto perché:

- Ospite: non riceve la proof of purchase che gli permette di recensire il prodotto in quanto non è registrato;
- Holder: non può recenire il prodotto perché non possiede la proof of purchase ad esso riferito.

Beetle non può forgiare la proof perché è una VC che contiene la firma dell'e-commerce creata con la sua chiave privata.

Le proprietà che l'azione malevola di Beetle coinvolge sono:

- **C5:** Beetle ospite non può recensire il prodotto perché non è registrato e Beetle holder non ha la giusta proof of purchase;
- **Unforgeability:** non è in grado di forgiare la proof da zero.

Hook:

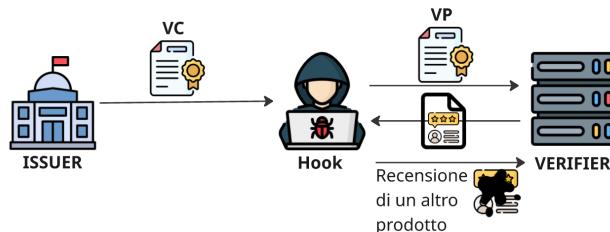


Figura 3.2: Hook

Hook viene battuto in quanto, per poter recensire un altro prodotto, deve modificare la proof of purchase aggiungendo un nuovo claim e la modifica comporta un cambio della firma del verifier e quindi quest'ultimo si accorge che la proof è stata manomessa. Le proprietà che l'azione malevola di Hook coinvolge sono:

- **C5:** non riesce a recensire il prodotto sia se ruba la VP sul canale di comunicazione, sia se viene a conoscenza della proof of purchase tramite data breach;
- **I4:** non riesce a recensire un altro prodotto perché la firma del verifier sarebbe invalida.

Mr.Smee:

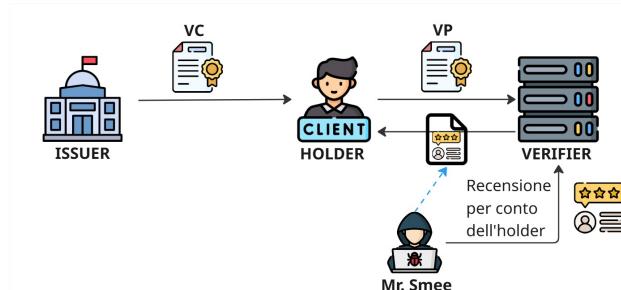


Figura 3.3: Mr.Smee

Supponendo che Mr.Smee venga a conoscenza della proof of purchase tramite un data breach, egli non è in grado di inviare una recensione poiché non riuscirebbe a replicare la firma dell'holder in quanto non conosce la sua chiave segreta. Stessa sorte accade se osserva il canale di comunicazione in cui viene usato tls 1.3 con https: non è in grado di reperire alcun tipo di informazione in quanto queste risulteranno cifrate e incomprensibili e in più non è in grado di firmare la challenge che ha inviato il verifier all'holder onesto.

Le proprietà che l'azione malevola di Mr.Smee coinvolge sono:

- **C5:** in alcun modo Mr.Smee è in grado di lasciare una recensione.
- **I2:** Mr.Smee non è in grado di modificare la proof of purchase (VC).

Dalì:



Figura 3.4: Dalì

Dalì, nel caso in cui volesse fare la recensione prima che il prodotto gli arrivi, viene battuto, in quanto solo alla sua ricezione, egli riceverà la proof of purchase; di conseguenza se effettua il reso prima di ricevere l'acquisto, non può emettere la recensione.

Se invece Dalì riceve il prodotto, ne fa il reso e inserisce una recensione falsa, NON viene battuto immediatamente dal sistema di controllo del verifier ma c'è la possibilità che a lungo termine, la rete di reviewer possa accorgersi del suo comportamento disonesto e quindi penalizzarlo fino ad essere inserito in una blacklist per non permettergli di fare altre recensioni.

La rete dei reviewer è interessata a fare questi controlli perché vuole far permanere l'affidabilità del meccanismo di recensioni in quanto gli genera introiti in termini di ETH e NFT.

In definitiva questo attacco può essere considerato battuto se viene fatto su larga scala (ossia acquista e recensisce tanti prodotti in maniera disonesta) ma non se viene fatto su scala ridotta (ossia acquista e recensisce pochi prodotti in maniera disonesta).

Le proprietà che l'azione malevola di Dalì coinvolge sono:

- **C5:** nel primo caso descritto, Dalì non riesce a fare la recensione perché non ha la proof of purchase; nel secondo caso, Dalì RIESCE a fare la recensione in quanto ha la proof of purchase e non viene battuto solo se fa recensioni di pochi prodotti.

Cosmo e Wanda:



Figura 3.5: Cosmo e Wanda

Cosmo e Wanda vengono battuti a larga scala in quanto se l'azienda Wanda paga una serie di utenti col ruolo di Cosmo per far sì che possa ricevere un vasto numero di recensioni positive ai propri prodotti e magari denigrare altri prodotti di altre aziende, queste potrebbero non essere in linea con la vera valutazione che i vari utenti Cosmo avrebbero voluto attribuirle; in questo modo la rete di reviewer onesti si accorge della situazione e contribuisce nell'inserirli in una blacklist in quanto il sistema prevede una serie di reward per gli utenti che si comportano onestamente. La proprietà che l'azione malevola di "Cosmo e Wanda" coinvolge sono:

- **R2:** a scala ridotta questo attacco va a buon fine in quanto la rete dei reviewer non è in grado di affermare con assoluta certezza se le recensioni rilasciate dai vari utenti Cosmo sono veritieri o meno.

Beagle Boys:

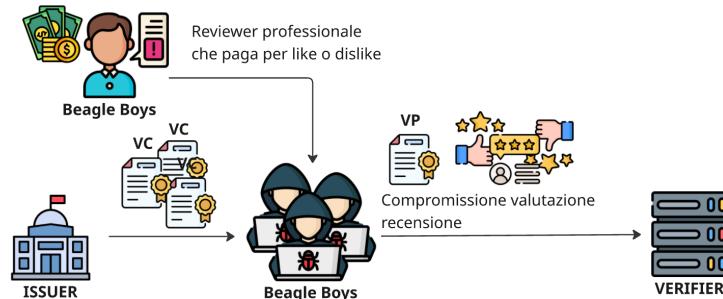


Figura 3.6: Beagle Boys

I Beagle Boys vengono battuti a larga scala in quanto se colludono tra di loro per far sì che possano inserire una serie di recensioni non veritieri a determinati prodotti, la rete di reviewer onesti si accorge della situazione e contribuisce nell'inserirli in una blacklist in quanto il sistema prevede una serie di reward per gli utenti che si comportano onestamente.

La proprietà che l'azione malevola di "Beagle Boys" coinvolge sono:

- **R2:** a larga scala questo attacco va a buon fine in quanto causa un'inaffidabilità delle recensioni che porta la rete onesta ad intervenire.

3.1.2 Forzare l'accesso o falsificare le credenziali

L'obiettivo di questi avversari è quello di ottenere l'accesso al sito web per utilizzare le sue funzionalità senza avere effettivamente i requisiti minimi per ottenere le credenziali certificate.

Pirandello:

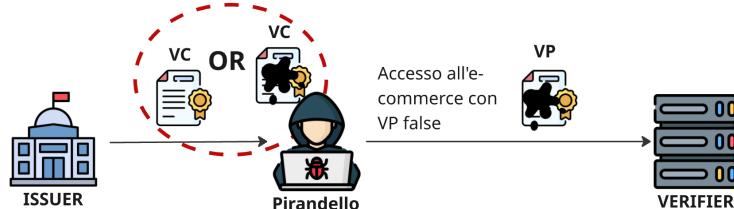


Figura 3.7: Pirandello

Pirandello viene battuto se prova a forgiare di suo pugno una VP firmata con la propria chiave privata, in quanto il verifier, applicando la chiave pubblica dell'issuer conservata nel suo did document, si accorgerà che quella presente nella VP è errata. Tale VP falsa può essere generata sia a partire dalla VC ricevuta dall'issuer sia a partire da una VC creata proprio da Pirandello. Le proprietà che l'azione malevola di Pirandello coinvolge sono:

- **C4:** non accederà all'e-commerce in quanto non ha VP proveniente da issuer affidabili;
- **I3:** non accederà all'e-commerce in quanto, invalidando la firma dell'Issuer la verifica non andrà a buon fine;
- **Unforgeability:** non è in grado di forgiare la firma valida della proof anche a partire da una VC ottenuta dall'issuer.

Sibilio:

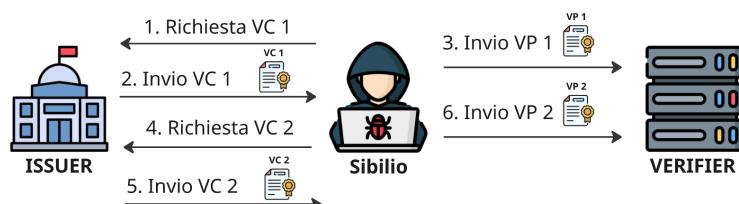


Figura 3.8: Sibilio

Sibilio viene battuto in quanto accede con il proprio SPID il quale garantisce che le stesse credenziali possano essere richieste, per il singolo utente, per una sola VC. Quello che accade è che Sibilio, richiedendo l'ottenimento di più VC con le stesse credenziali per accedere al servizio, questo gli verrà negato dall'issuer in quanto ha già precedentemente rilasciato una VC contenente le stesse credenziali.

Le proprietà che l'azione malevola di Sibilio coinvolge sono:

- **R1:** Sibilio non è in grado di creare più account fake perché l'issuer non fornisce più di una VC contenente le stesse credenziali per accedere al servizio.

3.1.3 Man in the middle (MITM) e replay attack: furto, reinvio di VC/VP

L'obiettivo di questi avversari è quello di ottenere l'accesso al sito web per utilizzare le sue funzionalità senza avere effettivamente i requisiti minimi per ottenere le credenziali certificate.

Tom:

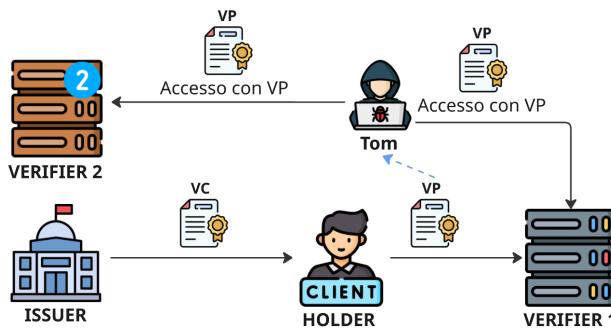


Figura 3.9: Tom

Tom viene battuto sulla base delle seguenti situazioni:

- se Tom viene a conoscenza della VP dell'holder tramite un data breach e se le policy di accesso di verifier 1 e verifier 2 sono le stesse, non è in grado di accedere in quanto nella firma dell'holder c'è un nonce e il timestamp ricevuto dal verifier durante l'apertura della connessione, che Tom non potrà replicare in quanto il verifier 2 invierà a Tom un diverso nonce e timestamp da firmare e quindi la firma risulterà essere differente da quella contenuta nella VP dell'holder;
- se osserva la VP sul canale in cui viene utilizzato tls 1.3 e https allora Tom non capirà niente in quanto è già il protocollo ad essere sicuro che protegge da eventuali replay attacks;

Le proprietà che l'azione malevola di Tom coinvolge sono:

- **C3:** il servizio non viene fornito neanche se le policy di accesso del verifier 2 permettono alla VP di contenere gli stessi claim che consentono l'holder di accedere al verifier 1;
- **C4:** non riesce ad accedere in quanto è sprovvisto di credenziali valide fornite da un issuer fidato;
- **I3:** non capendo niente dalla VC e provando a creare una VP con claim falsi, questi saranno incoerenti rispetto a quelli presenti nella VC;
- **Non-transferability:** in entrambe le situazioni descritte in precedenza, Tom non è in grado di accedere al verifier 2.

Jerry:

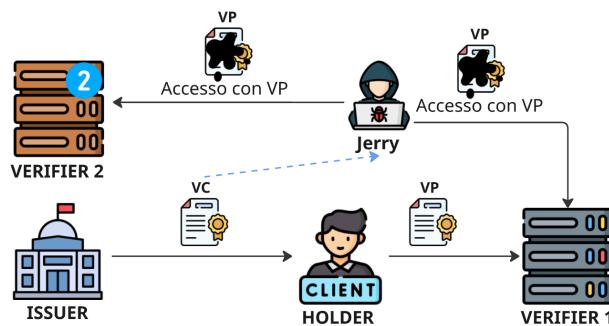


Figura 3.10: Jerry

Jerry viene battuto grazie al fatto che si assume che la comunicazione tra issuer e holder avvenga tramite https e usi tls 1.3 che previene attacchi di sniffing del traffico di rete. Quello che infatti Tom osserva è una VC cifrata dalla quale non riesce a trarne alcun beneficio. Conseguentemente la VP che crea è una VP che sia il verifier 1 che il verifier 2 non riterranno come valida in quanto Jerry non è in grado di replicare la firma dell'issuer.

Le proprietà che l'azione malevola di Jerry coinvolge sono:

- **C2:** non capisce niente in quanto il canale è cifrato;
- **C3:** non riesce ad accedere ai verifier;
- **C4:** non riesce ad accedere in quanto è sprovvisto di credenziali valide fornite da un issuer fidato;
- **I3:** non capendo niente dalla VC e provando a creare una VP con claim falsi, questi saranno incoerenti rispetto a quelli presenti nella VC;
- **Non-transferability:** non riesce ad accedere a nessuno dei due verifier per conto dell'holder.

Pooh:

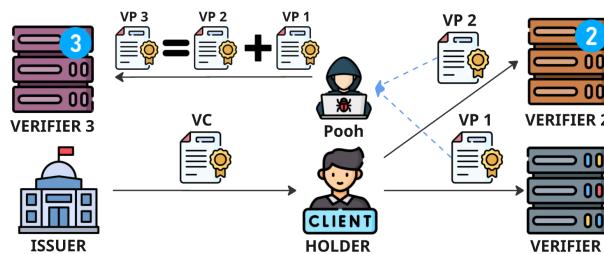


Figura 3.11: Pooh

Pooh viene battuto perché nella progettazione l'holder presenta due VP con did diversi per accedere ai due verifiers, quindi, supponendo che verifier 3 chiede credenziali diverse, Pooh non riuscirebbe ad accedervi combinando le due VP che ha ricevuto perché non è in grado di capire se queste due presentation provengano dalla stessa persona fisica.

Anche se l'holder non dovesse rispettare appieno il protocollo, ossia se usasse la VC con lo stesso did per creare le due VP e accedere ai due siti che richiedono credenziali diverse, allora pooh riuscirebbe solo ad ottenere più informazioni sull'holder (information leakage) ma comunque non riuscirebbe a combinarle per ottenere una terza VP valida in quanto non riuscirebbe a forgiare la firma dell'holder.

Le proprietà che l'azione malevola di Pooh coinvolge sono:

- **C4:** Pooh non è in grado di ottenere l'accesso al sito in quanto non è in grado di mettere assieme le due VP che osserva;
- **I2:** Pooh non è in grado di modificare le VP che osserva;
- **I3:** Pooh non riesce a combinare i claim provenienti da due VP diverse in quanto queste hanno did diversi;
- **Unlinkability:** Pooh, nella situazione in cui l'holder usa il protocollo correttamente, non è in grado di associare che i due did diversi appartengano alla stessa persona fisica;
- **Unforgeability:** Pooh non è in grado di forgiare la VP 3 a partire dalle VP 1 e VP 2 in quanto non può forgiare la firma che è ottenibile solo con la chiave privata dell'holder.

Giuda & the Apostles:

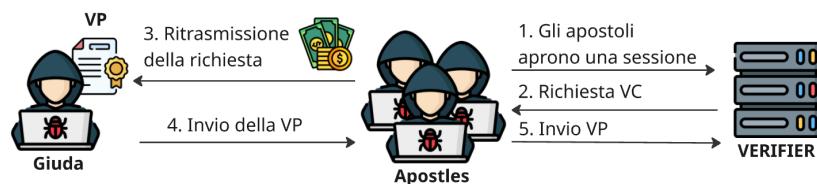


Figura 3.12: Giuda e gli apostoli

Giuda e gli apostoli non vengono battuti su scala ridotta in quanto, grazie alla loro collaborazione riescono ad accedere al verifier con una VP valida come se fosse Giuda a farlo. Questo perché gli apostoli si comportano come proxy tra Giuda e il verifier in cui quando gli apostoli aprono una sessione con il verifier, quest'ultimo invia una proof legata alla specifica sessione, utile nel calcolo della firma di Giuda che forgerà quando gli apostoli ritrasmettono la richiesta verso giuda il quale invierà la VP lecita con la sua firma valida agli apostoli che ritrasmetteranno al verifier per accedere all'e-commerce. Questo attacco a larga scala è pressoché limitato in quanto il verifier può inserire controlli sul fatto che per ogni did può essere attiva una sola sessione; in aggiunta, per come è definito il meccanismo di recensione, se gli apostoli inseriscono tante recensioni per conto di Giuda che sono considerate non attendibili per la rete di reviewer, allora questi ultimi inseriranno tanti dislike a queste recensioni portando a penalità per il did associato a Giuda fino ad inserirlo in una blacklist e quindi non permettendogli più di partecipare al meccanismo di recensioni.

Le proprietà che l'azione malevola di Giuda e gli apostoli coinvolge sono:

- **C3:** in questo caso il servizio che il verifier fornisce a Giuda è visibile anche agli apostoli;
- **Non-transferability:** gli apostoli sono in grado di accedere per conto di Giuda.

3.1.4 Erosione della privacy/de-anonimizzazione

L'obiettivo di questi avversari è quello di ottenere maggiori informazioni riguardo l'utente fisico associato alla sua identità digitale.

Dark Army:

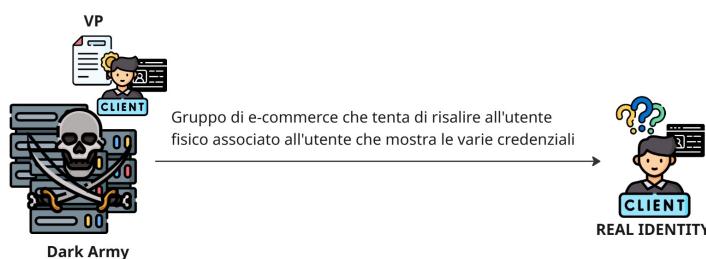


Figura 3.13: Dark Army

Dark Army viene battuto in quanto i verifier non riusciranno mai a risalire alla persona fisica poiché, anche se lo stesso utente ha fatto accesso a siti differenti, egli avrà diversi did e quindi non riconducibili alla stessa persona.

L'attacco funziona solo qualora l'utente non dovesse seguire il protocollo correttamente ossia usa la stessa VC per accedere a siti diversi: in questo caso più verifier possono colludere tra loro e associare allo stesso did più informazioni dello specifico utente fisico.

Le proprietà che l'azione malevola di Dark Army coinvolge sono:

- **Unlinkability:** l'attacco non avviene in quanto due did diversi non sono riconducibili alla stessa persona fisica a meno che l'utente non usi il protocollo correttamente.

3.2 Trasparenza ed efficienza

Nel sistema in esame, la trasparenza e l'efficienza vengono garantite dalla combinazione di blockchain Ethereum, smart-contract pubblici e storage distribuito IPFS.

- **T1 & T5: Protocolli noti:** Tutte le interazioni fra holder-issuer e holder-verifier avvengono su canali TLS 1.3 e seguono le specifiche W3C VC/VP: questi standard sono pubblici, versionati e verificabili da chiunque;
- **T2 & T6 – tracciabilità degli issuer e verifica delle decisioni del verifier:** Ogni credenziale contiene il DID dell'issuer; quando il verifier chiama il resolver ERC-1056 ottiene il DID Document e quindi la chiave pubblica corrispondente. L'intero processo è ancorato on-chain: chiunque può ispezionare i log del contratto, ricostruire le chiavi attive nel tempo e ripetere la verifica delle firme.
- **T3 – algoritmi e protocolli dell'issuer pubblici:** L'issuer usa algoritmi noti a tutti per il corretto funzionamento del sistema;
- **T4 – corretto rilascio delle credenziali:** supponendo che l'issuer sia affidabile le credenziali vengono rilasciate in maniera lecita e trasparente agli utenti;
- **T7 – gestione recensioni pubblica:** Grazie ad IPFS e alla sua trasparenza, i cid rilasciati al momento del rilascio di una recensione sono pubblicamente reperibili e consultabili una volta che vengono inseriti nella blockchain;
- **T8 – incentivi/penalità noti:** La logica di reward è codificata nello smart-contract: parametri come soglia 51 %, reward ETH, NFT al livello 100 ecc.) sono costanti e pubblici a tutti; l'esecuzione avviene interamente on-chain, dunque riproducibile.

Per quanto riguarda l'efficienza:

- **E1 – generazione rapida delle credenziali:** la velocità di creazione delle credenziali è garantita dal fatto che l'issuer usa protocolli di firma e rilascio efficienti che permettono un'elevata riduzione dell'overhead di attesa;
- **E2 – minimo traffico issuer-holder:** Dal momento che la VC è consegnata una sola volta l'holder non deve stabilire sessioni lunghe in quanto l'holder può usare in maniera autonoma le credenziali per accedere ai vari servizi;
- **E3 – login rapido:** Il verifier carica dal VDR il DID Document dell'issuer e verifica la VP localmente non effettuando alcuna chiamata a database esterni;
- **E4 – numero minimo di credenziali:** Grazie alla selective disclosure con BBS+ ogni VC è multi-uso: l'holder non ha bisogno di richiedere varianti ridotte a meno che non voglia accedere a siti che chiedono credenziali differenti; in tal caso dovrà ricontattare l'issuer e ricevere una nuova VC.

- **Minimization:** questa proprietà è garantita dal fatto che l'issuer fornisce solo la credenziale verificabile contenente le informazioni richieste dall'holder;
- **Predicate Disclosure:** questa proprietà è garantita dal fatto che viene utilizzato BBS+ per la Selective Disclosure;
- **Unobservability:** questa proprietà viene rispettata in quanto, quando l'issuer fornisce la VC all'Holder, è lui a decidere a quale Verifier e quando accedere con la relativa VP;
- **Untraceability:** sotto le ipotesi che l'Issuer sia non corrotto la proprietà viene rispettata in quanto non collude col Verifier.

Date le proprietà appena discusse è utile realizzare un grafico radar per visualizzarne in maniera diretta la loro copertura.

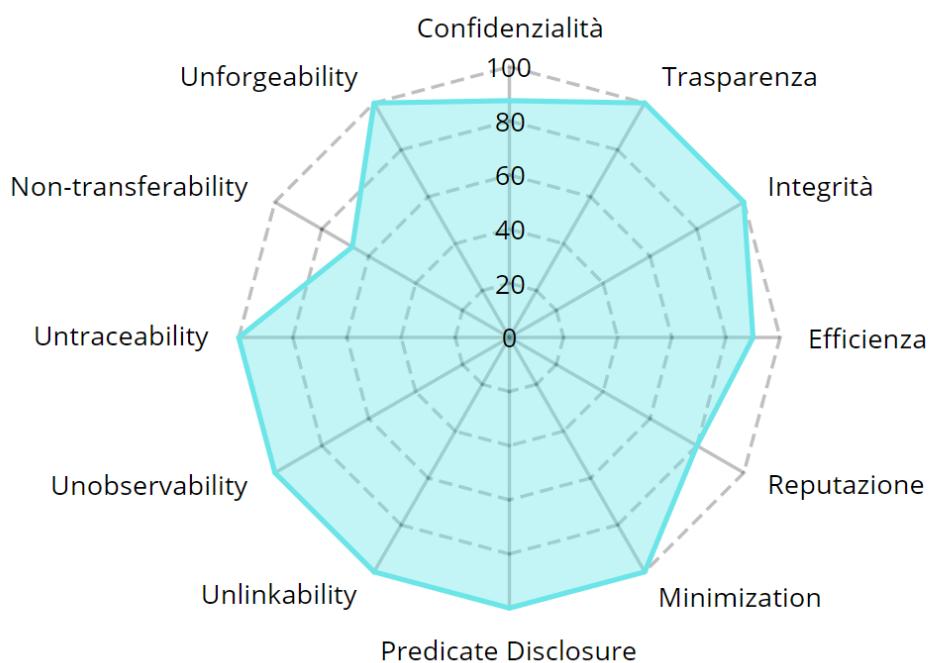


Figura 3.14: Grafico radar

Nota!

Nel calcolo della confidenzialità non sono considerati avversari che vengono battuti solamente con i protocolli standard come TLS e https, quindi la proprietà C1 non viene intaccata da nessun attaccante;

Nel calcolo dell'integrità non sono considerati avversari che vengono battuti solamente con i protocolli standard come TLS e https, quindi la proprietà I1 non viene intaccata da nessun attaccante.

WP4 - Implementazione e Prestazioni

In questo capitolo ci si soffermerà sulla implementazione delle funzionalità introdotte nel WP2. Si procederà nello stesso ordine del capitolo 2, partendo dalla fase di autenticazione, proseguendo con la fase di acquisto e concludendo con la fase di recensione e reward/penalizzazioni. Dove opportuno, la descrizione sarà accompagnata da brevi frammenti di codice che illustrano in modo conciso funzionalità particolarmente interessanti.

4.1 Setup iniziale

Come anticipato, Ethereum è stato scelto come blockchain di riferimento. A tal fine, è stata utilizzata **Ganache** come testnet per eseguire tutte le simulazioni. Il ledger così configurato funge da Verifiable Data Registry (VDR), ospitando esclusivamente i DID Document degli attori coinvolti e consentendo la risoluzione delle rispettive identità in modo verificabile, controllato e trasparente.

Come anticipato, è stato utilizzato **IPFS** per l'archiviazione delle recensioni. Una volta caricate sulla rete IPFS, per ciascuna recensione è stato generato un CID (Content Identifier) univoco. Questo CID è stato successivamente registrato sulla blockchain, rendendo le recensioni rintracciabili e accessibili a tutti gli utenti connessi alla rete in modo decentralizzato e verificabile.

Per poter compilare e deployare i diversi contratti sono stati creati due appositi file: il compile.js che ha il compito di compilare tutti gli smart contract scritti in Solidity (i file con estensione .sol) e il deploy.js che permette di poter deployare i contratti su blockchain ad uno specifico indirizzo per poterli contattare e usare i loro metodi.

E' stata inoltre utilizzata la libreria *did-jwt-vc* che espone alcune funzionalità chiave per la creazione/verifica di VC e VP.

Nota!

Nella file compresso c'è il file .txt che contiene il link per il drive delle cartelle della logica di progetto, quindi file di compile, deploy, diversi file .js e gli smart contract e un file .mp4 in cui è presente un breve video di simulazione in cui viene testata la **Decentralized Application (DApp)** dalla fase di registrazione alla fase di reward/penalizzazioni

4.2 Fase preliminare

Prima di analizzare le tre fasi descritte nel WP2, è stata prevista una fase preliminare in cui sono stati istanziati dieci utenti, rappresentati all'interno di un dizionario definito nel file User-Directory.js. Ogni utente è identificato da un codice univoco (userCode) e associato a diverse caratteristiche, come la nazione di residenza, la maggiore età (over18), il sesso, il possesso del porto d'armi e la condizione di studente o insegnante. Questa situazione può essere paragonata a un utente che, recandosi fisicamente presso un ufficio postale oppure online, ha richiesto l'attivazione dello SPID, identificandosi tramite la propria carta d'identità. A seguito di questa procedura, a ciascun utente è stato assegnato un userCode univoco.

4.3 Fase di registrazione

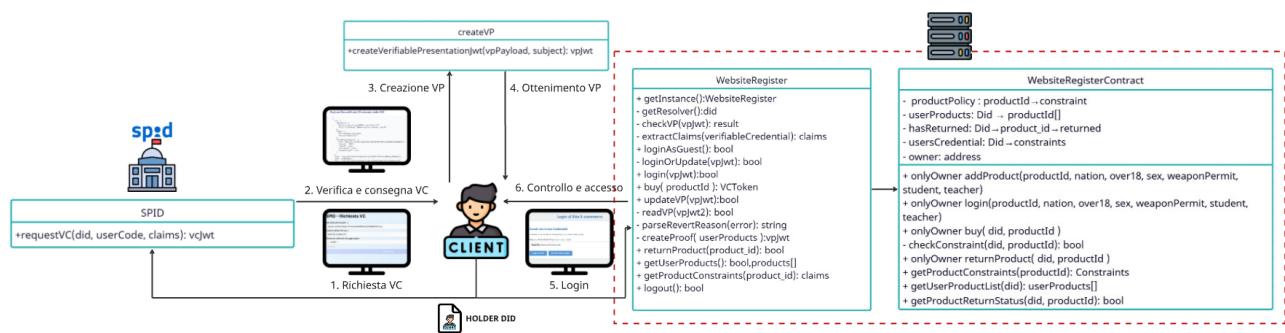


Figura 4.1: Fase di registrazione

Per accedere al sito di e-commerce, l'holder deve essere un utente verificato. A tal fine, può dimostrare la propria identità fornendo al sito una Verifiable Presentation. Il sito e-commerce, che in questo contesto assume anche il ruolo di Verifier, utilizzerà la presentazione per convalidare le credenziali dell'utente. L'utente quindi inizialmente dal proprio dispositivo accederà al sito dello SPID (l'issuer) per richiedere una VC. Il portale dello spid permette all'utente di poter inserire diverse informazioni utili mentre in backend usa un file .js (SPID.js) che espone la funzione di creazione della Verifiable Credential. Questa funzione prende in input:

- il did che l'utente ha precedentemente generato per conto proprio caricando il corrispondente DID document sul VDR in modo tale che il resolver possa effettuare la verifica del suo DID;
- un userCode univoco;
- i claim che l'utente vuole che vengano inseriti all'interno della VC.

Dopo che l'Issuer ha completato i controlli sull'identità dell'utente, rilascia a quest'ultimo una Verifiable Credential (VC) firmata digitalmente, che lo identifica in modo univoco.

Ora che l'utente è in possesso della propria VC, desidera accedere a un sito di e-commerce (il Verifier) per acquistare un prodotto. Per poterlo fare, deve presentare una Verifiable Presentation (VP).

Questa operazione può essere eseguita direttamente dall'utente tramite l'esecuzione dello script createVP.js, il quale utilizza la funzione createVerifiablePresentationJwt() fornita dalla libreria did-jwt-vc. Tale funzione genera una VP firmata dall'utente, contenente i claim precedentemente ottenuti dall'Issuer.

Nota!

Per motivi legati all'implementazione, il Selective Disclosure non è stato integrato, nemmeno nella sua variante basata su BBS+. Di conseguenza, il contenuto informativo della Verifiable Presentation (VP) risulta identico a quello della Verifiable Credential (VC). Ciò che distingue i due documenti non è l'informazione contenuta, ma esclusivamente la loro struttura, già definita secondo le specifiche, che differenzia formalmente una VP da una VC.

Ora che l'utente è in possesso della propria Verifiable Presentation (VP), può accedere alla pagina di login del sito di e-commerce, che richiede appunto la presentazione della VP.

L'utente clicca sul pulsante di login, che, lato backend, attiva la funzione `login()` definita nel file `WebsiteRegister.js`. Questa funzione riceve in input unicamente la VP.

Successivamente, il Verifier interroga il Resolver per recuperare le chiavi pubbliche associate ai DID dell'Issuer e del Verifier, tramite i rispettivi DID Document. Se la verifica delle firme ha esito positivo, l'utente ottiene l'accesso al sito e può procedere liberamente con i propri acquisti. Come già anticipato nel WP2, l'accesso al sito può essere effettuato anche da un utente ospite che a differenza di un holder non deve mostrare al verifier una VP in quanto non verrà registrato.

4.4 Fase di acquisto

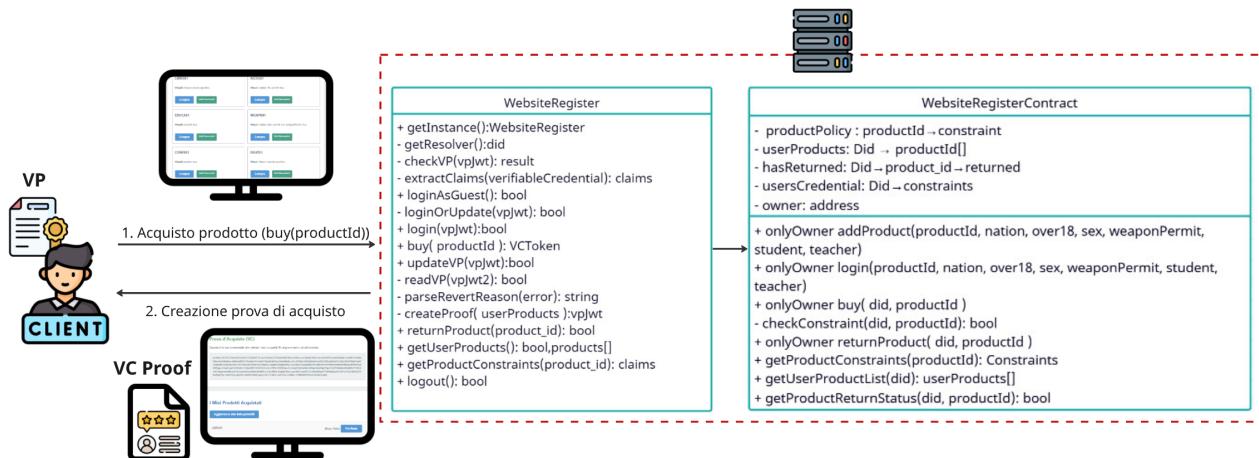


Figura 4.2: Fase di acquisto

Adesso che l'utente ha effettuato l'accesso al sito di e-commerce, nella home page può visualizzare l'elenco completo dei prodotti disponibili per l'acquisto.

Come anticipato, ogni prodotto espone a schermo i vincoli che l'utente deve soddisfare per poter procedere all'acquisto. Ad esempio, se un prodotto richiede il possesso del porto d'armi e l'utente non lo possiede, la procedura d'acquisto fallirà, mostrando un messaggio di errore.

Per acquistare un prodotto, l'utente clicca sull'apposito pulsante, che sul lato backend attiva la funzione `buy()` definita nel file `WebsiteRegister.js`. Questa funzione interagisce poi con il contratto `WebsiteRegisterContract.sol` per registrare l'acquisto, aggiungendo il prodotto al mapping `utente → lista di prodotti acquistati`.

Se l'acquisto va a buon fine, all'utente viene mostrata a schermo una *proof* di acquisto, ovvero una **Verifiable Credential proof** firmata dal *Verifier*, che certifica ufficialmente l'avvenuto acquisto del prodotto.

In questa fase, l'utente ha anche la possibilità di effettuare il *logout* oppure aggiornare la propria *Verifiable Presentation (VP)*. Questo può avvenire, ad esempio, nel caso in cui l'utente individui un prodotto con requisiti specifici che non può ancora soddisfare. In tal caso, può decidere di rivolgersi nuovamente all'*Issuer* per ottenere una nuova *Verifiable Credential (VC)*, da cui generare una VP aggiornata in grado di rispettare i vincoli richiesti.

In questa fase l'utente può anche effettuare un reso di un prodotto tramite l'apposita sezione.

Come già anticipato nel *WP2*, un utente ospite può effettuare l'acquisto di un prodotto, ma solo a condizione che quest'ultimo non presenti vincoli specifici. In pratica, se è un utente ospite a procedere con l'acquisto, il *Verifier* non emetterà alcuna prova di acquisto, in quanto — come già discusso — l'utente ospite non è abilitato a partecipare al meccanismo delle recensioni.

4.5 Fase di recensione

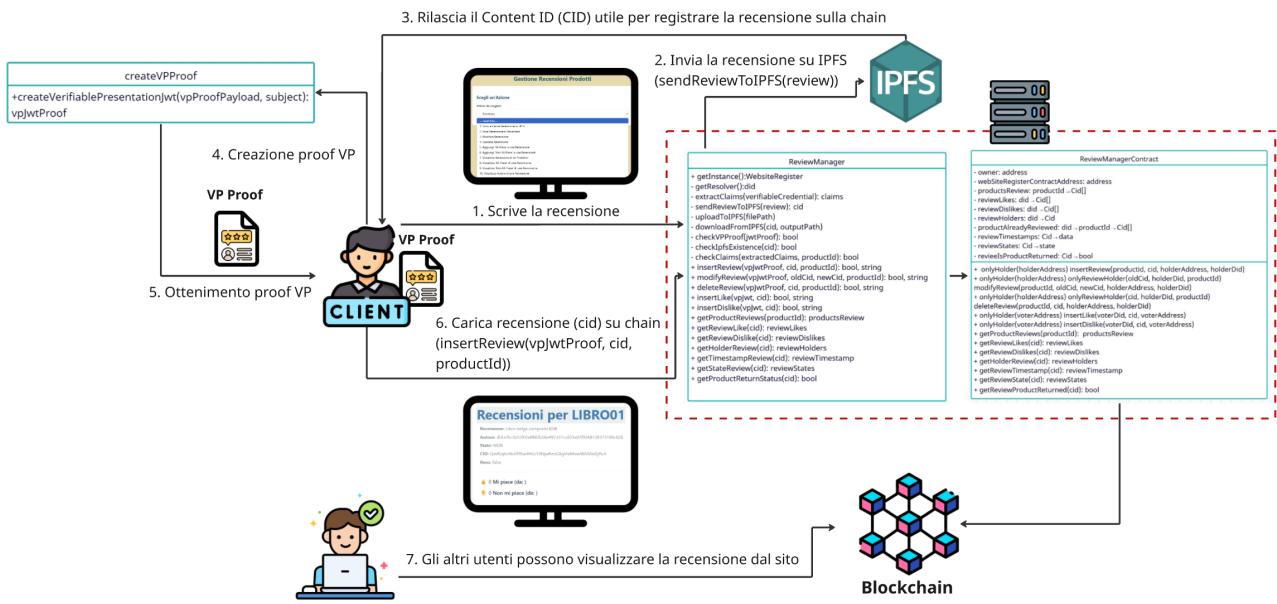


Figura 4.3: Fase di recensione

A questo punto, l'utente desidera recensire un prodotto precedentemente acquistato, ad esempio quello con **id = "LIBR001"**.

Per farlo, accede a una piattaforma distinta rispetto a quella dell'e-commerce, ma che, per semplicità e coerenza implementativa, si suppone appartenere anch'essa al *Verifier*.

Questa piattaforma, oltre alla possibilità di inserire una recensione, offre ulteriori funzionalità, tra cui: modificare o cancellare una recensione già esistente, esprimere un like o un dislike, visualizzare le recensioni associate a un determinato prodotto, identificare chi ha messo like o dislike a un prodotto, nonché visualizzare l'autore di ciascuna recensione.

L’utente inserisce la recensione in un apposito *textfield*, ma — come già indicato nel *WP2* — il contenuto della recensione non viene salvato direttamente sulla blockchain, per motivi di spazio ed efficienza. Invece, viene caricato su **IPFS** tramite la funzione `sendReviewToIPFS()` del file *ReviewManager.js*. IPFS restituisce un *Content Identifier (CID)*, che viene mostrato all’utente.

Per procedere con l'inserimento effettivo della recensione sulla blockchain, l'utente ha bisogno di una **VP proof**, ovvero una *Verifiable Presentation* generata a partire dalla *VC proof* precedentemente rilasciata dal *Verifier*, a dimostrazione dell'acquisto del prodotto. Questa VP proof serve quindi a garantire che la recensione provenga effettivamente dall'acquirente.

Una volta ottenuta la VP proof, l'utente può inserire la recensione on-chain tramite la funzione **insertReview()** definita nel file *ReviewManager.js*, specificando anche il **productID** a cui associare la recensione. (Alternativamente, sarebbe stato possibile utilizzare la tecnica di *Selective Disclosure* sulla VP proof per riferirsi solo al prodotto recensito, ma come già detto in precedenza, tale funzionalità non è stata implementata.)

La funzione **insertReview()** chiama l'omonima funzione del contratto *ReviewManagerContract.sol*, che si occupa di registrare la recensione sulla blockchain. Una volta pubblicata, la recensione sarà visibile a tutti gli utenti accedendo al sito di e-commerce, insieme alle sue principali caratteristiche: numero di like e dislike, chi ha espresso tali reazioni, stato della recensione, CID su IPFS e indicazione sull'eventuale restituzione del prodotto recensito.

Nota!

La funzione **insertReview()** deve essere invocata direttamente dall'*Holder*, poiché è fondamentale che quest'ultimo sia libero di caricare la propria recensione senza dover necessariamente passare attraverso i filtri del sito web.

Tuttavia, questo approccio introduce una possibile vulnerabilità: un attaccante potrebbe interagire direttamente con il contratto *ReviewManagerContract.sol*, chiamando la funzione **insertReview()** senza passare attraverso il modulo *ReviewManager.js*, che invece richiede la *proof* di acquisto (la **VP proof**). In questo modo, l'*Holder* può aggirare i controlli e pubblicare recensioni relative a prodotti che in realtà non ha mai acquistato.

Per mitigare questa vulnerabilità, nell'implementazione della funzione **insertReview()** sul contratto, è stato introdotto un controllo esplicito. In particolare, si utilizza la funzione **getUserProducts()** del contratto *WebsiteRegisterContract.sol* per verificare che l'utente abbia effettivamente acquistato il prodotto in questione. Tuttavia, questo rende la *proof* di acquisto una ridondanza, poiché il contratto stesso è già in grado di verificare autonomamente la validità della recensione.

Le soluzioni potrebbero quindi essere due:

- Togliere la *proof* perdendo in efficienza ma non in sicurezza;
- Mantenere la *proof* di acquisto, ma rimuovere il controllo interno alla funzione **insertReview()** che utilizza **getUserProducts()**. In questo modo si guadagna in efficienza, evitando verifiche on-chain ridondanti, pur non garantendo la sicurezza fin da subito. La sicurezza viene invece affidata alla rete di *reviewer* onesti, che agiscono come una vera e propria commissione di validazione. Se un *Holder* inserisce una recensione relativa a un prodotto che non ha effettivamente acquistato — come rilevabile pubblicamente dalla lista degli acquisti — i *reviewer* possono reagire assegnando *dislike*. L'accumulo di penalizzazioni può portare, come previsto dal protocollo, all'esclusione dell'utente dalla rete dei recensori. Questo meccanismo ricalca quanto avviene nella rete Ethereum, dove un attaccante può ad esempio proporre due blocchi in uno stesso slot (tentativo di *double spending*); in tal caso, la commissione di *validator* onesti può inserire nel blocco successivo la prova matematica dell'attacco, facendo scattare lo *slashing* — ovvero la penalizzazione e l'esclusione del validatore malevolo.

4.6 Fase di reward

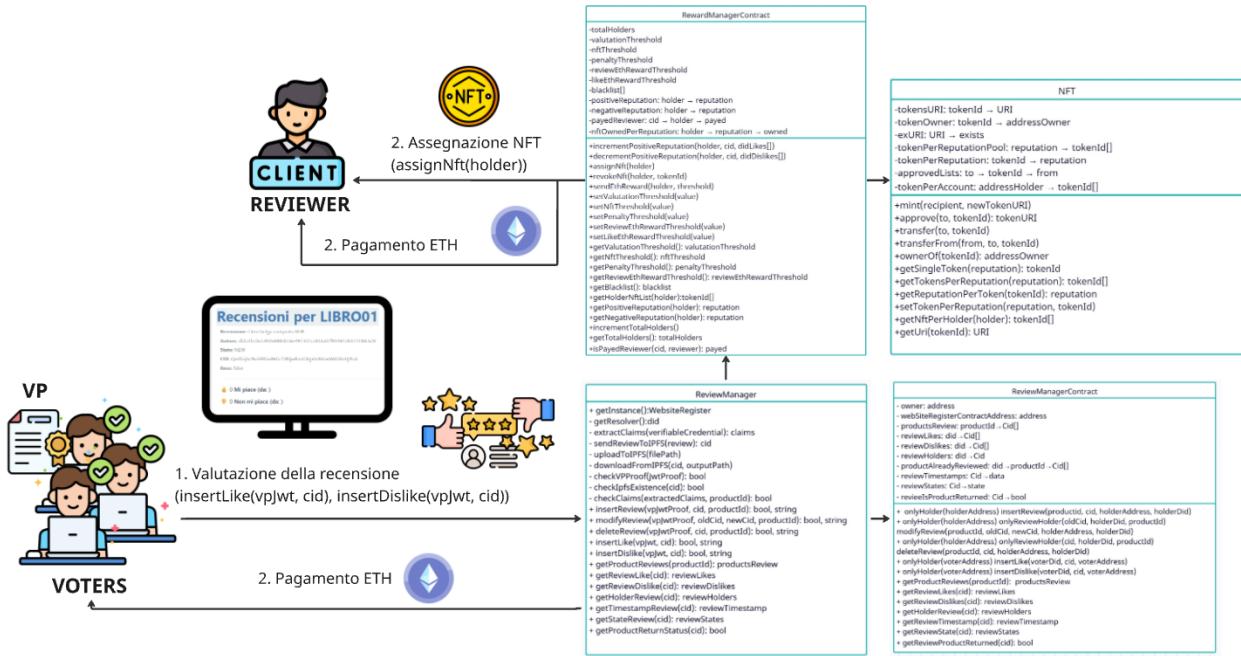


Figura 4.4: Fase di reward

L'ultima fase del sistema è quella di **reward**, nella quale gli utenti registrati al sito possono esprimere una valutazione positiva o negativa sulle recensioni pubblicate da altri utenti. Gli utenti che partecipano a questa fase vengono definiti *voters*, mentre l'autore della recensione è indicato come *reviewer*. I *voters* osservano le recensioni attraverso l'interfaccia del sito e-commerce e, se desiderano valutarne una, accedono alla piattaforma esterna per esprimere il proprio giudizio.

Un *voter* che intende valutare positivamente una recensione invoca la funzione `insertLike()` del modulo `ReviewManager.js`, che a sua volta chiama la funzione omonima nel contratto `ReviewManagerContract.sol`. Questa operazione comporta l'aggiunta del votante al mapping `cid → likers`, ovvero alla lista di utenti che hanno espresso un giudizio positivo su quella recensione. Lo stesso meccanismo vale anche per i *dislike*, che vengono gestiti tramite il mapping `cid → dislikers`.

Questo sistema di voto è fondamentale, poiché — come definito nella parte finale del *WP2* — quando una recensione raggiunge o supera il **51%** di valutazioni (positive o negative) rispetto al numero totale di utenti registrati alla rete, scatta il meccanismo di ricompensa in **Ethereum**. In dettaglio, il sistema prevede due soglie distinte di ricompensa:

- Il **reviewer**, ovvero l'autore della recensione, riceve una ricompensa di **0.01 ETH**.
 - Ogni **voter** che ha espresso il giudizio vincente (like o dislike, a seconda del superamento della soglia) riceve una ricompensa di **0.001 ETH**.

Ciò significa che, una volta raggiunto il consenso del **51%**, tutti i membri della maggioranza vengono premiati attraverso la funzione **sendEthReward(holder, threshold)** del contratto *RewardManagerContract.sol*. Questa funzione viene invocata automaticamente ogni volta che viene inserito un *like* o un *dislike* dalla funzione **insertLike()** di *ReviewManager.js*, con lo scopo di verificare se, con la nuova valutazione, sia stata superata la soglia stabilita. In tal caso, il sistema procede con l'erogazione dei premi.

Questo meccanismo incentiva non solo la produzione di recensioni di qualità da parte dei *reviewer*, ma anche la partecipazione attiva e onesta degli utenti nella loro valutazione.

Per evitare comportamenti opportunistici, come l'aggiunta e la rimozione sistematica di *like* o *dislike* con l'intento di ricevere ricompense multiple, è stato previsto che sia i *voters* sia il *reviewer*, una volta ricompensati per una determinata recensione, non possano ricevere ulteriori premi in ETH per la stessa recensione. Inoltre, nel momento in cui viene superata la soglia del **51%** di valutazioni (positive o negative), la reputazione del *reviewer* viene aggiornata:

- Se la soglia è superata da **like**, la reputazione positiva del reviewer viene incrementata di 1 punto.
- Se la soglia è superata da **dislike**, viene incrementata la reputazione negativa.

Per motivi pratici e dimostrativi, nel codice:

- La **soglia di reputazione positiva** è fissata a 3, per poter testare facilmente il meccanismo.
- La soglia critica di **reputazione negativa** è stata fissata a 2, invece di 25, per accelerare le simulazioni.

Come specificato nella parte finale del *WP2*, ogni volta che la **differenza tra reputazione positiva e negativa** supera il valore di 100, il reviewer riceve un **NFT** come ricompensa. Questo NFT può essere utilizzato, ad esempio, per accedere a sconti o a contenuti esclusivi sulla piattaforma. Nel contesto della demo, la differenza necessaria è ridotta a 3.

Infine, se un utente raggiunge una **reputazione negativa pari a 25** (valore reale, o 2 nella versione semplificata) e possiede già un NFT, esso gli verrà **revocato**. Se invece non possiede alcun NFT, verrà inserito in una **blacklist**, perdendo così la possibilità di partecipare al sistema delle recensioni e di accedere alla piattaforma e-commerce.

Elenco delle figure

1.1	Modello completo del sistema	11
1.2	Beetle	12
1.3	Hook	12
1.4	Mr.Smee	13
1.5	Dalì	13
1.6	Cosmo e Wanda	14
1.7	Beagle Boys	14
1.8	Pirandello	15
1.9	Sibilio	15
1.10	Tom	16
1.11	Jerry	16
1.12	Pooh	17
1.13	Giuda e gli apostoli	17
1.14	Dark Army	18
2.1	Fase di registrazione	25
2.2	Fase di acquisto	26
2.3	Fase di recensione	27
3.1	Beetle	31
3.2	Hook	32
3.3	Mr.Smee	32
3.4	Dalì	33
3.5	Cosmo e Wanda	34
3.6	Beagle Boys	34
3.7	Pirandello	35
3.8	Sibilio	35
3.9	Tom	36
3.10	Jerry	37
3.11	Pooh	38
3.12	Giuda e gli apostoli	39
3.13	Dark Army	39
3.14	Grafico radar	41
4.1	Fase di registrazione	43

4.2 Fase di acquisto	44
4.3 Fase di recensione	45
4.4 Fase di reward	47