



UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

TESI DI LAUREA

A Decentralized Web Authentication Approach leveraging Polygon ID

RELATORE

Prof. Christiancarmine Esposito

Università degli Studi di Salerno

CANDIDATO

Tommaso Sorrentino

Matricola: 0512108628

Anno Accademico 2022-2023

Abstract

L'evoluzione dell'identità digitale, da sistemi centralizzati e federati a soluzioni decentralizzate, segna un cambiamento significativo nel modo in cui gli utenti interagiscono con i servizi online. Questa tesi esplora il concetto di Self-Sovereign Identity (SSI) come alternativa rivoluzionaria ai modelli tradizionali, utilizzando tecnologie come la blockchain per garantire una gestione dell'identità più sicura, privata e interoperabile. Attraverso un'analisi dettagliata dell'infrastruttura offerta da Polygon ID, il lavoro mira a superare i limiti dei sistemi attuali proponendo un nuovo approccio basato sulla blockchain per l'autenticazione. Il focus è sull'analisi delle funzionalità di Polygon ID, quali la Zero-Knowledge Proof (ZKP), per consentire agli utenti di mantenere il controllo completo sui propri dati personali, promuovendo al contempo la fiducia e la privacy nelle interazioni online. Attraverso l'utilizzo di Verifiable Credential (VC) e l'esplorazione delle relazioni tra le entità chiave come il Titolare dell'Identità, l'Emittente e il Verificatore, la ricerca si propone di dimostrare come le tecnologie decentralizzate possano ridefinire l'autenticazione digitale, offrendo soluzioni innovative per la gestione dell'identità digitale in un'era sempre più connessa.

Indice

Indice	ii
Elenco delle figure	iv
Elenco delle tabelle	vi
1 Introduzione	1
1.1 Motivazioni	1
1.2 Obiettivi	2
1.3 Metodologia	2
2 Background	4
2.1 Autenticazione	4
2.2 Modello centralizzato	4
2.3 Modello federato	5
2.4 Self-Sovereign Identity	6
2.5 Soluzioni correnti per i wallet digitali	7
2.5.1 Soluzioni basate su DPKI:	7
2.6 Perchè Polygon ID?	10
3 Polygon ID	14
3.1 Componenti dell'infrastructure layer	15
3.1.1 Schemas	15
3.1.2 Rilascio	16

3.1.3	Verifica	18
3.2	Wallet	21
3.2.1	Wallet SDK	22
3.3	Iden3com	22
3.3.1	Claims	23
3.3.2	Identità	24
3.3.3	Genesis ID	25
3.3.4	Stato dell'Identità	26
3.3.5	Funzione di Transizione dello Stato d'Identità	27
3.3.6	Interazioni tra identità e claims	27
3.3.7	JSON Web Zero-Knowledge	31
4	Lavoro proposto	33
4.1	Definizione Dell'Issuer e utilizzo della Issuer-node-ui	33
4.1.1	Issuer-node-ui	33
4.1.2	Caso d'uso KYC Age Credential	33
4.2	Definizione del Wallet e utilizzo della Flutter-SDK	37
4.2.1	Vantaggi di Flutter e uso di polygonID	38
4.2.2	Caso d'uso KYC Age Credential	39
4.3	Definizione del Verifier e utilizzo della Verifier-SDK	40
4.3.1	Server del Verificatore	41
4.3.2	Client del Verificatore	41
4.3.3	Caso d'uso KnowYourAge credential	43
4.3.4	Estrazione della Prova dal Request	44
4.3.5	Erogazione del servizio richiesto dall'utente	45
5	Conclusioni	47
	Ringraziamenti	50

Elenco delle figure

2.1	Processo di autenticazione centralizzata	5
2.2	Processo di autenticazione nel modello federato	5
2.3	Processo di autenticazione SSI	6
3.1	Architettura nodo Issuer	15
3.2	Processo di erogazione di credenziali con Merkle Tree Root	16
3.3	Processo di erogazione delle credenziali con babyjubjub	17
3.4	Sequence diagram relativo al processo di verifica off-chain	20
3.5	Struttura dei claim	23
3.6	Struttura di un identità	25
3.7	Gestione dello stato di un identità	26
3.8	Diagramma stato di identità	28
4.1	Interfaccia grafica interattiva dell'issuer-node	35
4.2	Credential id della credenziale erogata	35
4.3	Generazione del file JSON da esporre al wallet	36
4.4	Qr-code della credenziale	36
4.5	Homepage del wallet	37
4.6	Homepage del wallet dopo aver generato il DID	38
4.7	Scansione del QR-code dall'app del wallet	40
4.8	Credenziale KYC Age presente sul wallet	40
4.9	Verifica della credenziale richiesta dal verifier	44

4.10 Immagine relative al wallet durante il processo di generazione della Zero Knoledge Proof e di comunicazione con il verifier	46
---	----

Elenco delle tabelle

2.1	Protocolli adottati dalle soluzioni attuali.	9
2.2	Costi delle operazioni su Ethereum e Polygon POS	12
2.3	Prezzi del gas e delle criptovalute al 27 febbraio 2023	12

CAPITOLO 1

Introduzione

1.1 Motivazioni

La gestione dell'identità digitale è un aspetto fondamentale della nostra vita online che influenza fortemente le interazioni che abbiamo con internet, dall'accesso alle email e ai social media, fino all'utilizzo di servizi bancari e di shopping online. In pratica, si tratta del modo in cui le piattaforme online riconoscono chi siamo, assicurandosi che siamo. Tradizionalmente, questo processo si basa su sistemi centralizzati, dove un'unica organizzazione, come un social network o un fornitore di email, mantiene un elenco di utenti e le loro password in un unico punto di controllo. Sebbene ciò possa semplificare il processo di accesso ai vari servizi, presenta notevoli rischi in termini di sicurezza e privacy, poiché un attacco a tale sistema potrebbe compromettere tutte le informazioni degli utenti [1].

Al contrario, i sistemi SSI si pongono come una rivoluzione nell'ambito dell'identità digitale, promuovendo un modello in cui l'individuo detiene il pieno controllo sui propri dati. Attraverso l'adozione di tecnologie decentralizzate, come la blockchain, la SSI elimina la necessità di intermediari centralizzati, riducendo così il rischio di attacchi centralizzati e aumentando la sicurezza e la privacy degli utenti. In questa nuova architettura, gli utenti possono condividere selettivamente le informazioni di identità con le parti richiedenti, senza rivelare più dati del necessario, promuovendo una gestione dell'identità più sicura, privata e interoperabile. L'identità auto-sovrana, infatti, si basa sulla condivisione e conservazione di rivendicazioni verificabili detenute al di fuori della blockchain. L'autenticità di queste

informazioni firmate è garantita attraverso l'hash memorizzato sulla blockchain. Quando gli utenti presentano queste rivendicazioni a una parte affidabile, l'hash può essere confrontato con il record disponibile sulla blockchain e verificato attraverso una firma integrata. Ciò consente alla parte affidabile di accettare rapidamente e precisamente la validità della rivendicazione. La blockchain, con la sua infrastruttura di chiave pubblica decentralizzata, offre non solo sicurezza e integrità degli oggetti dati, ma anche metodi robusti per crittografia e autenticazione. Inoltre, le caratteristiche chiave della blockchain, come l'immutabilità, l'usabilità e i bassi costi di transazione, aprono ampie opportunità per i sistemi di identità, contribuendo così a ridefinire il panorama dei sistemi distribuiti.

1.2 Obiettivi

L'obiettivo della tesi è investigare il funzionamento delle attuali piattaforme di autenticazione decentralizzata al fine di capire quali sono i punti deboli di esse. Una volta ottenuta un chiaro stato dell'arte sulle correnti implementazioni, verrà proposta un innovativo approccio basato su blockchain.

1.3 Metodologia

L'obiettivo della tesi è esaminare l'implementazione e l'utilizzo delle tecnologie offerte da Polygon ID per creare una soluzione innovativa nel campo delle identità digitali. Polygon ID fornisce un'infrastruttura per identità che facilita relazioni affidabili e sicure tra app e utenti, seguendo i principi dell'identità auto-sovrana e della privacy.

Attraverso l'analisi delle funzionalità offerte da Polygon ID, come le prove a conoscenza zero, con questo lavoro si intende esplorare le possibilità di utilizzare questa piattaforma per consentire agli utenti di dimostrare la propria identità senza dover rivelare informazioni sensibili.

Un focus particolare sarà rivolto alla comprensione dei concetti fondamentali di Polygon ID, come le Credenziali Verificabili, il Titolare dell'Identità, l'Emittente e il Verificatore. Esporerò come queste tecnologie possano essere utilizzate per garantire la privacy, la sovranità dell'utente e la fiducia transitoria tra le entità coinvolte.

Verrà esaminata l'integrazione delle funzionalità offerte da Polygon ID con lo sviluppo di un'applicazione utilizzando il Flutter SDK offerto da polyone, la messa in funzione di un

Verifier utilizzando il Verifier SDK e di un Issuer utilizzando l'agente che espone le degli end-point per realizzare i servizi di issuing.

Infine, si esplorerà il rapporto tra Polygon ID e Iden3, l'open-source protocol che sottende Polygon ID, per comprendere come questo protocollo influenzi l'implementazione e di Polygon ID.

La struttura della tesi sarà articolata nei seguenti capitoli, ciascuno dedicato a un aspetto fondamentale dello studio:

- **Background:** Questo capitolo delineerà il panorama attuale dell'autenticazione web, esplorando i concetti fondamentali e analizzando i protocolli che regolano i wallet digitali già esistenti. Verranno inoltre discusse le ragioni alla base della scelta di Polygon ID come fornitore di soluzioni ottimali per il progetto in questione.
- **Polygon ID:** Si entrerà nel dettaglio di Polygon ID, esaminandone l'architettura, i componenti chiave e i protocolli sottostanti che ne definiscono le operazioni. Questo capitolo mira a fornire una comprensione approfondita delle capacità e delle innovazioni portate da Polygon ID nel campo dell'autenticazione decentralizzata.
- **Lavoro proposto:** Verrà presentato il progetto proposto, evidenziando la sua struttura, le funzionalità e un caso d'uso pratico. Questa sezione illustrerà come Polygon ID viene implementato per risolvere problemi concreti nell'ambito dell'autenticazione web, fornendo al contempo un modello per applicazioni future.
- **Conclusioni:** Il capitolo conclusivo rifletterà sui risultati ottenuti e discuterà le potenziali aree di miglioramento e le direzioni future della ricerca e verranno considerate le implicazioni del lavoro svolto.

CAPITOLO 2

Background

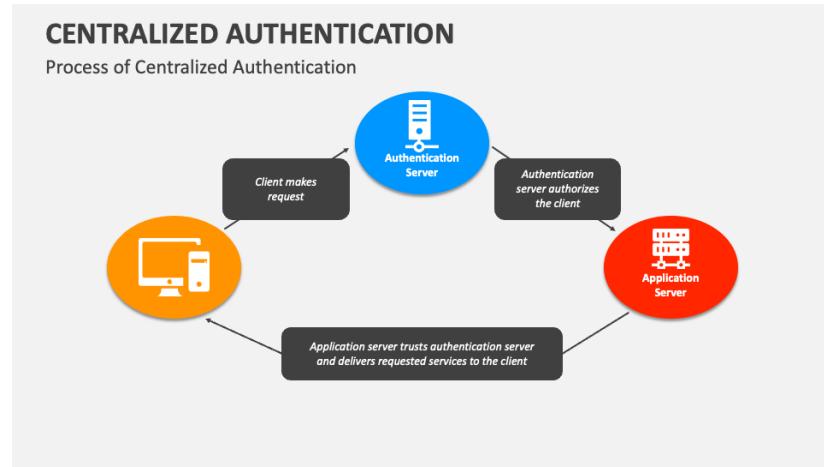
In questa sezione verranno introdotti elementi chiave per poter comprendere a pieno le tematiche trattate dal progetto e i motivi della scelta di Polygon ID

2.1 Autenticazione

L'autenticazione nel contesto della sicurezza informatica è un processo fondamentale che permette di verificare l'identità di un utente prima di concedergli l'accesso a risorse protette, come sistemi, applicazioni o dati sensibili. Questo processo è cruciale per garantire che solo gli utenti autorizzati possano accedere a informazioni o funzionalità specifiche, proteggendo così le risorse da accessi non autorizzati, frodi o abusi.

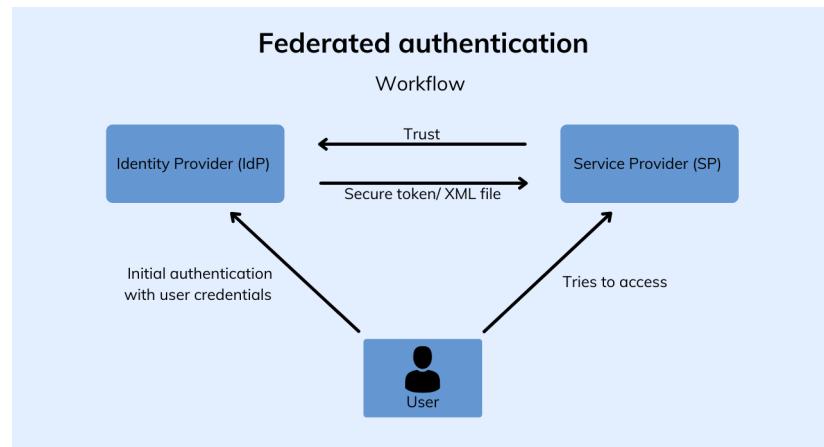
2.2 Modello centralizzato

I sistemi centralizzati gestiscono l'autenticazione tramite un server o un insieme di server che mantengono un database degli utenti e delle loro credenziali. Questo modello semplifica la gestione delle politiche di sicurezza ma introduce problemi significativi. La scalabilità diventa complessa con l'aumentare degli utenti [2], e un attacco riuscito al sistema centrale può compromettere tutte le credenziali utente. Inoltre, la centralizzazione pone seri rischi per la privacy, poiché un singolo ente ha accesso a un'ampia quantità di dati sensibili [3].

**Figura 2.1:** Processo di autenticazione centralizzata

2.3 Modello federato

Per superare i limiti dei sistemi centralizzati, sono stati introdotti i sistemi federati di autenticazione. I sistemi federati mitigano alcuni problemi dei sistemi centralizzati consentendo l'accesso a servizi diversi tramite un unico set di credenziali, grazie a un accordo di fiducia tra differenti entità (come aziende o servizi online). Questo approccio migliora l'esperienza utente riducendo il numero di password da ricordare e gestire [4]. Tuttavia, introduce complessità nella gestione della fiducia e nei protocolli di comunicazione [5]. Inoltre, se le credenziali federate vengono compromesse, l'attaccante potrebbe potenzialmente accedere a tutti i servizi connessi [6].

**Figura 2.2:** Processo di autenticazione nel modello federato

2.4 Self-Sovereign Identity

Al fine di risolvere i problemi di modello federato si è studiata l'autenticazione user-centric dove l'utente è al centro dell'autenticazione rimuovendo il bisogno per un'autorità terza. Nel modello relativo alla SSI emergono 3 principali attori:

- Issuer: Ente responsabile nell'erogare le credenziali all'utente.
- Wallet Holder: Utente che gestisce il proprio wallet contenenti le credenziali ricevute dall'issuer e generare delle presentazioni verificabili al verifier.
- Verifier: Ente che valida e verifica le credenziali dell'utente per poi erogare servizi di cui quest'ultimo è interessato.

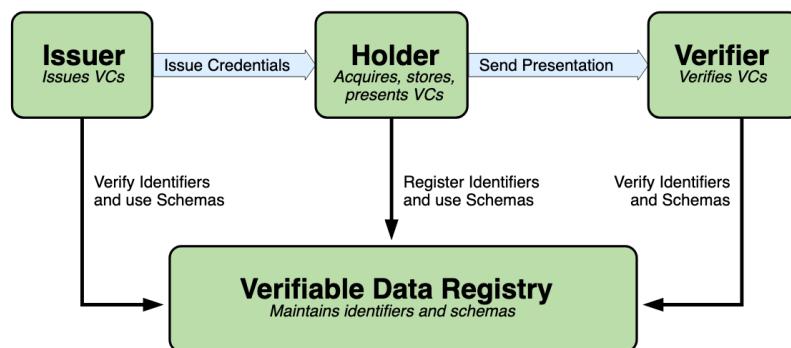


Figura 2.3: Processo di autenticazione SSI

A differenza della maggior parte dei precedenti sistemi di gestione delle identità, in cui il fornitore di servizi era al centro del modello di identità, SSI è user-centric [7]. La Self Sovereign Identity è un modello di gestione delle identità in cui l'Issuer della richiesta rilascia (almeno in parte) l'identità attestando alcuni attributi dell'utente. Questa identità è controllata dall'utente stesso. La blockchain sostituisce l'autorità di registrazione nei classici sistemi di gestione delle identità. A qualsiasi parte fidata che abbia bisogno di identificare l'utente verranno presentate le parti dell'identità controllata dall'utente che lo riguardano. Per accettare l'identità, la parte facente affidamento deve avere un rapporto di fiducia con l'Issuer della richiesta. L'identificatore è legato a un utente specifico tramite un metodo di autenticazione come la crittografia asimmetrica. Stabilendo un'accoppiata tra identificatore e chiave pubblica sulla blockchain, l'identificatore può essere verificato da chiunque legga la blockchain ponendo una sfida all'utente stesso o a un suo delegato. In questo modello la blockchain non solo funge da registro per gli identificatori di un'identità, ma contiene anche

le impronte crittografiche di tutte le rivendicazioni associate a un'identità. In questo processo non è necessario memorizzare informazioni sull'utente né presso l'Issuer né presso il Verifier, solo la fiducia tra l'Issuer e il Verifier deve essere stabilita in anticipo. Questo approccio offre numerosi vantaggi:

- **Maggiore Privacy e Controllo:** Gli utenti decidono quali informazioni condividere e con chi, riducendo la quantità di dati personali conservati dai fornitori di servizi [8].
- **Interoperabilità:** La SSI facilita l'interazione tra diversi sistemi e servizi, consentendo agli utenti di utilizzare le stesse credenziali in vari contesti [9].
- **Riduzione dei Rischi di Sicurezza:** Eliminando i punti centrali di fallimento, la SSI riduce il rischio di attacchi massicci e furti di dati. [10]
- **Efficienza e Riduzione dei Costi:** Riduce la necessità di controlli di autenticazione ripetuti e gestione delle password, semplificando i processi aziendali. [11]

La SSI rappresenta un importante passo avanti verso un ecosistema digitale più sicuro, efficiente e rispettoso della privacy degli utenti. Nonostante le sue promesse, la SSI è ancora in una fase relativamente iniziale di adozione, con sfide tecniche da superare.

2.5 Soluzioni correnti per i wallet digitali

Nell'ambito della mia ricerca di tesi, dedicata a esplorare la progettazione di una soluzione basata su Self-Sovereign Identity (SSI), ho condotto un'analisi approfondita dei wallet SSI esistenti. Questo studio mi ha permesso di comprendere i meccanismi sottostanti la loro realizzazione, identificando i protocolli crittografici adottati e valutando le funzionalità che apportano. Le soluzioni contemporanee per i wallet digitali in ecosistema SSI si basano principalmente sull'architettura Decentralized Public Key Infrastructure (DPKI).

2.5.1 Soluzioni basate su DPKI:

Le soluzioni basate su DPKI (Decentralized Public Key Infrastructure), distribuiscono la gestione della fiducia su una rete, spesso usando la tecnologia blockchain, per aumentare la sicurezza e ridurre la dipendenza da autorità centrali. Queste soluzioni possono offrire maggiore privacy e controllo agli utenti, ma possono anche affrontare sfide legate alla scalabilità. Le seguenti sono alcune delle soluzioni DPKI attualmente operative, ciascuna con le proprie caratteristiche distintive [12]:

- **Connect.Me:** È il primo portafoglio mobile SSI realizzato da Evernym sulla rete Sovrin, che impiega una blockchain pubblica autorizzata di Hyperledger Indy e prevede un'integrazione futura con la rete cheqd. Offre versatilità per vari casi d'uso, accesso a servizi pubblici e privati, e condivisione di dati d'identità riservati.[13]
- **Key Trust:** Portafoglio mobile sviluppato da Everis, gestisce credenziali digitali per diversi scopi, quali diplomi e documenti di accreditamento, con processi di convalida in tempo reale sulla rete Ethereum LACChain[14].
- **Talao:** Primo portafoglio mobile francese per l'Europa, sviluppato da Talao, segue le direttive ESSIF dell'UE e permette la raccolta di documenti vari, operando in conformità al GDPR[15].
- **Open:** Prototipo sviluppato dal governo olandese per un sistema SSI basato su IPv8, condividendo i dati in modo criptato e con autenticazione biometrica [16].
- **DIZME:** Progetto della Trust over IP Foundation che mira a connettere SSI con eIDAS, sfruttando InfoCert per fornire credenziali con valore legale [17].
- **Microsoft Authenticator:** Trasformato in un portafoglio SSI che gestisce credenziali verificate tramite Microsoft Entra, con supporto per standard SSI come W3C VC, DIDs e OpenID4VC [18].
- **Verse:** Portafoglio web SSI nato dal progetto EBSI4Austria, si concentra sull'implementazione di nodi EBSI in Austria per la gestione dei diplomi.

Soluzione	Tipo di Credenziale	Schema di Codifica	Protocollo di Scambio	Verifica
Connect.Me	Attribute-based credentials	JSON-LD, DIDComm WACI	OpenID4VCI / OpenID4VP	ZKP: ZKP-BBS+, DS: VC-LD Signature
KeyTrust	API per Reclami Verificabili	JSON, JSON-LD	OpenID4VCI / OpenID4VP	DS: VC-JWT
Talao	API per Reclami Verificabili	JSON-LD	OpenID4VCI / OpenID4VP	ZKP: ZKP-BBS+
IRMA	Attribute-based credentials	JSON	RestAPI (Protocollo IRMA)	ZKP: ZKP-CL, DS: ZKP-CL
Open	API per Reclami Verificabili	JSON	RestAPI	ZKP: ZKP-Intervallo & Prova basata sull'identità
DIZME	Attribute-based credentials	JSON-LD, DIDComm WACI	-	ZKP: ZKP-BBS+
Verse	API per Reclami Verificabili	JSON, JSON-LD	CHAPI	ZKP: VC-JWT, DS: VC-LD Signature
Microsoft Authenticator	API per Reclami Verificabili	JSON	-	ZKP: VC-JWT

Tabella 2.1: Protocolli adottati dalle soluzioni attuali.

Analizzando le diverse soluzioni per la gestione delle credenziali digitali [12], emergono alcune tendenze chiave nei protocolli e nelle specifiche tecniche adottate:

- **Tipologie di Credenziali:** Le "Verifiable Claims API" e le "Attribute-Based Credentials (ABC)" si affermano come le principali tipologie, evidenziando una preferenza per soluzioni che supportano la privacy e l'autenticazione basata su attributi con enfasi sulla verificabilità.

- **Schema di Codifica:** Gli schemi "JSON" e "JSON-LD" sono prevalenti, indicando una preferenza per formati leggeri e interoperabili, con "JSON-LD" che supporta semantica avanzata per la decentralizzazione.
- **Protocolli di Scambio:** "OpenID4VCI/OpenID4VP" emerge come standard di scambio dominante, allineandosi agli standard aperti per l'integrazione nell'ecosistema OpenID esistente.
- **Verifica e Prove a Conoscenza Zero:** La tecnologia "ZKP-BBS+" è ampiamente utilizzata, sottolineando l'importanza delle tecniche di minimizzazione dei dati per la privacy degli utenti.
- **Revoca delle Credenziali:** Meccanismi come "AV" (Accumulator-Based Revocation) e "Credential Status List" sono comuni, enfatizzando la necessità di gestire il ciclo di vita delle credenziali.
- **Firme Digitali e Sicurezza:** "VC-LD Signature" e "VC-JWT" vengono spesso impiegati per garantire l'integrità e l'autenticità delle credenziali digitali.

Queste osservazioni riflettono una tendenza verso l'adozione di standard aperti e tecnologie che garantiscono sicurezza e privacy.

2.6 Perchè Polygon ID?

Nell'ambito del lavoro proposto e dei trend tecnologici correnti, è stata adottata come solution provider Polygon ID in virtù della sua caratteristica chiave, ovvero la Zero-Knowledge-Proof, consentendo agli utenti di dimostrare la propria identità senza dover esporre le informazioni private. Ciò garantisce sia libertà di espressione soddisfando requisiti non funzionali legati alla tutela della privacy. Polygon ID garantisce anche l'interoperabilità utilizzando tecnologie neutre quali JSON-LD.

Polygon ID offre:

- Soluzione alla privacy per la verifica delle credenziali, grazie all'ausilio dei protocolli di Zero Knowledge Proofs.
- Comunicazione non diretta tra l'issuer e verifier.
- Verifica delle credenziali on-chain e off-chain, facilitandone il deploy per le piattaforme Web2 o Web3.

- Supporta l'erogazione massiva delle credenziali da parte dell'issur con un efficiente storage on-chain con costi di transazione irrisori.
- Adattabilità ad un ampio range di casi d'uso offrendo:
 - Protezione contro gli attacchi di Sybil per la privacy: Ciò viene realizzato utilizzando tecniche di nullificazione che assicurano che un'identità (essere umano) possa compiere un'azione solo una volta. Questo diventa una caratteristica chiave per casi d'uso specifici come i processi di voto.
 - Prove non riutilizzabili: garantiscono al Verifier che la prova presentata dall'utente non sia una copia di una prova generata da un'altra identità e consentono al Verifier di verificarne la validità.
 - Protezione contro il tracciamento delle impronte digitali.
- Un ecosistema esteso, molti enti lo stanno utilizzando.

Vantaggi per lo sviluppo

- **Polygon ID Wallet SDK**

Consente agli sviluppatori di creare wallet o integrare le soluzioni offerte da polygon. Come implementazione di riferimento di un portafoglio identità.

- **Polygon ID Issuer Node**

L'"issuer-node" è un'applicazione che gli Issuer possono ospitare autonomamente per mantenere il controllo dei propri dati. Espone metodi API che possono essere utilizzati dagli Issuer per rilasciare credenziali verificabili e fornisce un'interfaccia utente.

- **Polygon ID Verifier SDK**

Consente agli sviluppatori, ad esempio dApps, di impostare criteri di verifica e formulare interrogazioni per le credenziali degli utenti.

- **Polygon ID JS SDK**

Librerie JavaScript per creare applicazioni client come estensioni del browser, consentendo agli utenti di rilasciare credenziali su se stessi, di memorizzare credenziali e una chiave, e quindi di generare prove di possesso di tali credenziali.

- **Schema Builder**

Lo schema builder consente agli sviluppatori di creare schemi e renderli pubblici,

contribuendo alla standardizzazione e interoperabilità delle credenziali nell'ecosistema complessivo.

- **Query Builder**

Il Query Builder elimina la necessità per gli sviluppatori di avere una conoscenza approfondita del linguaggio zkQuery, fornendo un modo più intuitivo e user-friendly per creare interrogazioni.

Vantaggi nei costi di infrastruttura

Il nucleo della tecnologia di Polygon ID è disponibile con una licenza open source, gratuita. I costi nell'utilizzo di Polygon ID si verificano durante la scrittura sulla blockchain, che attualmente avviene solo quando un Issuer aggiorna lo stato nella blockchain o quando un Verifier on-chain esegue una verifica. Per tutte le altre interazioni, che avvengono in modalità di sola lettura sulla blockchain, i costi sono nulli o significativamente ridotti.

	Ethereum	Polygon POS
Transizione di stato	45	0,20
Verifica on-chain	18	0,08

Tabella 2.2: Costi delle operazioni su Ethereum e Polygon POS

Dettagli dei prezzi utilizzati per i calcoli:

	Ethereum	Polygon POS
Prezzo del gas	18 gwei	102 gwei
Prezzo del token	1.658 (Ether)	1,24 (MATIC)

Tabella 2.3: Prezzi del gas e delle criptovalute al 27 febbraio 2023

Dove le credenziali sono preservate?

Gli attori conservano le informazioni sulle credenziali nel modo seguente:

1. **Issuer:** l'Issuer può decidere dove preservare le credenziali. Di solito, gli issuer conservano le credenziali in un database locale. Tuttavia, l'issuer potrebbe anche scegliere di conservare le credenziali in altri repository, come in un database pubblico.
2. **User Wallet:** gli utenti (Identity Holders) preservano le credenziali nel loro portafoglio. Il portafoglio determina se le credenziali sono conservate localmente o in un vault esterno.
3. **Verifier:** i Verifier (dApp) preservano le informazioni sulle credenziali solo se viene utilizzata la cosiddetta "Selective Disclosure", in cui il Wallet Holder decide di condividere informazioni specifiche selezionate con il Verifier. Nel caso di un Verifier on-chain, le informazioni divulgate (o una versione hash di esse) saranno conservate on-chain.

È importante sottolineare che le credenziali complete NON vengono conservate sulla blockchain.

Che blockchain viene impiegata?

Polygon ID può essere eseguito su qualsiasi catena compatibile con l'EVM (Ethereum Virtual Machine). Attualmente, i contratti intelligenti necessari sono stati distribuiti sulla rete di test di Polygon (Mumbai) e sulla rete principale di Polygon PoS, ma potrebbero essere distribuiti su qualsiasi altra catena compatibile con l'EVM. Nell'ambito del lavoro proposto, è stato utilizzato l'ambiente di test di Mumbai per testare e sviluppare le funzionalità di Polygon ID.

CAPITOLO 3

Polygon ID

Polygon ID è un solution provider per la self sovereign identity, consente alle organizzazioni di rilasciare credenziali verificabili per utenti e altre organizzazioni ma al contempo fornisce agli enti la possibilità di verificare tali affermazioni attraverso una sofisticata suite di strumenti progettata per ciascun partecipante nell'ecosistema SSI. Polygon ID si caratterizza tra i tanti solution provider per l'utilizzo delle Zero Knowledge Proof che consentono agli utenti di provare la propria identità rispettando la privacy e evitando di trapelare le informazioni all'ente Verifier [19]. L'architettura del framework è composta da 3 entità:

1. Identity holder: un entità che detiene dei claims all'interno del suo wallet, oltre a questo genera delle zero knowledge proof a partire dai Verifiable Claims che sono presentate al Verifier che si preoccuperà dell'autenticità della prova.
2. Verifier: Il Verifier verifica le prove presentate dall'holder, richiede all'holder di generare le prove che possiede nel suo wallet. Nell'atto della verifica performa alcuni controlli, ovvero capire se la virtual claim è stata firmata/rilasciata da un determinato Issuer o che appunto rispetta alcuni requisiti definiti da lui. Il processo di verifica può avvenire sia on-chain che off-chain.
3. Issuer: L'Issuer è colui che rilascia le credenziali, sono firmate crittograficamente ed ogni credenziale proviene dall'Issuer.

3.1 Componenti dell’infrastructure layer

l’Issuer comprende le seguenti componenti:

- Vault: Il vault è basato su una soluzione preesistente chiamata Hashicorp, è utilizzata per proteggere i dati sensitivi quali le chiavi private degli utenti, i dati che vengono conservati in storage sono criptati con l’algoritmo di Baby JubJub.
- Redis: fa da storage per gli schemi, gli schemi vengono fetchati dagli ipfs e mantenuti da redis. Quando dovrà erogare la credenziale ad un holder sarà disponibile all’utilizzo.
- DB: utilizzato come fonte dati per il Nodo Issuer. Nell’ implementazione dell’Issuer node, viene utilizzato Postgres come database. È qui che vengono conservati tutti i dati relativi alle credenziali emesse.

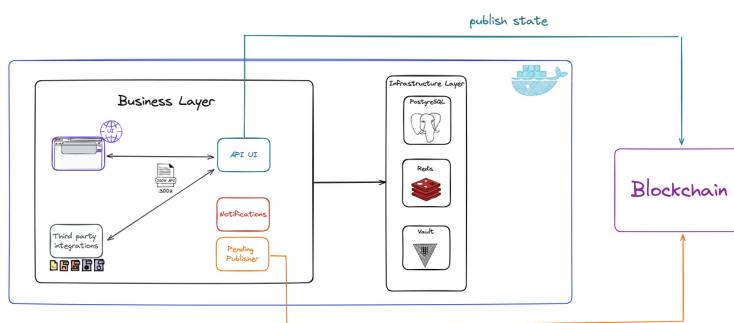


Figura 3.1: Architettura nodo Issuer

3.1.1 Schemas

Un *Schema Type* codifica la struttura di una particolare Credenziale Verificabile (VC), definendo il tipo, i campi che devono essere inclusi all’interno del VC e una descrizione per questi campi.

Gli schemi sono un componente cruciale che consente l’uso interoperabile di VC tra servizi diversi. Semplicemente analizzando uno schema, qualsiasi programma può interpretare il contenuto di una Credenziale Verificabile senza dover interagire con l’Ente Emettitore.

Per emettere Credenziali Verificabili, è possibile fare riferimento agli schemi esistenti. Se quelli esistenti non sono in grado di descrivere il tipo di dati desiderato, è necessario creare il proprio *Schema Type*.

Uno *Schema Type* è composto da due documenti:

1. Il Contesto JSON-LD, che contiene una descrizione del tipo e dei suoi campi. Di seguito è riportato un esempio di Contesto JSON-LD per lo *Schema Type* "CodingExperienceCredential".
2. Lo Schema JSON, che contiene le regole di convalida per il Nodo Emettitore. Di seguito è riportato un esempio di Schema JSON per lo *Schema Type* "CodingExperienceCredential".

3.1.2 Rilascio

Il possessore delle credenziali può provare ad un verifier che la propria credenziale è reale, senza rilevare alcuna informazione al di là della validità della credenziale stessa sfruttando la crittografia zkSNARK. Il verifier può richiedere delle prove alle informazioni contenute nelle credenziali e può riceverle per poi garantirne la correttezza evitando di accedere direttamente ed entrare in possesso di informazioni dell’utente.

Ci sono principalmente 2 modi in cui queste credenziali possono essere erogate:

- Merkle Tree Root Issuence.
- Baby Jub Jub Key Signature.

Merkle Tree Root Issuence

L’Issuer aggiorna lo stato della propria identità sulla Blockchain, per far sì che questo avvenga l’hash del suo albero di Merkle deve essere pubblico, poiché la funzione di transizione dello stato dell’identità possa essere eseguita. Si predilige questo metodo quando i contratti intelligenti fanno da Issuer. Il costo stimato per chiamare questa funzione è di circa 2 milioni di gas in media (0,36 MATIC nella rete principale di Polygon PoS a giugno 2023).

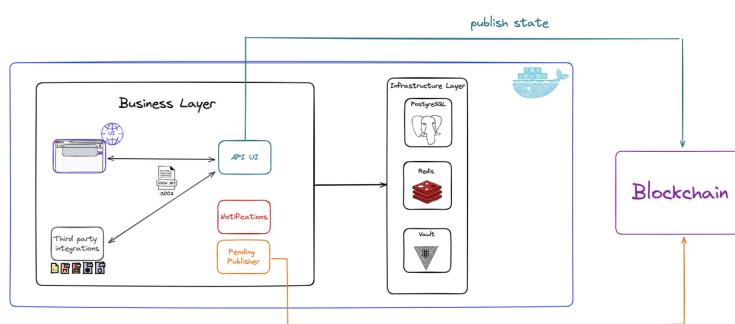


Figura 3.2: Processo di erogazione di credenziali con Merkle Tree Root

Baby Jub Jub Key Signature

La credenziale non viene aggiunta all’albero di Merkle dell’Issuer; al contrario, si utilizza una firma Baby Jubjub (BJJ) che viene poi verificata durante la presentazione. Con questo metodo, l’Issuer può rilasciare un gran numero di credenziali senza la necessità di spendere gas per emetterle.

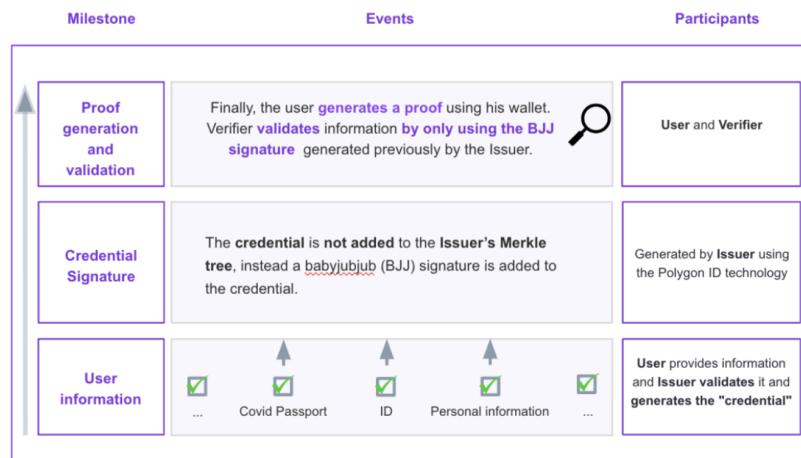


Figura 3.3: Processo di erogazione delle credenziali con babyjubjub

3.1.3 Verifica

Un Verifier è una piattaforma web2 o web3 che desidera autenticare gli utenti in base alle loro credenziali.

I Verifier possono configurare query basate sulle credenziali esistenti detenute dagli utenti ed erogate dagli Issuer. Una Query racchiude i criteri che un utente deve soddisfare per autenticarsi, come ad esempio "deve avere più di 18 anni" o "deve essere membro di XYZ DAO". Polygon ID offre agli utenti un’esperienza di autenticazione senza soluzione di continuità, personalizzata e basata sulla privacy.

La richiesta del Verifier è progettata utilizzando un query based language ed esposta all’utente sottoforma di Qr-Code. L’utente scannerizza il codice QR e il wallet successivamente genererà la prova necessaria al processo di verifica.

Nel processo di verifica delle credenziali non vi è alcuna interazione tra il Verifier e l’Issuer. Come parte della Query, il Verifier include gli identificatori degli Issuer fidati (Selective Disclosuer). Ad esempio, un Verifier dovrebbe aggiungere XYZ DAO come unico Issuer fidato quando verifica che un individuo sia membro di XYZ DAO. XYZ DAO non ha bisogno di accettare né interagire con il Verifier.

Alla fine del processo, il Verifier ottiene una prova crittografica che dimostra che l’utente soddisfa i criteri imposti dalla query, mentre l’utente condivide solo la quantità minima possibile di dati richiesta per l’interazione.

Verifier SDK

La SDK offre agli utenti un’esperienza di autenticazione completamente personalizzabile e basata sulla privacy. Il processo di verifica delle informazioni degli utenti in base alle loro credenziali può avvenire on-chain tramite un contratto intelligente o off-chain. Entrambi i processi coinvolgono lo stesso livello di privacy dell’utente e lo stesso grado di personalizzazione della query. La prova generata su mobile è la stessa per entrambi i casi; l’unica differenza è nel processo di verifica. La verifica on-chain avviene programmaticamente all’interno di un contratto intelligente. La verifica off-chain avviene all’interno di uno script che deve essere configurato dall’applicazione del Verifier (sia su un server o sul lato client):

1. Verifica off-chain fornisce tutti gli elementi per creare una Query personalizzata, configurare un Verifier e generare un codice QR (o deeplink) sul lato client per richiedere la prova dall’utente.
2. Verifica on-chain consente alle Dapp di verificare le credenziali degli utenti all’interno di un Contratto Intelligente utilizzando la crittografia a prova zero-conoscenza.

Verifica Off-chain

Ogni interazione off-chain tra un Verifier e il portafoglio di un utente segue questo flusso di lavoro:

1. Una web app rende disponibile all’utente una richiesta all’utente sottoforma di QR-code.
2. L’utente scansione il QR code utilizzato il suo wallet.
3. Il wallet fa il fetch dello stato di revoca delle credenziali di cui il verifier è interessato.
4. Il wallet genera un ZK proof sulla base delle richieste del verifier, la prova mostra al verifier anche che le credenziali non siano state revocate dall’Issuer.
5. L’utente spedisce queste informazioni al Verifier.
6. Il verifier utilizza le "verification API"
7. Il verifier controlla lo stato dell’Issuer e lo stato del wallet-holder per capire se le credenziali sono state revocate e la validità dei soggetti coinvolti.
8. Se la verifica ha buonfine, il verifier garantisce l’accesso all’utente e opzionalmente attivando meccanismi di logica custom.

Questa sezione fornisce tutti gli elementi necessari per integrare la verifica off-chain con Polygon ID.

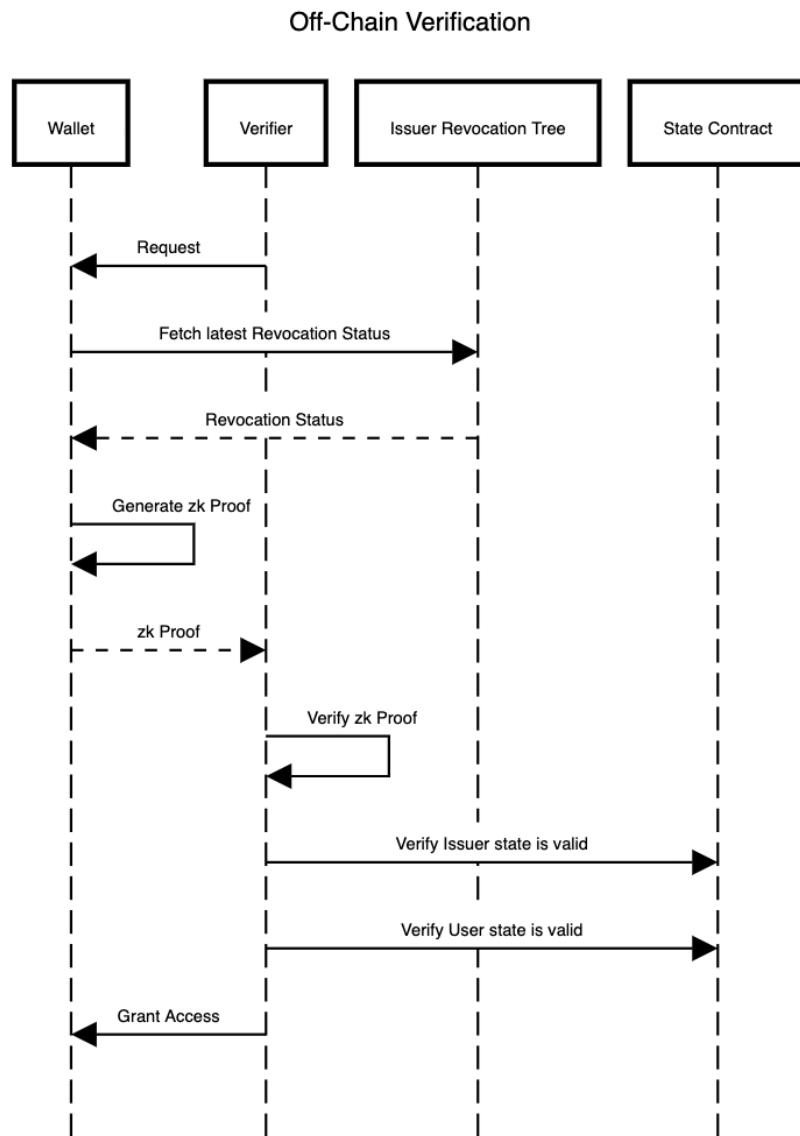


Figura 3.4: Sequence diagram relativo al processo di verifica off-chain

Supponiamo che la richiesta sia: "Hai più di 18 anni?". Il Verifier non ottiene mai accesso alle credenziali dell’utente. Invece, il Verifier riceve una prova crittografica che, alla verifica, fornisce una risposta "sì" o "no" alla domanda precedente.

ZK Query Language

I circuiti *Atomic Query Signature V2* e *Atomic Query MTP V2* sono stati progettati come circuiti generici per eseguire la verifica a prova zero-conoscenza (ZK) basata sulle affermazioni degli utenti.

Il Linguaggio di Query opera su questi circuiti per fornire un modo semplice per gli sviluppatori di progettare requisiti di autenticazione personalizzati per le credenziali di qualcuno. Finché l’utente possiede una credenziale di un tipo specifico, il Verifier può progettare una query basata su 6 operatori, ad esempio:

1. Deve essere un essere umano verificato per votare per una proposta specifica di un DAO - uguale (operatore 1).
2. Deve essere nato prima del 2000-01-01 per accedere a un sito web specifico - minore di (operatore 2).
3. Deve avere uno stipendio mensile superiore a 1000 per ottenere un prestito - maggiore di (operatore 3).
4. Deve essere un amministratore o un hacker di un DAO per accedere a una piattaforma - in (operatore 4).
5. Non deve essere residente in un paese presente nell’elenco dei paesi in lista nera per operare su uno scambio - not-in (operatore 5).
6. Non deve essere residente in un paese specifico - non-uguale (operatore 6).

La query è progettata dal Verifier e presentata all’utente tramite un codice QR (o deep-linking). A partire dalla prova generata dall’utente in risposta alla query, il Verifier è facilmente in grado di verificare se la query è soddisfatta o meno.

Il Linguaggio di Query segue le stesse regole, che la verifica sia implementata on-chain o off-chain, anche se la sintassi per definirle è leggermente diversa.

3.2 Wallet

Un portafoglio digitale è un’applicazione in grado di contenere e gestire le credenziali degli utenti. Basandosi sui principi dell’Identità Auto-Sovrana (SSI) e della crittografia, un portafoglio aiuta il suo Titolare a condividere dati con gli altri senza esporre ulteriori informazioni private sensibili. Solo il titolare del portafoglio ha il diritto di decidere quali informazioni condividere e cosa deve rimanere privato.

Polygon ID offre modi interessanti per la progettazione di un wallet.

3.2.1 Wallet SDK

Il Wallet SDK è uno SDK basato su Flutter che può essere utilizzato dagli sviluppatori per costruire applicazioni o integrare in modo trasparente le funzionalità del portafoglio con le loro app esistenti. Inizia con il Wallet SDK qui.

Questi sono i moduli (SDK) che mette a disposizione:

- polygonid-flutter-sdk [Plugin Dart]
- polygonid-ios-wrapper-sdk [Libreria Swift (Framework)]
- polygonid-android-wrapper-sdk [Libreria Kotlin (.aar)]
- Polygonid-react-native-wrapper-sdk [Libreria RN]

3.3 Iden3com

Polygon ID si appoggia al protocollo Iden3comm [20], è il responsabile che definisce l’insieme di regole e la sintassi dei dati che devono essere comunicati durante l’interazione con un Issuer e un Verifier. Il protocollo stabilisce le basi per la semantica e la sincronizzazione coinvolte durante la comunicazione tra due parti.

Il protocollo Iden3comm definisce la struttura dei messaggi, delle richieste e delle risposte che sono essenziali per lo scambio di informazioni tra il Titolare dell’Identità, l’Issuer e il Verifier. Gestisce i seguenti tipi di messaggi, richieste e risposte tra le entità coinvolte nel processo di comunicazione:

- Messaggi di Richiesta di Autorizzazione e Corpo della Risposta
- Messaggi di Risposta di Autorizzazione e Corpo della Risposta
- Richieste e Risposte di Prova a Conoscenza Zero
- Messaggi di Richiesta di Credenziali per l’Emissione e il Recupero di Credenziali
- Messaggi di Risposta di Credenziali per l’Emissione di Credenziali
- Messaggi Iden3 per Richieste di Recupero di Messaggi
- Messaggi di Richiesta dello Stato di Revoca
- Richieste di Registrazione del Dispositivo

3.3.1 Claims

Definizione dei claim

Un claim è un'affermazione fatta da un'identità su un'altra identità o su se stessa. Ogni dichiarazione è composta da due parti: la parte dell'indice e la parte del valore. I claim sono memorizzate sulle foglie di un albero di Merkle (MT). L'indice viene hashato ed è utilizzato per determinare la posizione della foglia dove verrà memorizzato il valore della dichiarazione. Una funzione speciale di validazione della transizione può essere utilizzata per limitare come le foglie sono memorizzate sull'albero di Merkle, ovvero rendere l'MT solo appendibile (le foglie possono solo essere aggiunte e non possono essere aggiornate o eliminate).

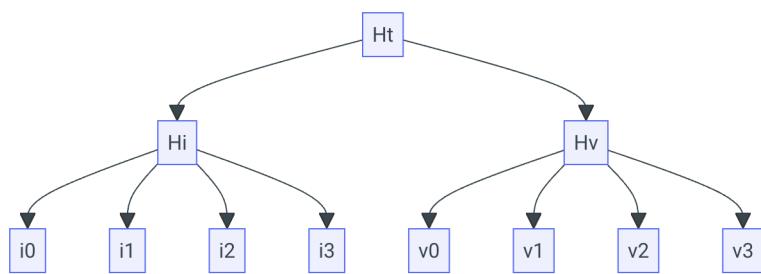


Figura 3.5: Struttura dei claim

Proprietà dei claims

- È impossibile generare una prova di un claim a nome di un'identità senza il suo consenso.
- i claims possono essere revocati.
- i claims possono essere aggiornati creando nuove versioni. Quando un claim viene revocato, non possono essere fatte ulteriori versioni. I claim possono essere impostati per essere aggiornabili o meno con un flag.
- I claim possono essere verificati. Ciò significa che è possibile dimostrare critograficamente che un claim è:
 - Emanato da un'identità specifica.
 - Non revocato.
 - Capire se è aggiornabile dall'ultima versione.

Ci sono due tipi di claim riguardo alla destinazione:

1. I claim relativi al possedimento di alcune proprietà relative all'identità. Esempio: Chiave Operativa, Indirizzo Ethereum, ecc.
2. I claim relativi ad altre proprietà dell'identità:
 - (a) (Un'altra) Identità ha una Proprietà: Relazione direzionale tra un'identità e una proprietà.
 - (b) Claim relativi ad un'altra: Relazione direzionale tra una proprietà e un'identità: identità memorizzata.

Affidabilità del Contenuto di una Claim

La correttezza di ciò che viene detto in un claim non è verificabile da un protocollo poiché ogni identità è libera di dichiarare ciò che vuole. Poiché è possibile sapere quale identità ha emesso il claim, la fiducia/reputazione che l'Issuer ha può influenzare la sua credibilità.

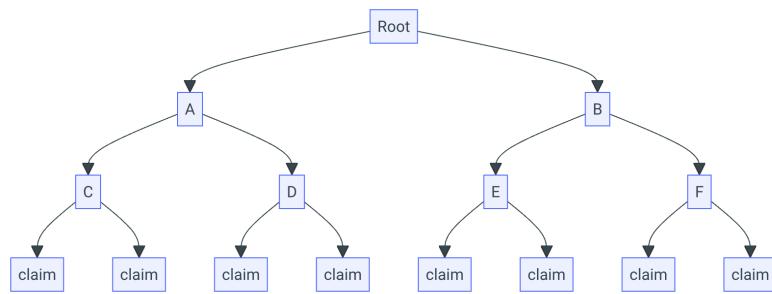
Tuttavia, il protocollo può garantire l'esclusività: non possono esistere due claim con lo stesso indice. Quindi è impossibile che un'identità dichiari che una proprietà (parte dell'indice del claim) sia collegata a due identità diverse (parte del valore del claim) allo stesso tempo.

3.3.2 Identità

Un'identità è caratterizzata dalle dichiarazioni che l'identità ha emesso e dalle dichiarazioni che l'identità ha ricevuto da altre identità. In altre parole, un'identità è costruita da ciò che l'identità ha detto e da ciò che gli altri hanno attestato sull'identità.

Ogni claim emesso da un'identità può essere crittograficamente provato e verificato. Per realizzare ciò (e altre proprietà), le identità sono costruite tramite alberi di Merkle, dove i claim sono posizionate nelle foglie, e la radice dell'albero è pubblicata sulla blockchain. Con questa costruzione, le identità possono emettere, aggiornare e revocare le dichiarazioni.

La costruzione del protocollo è progettata per abilitare zero-knowledge features. Ciò significa che le identità hanno la capacità di dimostrare il possesso delle proprietà dei claim in fase di emissione o di verificare che una particolare claim non sia stato revocato.

**Figura 3.6:** Struttura di un identità

3.3.3 Genesis ID

Descrizione

Ogni identità ha un identificatore unico determinato dallo stato iniziale dell’identità (hash delle radici del suo albero di Merkle). Questo identificatore è chiamato Genesis ID, sotto il quale vengono posizionate i claims iniziali (quelle contenute nello stato iniziale dell’identità).

Per l’implementazione iniziale del protocollo, l’albero Genesis dei claim conterrà almeno un claim di autorizzazione della chiave operativa che consente di operare a nome dell’identità. Finché un’identità non aggiunge, aggiorna o revoca claim dopo lo stato Genesis, il suo stato d’identità non deve essere pubblicato sulla blockchain, e le dichiarazioni Genesis possono essere verificate direttamente contro il Genesis ID. Questo perché il Genesis ID (calcolato con lo stato dell’identità come un hash della radice dell’albero delle dichiarazioni Genesis, una radice dell’albero di revoca vuota e una radice dell’albero delle radici vuota) è costruito dalla radice di Merkle che contiene quelle dichiarazioni.

Formato dell’identificatore

Un identificatore è determinato dal tipo di identità e dallo stato d’identità Genesis (chiamato anche Genesis ID). Questo è costruito creando un albero di Merkle che contiene i claim dello stato iniziale, calcolando la sua radice e facendo l’hash insieme a una radice dell’albero di revoca vuota e una radice dell’albero delle radici vuota. Composto con i primi 27 byte di questo risultato più 2 byte all’inizio (per specificare il tipo di identità) e 2 byte alla fine (per il checksum). Quindi, in totale, un identificatore è un array di byte di 31 byte, codificato in base58.

Tipo di identità

Un tipo di identità specifica le specifiche che un'identità segue (come la funzione hash utilizzata dall'identità). In questo modo, quando la funzione hash cambia, cambieranno anche gli identifieri delle identità, permettendoci di identificare il tipo di identità.

Struttura dell'identificatore ID (genesis): Base58 [tipo | stato_genesis | checksum]

- **tipo di identità:** 2 byte che specificano il tipo
- **stato_genesis:** I primi 27 byte dallo stato dell'identità (utilizzando l'albero di Merkle delle dichiarazioni Genesis)
- **checksum:** Somma (con overflow) di tutti i byte dell'ID Little Endian 16 bit ([tipo | stato_genesis])

3.3.4 Stato dell'Identità

Gli stati d'identità sono pubblicati sulla blockchain sotto l'identificatore, ancorando lo stato dell'identità con il timestamp in cui viene pubblicato. In questo modo, i claims dell'identità possono essere provate contro lo stato d'identità ancorato a un certo timestamp. Per passare da uno stato all'altro, le identità seguono le funzioni di transizione.

Gli stati d'identità possono essere pubblicati sulla blockchain in uno dei due modi: eseguendo direttamente la transazione per pubblicare la radice o indirettamente utilizzando un Relay.

Lo stato Genesis è lo stato iniziale di qualsiasi identità e non deve essere pubblicato sulla blockchain, poiché i claims sotto di esso possono essere verificate contro l'identificatore stesso (che contiene quello stato d'identità).

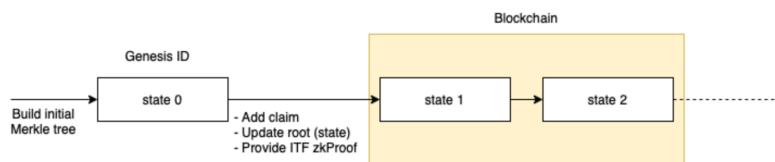


Figura 3.7: Gestione dello stato di un identità

3.3.5 Funzione di Transizione dello Stato d'Identità

La ITF (*Identity State Transition Function*) viene verificata ogni volta che uno stato viene aggiornato. Questo assicura che l'identità segua il protocollo durante l'aggiornamento.

Un albero di Merkle d'Identità è un albero binario sparso che consente solo l'aggiunta delle foglie (nessuna modifica o cancellazione). Aggiungere nuove dichiarazioni, aggiornarle attraverso versioni e revocarle deve essere fatto secondo l'ITF. Per garantire ciò, si utilizzano le cosiddette zero-knowledge proof in modo che quando un'identità sta pubblicando un nuovo stato al contratto intelligente, invia anche una zero-knowledge proof (π), dimostrando che il ϕ è soddisfatto seguendo l'ITF. In questo modo, tutti gli stati d'identità pubblicati sulla blockchain sono validati per essere conformi al protocollo.

3.3.6 Interazioni tra identità e claims

Aggiornamento dello Stato dell'Identità

L'Aggiornamento dello stato dell'identità è la procedura utilizzata per aggiornare le informazioni su ciò che l'identità ha dichiarato. L'aggiornamento ricade principalmente in questi 3 casi d'uso:

1. Aggiungere un claim.
2. Aggiornare un claim (incrementando la versione e cambiando la parte del valore della dichiarazione).
3. Revocare una claim.

Definizioni

- **IdState:** Stato dell'Identità
- **CIT:** Albero delle Dichiarazioni
- **CIR:** Radice dell'Albero delle Dichiarazioni
- **ReT:** Albero di Revoca
- **ReR:** Radice dell'Albero di Revoca
- **RoT:** Albero delle Radici
- **RoR:** Radice dell'Albero delle Radici

Lo **IdState** (Stato dell'Identità) è calcolato concatenando le radici dei tre alberi utente: $IdState = H(CIR\|ReR\|RoR)$ dove H è la funzione di hash definita dal Tipo di Identità (ad esempio, Poseidon). Tutti gli alberi sono SMT (alberi di Merkle sparsi) e usano la funzione di hash definita dal Tipo di Identità.

Diagramma dello Stato dell'Identità per Identità Diretta

Come visto nel diagramma, solo lo **IdState** è memorizzato sulla blockchain. Per risparmiare byte memorizzati sulla blockchain, è desiderabile che solo un "hash" rappresentante dello stato corrente dell'Identità sia memorizzato nel contratto intelligente. Questo "hash" unico è lo **IdState** (Stato dell'Identità), che è collegato a un timestamp e a un blocco sulla blockchain.

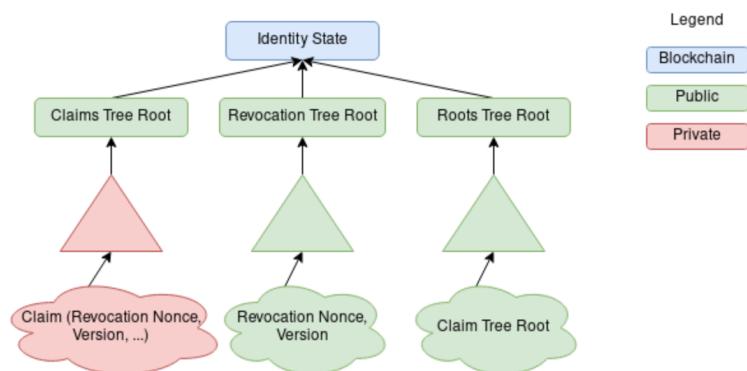


Figura 3.8: Diagramma stato di identità

Tutti i dati pubblici devono essere resi disponibili per qualsiasi detentore in modo che possano costruire prove "fresche" dell'albero di Merkle sia per ReT che per RoT. Questo permette al detentore di:

- Provare la recente non-revoca / versione corrente senza interazione con l'Issuer.
- Nascondere un particolare CLR da tutti gli altri CLR per evitare che un Issuer scopra un claim nascosto dietro una prova Zero Knowledge proof. A questo scopo, CLR è aggiunto a RoR.

Il luogo e il metodo per accedere ai dati pubblicamente disponibili sono specificati nel contratto intelligente dello Stato delle Identità. Genericamente si sceglie tra:

- **IPFS**, aggiungendo un link a un indirizzo IPNS (esempio: ipfs://ipns/Qm...), che contiene una struttura standardizzata dei dati.

- **HTTPS**, aggiungendo un link a un endpoint HTTPS (esempio: [https://kyc.iden3.io/api/v1/public-state/...](https://kyc.iden3.io/api/v1/public-state/)),
che offre i dati seguendo una API standardizzata.

Pubblicare claims

Il primo passo nella pubblicazione di un claim coinvolge l'aggiunta di una nuova foglia al CIT, che aggiorna il ClR dell'identità. I claim possono essere optionalmente pubblicate in lotti, aggiungendo più di una foglia al CIT in una singola transazione. Dopo che il CIT è stato aggiornato, l'identità deve seguire un aggiornamento dello stato dell'identità affinché tutti siano in grado di verificare i claim appena aggiunti. Questo comporta l'aggiunta del nuovo ClR al RoT, che a sua volta aggiornerà il RoR. Dopo di che, il nuovo IdState è calcolato e attraverso una transazione è aggiornato nel contratto intelligente dello Stato delle Identità (d'ora in poi, denominato "il contratto intelligente") sulla blockchain. Una volta che l'IdState aggiornato è nel contratto intelligente, chiunque può verificare la validità dei claim appena aggiunti.

La procedura per aggiornare l'IdState nel contratto intelligente può essere realizzata con i seguenti criteri:

- **Scalabilità scarsa** (no batch), **buona privacy** e **correttezza**: L'identità carica il nuovo IdState nel contratto intelligente con la prova di una transizione corretta dall'IdState vecchio a quello nuovo. Solo un claim viene aggiunta al CIT nella transizione.
- **Buona scalabilità** (batch), **buona privacy** e **correttezza**: Come prima, ma molte dichiarazioni vengono aggiunte (batch) nella transizione (con una singola prova per tutte le dichiarazioni appena aggiunte).
- **Buona scalabilità** (batch), **buona privacy** ma **senza correttezza**: L'identità carica il nuovo IdState nel contratto intelligente, senza dimostrare la correttezza nella transizione.

I criteri per la correttezza sono i seguenti:

- La revoca di un claim non può essere annullata.
- I claim aggiornabili vengono aggiornate solo con versioni crescenti, e solo una versione è valida alla volta.

Avere o non avere la garanzia della correttezza è specificato nel Tipo di Identità in modo che qualsiasi Verifier sia a conoscenza delle garanzie fornite dal protocollo per i claim dell'Issuer.

Nota

La **buona scalabilità** si riferisce al processo di verifica e ai costi relativi al contratto intelligente. Il batching con zkSNARKs può avere un carico computazionale elevato sul dimostrante(wallet).

Albero di Revoca

A volte, è desiderabile per un'identità invalidare una dichiarazione fatta tramite un claim. Per i claim regolari, ciò comporta la revoca (un processo idealmente irreversibile) e consente a qualsiasi Verifier di essere a conoscenza del fatto che un claim già pubblicata è reso invalido dall'identità dell'Issuer. Allo stesso modo, per i claim aggiornabili, deve esserci un meccanismo per invalidare le vecchie versioni quando ne viene pubblicata una nuova. Poiché confermare la validità attuale di un claim è un processo parallelo alla conferma che un claim è stata pubblicata in qualche momento, il processo di "validazione attuale" può essere separato. Separare questi due processi consente un design in cui il CIT (Albero dei claim) rimane privato, ma le informazioni sulla revoca/versione sono pubbliche, consentendo a un detentore di generare una prova fresca della "validità attuale" senza richiedere l'accesso al CIT privato. Per ottenere ciò, ogni Identità ha un CIT (Albero delle Dichiarazioni) e un ReT (Albero di Revoca) separato. Mentre l'Albero dei claim sarebbe privato e solo la radice pubblica, l'albero di revoca sarebbe completamente pubblico. Le radici di entrambi gli alberi (CIT e ReT) sono collegate tramite l'IdState (Stato dell'Identità) che è pubblicato nel contratto intelligente. L'Albero di Revoca potrebbe essere pubblicato in IPFS o altri sistemi di archiviazione pubblici.

Dimostrare che un claim è valido (e quindi non revocata/aggiornata) consiste in due prove:

1. Provare che i claim è stato emessa in un certo momento t (questa prova è generata una volta dall'Issuer e utilizza un IdState-CIR al tempo t memorizzato nel contratto intelligente).
2. Provare che i claim non è stato revocato/aggiornato di recente (questa prova è generata dal detentore con un recente ReR (Radice dell'Albero di Revoca) interrogando il ReT pubblico (Albero di Revoca) e verificata contro un IdState recente).

Revocare i claim

Per evitare di rivelare qualcosa sul contenuto dei claim nel ReT, i claim contengono un nonce di revoca venendo poi aggiunto come foglia nel ReT per revocare l'effettivo claim.

Per impedire l'annullamento della revoca di un claim, il ReT deve seguire alcune regole di transizione come il CIT, applicate da una prova ZK (per efficienza di spazio e verifica).

Oltre alla procedura di revoca, esiste un metodo per definire la validità di un claim basata sulla scadenza, impostando esplicitamente una data di scadenza nella dei claim. Revoca e Scadenza sono metodi compatibili per invalidare i claim.

Aggiornare claim

Per aggiornare un claim, una nuovo claim viene aggiunta al CIT con un valore di versione incrementato nella posizione dell'index del claim (la versione precedente claim non viene toccato). Quindi, viene aggiunta una foglia al ReT contenente il nonce di revoca e la versione più recente non valida (cioè, tutte le dichiarazioni con quel nonce e versione pari o inferiore a quella nella foglia sono invalide). Ciò significa che quando un claim viene aggiornato, lo stesso nonce di revoca viene utilizzato nel claim.

Per impedire il downgrade della versione di un claim e obbligare ad avere solo un claim aggiornabile valido alla volta, il ReT deve seguire le regole di transizione (come fa il CIT) applicate da una prova ZK (per efficienza di spazio e verifica).

Aggiornare e revocare sono metodi compatibili per invalidare le dichiarazioni: un claim aggiornabile può essere revocato, il che significa che nessun aggiornamento futuro (o passato) sarebbe valido.

Nel caso in cui un claim debba essere completamente revocato, senza la possibilità di aggiornare, la versione più recente e il nonce di revoca dovrebbero essere aggiunti al ReT.

3.3.7 JSON Web Zero-Knowledge

JSON Web Zero-Knowledge (JWZ) [19] è uno standard aperto per rappresentare messaggi comprovati con la tecnologia a prova zero-conoscenza. Basato sugli esistenti standard di messaggistica sicura, in particolare, JWM (JSON Web Message) e JWT (JSON Web Token), JWZ è un formato standard per rappresentare e inviare messaggi sicuri supportati dalla tecnologia a prova zero-conoscenza. È un modo innovativo di fornire l'interazione tra due parti che intendono scambiare messaggi mantenendo nascoste le chiavi pubbliche del mittente. La prova generata dal portafoglio è incapsulata nel formato JWZ e inviata al Verifier utilizzando callbackUrl. Nel Protocollo Iden3, JWZ è la primitiva principale per gestire le comunicazioni tra diverse parti. Un JWZ espande lo schema di firma dello standard JWT ampiamente utilizzato. Qualsiasi messaggio può essere incapsulato all'interno di un JWZ

mentre la prova garantisce l'integrità e la provenienza dei dati del messaggio, fornendo utili metadati insieme al messaggio.

CAPITOLO 4

Lavoro proposto

Il progetto proposto mira a dare un esempio delle potenzialità offerte da polygon, l’obiettivo del lavoro è mettere in funzione le funzionalità core proposte dal provider illustrando un caso d’uso specifico.

4.1 Definizione Dell’Issuer e utilizzo della Issuer-node-ui

4.1.1 Issuer-node-ui

Polygon ci offre un AgentEndpoint che espone le funzionalità di creazione e gestione ai wallet. Impiegheremo l’issuer-node-ui per richiedere i servizi evitando di richiedere in maniera programmatica le risorse all’agente.

4.1.2 Caso d’uso KYC Age Credential

Il caso d’uso documentato prevede la definizione e l’assegnazione di una credenziale ad un utente con la conseguente produzione del qr-code da poter scansionare dalla wallet-app.

Creazione della credenziale

La creazione delle credenziali avviene sfruttano l’endpoint di comunicazione "v1/credentials" includendo nel body della richiesta dei dati dei dati in formato JSON che specificheranno la credenziale da creare.

Listing 4.1: Dati in formato JSON

```

1 {
2   "credentialSchema ":" https://raw.githubusercontent.com/iden3/claim-schema-vocab/main/schemas/json/KYCAgeCredential-v3.json",
3   "type ":" KYCAgeCredential",
4   "credentialSubject ":{  

5     "id ":" did: polygonid:polygon:mumbai:2qNRDJZZzt3p73pdDLPGq4iS2jzBBuWHUrZhtRiuG",
6     "birthday ":"19960424",
7     "documentType":2
8   },
9   "expiration ":"2025-02-23T09:45:11.028Z",
10  "signatureProof":true ,
11  "mtProof":true
12 }

```

- **credentialSchema:** Rappresenta l’URL dello schema per questa credenziale. In questo caso, lo schema è situato all’indirizzo <https://raw.githubusercontent.com/iden3/claim-schema-vocab/main/schemas/json/KYCAgeCredential-v3.json>. Lo schema aiuta a definire la struttura e il formato della credenziale.
- **type:** Specifica il tipo di credenziale. In questo caso, il tipo è "KYCAgeCredential", indicando che si tratta di una credenziale KYC che include informazioni sull’età del soggetto.
- **credentialSubject:** Rappresenta l’oggetto che contiene le informazioni specifiche sul soggetto della credenziale. Questo oggetto contiene diversi campi:
 - **id:** Identificatore dell’utente a cui si assegna la credenziale.
 - **birthday:** Rappresenta la data di nascita del soggetto. Nel’esempio è specificato come 19960424, che corrisponde al 24 aprile 1996.
 - **documentType:** Specifica il tipo di documento associato al soggetto. In questo caso, è specificato come 23.
- **expiration:** Rappresenta la data di scadenza della credenziale. Nel’esempio è specificato come "2025-01-23T09:45:11.028Z", indicando che la credenziale scade il 23 gennaio 2025 alle 09:45:11 (tempo UTC).
- **signatureProof:** È un booleano che indica se la credenziale include una prova di firma.
- **mtProof:** È un booleano che indica se la credenziale include una prova di timestamp.

Dopo aver richiesto il servizio all’agente l’issuer node provvederà a fare il fetching relativo alla credenziale usando Redis e salvare le informazioni relative all’assegnamento della credenziali in modo sicuro con l’ausilio di Postgres e Vault. L’issuer node come response ci fornirà un id (cid) a cui è associata la credenziale, grazie alla quale è possibile richiedere ulteriori servizi.

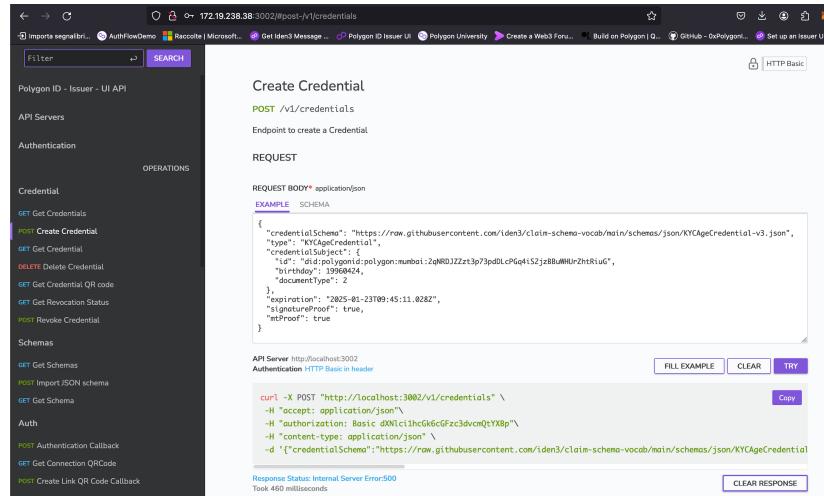


Figura 4.1: Interfaccia grafica interattiva dell’issuer-node

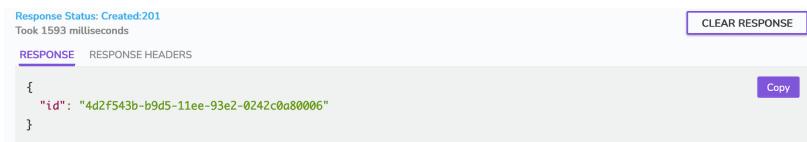


Figura 4.2: Credential id della credenziale erogata

Creazione del Qr-code

È possibile utilizzare l’endpoint di comunicazione "v1/credentials/cid" che darà per response dati in formato JSON utili per definire un qr-code per il wallet. il DID dell’Emittente (stringa identificativa recuperata chiamando l’end-point Create Identity) e l’identificatore della credenziale (o cid recuperato dall’end-point "Create Claim") vengono passati come variabili di percorso nell’URL della richiesta.

Il Nodo Emittente risponde inviando un messaggio di risposta che contiene un JSON che riporta i seguenti campi:

- **credentials:** contiene l’ID della credenziale (cid) e un collegamento allo schema associato alla credenziale.
- **url:** è l’indirizzo a cui il portafoglio dell’utente fa una chiamata al punto finale.
- **from:** è il did dell’Emittente.
- **to:** è il did del portafoglio dell’utente.
- **typ e type:** indicano il modo in cui il portafoglio dell’utente interagisce con l’issuer, impiegando il protocollo Iden3.

```

{
  "body": {
    "credentials": [
      {
        "description": "KYCAgeCredential",
        "id": "4d2f543b-b9d5-11ee-93e2-0242c0a80006"
      }
    ],
    "url": "http://172.19.238.38:3002/v1/agent"
  },
  "from": "did:polygonid:polygon:mumbai:2qFHeyB57ek6388WtfqNaGegkphvpbwYWAtHntLj",
  "id": "e78ea5bd-d852-49d7-ae97-3cf3ecf28931",
  "thid": "e78ea5bd-d852-49d7-ae97-3cf3ecf28931",
  "to": "did:polygonid:polygon:mumbai:2qR0JZZzt3p73pdDLcPGq4iS2jzBBuWHUrZhtRiuG",
  "typ": "application/iden3comm-plain-json",
  "type": "https://iden3-communication.io/credentials/1.0/offер"
}

```

Figura 4.3: Generazione del file JSON da esporre al wallet

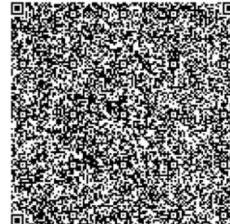


Figura 4.4: Qr-code della credenziale

Una volta generato il codice QR, l’utente può scansionarlo tramite l’app apposita sul cellulare e accettare la credenziale nel suo portafoglio.

4.2 Definizione del Wallet e utilizzo della Flutter-SDK

Nel corso dello sviluppo della mia applicazione, ho implementato il **Polygon ID Wallet SDK**, un set di strumenti basato su Flutter che include librerie utilizzate per creare l'app per il Wallet Holder. Questo SDK offre un'ampia gamma di funzionalità per la creazione e la gestione delle identità digitali, rendendolo ideale per integrare sistemi di Identità Digitale Autonoma (SSI) nelle applicazioni.

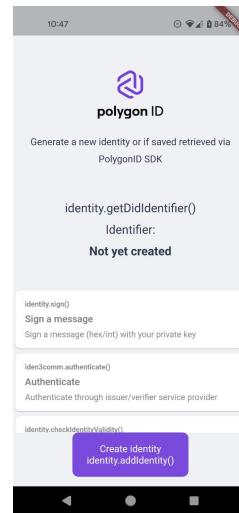


Figura 4.5: Homepage del wallet

4.2.1 Vantaggi di Flutter e uso di polygonID

L'app sfrutta la flessibilità di **Flutter SDK**, un framework open-source che supporta lo sviluppo di applicazioni mobile cross-platform. L'uso di Flutter, combinato con le funzionalità specifiche del Polygon ID Wallet SDK, ha permesso di creare un'app che non solo gestisce le identità digitali per più piattaforme ma anche rispettoso della privacy degli utenti. L'uso del **Polygon ID Wallet SDK** nella mia app ha portato numerosi vantaggi. Primo fra tutti, la capacità di creare e gestire identità per il portafoglio digitale. Questo si traduce nella possibilità di aggiungere, ripristinare o rimuovere identità, autenticarsi con emittenti e verificatori, e ricevere o aggiornare credenziali.

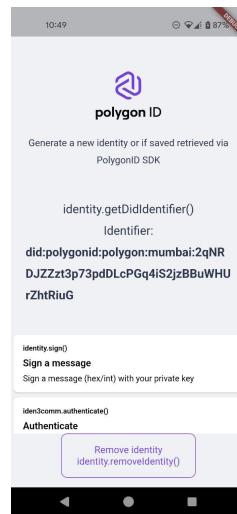


Figura 4.6: Homepage del wallet dopo aver generato il DID

4.2.2 Caso d’uso KYC Age Credential

L’ottenimento delle credenziali sul wallet si divide in:

- Generare l’entità CredentialRequestEntity utilizzando l’identificatore, callbackUrl, e i campi thid e from da iden3Message. thid è l’ID del messaggio e from rappresenta l’identificatore dal quale il Verificatore richiede la prova.
- Recuperare e salvare le credenziali utilizzando CredentialRequestEntity, l’identificatore, e la chiave privata associata al wallet.

Listing 4.2: Snippets per Recuperare e Salvare le Credenziali

```
1 Future<void> fetchAndSaveClaims({  
2   required Iden3MessageEntity iden3message,  
3   required String identifier,  
4   required String privateKey,  
5 }) async {  
6   Map<String, dynamic>? messageBody = iden3message.body;  
7  
8   // URL per il callback  
9   final String callbackUrl = messageBody['url'];  
10  // Credenziali  
11  List<dynamic> credentials = messageBody['credentials'];  
12  List<CredentialRequestEntity> credentialRequestEntityList =  
13    credentials.map((credential) {  
14      String credentialId = credential['id'];  
15      return CredentialRequestEntity(  
16        identifier,  
17        callbackUrl,  
18        credentialId,  
19        iden3message.thid,  
20        iden3message.from,  
21      );  
22    }).toList();  
23  
24  await sdk.iden3comm.fetchAndSaveClaims(  
25    credentialRequests: credentialRequestEntityList,  
26    identifier: identifier,  
27    privateKey: privateKey,  
28  );  
29}
```

L'iden3message è un oggetto ottenuto scansionando il qrcode esposto dall'issuer.

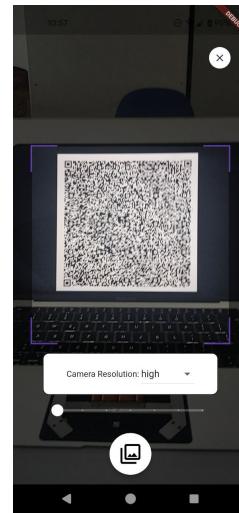


Figura 4.7: Scansione del QR-code dall'app del wallet

Una volta che le credenziali sono state salvate nella SDK, queste possono essere recuperate utilizzando credential.getClaims() con l'identificatore e la chiave privata utilizzati come parametri obbligatori e i filtri come parametro opzionale. I filtri consentono all'Integratore di ottenere le credenziali basate su criteri predefiniti.

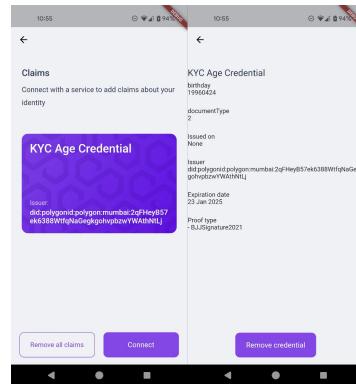


Figura 4.8: Credenziale KYC Age presente sul wallet

4.3 Definizione del Verifier e utilizzo della Verifier-SDK

Nella sezione dedicata alla verifica dell'identità nella mia tesi, ho esplorato e implementato un Verificatore sfruttando le capacità offerte dal Verifier-sdk. Un Verificatore, nel contesto delle identità digitali basate su blockchain, è un'entità che autentica gli utenti verificando le prove di identità fornite, la validazione di tali credenziali sarà effettuata da un contratto che

risiede sulla blockchain (off-chain). Questo processo si articola in due componenti principali: il Server del Verificatore e il Client del Verificatore.

4.3.1 Server del Verificatore

Il Server del Verificatore è il cuore del sistema di verifica. È responsabile per la generazione delle richieste di prova di conoscenza zero (Zero-Knowledge Proof, ZKP) in base ai requisiti specifici della piattaforma. Ho considerato due tipologie di autenticazione: l'Autenticazione di Base, utilizzata per piattaforme che necessitano di verificare l'identità dell'utente prima di concedere credenziali, e l'Autenticazione basata su Query, adatta per piattaforme che richiedono requisiti specifici come l'età dell'utente. Il mio Verificatore, attraverso il Server, elabora la verifica delle prove inviate dal Wallet, assicurando che le informazioni fornite corrispondano ai criteri richiesti.

4.3.2 Client del Verificatore

Il Client del Verificatore funge da interfaccia con l'utente. Nella sua forma più semplice, necessita di incorporare un codice QR che presenta la richiesta ZKP generata dal Server. Questo approccio facilita l'interazione utente, consentendo loro di scansionare il codice QR e generare localmente una prova basata sulla richiesta, che viene poi inviata al Server del Verificatore per l'effettiva verifica.

Nel contesto della mia applicazione, ho utilizzato queste informazioni per configurare un Verificatore che verifica le credenziali di tipo "KYCAgeCredential", concentrandomi su un attributo specifico come la data di nascita, per soddisfare un caso d'uso ipotetico in cui l'accesso è concesso solo agli utenti la cui data di nascita è inferiore alla data 01/01/2001.

L'implementazione pratica di questo sistema nel mio progetto si è basata sull'installazione di pacchetti specifici, la configurazione di endpoint sul server per gestire le richieste di autenticazione e la verifica delle callback contenenti le prove, seguendo le linee guida e gli esempi forniti dalla documentazione di Polygon ID. Questo approccio mi ha permesso di realizzare un sistema di verifica sicuro, dimostrando l'efficacia e la flessibilità del framework Polygon ID Identity nell'autenticazione off-chain degli utenti.

Implementazione della Verifica Zero-Knowledge

Nel contesto della mia tesi, ho esaminato l'impiego dei circuiti *Atomic Query Signature V2*, progettati per consentire verifiche Zero-Knowledge (ZK) basate sulle dichiarazioni degli

utenti. Questi circuiti, in combinazione con il *Query Language*, forniscono una metodologia robusta e flessibile per effettuare la validazione delle credenziali degli utenti.

credentialAtomicQuerySigV2

Questo circuito svolge un ruolo cruciale nell'ambito della verifica delle identità digitali, assicurando che una determinata dichiarazione sia stata effettivamente rilasciata da un emittente autorizzato per l'identità specificata. La procedura di verifica si articola nei seguenti passaggi:

1. **Verifica della Proprietà dell'Identità:** Il circuito inizia con la verifica che il dimostrante sia il legittimo proprietario dell'identità in questione.
2. **Conferma del Soggetto della Dichiarazione:** Successivamente, il circuito verifica che l'identità specificata sia effettivamente il soggetto della dichiarazione rilasciata, garantendo così la coerenza tra l'identità e la dichiarazione associata.
3. **Verifica della Firma dell'Emittente:** Un passaggio fondamentale consiste nella verifica che la dichiarazione sia stata firmata dall'emittente, confermando così l'autenticità e l'origine della dichiarazione stessa.
4. **Conformità dello Schema della Dichiarazione:** Il circuito controlla che lo schema della dichiarazione corrisponda esattamente a quello specificato nella query del verificatore, assicurando l'allineamento tra i requisiti di verifica e la struttura della dichiarazione.
5. **Validità della Dichiarazione:** Viene poi verificato che la dichiarazione non sia stata revocata dall'emittente e che non sia scaduta, garantendo la sua attuale validità e affidabilità.
6. **Soddisfazione della Query:** Infine, il circuito verifica che la dichiarazione soddisfi effettivamente la query posta dal verificatore, confermando che tutti i criteri richiesti siano rispettati dalla dichiarazione in esame.

Il *Query Language* si basa su questi circuiti per offrire agli sviluppatori un modo semplice di progettare requisiti di autenticazione personalizzati. Utilizzando sei operatori logici, il verificatore può formulare condizioni specifiche di autenticazione, come nel caso dell'esempio la data di nascita e conseguentemente accedere a determinati servizi o piattaforme.

4.3.3 Caso d’uso KnowYourAge credential

Esempio di Query Off-Chain

Per illustrare, si consideri una query che richiede all’utente di dimostrare di essere nato prima del 01/01/2001 per accedere a un servizio specifico:

```

1   id: 1,
2   const proofRequest = {
3     circuitId: 'credentialAtomicQuerySigV2',
4     query: {
5       allowedIssuers: ['*'],
6       type: 'KYCAgeCredential',
7       context: 'https://raw.githubusercontent.com/iden3/claim-schema-vocab/main/schemas/json-ld/kyc-v4.jsonld',
8       credentialSubject: {
9         birthday: {
10           $lt: 20010101,
11         },
12       },
13     },
14   };

```

Codice QR Corrispondente

Il codice QR per presentare questa query all’utente incorpora i dettagli della transazione e specifiche della query:

```

1  {
2    "id": "7f38a193-0918-4a48-9fac-36adfdb8b542",
3    "typ": "application/iden3comm-plain-json",
4    "type": "https://iden3-communication.io/proofs/1.0/contract-invoke-request",
5    "thid": "7f38a193-0918-4a48-9fac-36adfdb8b542",
6    "body": {
7      "reason": "airdrop participation",
8      "transaction_data": {
9        "contract_address": "<add your contract address here>",
10       "method_id": "b68967e2",
11       "chain_id": 80001,
12       "network": "polygon-mumbai"
13     },
14     "scope": [
15       {
16         "id": 1,
17         "circuitId": "credentialAtomicQuerySigV2OnChain",
18         "query": {
19           "allowedIssuers": ["*"],
20           "context": "https://raw.githubusercontent.com/iden3/claim-schema-vocab/main/schemas/json-ld/kyc-v4.jsonld",
21           "credentialSubject": {
22             "birthday": {
23               "$lt": 20010101
24             }
25           },
26           "type": "KYCAgeCredential"
27         }
28       }
29     ]
30   }
31 }

```

Il Qr-code verrà poi esibito dal client che lo esporrà al wallet holder. Questa metodologia non solo rende il processo di autenticazione più accessibile e gestibile per gli utenti finali, ma apre anche nuove prospettive nell'implementazione di meccanismi di autenticazione sicuri e rispettosi della privacy in vari ambiti applicativi.

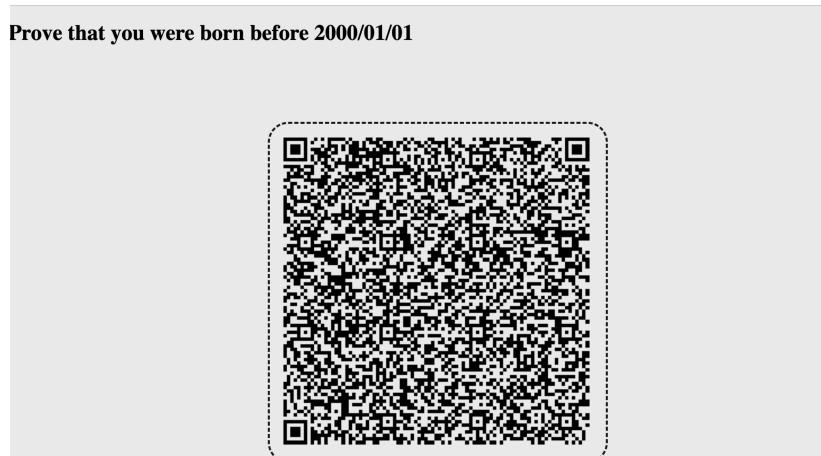


Figura 4.9: Verifica della credenziale richiesta dal verifier

l’utente scansionerà il qr-code ed invierà la zero knowledge proof prodotta dal portafoglio al client, che si proccuperà di inoltrarlo al server per l’effettiva verifica.

4.3.4 Estrazione della Prova dal Request

La prima fase del processo di verifica inizia con l'estrazione della prova inviata dal wallet dell'utente in risposta alla Richiesta di Autenticazione posta dal Verificatore. Questo viene fatto leggendo il contenuto del body della richiesta POST inviata dal wallet ottenendo il cosiddetto token JWZ.

```
const raw = await getRawBody(req);  
const tokenStr = raw.toString().trim();
```

Inizializzazione del Verificatore

Successivamente, ho inizializzato il Verificatore creando un resolver per recuperare lo stato dell'identità dallo Smart Contract di stato e un loader per le chiavi di verifica necessarie per verificare la Zero Knowledge proof prodotta dal wallet.

```
var verificationKeyloader = &loaders.FSKeyLoader{Dir: keyDIR}
resolver := state.ETHResolver{
    RPCUrl: ethURL,
    ContractAddress: common.HexToAddress(contractAddress),
}
resolvers := map[string]pubsignals.StateResolver{
    resolverPrefix: resolver,
}
verifier, err := auth.NewVerifier(
    verificationKeyloader, resolvers, auth.WithIPFSGateway("<gateway url>"))
```

Esecuzione della Verifica

Il passo finale nel processo di autenticazione è l'esecuzione della verifica della prova, per assicurare che la prova condivisa dall'utente soddisfi i criteri impostati dal Verificatore all'interno della richiesta iniziale.

```
authResponse, err := verifier.FullVerify(
    r.Context(),
    string(tokenBytes),
    authRequest.(protocol.AuthorizationRequestMessage),
    pubsignals.WithAcceptedStateTransitionDelay(time.Minute*5))
```

4.3.5 Erogazione del servizio richiesto dall'utente

La verifica si basa su una serie di passaggi che includono la verifica della prova di conoscenza zero, la verifica dello stato dell'identità on-chain, e la verifica degli input pubblici del circuito. Questi passaggi assicurano che l'utente sia effettivamente il "proprietario" dello stato utilizzato per generare la prova e che la prova soddisfi le regole richieste dal Verificatore.

In conclusione, il flusso di autenticazione che ho implementato permette al client web di autenticare l'utente utilizzando il suo DID dopo aver stabilito che l'utente controlla quell'iden-

tità e soddisfa la query presentata nella richiesta di autenticazione. Questo processo consente all'utente di accedere alla piattaforma senza divulgare alcuna informazione personale al client, eccetto il suo DID.

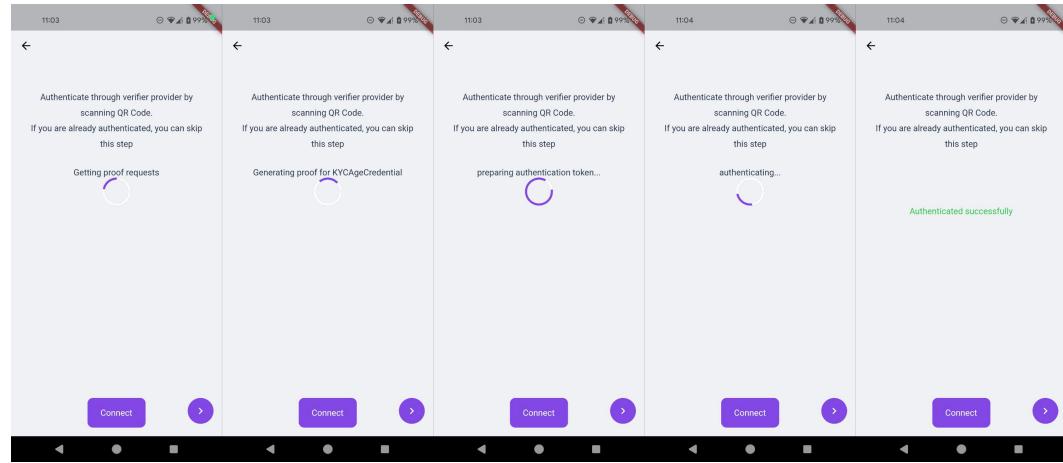


Figura 4.10: Immagine relativa al wallet durante il processo di generazione della Zero Knowledge Proof e di comunicazione con il verifier

CAPITOLO 5

Conclusioni

In conclusione, la presente tesi ha esplorato le potenzialità delle tecnologie decentralizzate per la gestione dell'identità digitale, con un focus particolare su Polygon ID. L'approccio proposto si distingue per la sua capacità di offrire agli utenti un controllo maggiore sui propri dati, incrementando la sicurezza e la privacy attraverso l'utilizzo di credenziali verificabili e meccanismi di autenticazione robusti basati sulla blockchain.

Una possibile direzione per migliorare ulteriormente il lavoro svolto consiste nell'integrare il sistema sviluppato con applicazioni specifiche per settori critici, come quello dell'health-care. Creando un client dedicato che comunica con l'Issuer Node UI e implementando dei Verifier appositamente progettati per rispondere alle esigenze uniche del dominio applicativo. Questo non solo aumenterebbe l'applicabilità pratica della soluzione proposta, ma potrebbe anche offrire nuove prospettive per la gestione dell'identità digitale in contesti in cui la sicurezza e la privacy dei dati sono di massima importanza.

Adottando un approccio più mirato, che consideri le specificità di settori come quello sanitario, il sistema potrebbe essere ottimizzato per gestire i requisiti normativi associati alla condivisione di dati sensibili. Un tale sviluppo potrebbe non solo rafforzare la fiducia degli utenti nel sistema, ma anche accelerare l'adozione di soluzioni di identità digitale auto-sovrane in ambiti critici, portando a miglioramenti significativi in termini di sicurezza e protezione della privacy.

Bibliografia

- [1] Hamed Tabrizchi and Marjan Kuchaki Rafsanjani. A survey on security challenges in cloud computing: issues, threats, and solutions. *The journal of supercomputing*, 76(12):9493–9532, 2020. (Citato a pagina 1)
- [2] M. D. Assunção, F. L. Koch, and C. B. Westphall. Grids of agents for computer and telecommunication network management. *Concurrency and Computation: Practice and Experience*, 16(5):413–424, 2004. (Citato a pagina 4)
- [3] Haifa Alanzi and M. Alkhatib. Towards improving privacy and security of identity management systems using blockchain technology: A systematic review. *Applied Sciences*, 2022. (Citato a pagina 4)
- [4] U. Habiba, Rahat Masood, and M. A. Shibli. Secure identity management system for federated cloud environment. pages 17–33, 2015. (Citato a pagina 5)
- [5] Hasnae L'Amrani, Y. Idrissi, and R. Ajhoun. Technical interoperability to solve cross-domain issues among federation systems. 7:21–40, 2020. (Citato a pagina 5)
- [6] Zubair Ahmad Khattak, S. Sulaiman, and J. Manan. A study on threat model for federated identities in federated identity management system. *2010 International Symposium on Information Technology*, 2:618–623, 2010. (Citato a pagina 5)
- [7] Ssi w3c standards, 2024. <https://www.w3.org/TR/did-core/>. (Citato a pagina 6)
- [8] Geovane Fedrecheski, J. Rabaey, L. Costa, P. Calcina-Ccori, William T. Pereira, and M. Zuffo. Self-sovereign identity for iot environments: A perspective. *2020 Global Internet of Things Summit (GIoTS)*, pages 1–6, 2020. (Citato a pagina 7)

- [9] Frederico Schardong and Ricardo Cust'odio. Self-sovereign identity: A systematic review, mapping and taxonomy. *Sensors (Basel, Switzerland)*, 22, 2021. (Citato a pagina 7)
- [10] Špela Čučko and Muhamed Turkanović. Decentralized and self-sovereign identity: Systematic mapping study. *IEEE Access*, 9:139009–139027, 2021. (Citato a pagina 7)
- [11] R. Belchior, B. Putz, G. Pernul, M. Correia, André Vasconcelos, and Sérgio Guerreiro. Ssibac: Self-sovereign identity based access control. *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 1935–1943, 2020. (Citato a pagina 7)
- [12] Zahra Ebadi Ansaroudi, Roberto Carbone, Giada Sciarretta, and Silvio Ranise. Control is nothing without trust a first look into digital identity wallet trends. pages 113–132, 2023. (Citato alle pagine 7 e 9)
- [13] Connect.me wallet, 2024. <https://www.evernym.com/blog/connect-me-sovrin-digital-wallet/>. (Citato a pagina 8)
- [14] Keytrust walet, 2024. <https://www.keytrust.com.au/>. (Citato a pagina 8)
- [15] Talao wallet, 2024. <https://talao.io/>. (Citato a pagina 8)
- [16] Open wallet, 2024. <https://openwallet.foundation/>. (Citato a pagina 8)
- [17] Dizme wallet, 2024. <https://www.infocert.it/grandi-aziende/dizme>. (Citato a pagina 8)
- [18] Microsoft authenticator, 2024. <https://www.microsoft.com/it-it/security/mobile-authenticator-app>. (Citato a pagina 8)
- [19] Polygon id documentation, 2024. <https://devs.polygonid.com/>. (Citato alle pagine 14 e 31)
- [20] Iden3 protocol documentation, 2024. <https://docs.iden3.io/protocol/spec/>. (Citato a pagina 22)

Ringraziamenti

Desidero esprimere la mia più profonda gratitudine e il mio sincero apprezzamento a tutte le persone che hanno contribuito a rendere possibile questo viaggio accademico e la realizzazione di questa tesi.

Innanzitutto, vorrei ringraziare mia madre e tutta la mia famiglia per il loro incondizionato supporto, amore e incoraggiamento in ogni fase della mia vita. Il loro sostegno costante è stato il faro che mi ha guidato attraverso le sfide e le difficoltà.

Un sentito ringraziamento è dedicato a Schizzo, il mio leale amico a quattro zampe, la cui presenza gioiosa e le distrazioni giocose hanno reso il percorso verso la conclusione di questa percorso meno gravoso e più allegro. La tua capacità di portare sorrisi nei momenti di stress non è stata solo apprezzata, ma anche profondamente necessaria.

Un ringraziamento speciale va a te Dyana, il mio cuore trabocca di gratitudine per te, sei stata la mia roccia nei momenti di dubbio e la mia luce nella tempesta, guidandomi con dolcezza attraverso ogni sfida. Il tuo amore incondizionato e la tua presenza mi hanno fornito la forza per perseguire i miei obiettivi, grazie a te ho superato ogni tipo di ostacolo e ho realizzato sogni che un tempo sembravano irraggiungibili.

Un'enorme grazie va anche a mio cugino Luca e al mio grande amico Francesco, per essere stati sempre lì quando avevo bisogno di una pausa o di una risata. Ma un grazie gigantesco va a Dario, che più di tutti è saputo stare al mio fianco, supportandomi e tirandomi su il morale nei momenti difficili. Non so cosa avrei fatto senza di voi!

Desidero rivolgere un sincero ringraziamento al Prof. Christiancarmine Esposito per l'opportunità unica di lavorare sotto la sua guida per la realizzazione di questa tesi.

Voglio anche esprimere la mia profonda riconoscenza al Dott. Biagio che mi ha guidato con dedizione e impegno. Biagio, la tua maestria nel navigare le complessità del nostro campo e la tua generosità nel condividere la tua conoscenza e la tua esperienza sono state la bussola

che mi ha orientato. La tua pazienza e la tua disponibilità per esplorare ogni mia curiosità, a risolvere ogni dubbio e a superare ogni ostacolo che il mio lavoro presentava sono state fonte di ispirazione e di ammirazione.

Infine, vorrei esprimere la mia gratitudine a tutto il personale del laboratorio di ricerca per l'ambiente stimolante e collaborativo.

A tutti voi, il mio più sincero ringraziamento per aver reso questo traguardo possibile e per avermi accompagnato in questo significativo capitolo della mia vita.