

UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA



CORSO DI LAUREA IN INFORMATICA

TESI DI LAUREA IN
INFORMATICA

**AUTENTICAZIONE DECENTRALIZZATA
CON BLOCKCHAIN NELLE SMART CITIES**

Candidato:

Nome e Cognome:
Gaetano Di Genio
Matr.: 0512106179

Relatore:

Prof. Christian Carmine
Esposito

Anno Accademico 2020/21

*Il primo dei tanti traguardi.
Dedico questo alla mia famiglia,
ai miei fedeli e cari amici,
a chi non c'è più e fa sentire la sua mancanza,
a me stesso.*

Indice

1	Introduzione	1
2	Città Smart, sfide e soluzioni	3
2.1	Città Smart	3
2.2	XACML-OAuth2-OpenID Connect	5
2.3	Primitive crittografiche fondamentali	7
2.4	Firma Digitale	8
2.5	Funzioni hash one-way	9
3	Self Sovereign Identity, il nuovo paradigma	11
3.1	Self Sovereign Identity (SSI)	11
3.2	Verifiable Credentials	12
3.3	Verifiable Credentials Model	13
3.4	Struttura di una Verifiable Credential	14
3.5	Verifiable Presentation	15
3.6	Utilizzi ed impatto delle Verifiable Credentials	16
3.7	Zero Knowledge Proof (ZKP)	18
3.8	Decentralized Identifiers (DIDs)	19
3.9	Struttura di un DID	19
3.10	Design Goals DIDs	20
3.11	Componenti architettura DID	21
3.12	DIDs pubblici e DIDs privati	23
4	Blockchain ed il progetto Hyperledger	25
4.1	La tecnologia Blockchain	25
4.2	Il progetto Hyperledger: Indy, Ursa ed Aries	27
4.3	Hyperledger Ursa	29
4.4	Hyperledger Indy	30
4.5	Hyperledger Aries	31
4.6	Protocolli comunicazione Aries	33
4.7	Implementazione Smart City con le tecnologie Hyperledger	34
	Bibliografia	37

Capitolo 1

Introduzione

La gestione delle politiche, delle infrastrutture, dei servizi, della sicurezza e del senso di efficienza trasmesso in una generica città possono essere di gran lunga migliorate attraverso l'utilizzo di tecnologie ICT e grazie alla cooperazione di differenti organizzazioni che insieme hanno come obiettivo quello di rendere la città più intelligente.

Tipicamente una piattaforma che supporti l'applicazione di una **Smart City** non è realizzata da una singola entità (ad es. il comune), bensì si integrano differenti infrastrutture ICT ognuna gestita da una diversa organizzazione creando di fatto un modello federato e non centralizzato.

Le sfide che questo modello porta sono numerose: è necessario che i dati gestiti dalle diverse organizzazioni siano protetti da accessi non autorizzati e da usi maliziosi, rendendo quindi necessario la creazione di politiche di accesso ai dati stessi le quali distinguono chi o cosa può accedere a quali dati ed in che modo.

Tipicamente le politiche di accesso e i dati inerenti alle identità dei soggetti da autorizzare sono memorizzate e rese disponibili da un singolo server, quindi seguendo un approccio centralizzato.

In un contesto federato come quello descritto sopra una soluzione del genere non è appropriata: ogni organizzazione ha infatti le proprie politiche di sicurezza, per cui è necessario che ognuna mantenga in un proprio database tutti i dati necessari.

È necessario inoltre che le organizzazioni siano a conoscenza delle politiche di sicurezza adottate dalle altre, così da poter utilizzare le stesse credenziali in modo continuo attraverso l'infrastruttura nel complesso (single sign-on).

Questo crea il problema di dover garantire consistenza e sicurezza delle informazioni replicate all'interno di un sistema distribuito, affrontato egregiamente dalla **tecnologia Blockchain**.

In questo lavoro di tesi si affronterà il problema di realizzare una soluzione di città smart, dapprima illustrando le metodologie utilizzate prima dell'avvento della tecnologia Blockchain, per poi vedere come quest'ultima possa essere utilizzata per realizzare anche il concetto di **Self Sovereign Identity (SSI)**.

La suddivisione dei capitoli è così effettuata: il capitolo 2 descrive i modelli e le tecnologie comunemente utilizzate per realizzare una soluzione di città smart, con tutti i problemi che derivano da queste implementazioni. Si introdurranno inoltre concetti base quali crittografia a chiave asimmetrica e firma digitale, particolarmente utili per comprendere i concetti mostrati nei successivi capitoli.

Il capitolo 3 introduce nel dettaglio la Self Sovereign Identity e tutti i componenti fondamentali che rendono possibile la sua realizzazione.

Infine nel capitolo 4 si guarda al progetto **Hyperledger** ed in particolare ad **Hyperledger Aries**, un tool che consente di creare applicazioni basate sul paradigma della SSI e che viene utilizzato in questo lavoro per realizzare una soluzione di città smart basata sull'uso della tecnologia Blockchain, la quale mira a risolvere la totalità delle problematiche presentate nel capitolo 2.

Capitolo 2

Città Smart, sfide e soluzioni

2.1 Città Smart

Il paradigma di **Città Smart** consiste nell'utilizzo di tecnologie informatiche per migliorare numerosi aspetti riguardanti la gestione di una città ed in generale la qualità della vita dei cittadini.

Questo paradigma viene tipicamente realizzato adoperando diverse tipologie di **sensori elettronici**, i quali si occupano di raccogliere dati di vario tipo: traffico stradale, inquinamento, temperatura, altro. Successivamente questa enorme raccolta di informazioni attraversa diverse fasi di processing ed analisi, allo scopo di inferire conoscenza utile al miglioramento della gestione di asset e risorse, oltre ad ottenere feedback sulle politiche di gestione attualmente adoperate nell'ambito della città.

I suddetti sensori possono essere installati su qualunque tipo di asset o dispositivo di proprietà dei cittadini in modo tale da poter monitorare ad esempio: scuole, ospedali, efficienza delle reti idriche, centrali energetiche ed altri servizi. [1]

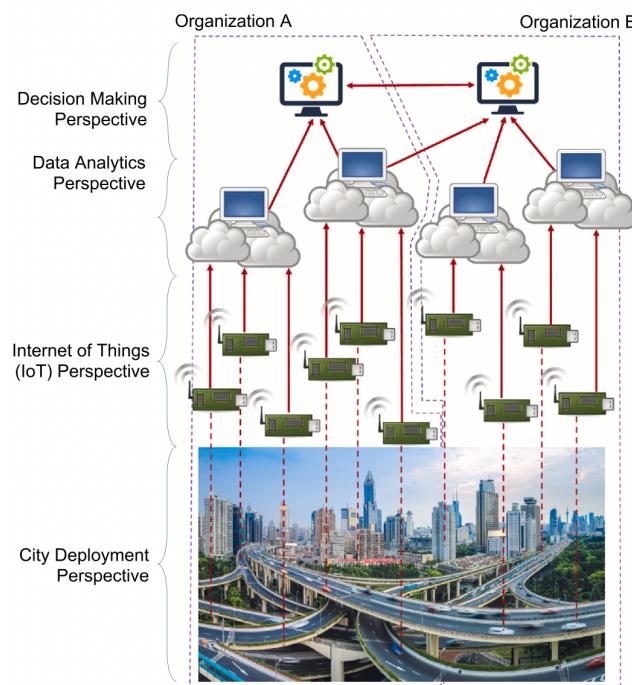


Figura 2.1: Schema di una smart city basata sull'utilizzo di un approccio federato con multipli decision-makers

Il tutto consente agli agenti di polizia o altre autorità di interagire in maniera istantanea e molto più efficiente con la comunità e le infrastrutture della città, permettendo anche di monitorare come certi fenomeni evolvono nel tempo.

È importante specificare che il paradigma di Città Smart non viene realizzato da una singola entità (ad esempio il comune) che a quel punto dovrebbe gestire autonomamente tutti i sensori, bensì l'approccio è **federato** (Figura 2.1): differenti organizzazioni collaborano al fine di rendere la città più intelligente, ognuna gestendo il proprio insieme di sensori .

Ad esempio, le aziende che si occupano di trasporti pubblici gestiscono i sensori piazzati sui propri mezzi di trasporto (bus, treni, tram), il comune gestisce i sensori disposti lungo le strade, le università e le scuole gestiscono le loro aule e i loro servizi e così via.

Il comune a questo punto riceve la mole di dati raccolti dalle differenti organizzazioni e funge da **decision-maker** centrale (è possibile avere anche multipli decision-makers).

È particolarmente importante che lo scambio dei dati in questa fase (in quanto sensibili e di grande importanza) avvenga in modo sicuro, permettendo soltanto a chi è autorizzato di poter accedere ad essi ed evitando inoltre ogni probabilità che avvengano fughe di informazioni (information leakage). [1]

La Figura 2.2 mostra le due principali tipologie di architetture di sicurezza:

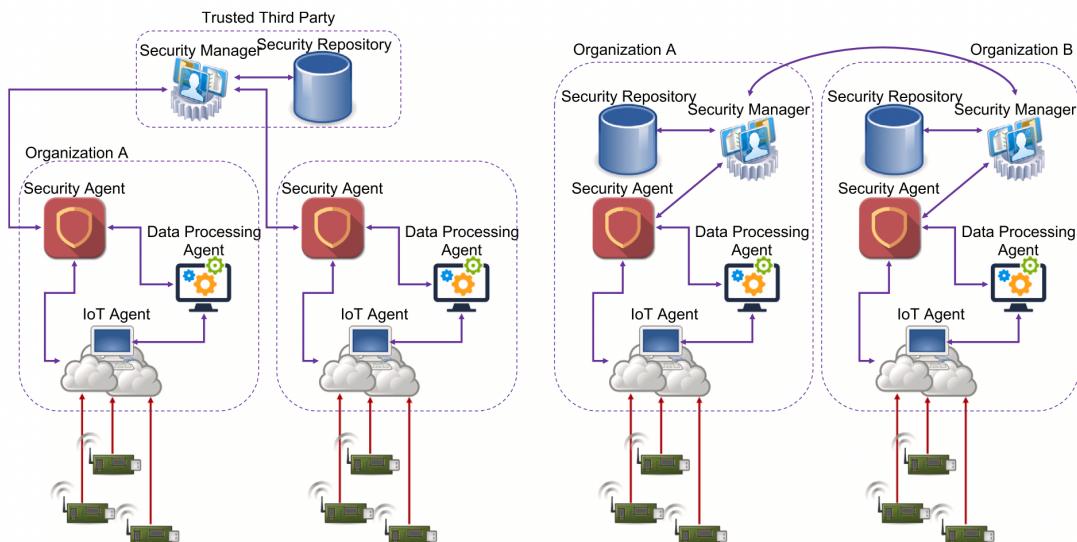


Figura 2.2: Architettura di sicurezza basata su approccio centralizzato (a) e approccio decentralizzato (b)

L'approccio illustrato nella Figura 2.2(a) prevede l'utilizzo di un singolo database nel quale sono memorizzate le politiche di controllo degli accessi e gli attributi riguardanti l'identità dei soggetti. Il punto debole di questo approccio è che risulta non adeguato avere un repository centrale e un singolo modello di policy in un contesto federato, poiché ogni organizzazione potrebbe aver definito politiche di sicurezza proprie ed utilizzare in generale architetture di sicurezza diverse da quelle utilizzate dalle altre organizzazioni.

Una soluzione potrebbe essere quella di federare i differenti repository, in modo che se qualche claim non possa essere verificato localmente si possa eseguire una query distribuita ai security manager delle altre organizzazioni direttamente connesse.

Tuttavia questo approccio è molto lento.

Sarebbe decisamente meglio, quindi, fare in modo che tutte le organizzazioni abbiano una copia locale

di tutte le varie politiche di sicurezza.

Questo richiede che le copie siano aggiornate continuamente per mantenere la consistenza delle informazioni al proprio interno, con tutti i possibili problemi in termini di sicurezza durante questa fase. Una soluzione ai problemi in quest'ultimo caso è l'utilizzo della tecnologia Blockchain. [1]

Nei seguenti capitoli si introducono il concetto di autorizzazione ed autenticazione, una tipica implementazione basata sull'utilizzo di **XACML-OAuth2-OpenID Connect**, infine si inizia ad introdurre una nuova tipologia di implementazione basata sull'utilizzo della Blockchain e del concetto di **Self Sovereign Identity**.

2.2 XACML-OAuth2-OpenID Connect

Si introducono brevemente il concetto di **Autenticazione** e di **Autorizzazione**:

- **Autenticazione** è il processo di validazione dell'identità e dei relativi attributi di un utente/macchina quando questo richiede l'accesso ad una specifica risorsa o funzionalità.
- **Autorizzazione** consiste invece nello stabilire chi è autorizzato/chi non è autorizzato ad accedere ad una certa risorsa in base alle politiche di controllo degli accessi (access control policies) stabilite.

Una maniera comune di realizzare il cosiddetto **Servizio di Autenticazione ed Autorizzazione (AAS)** consiste nell'utilizzare lo standard **XACML** (eXtensible Access Control Markup Language), il quale definisce un linguaggio per definire politiche di controllo degli accessi basato su XML.[1]

XACML definisce inoltre un'architettura per poter valutare queste politiche come un insieme di web services, in cui le interazioni tra le diverse componenti sono illustrate nella Figura 2.3:

Nello specifico: l'utente invia una richiesta al **Policy Enforcement Point (PEP)**, il quale agisce da front-end proteggendo la risorsa da richieste esterne. Estrae dalla richiesta dell'utente alcuni dettagli che invia al **Policy Decision Point (PDP)** che deve restituire al PEP la decisione finale circa il garantire o meno l'accesso dell'utente alla risorsa.

Tale decisione è presa interagendo a sua volta con il **Policy Information Point (PIP)** che mantiene diversi valori (risorsa, soggetto, ambiente) e il **Policy Retrieval Point (PRP)** che mantiene e gestisce le policy XACML.

Tali risorse sono amministrate infine dal **Policy Access Point (PAP)** che agisce in base a come l'amministratore decide di operare.

In combinazione con XACML è possibile utilizzare **OAuth2** che agisce come protocollo di autorizzazione, in cui l'utente interagisce con un authentication server ed ottiene, in caso di decisione positiva, un token di accesso che utilizzerà insieme alla richiesta inviandoli al PEP per ottenere l'accesso ad una specifica risorsa.

Il problema principale è che il token non contiene informazioni riguardanti l'identità del soggetto, per cui potrebbe essere compromesso ed utilizzato per attacchi di impersonificazione in cui si finge di essere un'altra persona ed ottenere l'accesso a specifiche risorse.

È necessario includere quindi OpenID, che in aggiunta include meccanismi crittografici per verificare asserzioni riguardanti l'identità, prevenendo la possibilità di questi suddetti attacchi.

Una implementazione di OpenID che poggia su OAuth2 prende il nome di **OpenID Connect**. [1] Insieme XACML, OAuth2 e OpenID Connect realizzano una comune implementazione di una soluzione per realizzare una città smart (Figura 2.1).

Questa soluzione, tuttavia, utilizza un singolo database per mantenere le policy di sicurezza e gli

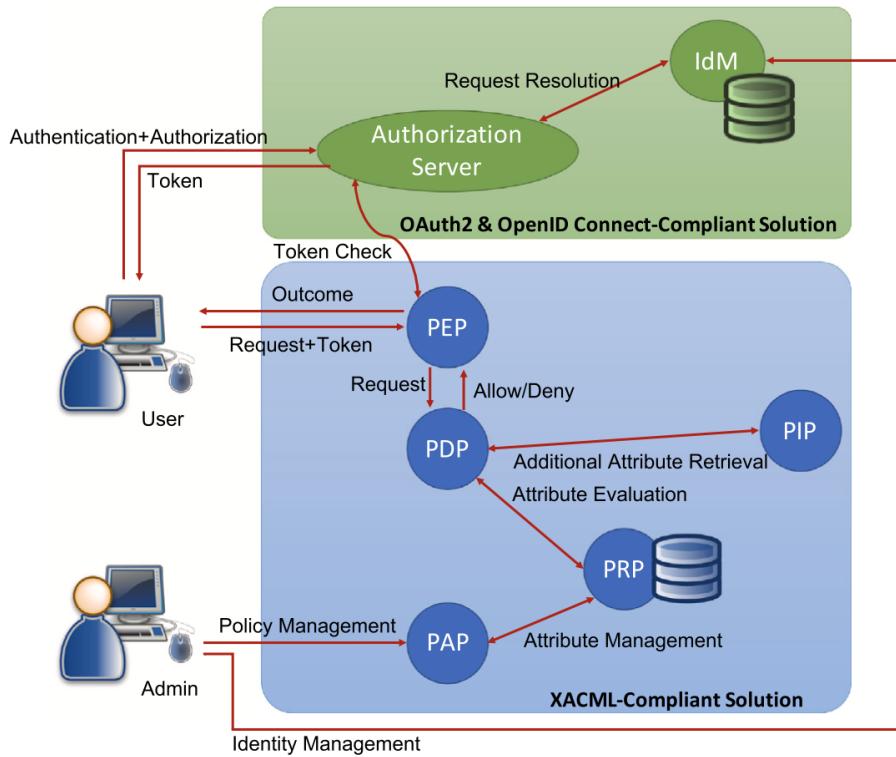


Figura 2.3: Architettura XACML ed integrazione con OAuth2 e OpenID Connect

attributi di identità e, come descritto in precedenza, non è una soluzione utilizzabile in un contesto federato.

Utilizzare una federazione di repository (uno per ogni organizzazione) è da un lato una possibile soluzione, dall'altro però impone diverse limitazioni: è più lenta a causa dei delay durante l'esecuzione delle query ed è inoltre necessario garantire interoperabilità tra i modelli utilizzati dalle diverse organizzazioni per memorizzare le policy e gli attributi.

Come ultima possibile soluzione si prevede l'utilizzo di una copia globale di tutte le policy e gli attributi per ogni organizzazione, dove però risulta difficile mantenere la consistenza di tutte le copie in quanto internet è un sistema asincrono in cui i delay sono impredicibili, inoltre è necessario che le informazioni scritte non siano in alcun modo manipolabili.

L'introduzione della **tecnologia Blockchain** può rendere fattibile quest'ultima soluzione: consiste in una catena di blocchi in cui i partecipanti decidono, mediante una politica di consenso, se aggiungere o meno un ipotetico nuovo blocco.

Garantisce immutabilità delle informazioni scritte nei blocchi, consistenza di dati replicati su più blocchi in un sistema distribuito e permette inoltre la realizzazione del concetto di **Self Sovereign Identity (SSI)** in cui l'individuo mantiene il totale controllo sulla propria identità e può decidere quando e come queste informazioni vengono condivise ad una terza parte, il tutto mediante l'utilizzo della crittografia [1].

Detto ciò, prima di introdurre il concetto di SSI e di come poterlo utilizzare per realizzare il concetto di città smart insieme alla tecnologia Blockchain, è doveroso dare alcune definizioni di base che torneranno utili nei successivi capitoli: **crittografia a chiave asimmetrica, firma digitale e funzione hash**.

2.3 Primitive crittografiche fondamentali

Si ipotizzi il seguente scenario: un mittente **Mitt** vuole comunicare con un destinatario **Dest** utilizzando un canale di trasmissione insicuro, cioè tale che altri possano intercettare i messaggi che vi transitano per conoscerli o alterarli.

Per proteggere la comunicazione i due agenti devono adottare un metodo di cifratura che permetta a **Mitt** di spedire un messaggio m sottoforma di crittogramma c , incomprensibile ad un ipotetico malintenzionato in ascolto sul canale, ma facilmente decifrabile da parte di **Dest** (**Figura 2.4**). [3]

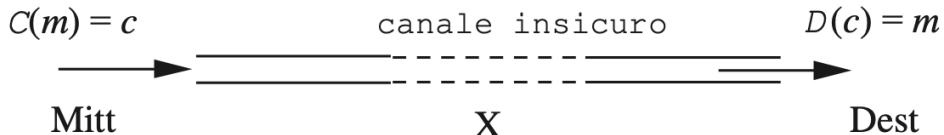


Figura 2.4: Scambio di un messaggio cifrato su un canale insicuro

Nella crittografia a chiave asimmetrica le operazioni di cifratura e decifratura prevedono l'utilizzo di due chiavi diverse: una chiave **pubblica** per cifrare i messaggi ed una chiave **privata** per decifrarli. La prima è nota a tutti, per cui chiunque può cifrare un messaggio per un certo destinatario utilizzando la chiave pubblica di quest'ultimo.

La seconda, invece, è segretamente custodita dal soggetto e viene utilizzata per decifrare il messaggio ricevuto precedentemente cifrato con la chiave pubblica del soggetto stesso.

La **Figura 2.5** mostra il funzionamento della crittografia a chiave asimmetrica: $k[prv]$ e $k[pub]$ sono le chiavi privata e pubblica dell'utente R.

Gli altri utenti U_1, U_2, \dots, U_n inviano a R i messaggi mi cifrati come $c_i = C(m_i, k[pub])$, $1 \leq i \leq n$. R decifra i crittogrammi calcolando $D(c_i, k[prv])$, mediante la chiave privata $k[prv]$ che solo lui conosce.

Un crittoanalista in ascolto sul canale insicuro non puo' ricavare alcuna informazione sui messaggi mi pur conoscendo C, D e $k[pub]$ [3].

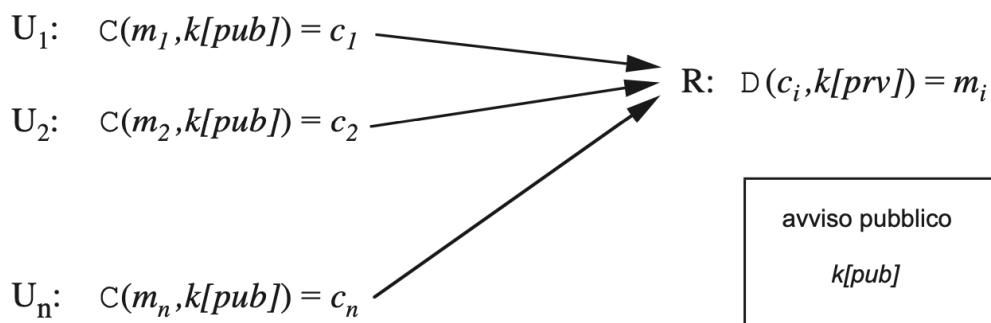


Figura 2.5: Crittografia a chiave asimmetrica.

2.4 Firma Digitale

La crittografia asimmetrica è particolarmente utilizzata per realizzare il meccanismo di **Firma Digitale** (Figura 2.6).

Prima di illustrarne il funzionamento è bene esaminare quali sono i motivi che rendono necessaria la firma manuale di un documento di qualsiasi genere e le proprietà che essa deve mantenere.

La **Firma manuale** apposta su un documento per siglare un accordo formulato in esso o per provare l'autenticità o la paternità dello stesso soddisfa alcune proprietà, quali:

- La firma è autentica e non è falsificabile, per cui prova che chi l'ha prodotta è veramente colui che ha sottoscritto il documento
- la firma non nè riutilizzabile, dunque risulta strettamente legata al documento su cui è stata apposta
- la firma non può essere ripudiata da chi l'ha apposta, quindi costituisce una prova legale di un accordo o di una dichiarazione contenuta nel documento
- il documento firmato non è alterabile, per cui chi ha prodotto la firma è sicuro che questa si riferirà solo al documento sottoscritto nella sua forma originale

Nonostante questi requisiti sono a volte annullati da impostori capaci di falsificare le firme manuali, essi sono validi nella maggior parte dei casi e la firma rimane il mezzo standard per l'autenticazione di un documento.

Creare una versione digitale di firma non può consistere nella semplice digitalizzazione di una firma manuale (ad esempio mediante uno scanner) poichè sarebbe fin troppo semplice effettuare dei copia e incolla della porzione di documento in cui figura la firma ed effettuare attacchi e frodi.

Si vuole invece che la firma digitale abbia una forma che dipende dal documento specifico così da essere inscindibile da questo.

Di seguito è mostrato il **protocollo di firma di Diffie-Hellman** basato sulla crittografia asimmet-

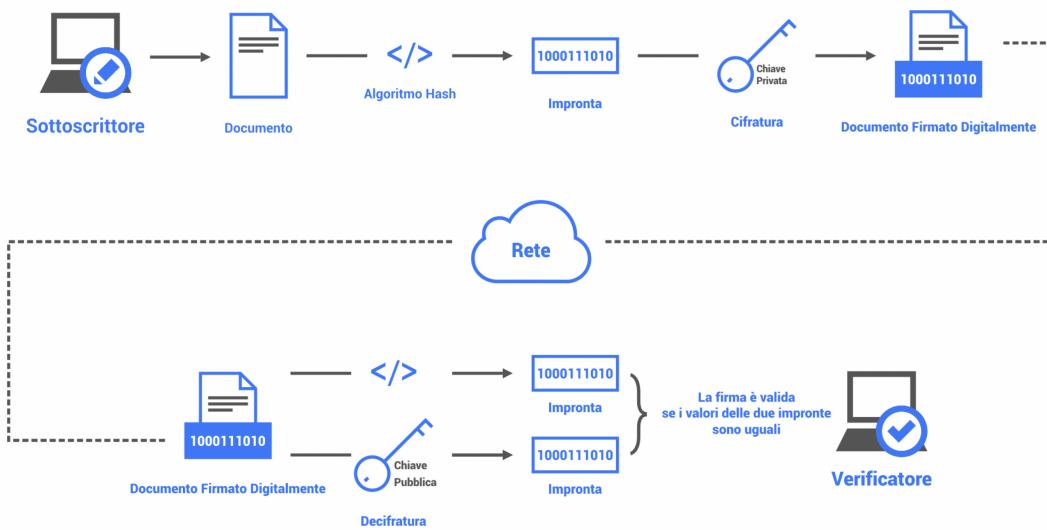


Figura 2.6: Funzionamento firma digitale.

trica, in cui la chiave privata del mittente è utilizzata da questi per produrre la firma e la sua chiave pubblica è utilizzata dal destinatario per verificarne l'autenticità.

Sia U il mittente e V il destinatario, $k_U[\text{prv}]$, $k_U[\text{pub}]$, $k_V[\text{prv}]$, $k_V[\text{pub}]$ le loro chiavi private e pubbliche, C e D le funzioni di cifratura e decifrazione del cifrario asimmetrico. Un possibile protocollo

(in cui si considera il messaggio m cifrato) è il seguente:

Firma e cifratura: Il mittente U genera la firma $f = D(m, kU[\text{prv}])$ per m, calcola il crittogramma firmato $c = C(f, kV[\text{pub}])$ con la chiave pubblica del destinatario e spedisce la coppia $\langle U, c \rangle$ a V.

Decifrazione e verifica: Il destinatario V riceve la coppia $\langle U, c \rangle$, decifra il crittogramma con la sua chiave privata, cioè calcola $D(c, kV[\text{prv}])$ ottenendo un valore pari a f, quindi cifra tale valore con la chiave pubblica di U ottenendo $C(D(m, kU[\text{prv}]), kU[\text{pub}]) = m$.

In tal modo V ricostruisce il messaggio m, attestando a un tempo l'identità del mittente se m è significativo. [3]

2.5 Funzioni hash one-way

Una funzione hash matematica $f: X \rightarrow Y$ è definita su un dominio X e codominio Y finiti e tale che $n = |X| \gg m = |Y|$.

Sono utilizzate per operare su elementi x appartenenti a X attraverso la loro immagine $y = f(x)$ appartenente a Y poiché la rappresentazione di y richiede $\log_2 m$ bit, quindi più breve di quella di x che ne richiede $\log_2 n$.

Quando le funzioni hash one-way sono adoperate nella crittografia (Figura 2.7), possiedono queste proprietà:

- per ogni elemento x appartenente a X è computazionalmente facile calcolare $f(x)$
- per la maggior parte degli elementi y in Y è computazionalmente difficile determinare un x tale che $f(x) = y$
- è computazionalmente difficile determinare una coppia di elementi x', x'' in X tali che $f(x') = f(x'')$

Esistono varie famiglie di funzioni hash utilizzate per scopi crittografici, tra cui la SHA e MD(Message Digest).

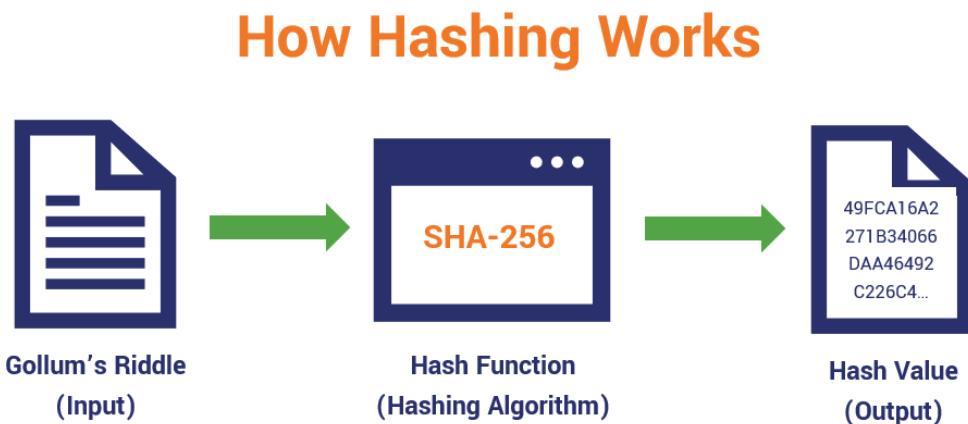


Figura 2.7: Funzionamento funzione hash one-way.

Definite le primitive crittografiche necessarie alla comprensione dei concetti successivamente espresi nel presente lavoro di tesi, si introducono nel successivo capitolo la Self Sovereign Identity in dettaglio e i componenti fondamentali che rendono possibile la realizzazione della stessa.

Capitolo 3

Self Sovereign Identity, il nuovo paradigma

3.1 Self Sovereign Identity (SSI)

La **Self Sovereign Identity** è l'idea secondo la quale un individuo o organizzazione controlla autonomamente la propria identità digitale, gestendo quando e come i dati sono forniti ad una terza parte e, quando questi vengono condivisi, che il tutto avvenga in modo sicuro ed affidabile.

Conferisce inoltre la capacità di gestire i propri identificatori e le relazioni con individui, aziende, governi, mantenendole separate le une dalle altre così da ottenere grossi benefici in termini di privacy. [2]

Con la SSI non esiste alcuna autorità centrale che funga da Identity Provider, ovvero che gestisca i dati degli utenti e la condivisione degli stessi ad entità terze.

Grazie all'utilizzo della crittografia e della tecnologia Blockchain, chiunque è in grado di presentare attributi riguardo la propria identità e gli altri possono verificarne la veridicità e la validità con estrema certezza.

I principi fondamentali della Self Sovereign Identity sono molteplici [5], vale a dire:

- **Rappresentabilità:** un ecosistema SSI fornisce i mezzi affinché qualsiasi entità - umana, legale, naturale, fisica o digitale - sia rappresentata da un numero qualsiasi di identità digitali
- **Interoperabilità:** un ecosistema SSI consente di rappresentare, scambiare, proteggere e verificare i dati dell'identità digitale di un'entità in modo interoperabile utilizzando standard aperti, pubblici e privi di royalty
- **Decentralizzazione:** un ecosistema SSI non deve affidarsi ad un sistema centralizzato per rappresentare, controllare o verificare i dati dell'identità digitale di un'entità
- **Controllo e rappresentanza:** un ecosistema SSI consente alle entità che hanno diritti naturali, umani o legali in relazione alle loro identità di controllare l'utilizzo dei dati legati alla loro identità digitale ad esercitare tale controllo impiegandolo e/o delegandolo ad agenti e tutori di loro scelta, inclusi individui, organizzazioni, dispositivi e software
- **Equità ed inclusione:** un ecosistema SSI non deve escludere o discriminare alcun soggetto nel suo ambito di governance

- **Usabilità, accessibilità e consistenza:** un ecosistema SSI deve massimizzare l'usabilità e l'accessibilità di agenti ed altri componenti SSI per il soggetto, inclusa la consistenza nella user experience
- **Portabilità:** un ecosistema SSI non deve limitare la capacità di un soggetto di poter spostare o trasferire una copia dei dati riguardanti la propria identità verso un agente o sistema di propria scelta
- **Sicurezza:** un ecosistema SSI consente ad un soggetto di mantenere i propri dati protetti, di controllare i propri identificatori e chiavi private/pubbliche e di utilizzare la crittografia end-to-end per tutte le interazioni
- **Verificabilità ed autenticità:** un ecosistema SSI consente ai soggetti di fornire prove verificabili dell'autenticità dei dati della loro identità digitale
- **Privacy e divulgazione minimale:** un ecosistema SSI consente ai soggetti di proteggere la privacy dei dati della propria identità digitale e di condividere soltanto i dati strettamente necessari durante un'interazione
- **Trasparenza:** un ecosistema SSI consente ai vari stakeholder di accedere e verificare facilmente le informazioni necessarie per comprendere gli incentivi, le regole, le politiche e gli algoritmi in base ai quali operano gli agenti e tutti gli altri componenti degli ecosistemi SSI

Realizzare la SSI prevede l'introduzione di alcune componenti fondamentali che, insieme, rispettino i principi fondamentali sopra elencati e rendano concreta la possibilità di sviluppare sistemi basati sulla SSI.

I componenti base sono **Verifiable Credentials (VCs)** e **Decentralized Identifiers (DIDs)**, di seguito illustrate in dettaglio.

3.2 Verifiable Credentials

Innanzitutto, una credenziale (generica) è un attestato di qualifica o competenza rilasciata ad un soggetto da parte di un'entità autorevole, la quale è riconosciuta come affidabile.

Ad esempio la patente di guida afferma che un certo soggetto è in grado di guidare correttamente un veicolo e di rispettare le indicazioni stradali, una laurea universitaria definisce il livello di istruzione di un individuo, il passaporto permette di viaggiare in altri paesi e così via.

L'utilizzo di queste credenziali nel mondo fisico consente facilmente di usufruire di numerosi servizi ma, nel mondo digitale, il loro utilizzo è molto più problematico: attualmente è difficile esprimere informazioni riguardo la propria identità quali livello di istruzione, dati finanziari, dati sulla salute in un modo che siano verificabili autonomamente da una macchina e che non siano in alcun modo alterabili.

Le **Credenziali Verificabili (Verifiable Credentials)** sono un tipo di credenziale utilizzabile sul Web, che garantiscono rispetto della privacy, possibilità di essere verificate da una macchina senza

la necessità di un intervento umano e garanzia di non poter essere alterate grazie all'utilizzo della crittografia.

L'obiettivo che si vuole raggiungere è il seguente: quando un'entità autorevole rilascia una particolare credenziale ad un soggetto nel mondo fisico, si vuole che la medesima autorità rilasci la stessa credenziale anche sotto forma di Verifiable Credential, in modo che l'individuo possa tenerla memorizzata nel suo Wallet digitale e successivamente utilizzarla per provare attributi riguardo la propria identità sul web.

Questo è il cosiddetto **Modello a Credenziali Verificabili (Verifiable Credentials Model)**. [5]

3.3 Verifiable Credentials Model

Nel mondo fisico una credenziale è solitamente composta da:

- informazioni relative al soggetto della credenziale (es. nome, numero identificativo)
- informazioni relative all'autorità che rilascia la certificazione (es. comune, motorizzazione)
- informazioni relative al tipo di credenziale in questione (es. patente, passaporto)
- altre informazioni relative alla credenziale (es. data di scadenza)
- informazioni relative agli specifici attributi riguardanti il soggetto (es. data di nascita, nazionalità, classe di veicoli abilitato a guidare)

Una Credenziale Verificabile permette di rappresentare le stesse informazioni rappresentate da una credenziale fisica ma, mediante la firma digitale ed altre tecnologie, queste risultano più affidabili e non soggette a falsificazione rispetto alla controparte fisica.

Inoltre è possibile trasmetterle con molta rapidità, rendendole più convenienti rispetto alle credenziali fisiche quando si tratta di doverle utilizzare per dimostrare attributi riguardo l'identità su grandi distanze geografiche.

Gli attori coinvolti nel Modello a Credenziali Verificabili sono i seguenti [4]:

- **Issuer:** entità che ricopre il ruolo di rilasciare credenziali ad un Holder
- **Holder:** entità che riceve le credenziali da un'autorità e le mantiene nel proprio portafogli digitale (Wallet) per poi utilizzarle in seguito per presentarle ad un Verifier (es. studente, impiegato, persona)
- **Soggetto:** entità per la quale sono realizzate le credenziali. Ad esempio umani, animali e cose. Solitamente il soggetto coincide con l'Holder, ma non è sempre così: ad esempio un padre (l'Holder) potrebbe mantenere le credenziali del figlio (il Soggetto), il proprietario di un canile (l'Holder) potrebbe mantenere le credenziali di tutti i cuccioli (i Soggetti) e così via.
- **Verifier:** entità terza che riceve una o più credenziali verificabili da un Holder ed ha lo scopo di processarle per poi poter fornire un eventuale servizio.
Ad esempio siti web, datori di lavoro, addetti alla sicurezza, altro.

- **Verifiable Data Registry:** componente fondamentale utilizzato per memorizzare chiavi pubbliche, identificatori, schemas ed altri. Può consistere in un database, una Blockchain, un file system distribuito o altre tipologie di storage affidabili.

In questo lavoro di tesi si farà utilizzo di una Blockchain per la memorizzazione di queste informazioni.

La Figura 3.1 illustra i seguenti attori:

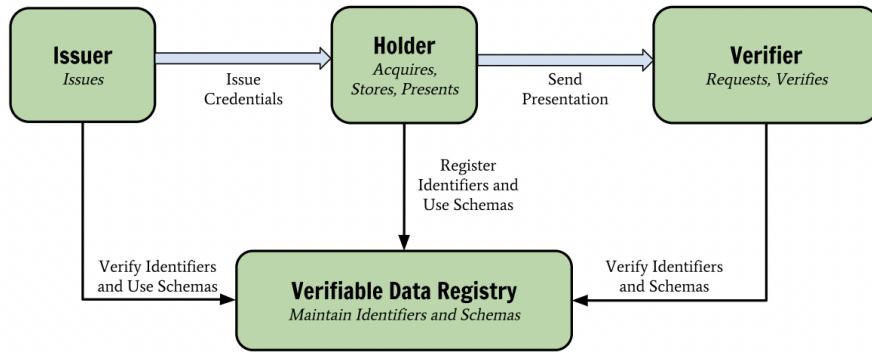


Figura 3.1: Il modello a credenziali verificabili

3.4 Struttura di una Verifiable Credential

Una Credenziale Verificabile è composta dai seguenti campi [4]: **Claim(s)**, **Metadati**, **Proof(s)**. Nel dettaglio:

- **Claim(s):** è un'asserzione riguardante un soggetto, espressa sotto forma di relazione **Soggetto-Proprietà-Valore** come mostrato nella Figura 3.2.
- Una Verifiable Credential è costituita da un numero arbitrario di Claims.
- Un esempio potrebbe essere il seguente: Mario (Soggetto) lavora presso (Proprietà) Google (Valore).

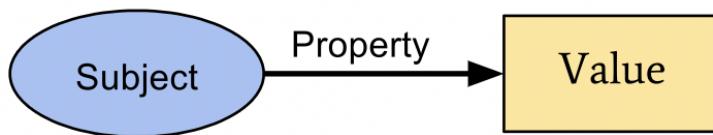


Figura 3.2: Struttura base di un Claim

- **Metadati:** sono informazioni riguardanti la credenziale in questione, quali ad esempio: tipo della credenziale, data di scadenza della stessa, informazioni sull'Issuer.

- **Proof(s)**: la prova crittografica utilizzata per verificare che la Credenziale in questione sia stata effettivamente rilasciata dallo specifico Issuer e che non sia stata manomessa.
- Soltanamente tale prova crittografica consiste in un meccanismo di firma digitale, dove il funzionamento è il seguente: l'Issuer firma digitalmente la credenziale che sta per rilasciare ad un Holder mediante la propria chiave privata.
- L'holder, d'altro canto, può verificare mediante la chiave pubblica dell'Issuer che la credenziale appena ricevuta sia stata effettivamente firmata e rilasciata da questo, verificandone al contempo l'integrità.

La Figura 3.3 raffigura la struttura di una Credenziale Verificabile.

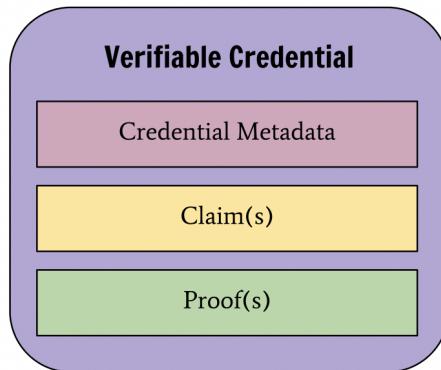


Figura 3.3: Struttura di una Credenziale Verificabile con i relativi campi

3.5 Verifiable Presentation

L'Holder, una volta ricevuta una Verifiable Credential da un Issuer ed averla memorizzata nel suo Wallet, può utilizzarla per provare ad un Verifier attributi riguardo la propria identità per usufruire di un certo servizio.

Ciò non viene fatto presentando direttamente la Verifiable Credential all'entità terza così com'è, bensì l'Holder crea una **Presentazione Verificabile (Verifiable Presentation)** derivata dalla prima.

Questo per i seguenti motivi: creando una Presentation è possibile includere e mostrare alla terza parte soltanto un sottoinsieme di Claims pertinenti alla specifica situazione, evitando di comunicare informazioni superflue (es. è superfluo inserire informazioni riguardo il proprio domicilio se lo scopo è soltanto quello di dimostrare di possedere un'età superiore ai 18 anni).

Ciò porta ad un enorme passo in avanti per la Privacy del soggetto.

Inoltre, è possibile aggregare nella Presentation informazioni riguardanti lo stesso soggetto anche rilasciate da diversi Issuer.

Resta, naturalmente, l'affidabilità e la verificabilità garantita dall'utilizzo della crittografia, esattamente allo stesso modo delle Verifiable Credentials. [4]

La struttura di una Verifiable Presentation è molto simile alla struttura di una Verifiable Credential descritta in precedenza, dove i campi questa volta sono: **Verifiable Credential(s)**, **Metadati**, **Proof(s)**.

In dettaglio:

- **Verifiable Credential(s)**: un numero arbitrario di Credenziali rilasciate anche da diversi Issuers
- **Metadati**: informazioni quali data di scadenza ed altre associate alla Presentation
- **Proof(s)**: prova crittografica solitamente consistente in firma digitale, dove in questo caso è l'Holder a firmare digitalmente la Presentation con la propria chiave privata, mentre il Verifier può verificare con la chiave pubblica del primo che la Presentazione ricevuta sia integra, non sia stata manomessa e sia stata creata effettivamente dal soggetto.

Lo schema è raffigurato dalla Figura 3.4:

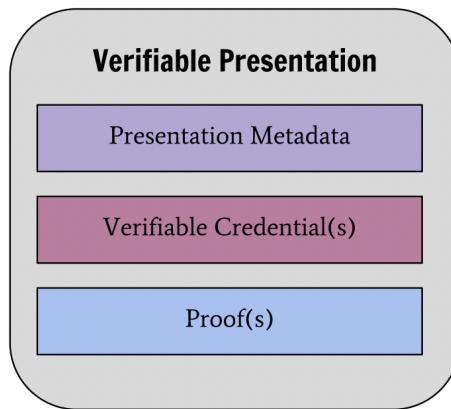


Figura 3.4: Struttura di una Presentazione Verificabile con i relativi campi

3.6 Utilizzi ed impatto delle Verifiable Credentials

Nell'utilizzo pratico di tutti i giorni, i benefici apportati dalle Verifiable Credentials sono numerosi ed abbracciano aspetti quali **privacy**, **semplicità** ed **immediatezza d'uso**, **affidabilità** e **sicurezza**. Si ipotizzi il seguente scenario: un utente internet ha intenzione di acquistare una bottiglia di prestigioso vino da un sito specializzato, il quale ha bisogno di verificare alcune informazioni quali età, indirizzo di residenza, altre.

Normalmente l'utente dovrebbe inserire manualmente i vari campi dell'ipotetico form (nome, cognome, numero di un documento di riconoscimento (patente, passaporto, carta d'identità), indirizzo di residenza, data di nascita) ed inviare il tutto al sito.

Ciò tuttavia potrebbe non bastare, per cui il sito richiede l'invio di una scansione di un documento fisico per poter verificare la veridicità delle informazioni inserite.

Fatto questo, un addetto alle verifiche dei documenti analizzerà manualmente i dati e darà un riscontro, solitamente nel giro di qualche ora o giorni.

Con le Verifiable Credentials, invece, è possibile fornire una Presentation con tutti i dati richiesti ed inviarla al sito, il quale senza bisogno di un intervento umano effettuerà le varie verifiche attraverso l'uso della crittografia ed in pochi istanti si può ottenere l'esito della transazione.

Ciò comporta un grande risparmio in termini di tempo e costo.

Inoltre, se qualcuno dovesse conoscere il numero del documento di identità e tutte le informazioni private di un utente, non potrà in alcun modo impersonificarlo in quanto le informazioni che comunicherebbe non sarebbero derivanti da una Verifiable Credential rilasciata da un'entità autorevole.

Un altro grande vantaggio è dato dal fatto che una Presentation è fornita ad un Verifier senza la necessità da parte dell'Holder di dover richiamare l'Issuer, per cui l'interazione avviene tra Holder e Verifier, non tra Holder, Issuer e Verifier.

Questo riduce il numero di **interazioni tra gli attori**, mentre tutto ciò che è necessario per effettuare le verifiche crittografiche proviene dalla Blockchain.

L'Holder, quindi, decide quando e come i suoi dati vengono condivisi a terzi anche nel mondo digitale, esattamente come avviene attualmente nel mondo fisico, grazie al Verifiable Credentials Model.

Recentemente, col verificarsi della situazione **COVID-19**, è stato necessario introdurre un qualche meccanismo per poter provare il proprio status di "vaccinato" (Green Pass), il tutto in rapidità e semplicità.

Per questo scopo, si è anche dovuto tenere conto del fatto che in una qualsiasi nazione c'è sempre una porzione della popolazione che non possiede uno smartphone o altri device digitali per cui è stata utilizzata una Verifiable Credential stampabile su un pezzo di carta impossibile da manomettere e falsificare.

La soluzione è stata incorporare una Verifiable Credential in **un codice QR** come mostrato nella Figura 3.5.



Figura 3.5: Generico codice QR

Tuttavia questo approccio non è esente da problematiche: c'è un limite alla capacità di informazioni che un codice QR può contenere, tipicamente 400-500 bytes, un numero di bytes maggiore porterebbe ad avere un QR illegibile da una fotocamera in quanto troppo denso di dettagli.

In alternativa è possibile inserire nel QR una **URL** ad un web service che si occuperebbe di restituire la Presentation.

Questo risolverebbe il problema della capacità, ma porterebbe lo svantaggio di dover avere un URL per ogni credenziale e l'Issuer potrebbe venire a conoscenza di tutte le volte che una particolare credenziale è utilizzata.

Questi problemi possono essere mitigati, ma è importante tenerne conto sempre e comunque.

In generale le Credenziali Verificabili stampate su carta eliminano il problema della falsificazione dei documenti e consentono anche a chi non è in possesso di un dispositivo digitale di farne uso, tuttavia è bene considerare che non è possibile per il Verifier richiedere informazioni non strettamente contenute sul foglio, per cui parte della straordinaria potenza delle Verifiable Credentials è perduta. [5]

3.7 Zero Knowledge Proof (ZKP)

La **Zero Knowledge Proof** è un metodo basato sull'utilizzo della crittografia che consente ad un'entità di provare ad una terza parte di essere a conoscenza di una certa informazione, senza di fatto rivelarne il contenuto.

Ad esempio è possibile per un soggetto dimostrare che una certa università gli ha rilasciato una laurea, senza rivelare informazioni personali contenute nella stessa.

Il meccanismo ZKP ha reso realizzabile numerose possibilità per l'Holder, alcune delle quali già presentate precedentemente:

- combinare un numero arbitrario di Verifiable Credentials rilasciate anche da differenti Issuers in una singola Verifiable Presentation, senza rivelare in alcun modo identificatori del soggetto o informazioni private al Verifier.
Questo elimina la possibilità di risalire all'identità del soggetto effettuando delle correlazioni da parte di multipli Verifiers.
- selezionare un sottoinsieme di Claims da una o più Verifiable Credentials, in modo da fornire al Verifier soltanto le informazioni strettamente necessarie e nulla di più
- formattare la Verifiable Credential secondo lo schema accettato dal Verifier, anche se questo risulta diverso rispetto allo schema utilizzato dall'Issuer quando ha rilasciato la Credenziale.
Il tutto viene fatto dall'Holder senza necessità di dover coinvolgere nuovamente l'Issuer, dimostrando quindi grande flessibilità nell'utilizzo

L'Holder quindi, una volta ricevuta la Verifiable Credential dall'Issuer, può provare ad un Verifier la validità della firma digitale apposta dall'Issuer sulla Credenziale, senza rivelare il contenuto effettivamente firmato.

Esiste, tuttavia, un approccio non basato su ZKP, detto appunto approccio non-ZKP.

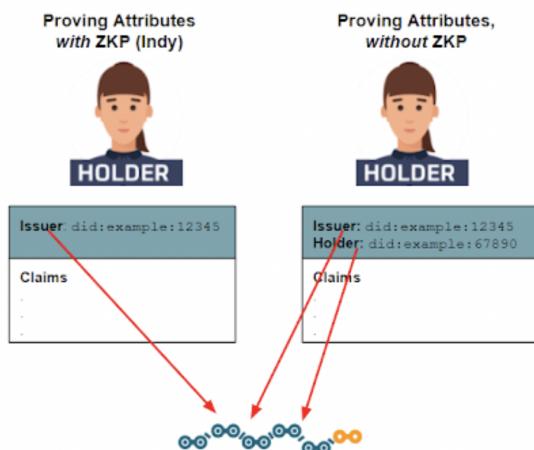


Figura 3.6: DID dell'Holder inserito nella credenziale dall'Issuer con l'approccio non-ZKP (destra)

In questo modello l'Holder prova all'Issuer di controllare uno specifico **identificatore decentralizzato (DID)**, che viene quindi inserito dall'Issuer nella Credenziale rilasciata.

Più avanti, quando il Verifier estrae dalla Presentation l'identificatore, deve verificare che l'Holder mantiene ancora il controllo su quest'ultimo.

Se il controllo va a buon fine vuol dire che la Credenziale è stata effettivamente rilasciata a quell'Holder.

L'approccio non-ZKP consente a diversi Verifier di poter risalire all'identità del soggetto mediante correlazioni, tuttavia l'approccio è più semplice da realizzare rispetto al modello basato su ZKP (Figura 3.6). [5]

3.8 Decentralized Identifiers (DIDs)

Individui ed organizzazioni fanno uso, oggiorno, di identificatori globalmente univoci in numerosi e differenti ambiti.

Si pensi, ad esempio, a identificatori utilizzati come indirizzo di comunicazione (numero telefonico, indirizzo email, username sui social media), a ID (passaporto, patente di guida, carta d'identità), a identificatori utilizzati per descrivere prodotti (numero seriale, codice a barre), a indirizzi per pagine web (URL).

I problemi riguardanti questi identificatori sono molteplici: la maggior parte di questi non sono sotto controllo di chi li utilizza, bensì sono rilasciati da un'entità autorevole che decide a chi o a cosa si riferiscono e possono effettuarne la revoca degli stessi in ogni momento.

Potrebbero inoltre mostrare informazioni personali senza una vera necessità e diventare inutilizzabili nel caso in cui l'organizzazione che li ha creati e rilasciati fallisce.

Infine, non per importanza, possono potenzialmente essere riprodotti da persone malintenzionate che potrebbero quindi impersonificare un soggetto per compiere atti illeciti.

Gli **Identificatori Decentralizzati (Decentralized Identifiers (DIDs))** sono un nuovo tipo di identificatore globalmente univoco fondamentali per poter utilizzare le Verifiable Credentials, in fase di diventare uno standard World Wide Web Consortium (W3C).

Questi identificatori non sono creati da un'organizzazione autorevole (approccio centralizzato), bensì ogni individuo (o azienda) può generare il proprio in modo sicuro e veloce, e può dimostrare di averne il controllo mediante meccanismo di autenticazione con di firma digitale (approccio decentralizzato).

Ogni entità può avere il controllo di uno o anche più DIDs in modo da poterli utilizzare in differenti contesti o situazioni, mantenendo una separazione delle diverse interazioni e identità con altri soggetti. [4]

3.9 Struttura di un DID

Un Identificatore Decentralizzato è di fatto una semplice stringa costituita da: **DID schema**, **DID Method**, **Identificatore del DID Method**.

Così come un URL di un sito web inserito in una barra di ricerca di un browser viene risolto in una pagina web, un DID viene risolto (da un DID Resolver) in un **DID Document (DIDDoc)** che contiene informazioni importanti quali:

- **chiavi pubbliche del soggetto**: per effettuare le verifiche della firma digitale
- **endpoint del soggetto**: per potergli instaurare una comunicazione con il soggetto

La struttura di un DID in dettaglio è la seguente:

- **DID schema:** descrive la sintassi, genericamente è "did".
- **DID Method:** descrive come il DID ed il DIDDoc sono creati, aggiornati, risolti e disattivati.
- **Identificatore del DID Method:** identificatore del DID Method



Figura 3.7: Struttura base di un DID

Un DID garantisce:

- **Alta disponibilità:** un DID, essendo comunemente memorizzato su una Blockchain ed essendo la Blockchain comunemente duplicata su numerosi e diversi server in giro per il mondo, può essere risolto anche in caso di malfunzionamento di alcuni server, proprio per via del fatto che ci saranno altri server disponibili con la stessa copia della Blockchain.
In altre parole, c'è una tolleranza ai guasti.
- **Verificabilità:** è possibile controllare che un certo DID sia sotto il controllo di una certa persona o azienda, richiedendo la verifica di controllo della chiave privata associata alla chiave pubblica trovata nel DIDDoc.

Per chiarire: il DID viene risolto in un DIDDoc, il quale contiene le chiavi pubbliche e endpoint di un soggetto.

Per verificare che quel soggetto ha effettivamente creato e controlla quel DID, un Verifier utilizza la chiave pubblica trovata (nel DIDDoc) per cifrare un messaggio da inviare al soggetto all'endpoint trovato (nel DIDDoc).

Se il soggetto è in grado di leggere e rispondere alla richiesta contenuta nel messaggio, vuol dire che possiede la chiave privata associata alla chiave pubblica trovata in precedenza, dimostrando quindi di essere lui ad aver creato e a controllare il DID in questione. [5]

3.10 Design Goals DIDs

Gli Obiettivi di Design (Design Goals) realizzati per i DIDs sono molteplici [8]:

- **Decentralizzazione:** rimossa la necessità di un'autorità centralizzata che si occupa di creare ed assegnare identificatori ai diversi individui. Come descritto in precedenza, ogni entità crea il proprio Identificatore Decentralizzato senza necessità di un terzo

- **Controllo:** come diretta conseguenza del primo obiettivo, un individuo ha un controllo totale del proprio identificatore, decidendo quando e se revocarlo
- **Privacy:** l'individuo, grazie al maggiore controllo sui propri identificatori, può gestire al meglio la privacy dei propri dati personali, gestendo quali e quando questi vengono condivisi ad una terza parte
- **Sicurezza:** l'utilizzo di un DIDDoc garantisce un sufficiente livello di sicurezza per i Verifier che hanno bisogno di identificare ed accettare informazioni personali di un certo soggetto
- **Proof-based:** la crittografia consente agli individui di fornire prove di vario tipo durante l'interazione con altre entità
- **Rilevabilità:** rende possibile a delle entità di scoprire DIDs di altre, per poterci instaurare una connessione ed interagire
- **Interoperabilità:** l'infrastruttura basata su DID fa uso di standard condivisi da molti, così che sia possibile realizzare applicazioni utilizzando tool e librerie apposite per garantire l'interoperabilità
- **Portabilità:** è garantita l'indipendenza da specifici sistemi o reti, permettendo alle entità di utilizzare i loro identificatori decentralizzati con qualunque sistema supporti DID e DIDDoc, a prescindere dalla sua struttura specifica
- **Semplicità:** la tecnologia è semplice da comprendere, implementare ed utilizzare
- **Estensibilità:** possibilità di estendere il sistema, purchè non si interferisca con i principi di interoperabilità, semplicità e portabilità

3.11 Componenti architettura DID

La Figura 3.8 illustra le componenti di una tipica architettura DID:

Di seguito sono descritte in dettaglio le singole componenti [8]:

- **DIDs e DID URLs:** come descritto in precedenza, un DID è un particolare identificatore che viene risolto da un DID Resolver in un DID Document (DIDDoc).
Un DID URL estende la sintassi del DID incorporando altre informazioni quali percorso, fragment o query per poter identificare una specifica risorsa (es. una chiave pubblica contenuta nel DIDDoc o altro).
La sintassi è del tipo **did:method:methodidentifier/percorso/per/la/risorsa**
- **DID Subject:** specifica l'entità identificata dal DID (persona, organizzazione, cosa). Il soggetto potrebbe coincidere con chi controlla effettivamente il DID (Controller), ma non necessariamente

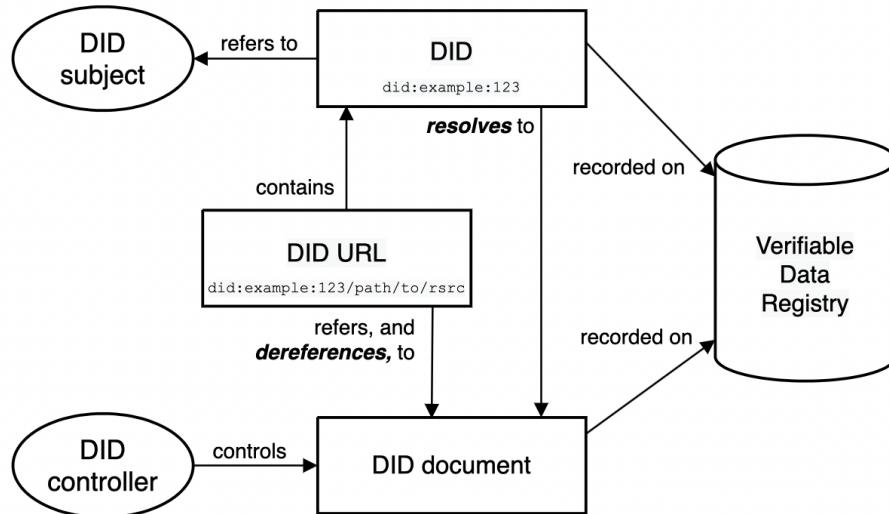


Figura 3.8: Componenti dell'architettura DID

- **DID Controller:** specifica l'entità che ha la capacità di poter modificare un certo DID Document. Un DID può avere uno o più Controller, i quali a loro volta possono essere diversi dal soggetto identificato dal DID, come detto precedentemente
- **Verifiable Data Registry:** già presentata in precedenza, è una Blockchain (può anche trattarsi di altre tipologie di storage) sulla quali viene memorizzato il DID e il DIDDoc
- **DID Document:** contiene informazioni pubbliche quali chiavi pubbliche, endpoint del soggetto. Può essere modificato da un DID Controller.
- **DID Methods:** meccanismi mediante i quali i DID e i DIDDoc vengono creati, aggiornati, risolti e disattivati, come descritto in precedenza
- **DID Resolver e DID Resolution:** il Resolver è una componente del sistema che prende in input un DID e produce in output il relativo DIDDoc.
Questo processo è chiamato DID Resolution (risoluzione del DID)
- **DID URL dereferencers e DID URL dereferencing:** il dereferencer prende in input un DID URL e restituisce in output la risorsa effettivamente richiesta.
Questo processo è denominato URL dereferencing (dereferenziazione dell'URL)

3.12 DIDs pubblici e DIDs privati

Un DID Pubblico (Figura 3.9) è realizzato per essere altamente disponibile e visibile da chiunque. Questi identificatori, essendo utilizzati da tante entità, permettono di risalire all'identità del possessore mediante correlazioni.

In questo caso però, le correlazioni non sono una cosa negativa.

Basti pensare che tipicamente questi vengono utilizzati da aziende e governi per poter essere contattati da qualunque individuo o da parte di entità che ricoprono il ruolo di Issuer (che rilasciano credenziali).

Un Holder che deve mostrare qualche attributo riguardo la propria identità ad un soggetto terzo, incorpora nella Presentation il DID pubblico dell'Issuer, in modo che sia possibile per il Verifier investigare eventualmente sull'identità ed affidabilità di chi ha rilasciato quelle particolari credenziali.

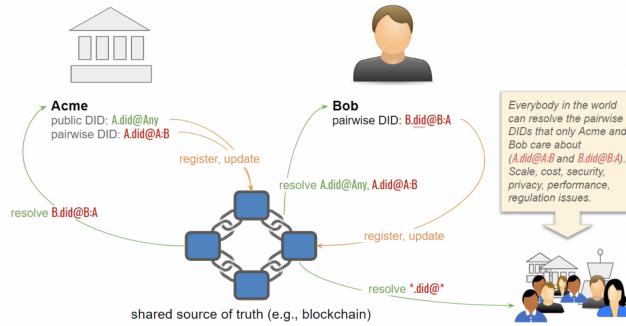


Figura 3.9: Schema DID pubblico

I DID privati (Figura 3.10), d'altro canto, sono utilizzati per realizzare connessioni tra due o più entità che vogliono comunicare tra loro privatamente.

In questo secondo caso, DID e DIDDoc non sono messi in pubblico su una Blockchain, bensì sono creati ad hoc ed inviati direttamente alla controparte con la quale si vuole comunicare.

In questo modo nessun altro, eccetto le entità che comunicano, è in grado di vedere i DID utilizzati. Va considerato che siccome i DID privati non sono scritti sulla Blockchain (e scrivere sulla Blockchain ha un costo), questi permettono anche di risparmiare ulteriori costi. [5]

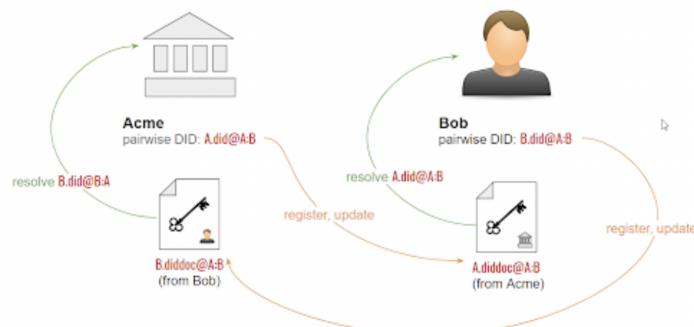


Figura 3.10: Schema DID privato

Capitolo 4

Blockchain ed il progetto Hyperledger

4.1 La tecnologia Blockchain

Tradizionalmente, le transazioni che avvengono in Internet (ad esempio monetarie) sono legate ad istituzioni finanziarie che fungono da terza parte fidata e controllano, salvaguardano e preservano gli scambi di denaro (Figura 4.1).

Questo approccio centralizzato non è esente da problemi: poichè una piccola probabilità di frodi è sempre presente è necessario pagare commissioni piuttosto elevate per ogni singola transazione ed inoltre l'intermediario viene a conoscenza di ogni movimento, per cui la privacy non è preservata.

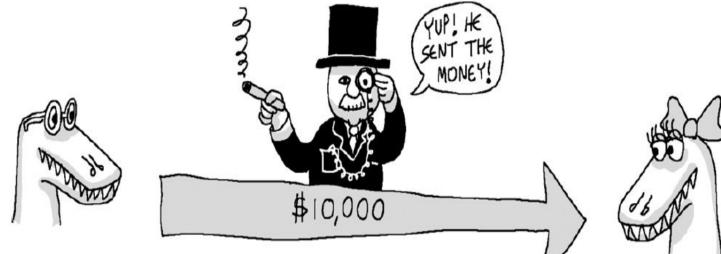


Figura 4.1: Esempio di transazione (finanziaria) che si affida ad una terza parte fidata (es. Banca)

Un approccio decentralizzato prevede che lo scambio di denaro o in generale una transazione avvenga senza la necessità di una terza parte, ovvero mittente e destinatario comunichino direttamente l'uno con l'altro.

Ogni transazione viene firmata digitalmente dal mittente con la sua chiave privata e cifra il messaggio con la chiave pubblica del destinatario.

In ricezione il destinatario decifra il tutto e verifica la firma con la chiave pubblica del mittente.

Ogni transazione viene poi inoltrata a tutti i nodi e scritta sul registro pubblico dopo la verifica.

È importante precisare che le transazioni non arrivano nell'ordine in cui sono generate, per cui è necessario ideare un sistema per assicurarsi che non si verifichi una doppia spesa della criptovaluta.

La tecnologia Blockchain risolve il problema ordinando le transazioni inserendoli in blocchi e collegandoli a formare una catena [7].

In particolare, ogni blocco contiene l'hash del blocco precedente (Figura 4.2).

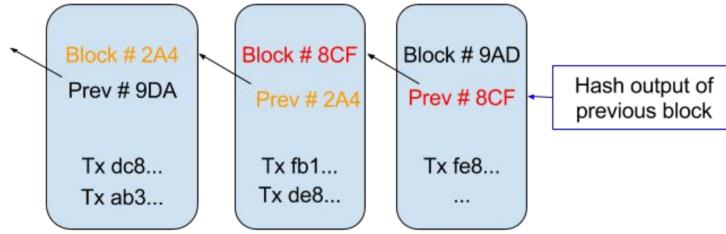


Figura 4.2: Catena di blocchi collegati mediante l'hash del blocco precedente

Come si decide qual è il prossimo blocco della catena?

Non si può, ancora una volta, prendere una decisione del genere facendo riferimento all'ordine di arrivo dei blocchi, per cui è stato ideato il cosiddetto meccanismo della **Proof of Work (PoW)**. Un nuovo ipotetico blocco viene accettato nella Blockchain solo dopo aver risolto un particolare puzzle crittografico.

Nello specifico, un nodo che genera un certo blocco deve dimostrare di aver investito risorse di calcolo sufficienti per poter risolvere quel puzzle, dopodiché la soluzione è inserita nel blocco e quest'ultimo è aggiunto alla catena.

Ad esempio si può richiedere ad un nodo di trovare un nonce che, quando viene eseguito l'hashing con le transazioni e l'hash del blocco precedente, produce un hash con un certo numero di zeri iniziali.

Lo sforzo medio richiesto è esponenziale nel numero di bit zero richiesti, invece il processo di verifica è molto semplice e rapido.

Questo puzzle non è banale da risolvere e la complessità del particolare problema può essere regolata affinché occorrono almeno 10 minuti per generare un blocco valido.

I nodi che investono risorse di calcolo per risolvere questi puzzle sono chiamati **Miners**, i quali vengono remunerati finanziariamente.

La rete accetta solo la catena di blocchi più lunga, per cui è quasi impossibile per un malintenzionato introdurre una transazione fraudolenta poiché non deve solo generare un blocco risolvendo un puzzle, ma deve anche generare tutti i blocchi successivi gareggiando contro i nodi buoni. [7]

La Figura 4.3 mostra il meccanismo della Proof of Work:

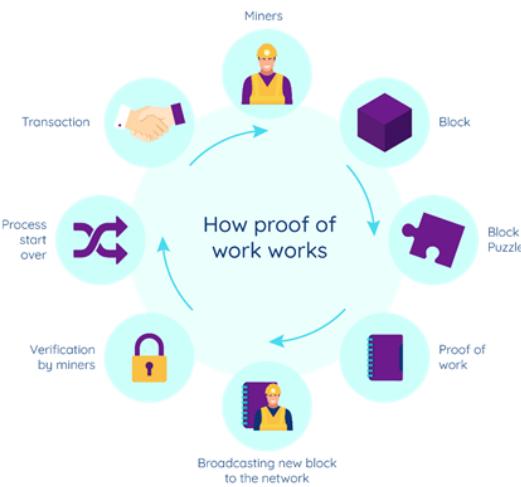


Figura 4.3: Proof of Work

4.2 Il progetto Hyperledger: Indy, Ursa ed Aries

La fondazione **Hyperledger** è un'organizzazione senza fini di lucro che si occupa di fornire l'insieme delle infrastrutture e delle risorse necessarie per la creazione di ecosistemi, applicazioni basati sulla tecnologia Blockchain.

I progetti sono numerosi (Figura 4.4) e sono rivolti ad aziende, accademici, start-up, fornitori di servizi ed altri per aiutarli nella creazione di soluzioni commerciali o di reti blockchain. [9]



Figura 4.4: Progetti della fondazione Hyperledger

In questo capitolo si guardano nel dettaglio tre progetti: **Indy**, **Ursa** ed **Aries**, che insieme permettono di implementare ed utilizzare Verifiable Credentials, DIDs ed in generale rendere concreto il paradigma SSI.

Hyperledger Indy (Figura 4.5) è stato il primo framework Blockchain incentrato sull'identità digitale entrato a far parte di Hyperledger nel 2017. [5]



Figura 4.5: Il progetto Hyperledger Indy

È un distributed ledger appositamente realizzato per supportare tutte le funzionalità introdotte nei precedenti capitoli, quali Verifiable Credentials basate su ZKP, DIDs.

Fornisce inoltre un **software development kit (SDK)** per creare agenti ed implementa un public, **permissioned distributed ledger**.

Con l'evolvere del progetto Indy, ci si è resi conto che le funzioni crittografiche utilizzate in ques'ultimo sarebbero potute essere utilizzate anche in altri progetti o al di fuori di Hyperledger.

Nel 2018 si è deciso di prendere tutta la parte riguardante la crittografia da Indy e di creare un progetto separato, vale a dire **Hyperledger Ursa** (Figura 4.6). [5]



Figura 4.6: Il progetto Hyperledger Ursa

Gli algoritmi crittografici sono progettati e realizzati esclusivamente da esperti del settore i quali grazie ad Ursa possono lavorare con le primitive crittografiche, implementare algoritmi, testarli e renderli infine disponibili agli sviluppatori i quali possono usufruirne per realizzare agenti utilizzando l'SDK offerto da indy.

Il repository **indy-sdk** è il software Indy che consente di creare agenti che possono interagire con un ledger Indy e tra loro per scambiarsi informazioni, credenziali ed altro.

Inizialmente non era garantita l'interoperabilità tra agenti costruiti da differenti organizzazioni, per cui si è pensato di creare il progetto **Hyperledger Aries**, allo scopo di rendere l'agente indipendente dalla specifica implementazione e quindi interoperabile con quelli gestiti da altri. [5]

Aries è un toolkit progettato per creare soluzioni incentrate sulla creazione, trasmissione, archiviazione ed utilizzo di Verifiable Credentials.

Definisce un insieme di protocolli per instaurare una connessione protetta tra due agenti e scambiare messaggi in modo sicuro.

È interamente incentrato sul garantire interazioni peer to peer tra agenti controllati da entità diverse: persone, organizzazioni o cose. [9]

Insieme, Indy, Ursa ed Aries formano lo stack Hyperledger per l'identità digitale (Figura 4.7).



Figura 4.7: Lo stack Hyperledger per l'identità digitale

Nelle successive sezioni si descrivono in dettagli questi tre progetti, partendo dal basso dello stack in Figura 4.7 verso l'alto.

4.3 Hyperledger Ursas

Il progetto Hyperledger Ursas mette a disposizione package di funzioni crittografiche utilizzabili a livelli più alti, ad esempio da Indy ed Aries, per poter: generare chiavi private/pubbliche, cifrare/decifrare, firmare digitalmente e fare la verifica della firma, generare hash e fare la verifica, utilizzare la tecnologia ZKP ecc.

Il processo di creazione dei package comincia dal basso, vale a dire dalla **ricerca** ad opera di matematici.

Questi progettano e successivamente dimostrano la fattibilità degli algoritmi usando la matematica pura (Figura 4.8), per cui lavorano ad un livello molto più basso rispetto ad uno sviluppatore che poi farà uso di questi algoritmi.

Lo scopo è anche quello di migliorare e rendere più efficienti algoritmi già esistenti, controllandone la resistenza ad eventuali attacchi (sia di forza bruta che di altro tipo).

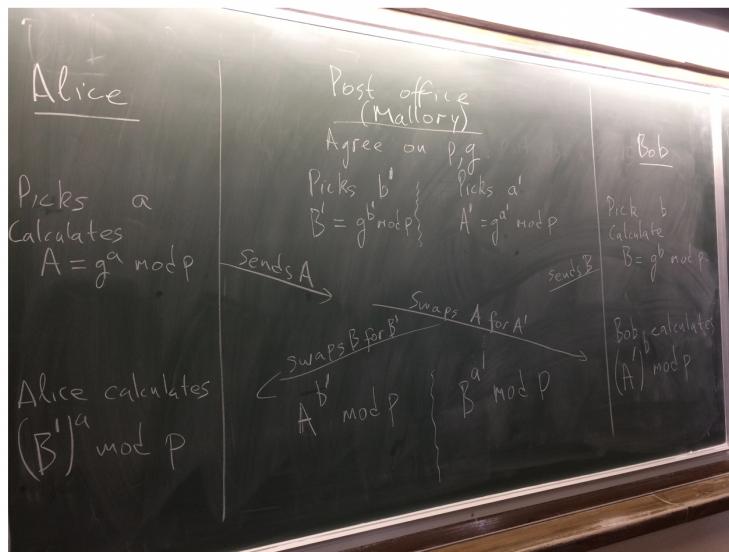


Figura 4.8: Ricerca matematica per creare algoritmi crittografici

Una volta progettato e testato un algoritmo utilizzando la pura matematica, bisogna implementarlo trasformando le equazioni matematiche in codice.

Anche in questo caso si lascia fare il lavoro ad esperti del settore, i quali possono testare anche la validità dell'implementazione e risovare eventuali falle.

Successivamente si impacchetta il codice in package open source e si rendono quindi disponibili all'uso a chiunque ne abbia la necessità all'interno dello sviluppo del proprio software, ad esempio in Indy ed Aries.

Ursa si occupa proprio di fornire questi package contenenti i diversi algoritmi crittografici: quando si invia un messaggio o una credenziale da un agente ad un altro ad esempio, le informazioni vengono cifrate attraverso una chiamata ad una funzione Ursa e decifrate attraverso una chiamata ad un'altra funzione Ursa.

Allo sviluppatore risulta molto semplice utilizzare gli algoritmi messi a disposizione, così da potersi concentrare sullo sviluppo di altri dettagli e non preoccuparsi di dover scrivere funzioni crittografiche in proprio. [5]

4.4 Hyperledger Indy

Hyperledger Indy implementa (basandosi sulla crittografia messa a disposizione da Urs) una Blockchain appositamente creata per l'identità digitale, permettendo di garantire certezza riguardo chi sta parlando a chi in una transazione.

Essendo sviluppata per un motivo così specifico, in che modo è realizzata e come si differenza da altre Blockchain?

Innanzitutto, come tutte le altre Blockchain anche Indy è immutabile, per cui una volta scritti i dati nel registro questi non possono essere più modificati.

I nodi comunicano tra di loro per raggiungere il consenso circa quali transazioni vanno scritte sul registro, inoltre questi devono dimostrare non solo di essere autorizzati a scrivere sul registro, ma anche di essere autorizzati a scrivere quella particola transazione (es. aggiornare un DIDDoc per cambiare una chiave pubblica al suo interno richiede un'autorizzazione).

Essendo costruita per uno specifico motivo (l'identità digitale) questa non supporta smart contracts o scambio di criptovalute, se non un token utilizzato per supportare particolari tipologie di pagamenti (es. scrittura sul registro, emettere una credenziale ecc.), oltre ad essere utile per prevenire attacchi Dos dove un malintenzionato può inondare la rete di scritture sul registro, impedendo altre operazioni. Indy è una Blockchain di tipo pubblico (chiunque può visualizzare le transazioni scritte sul registro) e permissioned (solo chi autorizzato può partecipare al consenso), a differenza di altre Blockchain in cui chiunque può partecipare al consenso (Permissionless, es. Blockchain) (Figura 4.9). [5]

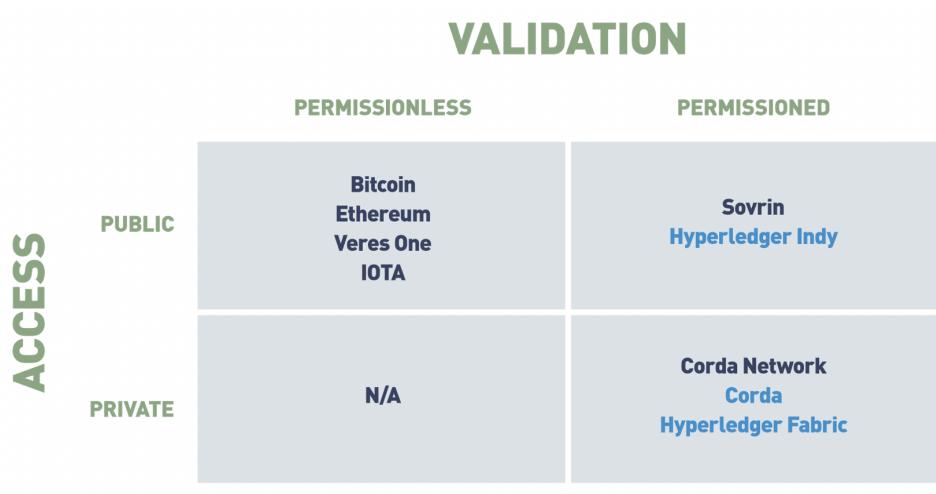


Figura 4.9: Tipologie varie di Blockchain

Il fatto di essere di tipo pubblico pone dei vincoli circa quali dati possono essere scritti sul registro. In particolare, solo i dati pubblici possono essere scritti (chiavi pubbliche, endpoint, schemas) e nessun altro, anche se cifrati.

I dati privati (es. credenziali, chiavi private) non vanno scritti sul registro pubblico poiché gli algoritmi di cifratura attualmente in uso potrebbero essere rotti nel futuro e siccome le informazioni sulla

blockchain sono pubbliche e non possono essere cancellate, queste potrebbero poi diventare di dominio pubblico.

A cosa serve quindi la Blockchain nel contesto dell'identità digitale?

Quando un Holder mostra una presentazione delle informazioni sulla propria identità ad un Verifier, si vuole che quest'ultimo non debba mettersi in contatto con l'Issuer per verificare l'autenticità delle informazioni, bensì questo deve prendere da un registro pubblico ed immutabile (la Blockchain) tutto ciò che occorre (chiavi pubbliche ed altro) per poter verificare autonomamente i dati mostratogli.

In dettaglio, ciò che dev'essere scritto sulla Blockchain (Figura 4.10) [5] è:

- **DIDs pubblici:** che vengono poi risolti in un DIDDoc contenente chiavi pubbliche ed endpoints. I DID pubblici appartengono solitamente agli Issuer, usati poi dai Verifier per effettuare le verifiche necessarie
- **Schemas:** definisce i campi della credenziale che verrà emessa dall'Issuer
- **Definizione delle credenziali:** specificano gli schemas usati, le chiavi pubbliche e i DIDs utilizzati
- **Registri per revoca:** opzionali, utilizzati solo in caso si voglia che l'Issuer possa anche revocare una credenziale precedentemente rilasciata

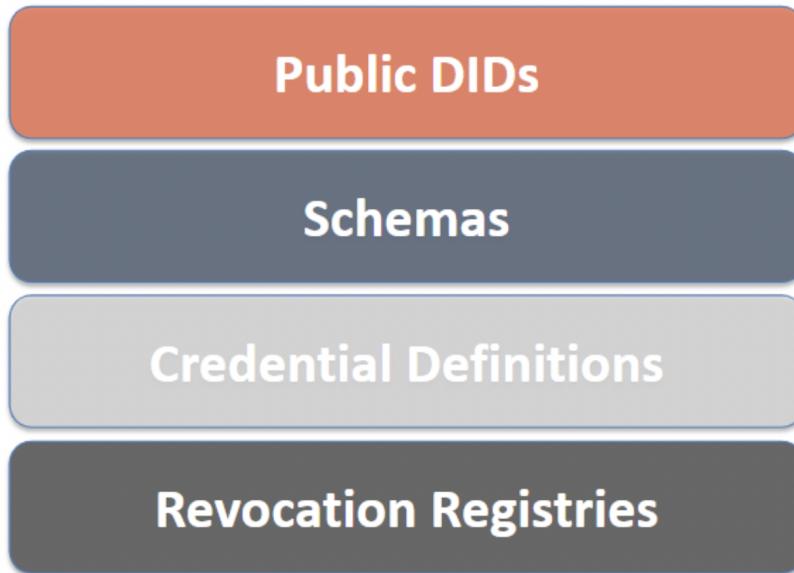


Figura 4.10: Cosa viene scritto sulla Blockchain

4.5 Hyperledger Aries

Visti in dettaglio la parte crittografica (Ursa) e la Blockchain (Indy), in questa sezione si guarda Aries nel dettaglio, iniziando dalla definizione di **agente**.

Un agente Aries è un software che consente ad una entità (persona, organizzazione o cosa) di assumere uno o più ruoli all'interno del Verifiable Credentials Model (Issuer, Holder o Verifier) e di interagire con altri agenti per scambiare credenziali o messaggi in generale utilizzando DID privati per abilitare comunicazioni peer to peer affidabili.

Esistono diverse tipologie di agenti, tra cui [5]:

- **Agenti personali:** utilizzati da persone normali, tipicamente consistono in applicazioni mobile o desktop e consentono al soggetto di custodire le proprie credenziali, trasmettere a terzi per la verifica, scambiare messaggi con altri agenti ecc.
- **Agenti Enterprise:** utilizzati da aziende o governi ed hanno lo scopo di rilasciare credenziali ai propri clienti, verificare le prove di questi ultimi ecc.
- **Agenti su Device:** agenti installati direttamente su sensori IoT, ad esempio è possibile installare un agente sulla propria auto che generino credenziali verificabili circa il kilometraggio, per poi poterle verificare in fase di vendita
- **Agenti di Routing:** fungono da intermediari per facilitare il flusso di messaggi tra altri agenti

Nonostante esistano diverse tipologie di agente, tutti condividono la stessa architettura di base, formata dalle seguenti componenti (Figura 4.11) [5]:

- **Servizio di gestione delle chiavi:** vale a dire un servizio di archiviazione sicura (wallet) nel quale si memorizzano le informazioni private e cifrate del soggetto (chiavi private, credenziali, altro).
Consiste in un database (solitamente un'implementazione SQLite) per agenti personali e PostgreSQL per agenti enterprise
- **Interfaccia per il messaging:** consente ad un agente di stabilire e gestire connessioni con altri agenti e di inviare/ricevere messaggi.
Aries usa il protocollo DIDComm
- **Interfaccia per il ledger:** consente all'agente di interagire col ledger, ad esempio per scrivere, leggere da esso.
Aries può essere implementato anche con altre implementazioni Blockchain, non soltanto Indy
- **Controller:** gestisce la logica di business dell'agente, ovvero come questo reagisce agli eventi e chi deve rappresentare

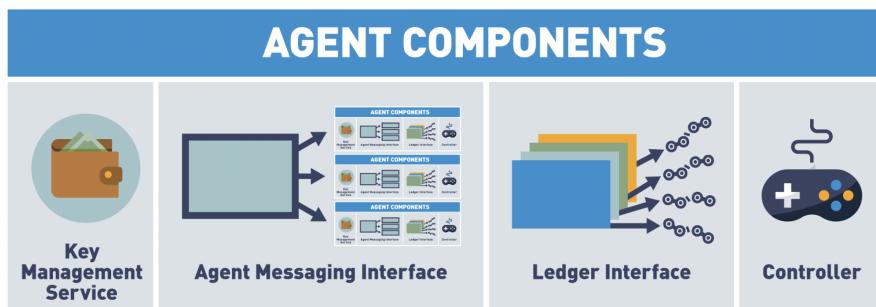


Figura 4.11: Architettura di un agente Aries

4.6 Protocolli comunicazione Aries

Per instaurare una connessione tra due agenti Aries si comincia da un **invito** (Figura 4.12). Quest'ultimo solitamente consiste in un codice qr scansionabile da un agente mobile o in una stringa JSON **in chiaro**, contenenti una serie di informazioni (chiavi pubbliche, endpoint) per poter cifrare un primo messaggio da inviare all'entità invitante.

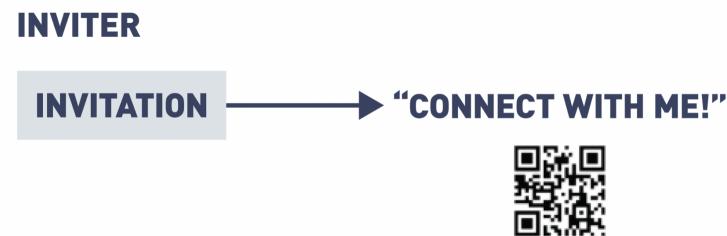


Figura 4.12: Richiesta dell'invitante a connettersi con lui mediante qr

Ottenuto l'invito, se l'entità invitata ha intenzione di accettarlo crea un nuovo DID e DIDDDoc per la relazione che si sta formando ed inserisce questi due in un messaggio "**richiesta di connessione**" che cifra con la chiave pubblica ed invia all'endpoint specificati nell'invito in chiaro di partenza (Figura 4.13).

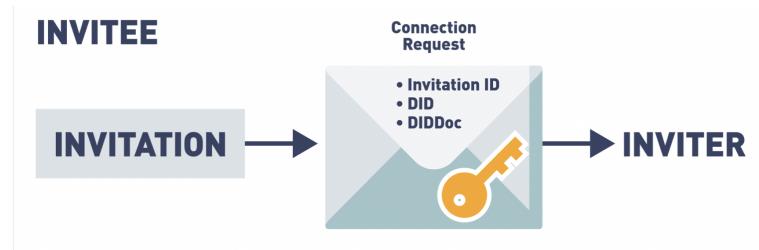


Figura 4.13: Richiesta di connessione da parte dell'invitato dopo aver accettato l'invito dell'invitante

L'invitante riceve la richiesta di connessione dall'invitato ed a questo punto crea anche lui un DID e un DIDDDoc e li inserisce in una "**risposta alla connessione**" ed invia all'invitato (Figura 4.14).

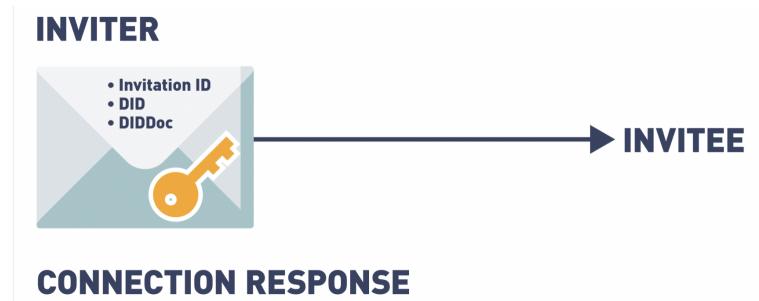


Figura 4.14: Risposta alla connessione da parte dell'invitante

A questo punto i due agenti sono connessi con un canale cifrato end-to-end e possono scambiarsi messaggi in modo sicuro. [6]

4.7 Implementazione Smart City con le tecnologie Hyperledger

In questa ultima sezione si descrive nel dettaglio come si è realizzata una soluzione per l'implementazione di una Città Smart che risolvesse i problemi presentati nel Capitolo 2 e che utilizzasse il paradigma SSI utilizzando la tecnologia Blockchain.

Tecnologie e progetti utilizzati:

- **von-network**: una rete Indy per il development di soluzioni
- **Aries clouagent python (ACA-py)**: framework per realizzare agenti Aries non-mobile
- **Docker**: piattaforma software per il deployment delle applicazioni in container (utilizzata per deployare la rete Indy e gli agenti Aries)
- **Arduino**: scheda programmabile utilizzata per simulare un sensore-server che raccoglie dati e li rende disponibili utilizzando i servizi REST

Nel dettaglio si descrivono le componenti del progetto:

von-network è una rete Indy (Figura 4.15) utilizzata per lo sviluppo di applicazioni che include anche un'interfaccia web per poter tenere sotto controllo lo stato dei nodi della rete.

```

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
gaetanodigenio@MacBook-Air-di-Gaetano von-network % ./manage start
WARNING: The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested
Starting von_node2_1 ... done
Starting von_node1_1 ... done
Starting von_webserver_1 ... done
Starting von_node3_1 ... done
Starting von_node4_1 ... done
Want to see the scrolling container logs? Run "./manage logs"
gaetanodigenio@MacBook-Air-di-Gaetano von-network %

```

Figura 4.15: Istanza di una rete Indy lanciata ed in esecuzione

ACA-py permette di realizzare gli agenti Aries e di scrivere la logica di business degli stessi nel controller.

In particolare si è fatto uso, in questo progetto, di tre agenti Aries: **Ente**, **Sensore** e **Municipalità**. **Ente**: rappresenta un'entità autorevole che si occupa di rilasciare credenziali (Issuer) contenenti una certificazione a specifici sensori.

La certificazione rilasciata è riconosciuta globalmente e garantisce che il sensore che l'ha ottenuta restituisca misurazioni affidabili e non alterate in alcun modo.

Sensore: è l'agente che rappresenta un sensore che prende misurazioni del vento.

L'entità sensore, in realtà, è formata da due componenti: il **sensore fisico**, realizzato utilizzando una scheda Arduino che mantiene un insieme di misurazioni in formato JSON e funge da server per poter ottenere queste misurazioni mediante servizi REST.

La seconda componente è l'**agente Aries Sensore**, che si occupa di interagire con gli altri agenti, scambiare credenziali e fornire presentazioni.

Municipalità: agente Aries che rappresenta il comune il quale, secondo il paradigma Smart City spiegato nel capitolo 2, funge da decision maker centrale ricevendo tutti i dati raccolti dai differenti sensori delle diverse organizzazioni.

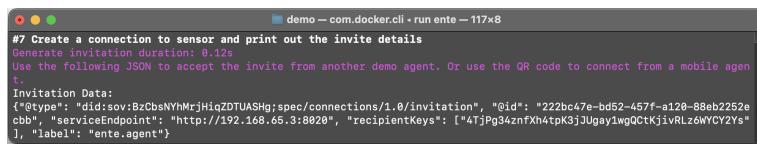
Lo scopo del progetto è il seguente: l'agente Ente rilascia una credenziale all'agente Sensore contenente una certificazione.

L'agente Sensore a sua volta instaura una connessione con l'agente Municipalità, il quale si comporta nel modo seguente: richiede a Sensore una presentazione in cui figura la certificazione precedentemente rilasciata da Ente e, nel caso la verifica andasse a buon fine, Municipalità utilizza i servizi REST per ottenere i dati dal sensore fisico (Arduino) e li stampa a schermo.

Naturalmente in caso in cui la verifica non dovesse avere successo, Municipalità non prenderà i dati dal server in quanto non può fidarsi della validità del sensore e dei dati raccolti da esso.

Nel dettaglio i vari step:

Inizialmente l'agente Ente mostra un invito in formato JSON (Figura 4.16)



```
#7 Create a connection to sensor and print out the invite details
Generate invitation duration: 0.12s
Use the following JSON to accept the invite from another demo agent. Or use the QR code to connect from a mobile agent.
Invitation Data:
{ "@type": "did:sov:BzCbsNYhMrjHiqZDTUAShG;spec/connections/1.0/invitation",
  "@id": "222bc47e-bd52-457f-a120-88eb2252e
ccb",
  "serviceEndpoint": "http://192.168.65.3:8020",
  "recipientKeys": ["4TjPg34znfXh4tpK9jJUgawQOtkjivRlzwCY2Ys"],
  "label": "ente.agent"}
```

Figura 4.16: Invito in formato JSON contenente chiave pubblica ed endpoint di Ente

L'agente Sensore copia ed incolla l'invito e si instaura la connessione tra i due (Figura 4.17), i quali ora possono scambiarsi messaggi cifrati end-to-end o rilasciare una credenziale (da parte di Ente)

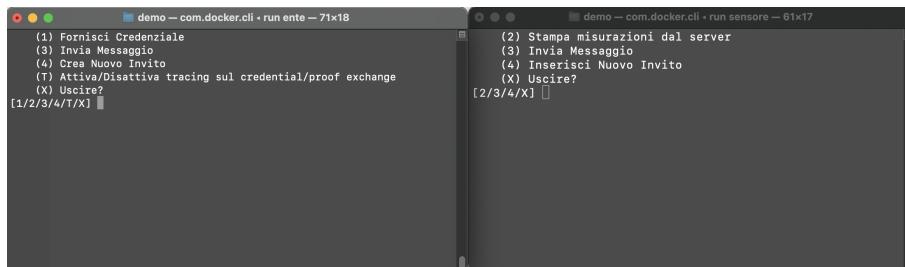
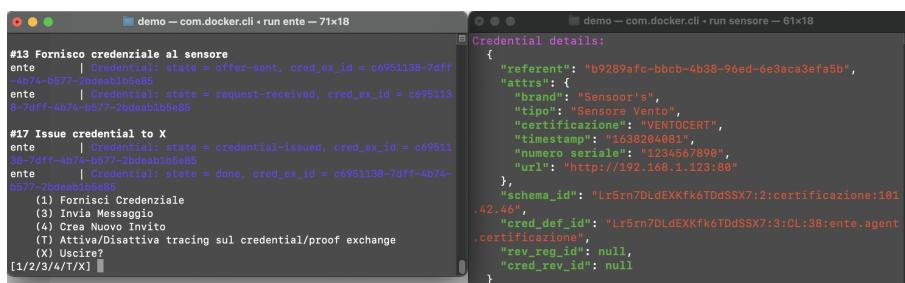


Figura 4.17: Funzionalità dell'agente Ente (schermata di sinistra) e dell'agente Sensore (schermata di destra)

Quando Ente rilascia la credenziale (Opzione 1 nella Figura 4.12), Sensore la riceve e la stampa a schermo (Figura 4.18)



```
#13 Fornisco credenziale al sensore
ente    | Credential: state = offer-sent, cred_ex_id = c6951138-7dff-4b74-b577-2bdeab1b5e85
ente    | Credential: state = request-received, cred_ex_id = c6951138-7dff-4b74-b577-2bdeab1b5e85
#17 Issue credential to X
ente    | Credential: state = credential-issued, cred_ex_id = c6951138-7dff-4b74-b577-2bdeab1b5e85
ente    | Credential: state = done, cred_ex_id = c6951138-7dff-4b74-b577-2bdeab1b5e85

Credential details:
{
  "referent": "b9289afc-bbbc-4b38-96ed-6e3aca3efaf5b",
  "atrs": {
    "brand": "Sensore Vento",
    "tipo": "Sensore Vento",
    "certificazione": "VENTOCERT",
    "timestamp": "1638204981",
    "numero_serie": "1234567890",
    "url": "http://192.168.1.123:89"
  },
  "schema_id": "Lr5rn7DLdEXKFk6T0dSSX7:2:certificazione:101
42_46",
  "cred_def_id": "Lr5rn7DLdEXKFk6T0dSSX7:3:CL:38:ente.agent
.certificazione",
  "rev_reg_id": null,
  "cred_rev_id": null
}
```

Figura 4.18: Agente Ente (sinistra) rilascia credenziale all'agente Sensore(destra)

Da questo momento in poi l'interazione avviene tra Sensore (che ha ricevuto la credenziale da Ente) e Municipalità, i quali instaurano una connessione sempre mediante invito (mostrato da Municipalità), la Figura 4.19 mostra le funzionalità disponibili per gli agenti.

The figure shows two terminal windows. The left window, titled 'demo — com.docker.cli • run sensore — 61x18', displays the following text:

```
"rfc23_state": "request-sent",
"connection_id": "6ff9fe9a2-115d-4373-8d96-f79f787eed32",
"accept": "auto",
"request_id": "3b08942db-e954-4571-9d91-f0c1e392e34b",
"state": "request"
}

Connect duration: 0.12s
Waiting for connection...
sensore.agent handle_connections response response-received
sensore.agent handle_connections active completed
Sensore | Connessione | Durata connessione: 0.55s
(2) Stampa misurazioni dal server
(3) Invia Messaggio
(4) Inserisci Nuovo Invito
(X) Uscire?
[2/3/4/X]
```

The right window, titled 'demo — com.docker.cli • run municipalita — 72x20', displays the following text:

```
Waiting for connection...
municipalita.agent handle_connections request request-received
municipalita.agent handle_connections response response-sent
municipalita.agent handle_connections active completed
Municipalita | Connessione
Municipalita | Connected
Municipalita | Richiesta Certificazione/Ottieni Dati Sensore
(2) Fornisci Credenziale
(3) Invia Messaggio
(4) Crea Nuovo Invito
(X) Uscire?
[1/2/3/4/X]
```

Figura 4.19: Funzionalità agente Sensore (sinistra) e agente Municipalità (destra) dopo aver instaurato la connessione

Selezionando l'opzione 1 nella Figura 4.19 nella schermata dell'agente Municipalità (destra), quest'ultimo richiede una prova che il Sensore sia stato certificato.

Quest'ultimo crea la presentazione contentente le informazioni richieste e la invia all'agente Municipalità, il quale effettua che sia tutto in regola ed a quel punto prende i dati dal sensore fisico (Arduino) e li stampa a schermo.

La figura 4.20 mostra il tutto

The figure shows two terminal windows. The left window, titled 'demo — com.docker.cli • run sensore — 61x18', displays the following text:

```
#24 Query for credentials in the wallet that satisfy the proof request
#25 Generate the proof

#26 Send the proof to X: {"indy": {"requested_predicates": {}, "requested_attributes": {"0_numero seriale_uuid": {"cred_id": "43836512-bc2f-4a51-b5dc-2238dbfbfab", "revealed": true}, "0_certificazione_uuid": {"cred_id": "43836512-bc2f-4a51-b5dc-2238dbfbfab", "revealed": true}, "0_url_uuid": {"cred_id": "43836512-bc2f-4a51-b5dc-2238dbfbfab", "revealed": true}}, "self_attested_attributes": {}}}
Sensore | Presentazione: state = presentation-sent, pres_ex_id = 2e3a575b-3496-404f-8d93-26add521787b
Sensore | Presentazione: state = done, pres_ex_id = 2e3a575b-3496-404f-8d93-26add521787b
(2) Stampa misurazioni dal server
(3) Invia Messaggio
```

The right window, titled 'demo — com.docker.cli • run municipalita — 72x37', displays the following text:

```
#20 Richiesta proof di certificazione dal sensore
Municipalita | Presentazione: stato = request-sent, pres_ex_id = 01f04c4d-db73-48f2-863a-4d4bc08613f1
Municipalita | Presentazione: stato = presentation-received, pres_ex_id = 01f04c4d-db73-48f2-863a-4d4bc08613f1
#21 Processo la proof presentata dal sensore
#22 Controlla se la proof è valida, in tal caso prendo i dati dal server del sensore
Municipalita | Presentazione: stato = done, pres_ex_id = 01f04c4d-db73-48f2-863a-4d4bc08613f1
Municipalita | Proof = true

#23.1 Ricevuta proof di certificazione, check dei claims
Municipalita | numero seriale: 1234567890
Municipalita | certificazione: VENTOCERT
Municipalita | url: http://192.168.5.94:80
Municipalita | schema_id: 5jHhK2VorgFj1Nh5mYKKsJ:2:certificazione:47.6.3
Municipalita | cred_def_id 5jHhK2VorgFj1Nh5mYKKsJ:3:CL:48:ente.agent.certificazione
Stampo dati server sensore
{'luogo': 'paestum', 'velocità': '3 km/h', 'data': '30-07-2021', 'ora': '09:00'}
{'luogo': 'paestum', 'velocità': '5 km/h', 'data': '30-07-2021', 'ora': '10:00'}
{'luogo': 'paestum', 'velocità': '5 km/h', 'data': '30-07-2021', 'ora': '11:00'}
{'luogo': 'paestum', 'velocità': '6 km/h', 'data': '30-07-2021', 'ora': '12:00'}
{'luogo': 'paestum', 'velocità': '5 km/h', 'data': '30-07-2021', 'ora': '13:00'}
{'luogo': 'paestum', 'velocità': '5 km/h', 'data': '30-07-2021', 'ora': '14:00'}
{'luogo': 'paestum', 'velocità': '5 km/h', 'data': '30-07-2021', 'ora': '15:00'}
```

Figura 4.20: Proof di certificazione inviata e confermata, dati ottenuti dal server e stampati a schermo

Il progetto così terminato, ha mostrato come poter autenticare un'entità in un contesto decentralizzato grazie all'utilizzo della tecnologia Blockchain e della Self Sovereign Identity.

Bibliografia

- [1] CHRISTIAN ESPOSITO, MASSIMO FICCO, BRIJ BHOOSHAN GUPTA: *Blockchain-based authentication and authorization for smart city applications.*
Information Processing and Management, 2021.
- [2] HEATHER VESCENT, KALIYA YOUNG, KIM HAMILTON DUFFY, MARKUS SABADELLO, DMITRI ZAGIDULIN, JUAN CABALLERO: *A comprehensive guide to Self Sovereign Identity.*
The Purple Tornado, 2019.
- [3] ANNA BERNASCONI, PAOLO FERRAGINA, FABRIZIO LUCCIO: *Elementi di Crittografia.*
Pisa University Press, 2015.
- [4] VERIFIABLE CREDENTIALS DATA MODEL v1.1 .
URL = <https://www.w3.org/TR/vc-data-model/>.
W3C Recommendation, 2021.
- [5] LFS172X: INTRODUCTION TO HYPERLEDGER SOVEREIGN IDENTITY BLOCKCHAIN SOLUTIONS: INDY, ARIES URSA.
URL = <https://learning.edx.org/course/course-v1:LinuxFoundationX+LFS172x+2T2021/home>.
Linux Foundation, 2021.
- [6] LFS173X: BECOMING A HYPERLEDGER ARIES DEVELOPER
URL = <https://learning.edx.org/course/course-v1:LinuxFoundationX+LFS173x+3T2021/home>.
Linux Foundation, 2021.
- [7] MICHAEL CROSBY (GOOGLE), NACHIAPPAN (YAHOO), PRADHAN PATTANAYAK (YAHOO), SANJEEV VERMA (SAMSUNG RESEARCH AMERICA), VIGNESH KALYANARAMAN (FAIRCHILD SEMICONDUCTOR).
BlockChain Technology .
Sutardja Center for Entrepreneurship Technology, Berkley University of California, 2015.
- [8] DECENTRALIZED IDENTIFIERS (DIDs) v1.0 .
URL = <https://www.w3.org/TR/did-core/>.
W3C Recommendation, 2021.
- [9] NATHAN GEORGE, CTO, SOVRIN FOUNDATION, AND HYPERLEDGER ARIES SPONSOR AND CONTRIBUTOR
Announcing Hyperledger Aries, infrastructure supporting interoperable identity solutions!
URL = <https://www.hyperledger.org/blog/2019/05/14/announcing-hyperledger-aries-infrastructure-supporting-interoperable-identity-solutions>.
Hyperledger Aries Blog, 2019