Politecnico
di Torino

Politecnico di Torino
Ingegneria Matematica
A.a 2023/2024

Gestione sicura e flessibile dei dati sanitari mediante la tecnologia blockchain

Relatore:
Prof. Danilo Bazzanella

Correlatore:
Marco Bazzani

Candidato:
Francesca Portadibasso

Abstract

Nel contesto in continua evoluzione dovuto all'innovazione delle tecnologie digitali, il settore sanitario ha subito una profonda trasformazione. Questo cambiamento ha portato nuove prospettive e sfide nella gestione e condivisione delle informazioni mediche. In questo scenario, l'uso delle **Electronic Health Record** (EHR) ha rivoluzionato il modo in cui le tali informazioni vengono registrate, archiviate e condivise tra operatori sanitari e pazienti. Rimane cruciale tuttavia affrontare le questioni legate alla sicurezza, all'integrità e alla protezione della privacy di tali dati.

Le tecnologie legate alla **blockchain** emergono come una soluzione promettente per affrontare le sfide legate all'archiviazione e alla condivisione sicura degli EHR. Questa tesi esplora l'applicazione della tecnologia blockchain alla gestione affidabile e trasparente dei dati sanitari, concentrando l'attenzione su **Hyperledger Fabric**, come infrastruttura per veicolare le informazioni.

Saranno esaminate le nozioni fondamentali relative agli EHR, inclusi i loro vantaggi, verrà analizzato il ruolo della blockchain, in particolare come Hyperledger Fabric possa essere utilizzato per costruire una rete ideale. La scelta per la gestione delle EHR è ricaduta su Hyperledger Fabric in quanto permette di progettare una blockchain autorizzata, la quale offre diversi vantaggi, tra i quali la trasparenza, l'immutabilità e la privacy. Ogni nodo presente all'interno della rete non è anonimo, favorendo, in tal modo, la rintracciabilità di eventuali attacchi e, nello stesso tempo, un incentivo a svolgere correttamente il proprio ruolo. Inoltre, dal momento che ogni azione portata avanti sulla blockchain deve essere convalidata e approvata da coloro che ne hanno diritto, il sistema consente di riconoscere in anticipo chi non ha buone intenzioni e di evitare che chiunque abbia accesso ai dati dei pazienti.

Nel caso di studio che verrà illustrato, verrà presentata una rete blockchain composta da 2 ospedali, che utilizzano un canale per lo scambio degli *smart contract*, legati alla creazione, all'acquisizione e all'archiviazione delle EHR, azioni che in un contesto reale potrebbero essere molto utili per velocizzare e rendere più sicuri tutti i processi che prevedono la comunicazione tra due o più istituzioni sanitarie. Sarà, quindi approfondito il processo di progettazione e di implementazione degli smart contract per garantire il trasferimento sicuro degli EHR all'interno di una rete sanitaria, in quanto rappresentano uno strumento fondamentale all'interno di una rete Fabric. Si continua allargando la rete per rendere il tutto più verosimile, aggiungendo un terzo ospedale, ovvero una terza organizzazione, alla rete e anche un secondo canale, per permettere comunicazioni private, sfruttando così tutte le potenzialità della blockchain scelta.

La combinazione degli EHR e della blockchain costituisce un ambito di ricerca altamente promettente, con il potenziale per rivoluzionare la gestione dei dati nel settore sanitario.

Indice

Capitolo 1

Introduzione

Il settore medico sanitario è sempre stato argomento di discussione nell'ambito scientifico. In particolare, il progresso tecnologico ha modificato profondamente questo settore, sia per quanto riguarda l'applicazione delle soluzioni nell'ambito medico-scientifico, sia nell'ambito organizzativo del sistema. L'oggetto di discussione della tesi riguarda proprio questo secondo aspetto, e cioè l'utilizzo di tecnologie emergenti per facilitare e rendere più sicura e affidabile la circolazione delle informazioni poste al servizio della medicina e della ricerca scientifica ospedaliera.
Tra le varie tecnologie presenti, la **blockchain**, ovvero una tecnologia di registro distribuito che consente la creazione di un registro sicuro e immutabile di transazioni, è tra quelle che sta riscontrando il maggiore successo. Questa tecnologia trova applicazione in vari ambiti e settori scientifici e disciplinari, ma molto interessante appare la sua applicazione in ambito medico sanitario. [1]:

- **Electronic Health Record**: la blockchain consente di mettere in connessione sicura tutta una serie di dati e di informazioni che il sistema sanitario (ma anche più semplicemente singole strutture ospedaliere) generalmente produce; ed inoltre, consente ai diversi fornitori di assistenza sanitaria (medici, paramedici, ricercatori), di avere accesso a tali dati anche se non affiliati tra loro. In questo modo, si migliora il coordinamento dell'assistenza sanitaria, consentendo un accesso più efficiente e sicuro alle informazioni mediche cruciali dei pazienti da parte dei professionisti sanitari.

- **Automazione delle transazioni sanitarie**: la blockchain può automatizzare il processo di pagamento delle transazioni sanitarie mediante l'utilizzo degli smart contract,

- **Interoperabilità dei dati dei pazienti**: la blockchain consente una raccolta più efficace delle informazioni necessarie per supportare le iniziative di salute pubblica della popolazione nei grandi sistemi sanitari.

- **Accesso online alle cartelle cliniche dei pazienti**: la blockchain consente ai pazienti di accedere in modo più semplice, sicuro ed efficace alle proprie cartelle cliniche direttamente online.

- **Gestione delle operazioni mediche e della logistica sanitaria**: la blockchain può migliorare la gestione dei contratti sanitari e ridurre i costi di transazione consentendo funzionalità come il monitoraggio e l'esecuzione dei contratti in tempo reale, migliorando così l'efficienza delle operazioni mediche e logistiche nel settore sanitario.

Capitolo 2

Electronic Health Record

Le **Electronic Health Record** (EHR) rappresentano lo strumento tecnologico-informativo che ha maggiore probabilità di successo. Esse contengono la storia clinica di un paziente, ma anche dati, previsioni e informazioni riguardanti una possibile cura a cui il paziente in questione è sottoposto.[1]

La tecnologia blockchain fornisce una solida soluzione per la costruzione di una banca dati condivisa, in cui ogni soggetto autorizzato, compreso il paziente stesso, può accedere, esaminare e aggiornare un registro delle informazioni sanitarie protette (PHI), indipendentemente dalla rete a cui sono collegati. [2]

Attualmente, le informazioni sui pazienti sono conservate in modo differente tra diverse istituzioni sanitarie, ospedali e compagnie di assicurazioni, spesso senza un accesso completo a un database condiviso per i pazienti. Utilizzando la blockchain per archiviare le cartelle cliniche, il sistema potrebbe mettere a disposizione dei ricercatori i dati strutturati dei pazienti. Questo potrebbe portare a una riduzione dei costi di archiviazione e a un aumento dell'efficienza complessiva del sistema.[3]

Archiviare le EHR sulla blockchain permette di trovare soluzioni trasparenti in grado di infondere fiducia tra quanti operano sulla catena delle informazioni, in quanto la cronologia delle transazioni sarà condivisa e immutabile da un lato, e i dati sensibili di un paziente gestiti in modo sicuro e affidabile, dall'altro. Inoltre, se un paziente ha necessità di cambiare ospedale per un qualsiasi motivo, il trasferimento di tutte le informazioni che lo riguardano avviene in maniera lenta, in quanto i dati medici non vengono inviati per email poiché è un rischio per la sicurezza. Avere quindi un'entità decentralizzata come la blockchain, condivisa, immutabile e trasparente, è lo strumento che permettere di raggiungere il consenso senza fare affidamento su un unico soggetto, garantisce maggiore sicurezza dei dati.
Quindi, una tale tecnologia, può far fronte a problemi come *frequenti visite a ospedali non collegati*, *ripetizione di test a causa della mancanza di accesso ai dati*, *gestione complessa e riservatezza dei dati sensibili* o ancora *difficoltà nella condivisione dei dati tra operatori sanitari.*

2.1 Problemi con le Electronic Health Record

Al giorno d'oggi è ancora difficile mettere in pratica in maniera efficiente delle soluzioni basate sulla blockchain, in quanto sorgono alcuni problemi. Ad esempio vi è mancanza di interoperabilità tra diverse soluzioni basate su blockchain, in quanto non esistono standard comuni; o ancora la scalabilità per gestire grandi volumi di dati clinici; la partecipazione dei pazienti (non tutti sono in grado o disposti a gestire i propri dati) e anche la mancanza di incentivi[4].
Il problema però di maggiore rilievo, che le blockchain affrontano, riguarda il rischio di accesso

non autorizzato ai dati, dovendo spesso fare affidamento su terze parti per le transazioni. In tal caso si pone la necessità di implementare e di mettere a punto le blockchain, al fine di garantire la riservatezza dei dati quando questi vengono trasferiti al di fuori della catena.

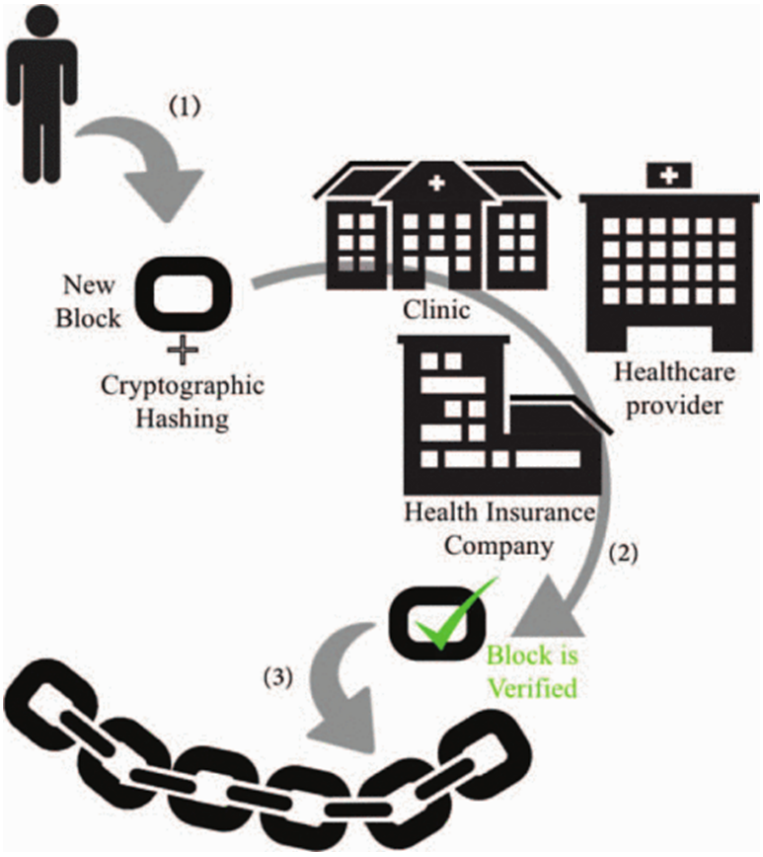

Figura 2.1: Schema[5]

Del resto, come si può vedere nello schema riportato nella figura 2.1 , aggiungere un nuovo blocco ad una EHR comporta il coinvolgimento di diversi soggetti, dal medico che chiede l'aggiunta del blocco, agli enti sanitari selezionati che verificano e autorizzano l'inserimento, alle compagnie assicurative sanitarie, fino all'aggiunta del blocco alla catene delle informazioni.

Questo può portare a problemi di sicurezza, come l'attacco **Sybil**, nel quale un singolo aggressore (o un gruppo di aggressori) assume il controllo della rete fingendo di essere più nodi. In questo modo sarebbe possibile che, chi è coinvolto nell'attacco, assuma il controllo delle rete facendo si che le sue decisioni prevalgano poiché, al nodo attaccato dalla rete della catena, viene impedito di partecipare a qualsiasi attività. Un'altra preoccupazione riguarda la deduzione di dati privati: se i valori hash delle chiavi private dei pazienti fossero archiviati nel blocco stesso, un medico potrebbe dedurre informazioni sensibili, compromettendo così la privacy dei pazienti. [5]

In letteratura però, si trovano soluzioni a queste sfide:

- per gestire il grande volume di dati clinici, si è iniziato a considerare l'archiviazione dei dati effettivi sul cloud, mentre sulla blockchain vengono memorizzati solo i puntatori ai dati.[4]
- per cercare di affrontare il problema della scalabilità, si parla di archiviare i dati *off-chain*, in questo modo all'interno della blockchain si trovano solo informazioni sui dati e su come è possibile avervi accesso. Sempre grazie a questo sistema, i dati possono essere cancellati permanentemente e si rispetta il diritto all'oblio: sulla blockchain rimarrà solo il puntatore a tali dati che sarà privo di utilità.[4]
- per migliorare le prestazioni e quindi permettere un'elaborazione dei dati più veloce, è possibile dare le autorizzazioni di convalida solo ad alcuni nodi.[4]
- è possibile implementare una blockchain come lista di controllo degli accessi, in questo modo si possono prevenire gli accessi non autorizzati ed eliminare la deduzione dei dati privati dei pazienti.[5]
- per affrontare l'attacco Sybil, che non può essere completamente eliminato ma può essere mitigato, si è pensato di costringere ogni nodo minerario a competere nella risoluzione della *Proof of Work* prima di poter aggiungere un nuovo blocco alla blockchain; così facendo, da un lato, sarà più difficile per un attaccante assumere il controllo della rete e, dall'altro, si consentirà di rilevare un attacco Sybil, poiché l'avversario dovrebbe controllare più del 50% della rete per avere successo.[5]
- utilizzando la blockchain autorizzata, o una blockchain privata o di consorzio al contrario di una pubblica, si proteggono maggiormente i dati e la privacy, oltre a prevenire diverse minacce alla sicurezza.[4]

D'altro canto, dal momento che i dati sanitari dei pazienti sono estremamente delicati e anche la semplice comunicazione tra medici e pazienti potrebbe rivelare informazioni sensibili, tali da minacciarne la privacy, utilizzare una b. autorizzata presenta, soprattutto da questo punto di vista, numerosi vantaggi, di seguito sintetizzati:[6]:

- in un sistema senza autorizzazione, ad esempio l'anonimato degli utenti potrebbe portare a un trattamento impersonale e all'abuso dei dati sensibili.
- la tempestività degli aggiornamenti delle informazioni sul trattamento dei pazienti è cruciale per garantire un'assistenza efficace.
- l'esecuzione delle transazioni, come l'aggiornamento delle autorizzazioni per l'accesso ai dati sanitari o la condivisione di informazioni per la ricerca, potrebbe comportare costi che limitano l'accessibilità del sistema.
- l'implementazione del sistema potrebbe prevedere l'estensione dei ruoli e delle funzionalità oltre a medici e pazienti, a seconda delle esigenze specifiche.
- la progettazione di un'architettura basata su blockchain per la gestione dei dati nella radioterapia oncologica.
- il servizio Membership gestisce la registrazione degli utenti, verificando l'autenticità dei medici tramite la Banca Dati Nazionale dei Medici.
- i dati del paziente sono crittografati per garantire la riservatezza, e le chiavi di crittografia vengono condivise solo con gli utenti autorizzati.

- i dati del paziente sono memorizzati sia on−chain che off−chain, garantendo sicurezza e disponibilità.
- le transazioni e i dati sono protetti tramite firme digitali e crittografia, garantendo l'integrità e la riservatezza delle informazioni.
- una piattaforma cloud permette l'archiviazione dei dati e le API basate sui ruoli consentono l'accesso ai dati da qualsiasi nodo nella rete.
- il sistema è progettato per gestire un grande numero di utenti e nodi, con protocolli di consenso e dimensioni dei blocchi regolabili per garantire la scalabilità.

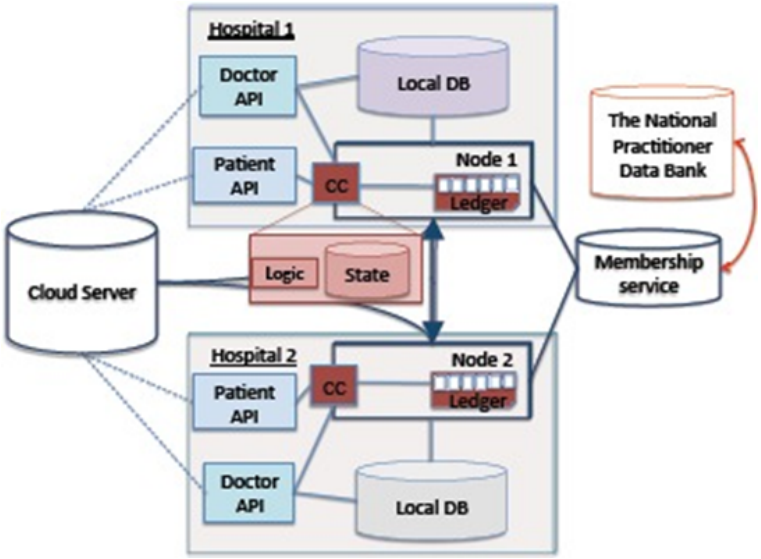

Figura 2.2: Blockchain applicata ad un caso di studio [6]

Attualmente, nel settore sanitario, ci sono diverse lacune riguardanti le cartelle cliniche, soprattutto per quanto riguarda l'accesso, l'elaborazione e l'analisi dei dati sanitari provenienti da diverse fonti da parte dei vari attori del settore. Ad esempio, non esistono cartelle cliniche universali che permettano il trasferimento agevole di test di laboratorio, immagini diagnostiche o informazioni sui farmaci di un paziente tra le diverse visite. Anche se la maggior parte dei documenti medici è passata dalla forma cartacea a quella digitale, i dati risiedono attualmente in silos[1], rendendo difficile per i pazienti e gli operatori sanitari conciliare le informazioni tra i vari fornitori. Questa frammentazione è complicata dalla mancanza di una fonte centralizzata che colleghi tutti i dati sanitari di un individuo e dall'assenza di un ordine definito in cui sono stati registrati.
Inoltre, con l'aumento dei dati sanitari digitali, la sicurezza delle informazioni è diventata una priorità urgente. Le violazioni dei dati e i furti di identità medica sono in aumento, spingendo gli operatori sanitari a cercare modi più efficaci per proteggere le informazioni personali dei pazienti[1].

[1]I silo sono sistemi di storage e di gestione dei dati separati tra loro che risultano inaccessibili ad altre aree dell'azienda. Si formano quando i dati vengono conservati in sistemi o database distinti che non sono in grado di comunicare o condividere le informazioni tra loro. In questo caso, si intende che i dati sanitari sono dispersi e non integrati tra diverse fonti o sistemi, creando una mancanza di coesione e interoperabilità.[7]

2.2 Cos'è una Electronic Health Record

Le **Electronic Health Record** (EHR) sono dei dossier digitali che contengono informazioni sensibili e cruciali per la diagnosi e il trattamento medico. Questi dati rappresentano una risorsa preziosa per migliorare la qualità e l'efficienza dei servizi sanitari. Una Electronic Health Record, creata e mantenuta per tutta la vita del paziente, viene solitamente conservata e condivisa tra diverse strutture sanitarie, come ospedali, cliniche e medici.

Per comprendere al meglio la struttura e la gerarchia di una EHR basata su blockchain, è possibile fare affidamento alla figura 2.3:

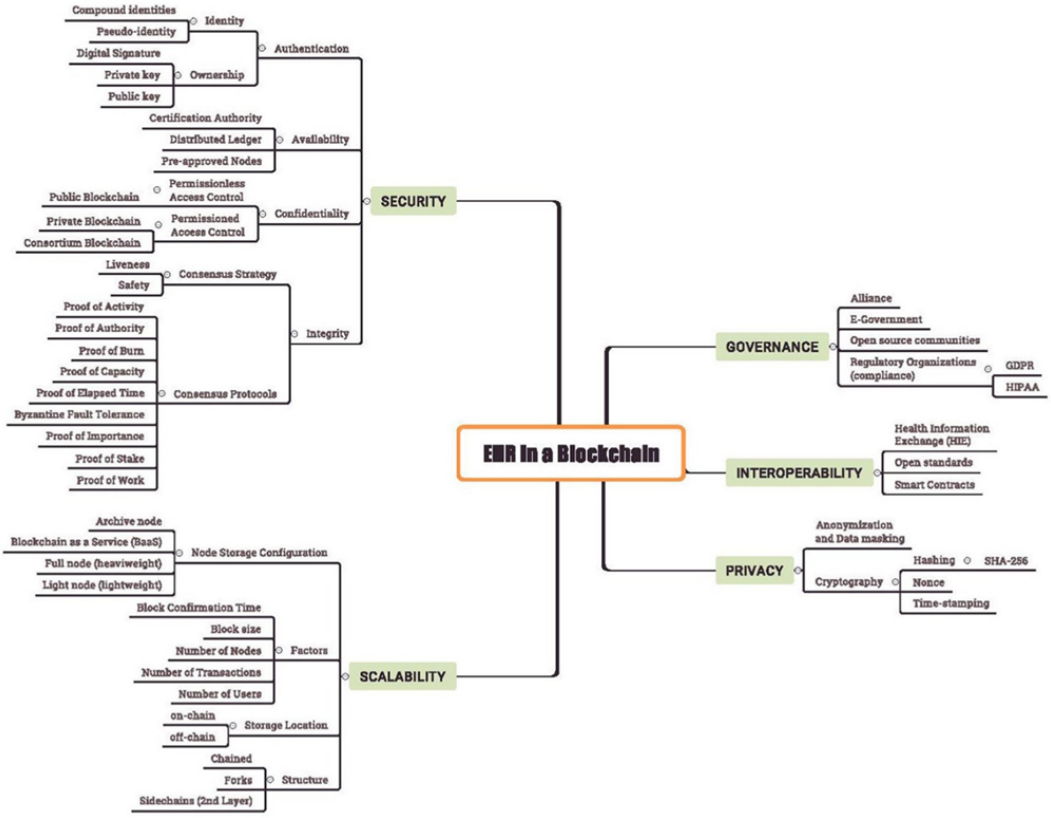

Figura 2.3: Tassonomia di un'EHR[8]

La classificazione proposta[8] si concentra su cinque caratteristiche principali: governance, interoperabilità, privacy, scalabilità e sicurezza. Ognuna di essa è ulteriormente dettagliata per affrontare le proprietà specifiche delle Electronic Health Record.

La letteratura in materia declina ogni singola caratteristica, delineando per ciascuna di essa non solo la definizione ma anche gli attori chiamati a rendere operativo, concreto ed efficace il sistema.

GOVERNANCE:

- **Alliance:** un accordo o una partnership tra due o più entità, come le organizzazioni.
- **E-government:** o governo elettronico, fa riferimento all'uso delle tecnologie dell'informazione e della comunicazione (TIC) per fornire servizi pubblici ai cittadini o alle imprese e ad altre agenzie governative.
- **Open source communities:** le comunità open source sono gruppi di persone che collaborano per sviluppare, migliorare e supportare il software o altri progetti.
- **Regulatory organization (compliance):** organizzazione incaricata di stabilire e far rispettare le norme e i regolamenti in un determinato settore o ambito.

INTEROPERABILITÁ:

- **Health Information Exchange (HIE):** scambio di informazioni sanitarie; si riferisce al processo di condivisione elettronica sicura e interoperabile delle informazioni sanitarie tra diverse organizzazioni sanitarie, come ospedali, cliniche, laboratori diagnostici, farmacie e altri fornitori di assistenza sanitaria.
- **Open Standard:** fanno riferimento a specifiche tecniche o protocolli che sono pubblicamente disponibili e possono essere utilizzati liberamente da chiunque senza restrizioni di licenza o proprietà.
- **Smart Contract:** è un codice informatico che viene eseguito automaticamente quando sono soddisfatte determinate condizioni predefinite. Questi contratti sono immutabili e vengono eseguiti su una blockchain, consentendo alle transazioni di essere automatizzate, tracciate e verificate in modo sicuro e trasparente senza la necessità di intermediari.

PRIVACY:

- **Anonymization and Data Masking:** tecniche utilizzate per proteggere la privacy e l'anonimato dei dati sensibili.
- **Crittografia:** processo di conversione di dati leggibili in un codice indecifrabile chiamato crittogramma, utilizzando algoritmi matematici complessi e chiavi di crittografia. Viene utilizzata per garantire la confidenzialità e la sicurezza dei dati durante la loro trasmissione o archiviazione, impedendo agli utenti non autorizzati di accedere o interpretare i dati.

SCALABILITÀ:

- **Node Storage Configuration:** fa riferimento alla configurazione di memorizzazione dei nodi all'interno di una rete blockchain o di un sistema distribuito. Questa configurazione determina come e dove vengono archiviati i dati su ciascun nodo della rete.
- **Factors:** vari elementi o fattori che influenzano la progettazione, l'implementazione o l'utilizzo di un sistema EHR.
- **Storage Location:** fa riferimento al luogo fisico o virtuale in cui vengono memorizzati i dati relativi alle Electronic Health Record. Può essere un server locale presso un'organizzazione sanitaria, un servizio di cloud storage o un sistema distribuito come una blockchain.
- **Structure:** fa riferimento alla disposizione o all'organizzazione dei dati all'interno di una Electronic Health Record o di un sistema EHR. Questa struttura determina come i dati sono organizzati, categorizzati e presentati agli utenti autorizzati.

SICUREZZA:

- **Riservatezza:** fa riferimento alla protezione dei dati sensibili e personali contenuti nelle EHR. Garantire la riservatezza significa assicurare che solo le persone autorizzate possano accedere a tali informazioni e che esse non vengano divulgate o accessibili a individui non autorizzati.
- **Integrità:** fa riferimento alla precisione, completezza e coerenza delle informazioni contenute nelle Electronic Health Record. Garantire l'integrità dei dati significa assicurarsi che le informazioni non siano state alterate, danneggiate o modificate in modo non autorizzato.
- **Disponibilità:** fa riferimento alla capacità di accedere e utilizzare le informazioni contenute nelle Electronic Health Record quando necessario. Assicurare la disponibilità dei dati significa garantire che le informazioni siano accessibili agli utenti autorizzati in modo tempestivo e affidabile.
- **Autenticazione:** processo attraverso il quale viene verificata l'identità di un utente o di un sistema prima di concedere l'accesso alle informazioni. Garantire l'autenticazione significa assicurarsi che gli utenti siano chi dicono di essere e che abbiano il permesso di accedere alle Electronic Health Record.

2.2.1 CCE, EMR, EHR ed FSE

Quando si affronta un tema così complesso come quello della EHR, è importante definire non solo gli ambiti operativi che abbiamo appena visto, ma è necessario anche stabilire una tassonomia condivisa a livello nazionale ed internazionale, sia nell'ambito scientifico che medico-sanitario. Spesso, soprattutto in Italia, termini come CCE, EMR, EHR ed FSE sono usati per indicare lo stesso oggetto e sono chiamate a svolgere funzioni diverse.[9].

Cartella clinica elettronica (CCE)

In ambito ospedaliero, la **Cartella Clinica Elettronica (CCE)** rappresenta un passo avanti rispetto alla tradizionale Cartella Clinica Cartacea (CCC); è uno strumento che consente la gestione strutturata e organizzata dei dati relativi alla storia clinica di un paziente, sia durante il ricovero che in ambito ambulatoriale. Essa supporta i processi clinici, inclusi quelli diagnostici e terapeutici, così come l'assistenza fornita durante ogni singolo episodio di cura.

Le funzioni principali della Cartella Clinica Elettronica possono essere riassunte nelle seguenti attività fondamentali:

- **Supporto alla pianificazione e valutazione delle cure:** aiuta a predisporre il piano diagnostico-terapeutico-assistenziale e a valutare l'appropriatezza delle cure erogate rispetto agli standard.
- **Strumento di comunicazione e integrazione operativa:** serve come strumento di comunicazione per facilitare l'integrazione tra i professionisti sanitari coinvolti in un piano di cura specifico, garantendo così la continuità assistenziale.
- **Fonte dati per la ricerca e la formazione:** fornisce dati per studi scientifici, ricerche cliniche, formazione e aggiornamento degli operatori sanitari, valutazione delle attività assistenziali ed esigenze amministrativo-legali.

- **Protezione legale degli interessi:** consente di tracciare tutte le attività svolte per proteggere gli interessi del paziente, dei medici e dell'azienda sanitaria, garantendo la rintracciabilità delle responsabilità, della cronologia e delle modalità di esecuzione.

La CCE è un sistema informatico che contiene tutte le informazioni necessarie per gestire un processo diagnostico-terapeutico-assistenziale. Queste informazioni includono:

- Assessment clinico e infermieristico.
- Esame obiettivo.
- Diario clinico integrato (medico e infermieristico).
- Referti di prestazioni ambulatoriali e di altri esami diagnostico-specialistici.
- Gestione del ciclo del farmaco e delle attività di nursing.
- Gestione del percorso chirurgico.
- Gestione della lettera di dimissione con eventuali suggerimenti per il Medico di Medicina Generale e Pediatra di Libera Scelta e di continuità assistenziale.
- Documenti amministrativi come i consensi informati.

Un errore comune è pensare che implementare la Cartella Clinica Elettronica (CCE) equivalga semplicemente a trasferire la tradizionale Cartella Clinica Cartacea in formato digitale, ma questa ipotesi è estremamente riduttiva poiché essa è concepita come un sistema informatico integrato aziendale, trasversale alle varie tipologie di regime clinico-sanitario e ai processi di cura. Deve rispettare i requisiti e le funzioni della cartella clinica cartacea e risolvere le criticità ad essa associate, offrendo l'opportunità di aumentarne il valore attraverso l'integrazione con altri strumenti informatici.

Electronic Medical Record (EMR)

L'**Electronic Medical Record (EMR)** è più di una semplice Cartella Clinica Elettronica (CCE). È un'architettura progettata per gestire in modo integrato i flussi informativi nell'ambito clinico-sanitario di un'azienda, al fine di garantire un governo completo del percorso diagnostico-terapeutico-assistenziale. L'EMR permette di gestire tutti i contatti di un cittadino con i servizi forniti dall'azienda sanitaria o da un'azienda che ha un unico titolare del trattamento. Un **Electronic Medical Record (EMR)** è essenzialmente la versione digitale della cartella clinica di un singolo paziente. Contiene la storia medica e terapeutica del paziente ed è limitato alla struttura sanitaria in cui è stato creato. Le informazioni contenute in un EMR sono standard e sono principalmente utilizzate per la gestione delle cure interne all'organizzazione sanitaria.

Electronic Health Record (EHR)

Un'**Electronic Health Record (EHR)** è una piattaforma informatica che offre una panoramica più ampia della salute di un individuo. Questo sistema integrato va oltre la semplice raccolta di dati clinici standard provenienti da una singola struttura sanitaria: un EHR può includere informazioni provenienti da diverse strutture sanitarie e offre una visione più completa e condivisa della storia clinica del paziente. Le informazioni contenute al suo interno sono accessibili in tempo reale e in modo sicuro solo agli utenti autorizzati e possono includere non solo dati medici, ma anche informazioni sulla salute generale dell'individuo. Un'EHR rappresenta un archivio digitale

che contiene una vasta gamma di informazioni sulla salute di un individuo, provenienti da eventi clinici e prestazioni erogate sia dal Servizio Sanitario Nazionale che da strutture sanitarie private. Le sue principali finalità, come definite dal Ministero della Salute, includono:

- **prevenzione, diagnosi, cura e riabilitazione**, in quanto supporta il processo di assistenza sanitaria, fornendo agli operatori sanitari informazioni cruciali per la prevenzione, diagnosi, trattamento e riabilitazione dei pazienti;
- **studio e ricerca scientifica**, in quanto i dati contenuti al suo interno possono essere utilizzati per studi e ricerche scientifiche nel campo medico, biomedico ed epidemiologico, contribuendo così alla conoscenza e alla progressione della medicina;
- **programmazione sanitaria e valutazione dell'assistenza**, in quanto fornisce dati utili per la programmazione delle risorse sanitarie, la verifica della qualità delle cure e l'efficacia dell'assistenza sanitaria, consentendo una valutazione accurata delle prestazioni sanitarie.

Le informazioni raccolte nell'EHR e rese disponibili al paziente e, con il suo consenso, al personale sanitario includono:

- Informazioni personali e amministrative del paziente.
- Dati relativi al medico di famiglia scelto dal paziente.
- Dettagli sulle prescrizioni mediche emesse da specialisti e farmacisti.
- Risultati di test di laboratorio e analisi cliniche.
- Riassunto della storia clinica del paziente e della sua situazione attuale, redatto e aggiornato dal medico curante.

Inoltre, il paziente stesso ha la possibilità di aggiungere alla propria EHR ulteriori informazioni e documenti personali rilevanti per la cura sanitaria, come ad esempio una lettera di dimissione ospedaliera, un verbale di pronto soccorso, dei referti di radiologia, vaccinazioni, un dossier farmaceutico ad altro ancora. Così facendo, si ha la possibilità di gestire attivamente il proprio archivio sanitario.

Quindi l'EHR è un sistema completo per la gestione delle informazioni sanitarie e supporta attivamente il processo di cura che si fonda sulla gestione integrata degli strumenti sopra descritti e che si riassumono brevemente:[9].

- **Cartella Clinica Cartacea (CCC):** Si riferisce alla tradizionale cartella clinica in formato cartaceo utilizzata per registrare la storia clinica di un paziente. È un sistema basato su supporti fisici, come fogli di carta, per documentare le informazioni mediche e terapeutiche.
- **Cartella Clinica Elettronica (CCE):** È un'evoluzione digitale della CCC, progettata per gestire in modo organico e strutturato i dati relativi alla storia clinica di un paziente. La CCE supporta i processi clinici, diagnostici e terapeutici e favorisce la continuità delle cure attraverso la condivisione e il recupero dei dati tra diverse strutture sanitarie.
- **Electronic Medical Record (EMR):** Si riferisce alla versione digitale della cartella clinica di un singolo paziente all'interno di una stessa struttura sanitaria.

- **Electronic Health Record (EHR):** È un sistema informatizzato più completo e avanzato progettato per raccogliere e gestire in modo integrato le informazioni sanitarie di un paziente provenienti da diverse fonti e strutture sanitarie. L'EHR va oltre i dati clinici standard e include una visione più ampia dell'assistenza sanitaria di un paziente. È utilizzato attivamente dai professionisti sanitari per erogare cure, supportare le decisioni cliniche e ottimizzare i flussi di lavoro.

In sintesi, la principale differenza tra questi concetti riguarda il livello di *completezza*, *integrazione* e *funzionalità* del sistema informatizzato per la gestione delle informazioni sanitarie.

2.2.2 Classificazione delle EHR

A prescindere dalla tecnologia utilizzata per il loro sviluppo, le Electronic Health Record possono essere classificate in base agli obiettivi e alla natura delle loro funzionalità.

Gartner[2] ha proposto una classificazione delle diverse generazioni di EHR, come si può vedere nella figura 2.4:

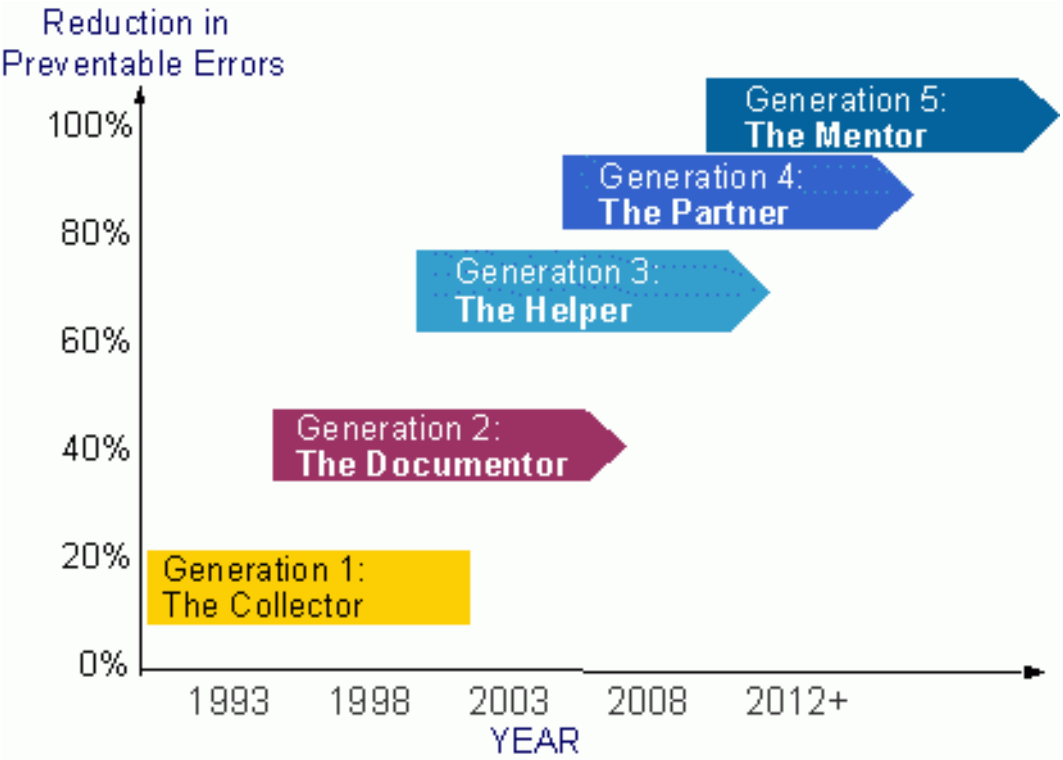

Figura 2.4: Classificazione di una EHR[11]

[2]Gartner è una delle principali società di ricerca e consulenza nel settore dell'informatica e della tecnologia dell'informazione. Fondata nel 1979, Gartner fornisce analisi, ricerche, consulenze e servizi di consulenza a imprese e organizzazioni in tutto il mondo per aiutarle a prendere decisioni informate relative alle tecnologie, alle strategie aziendali e all'innovazione.[10]

Come si può vedere dalla figura 2.4, le cartelle vengono suddivise in cinque generazioni che si evolvono nel tempo, e che assumono caratteristiche differenti tali da accrescere la solidità del sistema informativo su cui si basano. [11]:

Generazione 1: Il Collector

Questo tipo di software consente l'accesso ai dati clinici basandosi sull'episodio clinico. La EHR in questa fase non è fortemente integrata con altri sistemi, ma mostra alcuni tipi di dati. È principalmente utilizzata come strumento per la segnalazione dei risultati e offre accesso ai dati clinici a diversi utenti.

Generazione 2: Il Documentor

In questa fase, il software fornisce agli utenti la documentazione clinica relativa alla cura del paziente. Offre ai medici la possibilità di aggiungere (archiviare/scrivere) e modificare i dati nella cartella clinica del paziente. Consente anche l'invio di messaggi ai medici delle cure primarie.

Generazione 3: L'Helper

In questa fase, il software monitora i dati per aiutare gli utenti a supportare i loro processi. La EHR fornisce supporto alle attività relative al processo assistenziale come la gestione degli ordini, i piani di assistenza infermieristica e i percorsi clinici. Tuttavia, non vi è una grande quantità di "intelligenza" applicata ai dati. Nel sistema vengono inserite solo alcune regole e protocolli di base.

Generazione 4: The Partner

In questa fase, il software funge da partner, aiutando a prevedere e predire il flusso di lavoro dell'utente. Vengono memorizzati molti dati, e c'è una base di conoscenza che la CCE utilizza per fornire suggerimenti concreti per migliorare il benessere del paziente.

Generazione 5: Il Mentor

La generazione più sofisticata documentata da Gartner. Questo tipo di EHR è in grado di guidare gli utenti nella cura dei pazienti, offrendo un supporto avanzato e consulenza basata su dati e conoscenze approfondite.

Generazione 6: Il Seer

Introduce funzioni di analisi predittiva per supportare i medici nella valutazione dei rischi e delle possibili evoluzioni della salute dei pazienti. Uno degli obiettivi principali di queste evoluzioni è la riduzione degli errori prevenibili nel processo di cura.

In sintesi, le diverse generazioni di EHR si evolvono nel fornire sempre maggiore supporto e consulenza ai medici e agli operatori sanitari. Partendo dalla semplice raccolta e visualizzazione delle informazioni, passano poi a rendere le informazioni interattive e ad applicare regole di base. Successivamente, modificano il flusso di lavoro dell'utente finale in base alle informazioni raccolte e forniscono consulenza durante il processo di cura.

2.3 Alcuni esempi di blockchain

Poiché la dimensione dei record sanitari è elevata, la soluzione di archiviarli nel cloud offre vantaggi come un accesso facilitato e la possibilità di condividere le informazioni tra le parti coinvolte; inoltre, le misure di sicurezza e privacy nel cloud aumentano la protezione dei dati. Di fatto i dati prodotti da una struttura sanitaria vengono memorizzati su server esterni accessibili via web da medici, direttori sanitari e altri operatori autorizzati per mezzo di un computer e una con-

nessione internet protetta. Nonostante ciò, il cloud non garantisce l'interoperabilità tra i diversi fornitori di servizi sanitari e può presentare sfide riguardo all'integrità e all'autenticità dei dati. É proprio per far fronte a questi problemi che la tecnologia blockchain emerge come una soluzione promettente. Infatti, una rete blockchain tutela la sicurezza dei dati sensibili in quanto monitora e garantisce l'accesso alle cartelle cliniche solo ai membri autorizzati. Vi è anche maggiore protezione da eventuali manomissioni dei referti in quanto questa tecnologia si comporta come un database distribuito, coinvolgendo diversi attori come operatori sanitari ed ospedali.
La blockchain utilizza la crittografia e i meccanismi di hashing, per preservare la riservatezza, l'integrità e la disponibilità dei dati, mentre i dati dei pazienti vengono aggiunti alla rete in base al consenso della maggior parte dei partecipanti. Inoltre, la blockchain consente ai professionisti sanitari di fornire raccomandazioni personalizzate ai pazienti tramite alias crittografati, mantenendo la privacy: quindi il trasferimento di informazioni tra i membri della blockchain è sicuro e conveniente, questo proprio grazie agli smart contract, i quali permettono appunto lo scambio di dati in modo trasparente e senza intermediari, garantendo l'affidabilità delle transazioni[5].

Perciò la blockchain può essere utile per sopperire a sistemi che richiedono operazioni complesse, quali archiviazione di backup, meccanismi di ripristino e aggiornamento costante dei dati, operazioni comuni nella gestione dei sistemi informativi sanitari. Tale rete, essendo distribuita, elimina un singolo punto di errore, fornendo quindi già un backup in quanto ogni nodo della blockchain contiene una copia unica dei dati, riducendo il numero di transazioni tra i sistemi informativi e alleggerendo il carico sull'ecosistema sanitario. Nella tabella 2.5 sono riportate una serie di blockchain che si stanno sviluppando, grazie ai progressi attuati nella raccolta delle informazioni mediche, e che possono essere utilizzati nella gestione dei dati sanitari e nella condivisione di questi con i pazienti interessati[12] :

	Blockchain company		
	Name	Country	Website
EMR data management	PokitDoc	USA	http://pokitdoc.com
	Gem	USA	http://enterprise.gem.co/health
	YouBase	USA	http://www.youbase.io
EHR data management	Medicalchain	USA	http://www.medicalchain.com
	HealthWizz	USA	http://www.healthwizz.com
	Curisium	USA	http://www.curisium.com
	Hearthy	Spain	http://hearthy.co
	Iryo	Slovenia	http://iryo.io
	Robomed	Russia	http://www.robomed.io
PHR data management	Medcredits	USA	https://medcredits.io
	MyClinic	UK	https://myclinic.com
Point-of-care genomics	Nebula Genomics	USA	http://www.nebula.org
	Genomes.io	USA	http://www.genomes.io
	TimiCoin	USA	http://www.timicoin.io
	Shivom	Switzerland	http://shivom.io
Oncology patients network	OncoPower	USA	http://oncopower.org
Pharma & drug development	Embleema	France	http://www.embleema.com
	BlockPharma	France	http://www.blockpharma.com
	Chronicled MediLedger	USA	http://www.mediledger.com

Figura 2.5: Blockchain in campo medico/sanitario[12]

In letteratura sono presenti numerosi esempi di applicazioni della blockchain nella gestione dei record medici elettronici (EMR), tra questi si trovano:

Guardtime è un'azienda che sfrutta una piattaforma basata su blockchain per garantire la sicurezza delle cartelle cliniche di più di un milione di pazienti in Estonia.[4]

MedRec è un progetto sviluppato dal *MIT Media Lab* e dal *Beth Israel Deaconess Medical Center* che si propone di garantire ai pazienti il controllo sui propri dati sanitari. Questo progetto utilizza in particolare la piattaforma *Ethereum*, per consentire ai pazienti di determinare chi può accedere ai propri dati attraverso specifici permessi di accesso registrati sulla blockchain[4]. Rappresenta un nuovo sistema decentralizzato per la gestione dei record medici elettronici, offrendo ai pazienti un registro completo e immutabile con facile accesso alle loro informazioni mediche tra diversi fornitori e centri di trattamento. Grazie alle proprietà uniche della blockchain, tale progetto gestisce in modo efficace l'autenticazione, la riservatezza, la responsabilità e la condivisione dei dati[1].

Gem Health Network (GHN) è stato sviluppato dalla start up statunitense *Gem* utilizzando la tecnologia blockchain di Ethereum. Questa piattaforma consente a diversi operatori sanitari di accedere in modo condiviso agli stessi dati, facilitando la collaborazione e la condivisione delle informazioni nel settore sanitario[4].

Healthbank è un'azienda svizzera nel settore della sanità digitale che sta sviluppando soluzioni simili per consentire ai pazienti di avere il pieno controllo dei propri dati utilizzando la tecnologia blockchain.[4]

Healthcoin è un progetto che ha come obiettivo costruire un sistema EMR globale[4].

Ancile è un framework basato su blockchain che utilizza gli smart contract costruito sulla piattaforma blockchain di Ethereum per ottenere il controllo degli accessi, la sicurezza dei dati, la privacy e l'interoperabilità delle cartelle cliniche elettroniche[4].

BlockHIE è una piattaforma basata su blockchain per lo scambio di informazioni sanitarie. Utilizza due blockchain, EMR-Chain e PHD-Chain, per archiviare separatemene EMR e PHD[3]. Il sistema propone due algoritmi di confezionamento delle transazioni basati sull'equità per migliorare il throughput del sistema e garantire l'equità tra gli utenti[14].

Health Wizz è una società che sta sperimentando un'app mobile aggregatrice EHR abilitata per blockchain e FHIR[4], che utilizza la blockchain per tokenizzare i dati, permettendo ai pazienti di aggregare, organizzare, condividere, donare e/o scambiare in modo sicuro le proprie cartelle cliniche personali. L'obiettivo è di consentire alle persone di controllare i propri dati sanitari per favorire una migliore comunicazione tra le organizzazioni sanitarie e gli operatori sanitari e promuovere uno standard di cura più elevato[12].

MediBchain, la quale utilizza funzioni crittografiche per deidentificare i dati dei pazienti nei sistemi EMR basati su blockchain[4].

Healthcare Data Gateway (HDG) sta lavorando ad un'applicazione EMR basata su blockchain che consente ai pazienti di possedere, controllare e scegliere come condividere i propri dati preservando la privacy. Viene proposta un'architettura correlata per la gestione

[3]PHD "Personal Health Data" o "Patient Health Data" è un termine che si riferisce ai dati personali relativi alla salute dei pazienti.[13]

[4]FHIR "Fast Healthcare Interoperability Resources", ovvero uno standard di interoperabilità dei dati sanitari che definisce la struttura dei dati e i metodi per scambiare informazioni tra sistemi informativi sanitari[15].

e la condivisione dei dati medici dei pazienti diabetici utilizzando contratti blockchain multi-firma per ottenere il controllo degli accessi e la riservatezza dei dati[4].

Medicalchain è un progetto che mira a introdurre la tecnologia blockchain nel settore sanitario. Attraverso la sua piattaforma ha come obiettivo facilitare la condivisione delle cartelle cliniche dei pazienti tra diverse istituzioni sanitarie internazionali, consentendo ai vari attori sanitari, come medici, ospedali, laboratori, farmacisti e assicuratori, di richiedere l'autorizzazione per accedere e interagire con le cartelle cliniche dei pazienti. Ogni interazione su Medicalchain è verificabile, trasparente e sicura, in quanto viene registrata come transazione sul suo registro distribuito. La tecnologia blockchain che utilizza archivia in modo sicuro i dati sanitari e assicura l'integrità e l'unicità delle informazioni[1].

HealthChain è un'applicazione EMR sviluppata come rete blockchain privata e autorizzata utilizzando Hyperledger Fabric e distribuita su Bluemix. Presenta un'architettura a 3 livelli: il livello superiore dell'architettura è popolato con un'interfaccia web in cui gli utenti interagiscono con HealthChain[2]. Questa pagina Web è ospitata sullo strato intermedio, in cui un server NodeJs è responsabile della comunicazione con la struttura Hyperledger e del codice della catena nella figura 2.6.

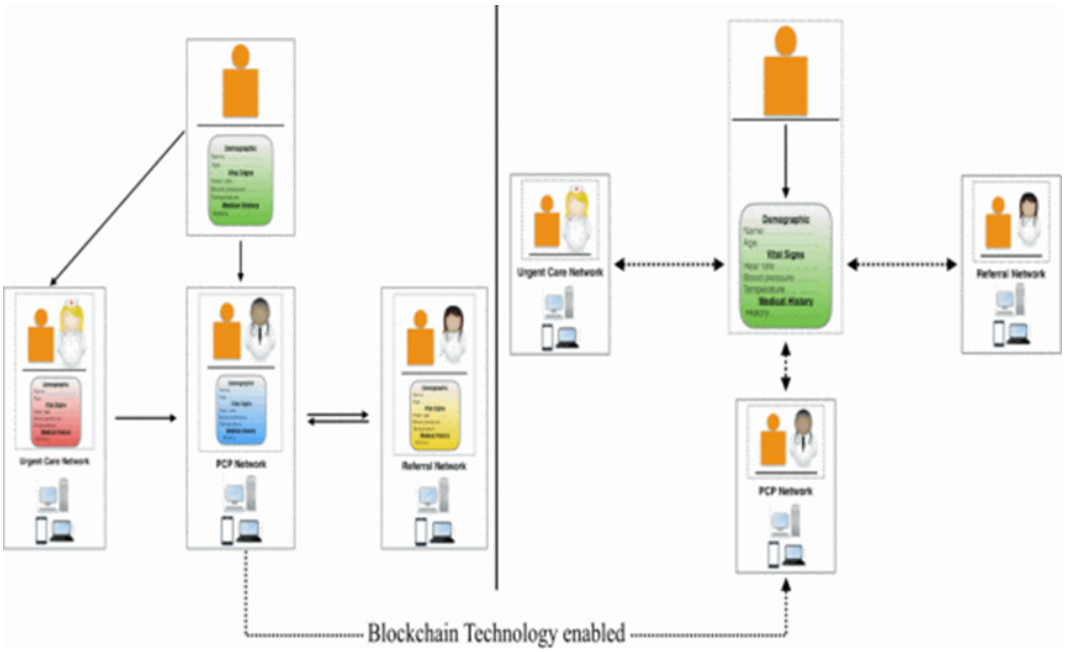

Figura 2.6: Health Chain[2]

HealthChain mira a ottimizzare diversi aspetti del ciclo di vita delle PHI[5] contemporaneamente. Durante la visita iniziale presso uno dei fornitori di assistenza, il paziente crea la prima versione del suo record PHI. Questa versione iniziale viene quindi caricata sulla blockchain, diventando la risorsa digitale primaria all'interno della catena. Gli smart contract garantiscono che il paziente possa solo creare la versione iniziale delle sue informazioni sanitarie e caricarle sulla blockchain. L'architettura modulare di Hyperledger Fabric consente a

[5]Le PHI "Informazioni Protette sulla Salute" sono informazioni di identificazione personale che si riferiscono a dati anagrafici, condizioni mediche e psicologiche, prestazioni e pagamenti di servizi medici[16]

HealthChain di ottenere riservatezza, scalabilità e sicurezza dei dati sanitari. HealthChain incorpora anche chaincode che controllano le autorizzazioni e i privilegi di accesso sulla rete blockchain[4].

- **Cardea** è un ecosistema completo di credenziali verificabili, sviluppato come progetto open source per facilitare la condivisione di informazioni sanitarie nel rispetto della privacy dei pazienti; è nato da un'iniziativa volta a condividere la prova dello stato del test Covid-19 per l'accesso ad Aruba. L'obiettivo è consentire alle autorità sanitarie pubbliche di sviluppare e implementare soluzioni sanitarie digitali open source, incentrate sulla privacy e facili da utilizzare. Offre un insieme completo di strumenti per emettere, condividere e verificare una vasta gamma di informazioni sanitarie utilizzando identificatori decentralizzati, un'applicazione mobile dedicata e un registro distribuito. Si basa sui framework Hyperledger Indy e Hyperledger Aries[17].

  Factom, **HealthCombix**, **Patientory**, **SimplyVital**, **IBM Watson**, **BurstIQ**, **Bowhead**, **QBRICS**, **Nuco**, **MedBlock**, **BlockHIE**, **FHIRCain**, **MedBlock**, **FHIRChain**, **MeDShare**: sono altri progetti che si trovano in letteratura che lavorano a soluzioni di reti blockchain per gestire informazioni mediche[4].

Nel contesto della protezione e della privacy dei dati sensibili archiviati sui registri sanitari elettronici (EMR) basati su blockchain, diversi schemi crittografici sono stati proposti per migliorare la sicurezza e la validità di tali EMR archiviati sulla blockchain. Ad esempio, Hussein et al. propone un metodo che utilizza la trasformazione della lunghezza d'onda discreta e l'algoritmo genetico per potenziare la sicurezza e ottimizzare le prestazioni del sistema. Viene anche suggerito uno schema di firma basato su attributi con autorità multiple, che consente al paziente di approvare un messaggio da aggiungere alla blockchain in base agli attributi del messaggio, senza rivelare informazioni sensibili[4].
Un altro approccio utilizza una combinazione di crittografia basata sugli *attributi ABE* (Attribute-Based Encryption) [ovvero una forma di crittografia che consente di assegnare attributi ai dati e di definire regole di accesso basate su tali attributi], crittografia basata sull'*identità IBE* (Identity-Based Encryption) [sistema nel quale, l'identità di un destinatario, come ad esempio un indirizzo email o un nome utente, viene utilizzata direttamente come chiave pubblica per crittografare i dati] e firma basata sull'*identità IBS* (Identity-Based Signature) [sistema nel quale un individuo può firmare un messaggio utilizzando la propria identità invece di una chiave privata.] insieme alla tecnologia blockchain[4].
Altre proposte per la sicurezza degli EMR basati su blockchain includono gli schemi di *gestione delle chiavi di Zhao et al.* e l'*architettura GAA-FQ* (Granular Access Authorization supporting Hydraulic Queries) proposta da Zhang e Poslad. Quest'ultima fornisce un'autorizzazione sicura a diversi livelli di granularità senza richiedere un'infrastruttura a chiave pubblica, mentre per quanto riguarda la privacy, viene proposto uno schema sicuro e rispettoso della privacy per gli EMR basato su blockchain. Questo schema utilizza blockchain private e consorziate per archiviare rispettivamente gli EMR effettivi e i puntatori agli EMR[4].

Capitolo 3

Blockchain

La tecnologia blockchain è stata introdotta per la prima volta da Satoshi Nakamoto nel suo famoso articolo "*Bitcoin: A Peer−to−Peer Electronic Cash System*"[18], il quale ha fornito le fondamenta matematiche per la criptovaluta **bitcoin**. Oltre ad essere la base di tutte le criptovalute, la tecnologia blockchain ha trovato una vasta applicazione anche in altri settori e ha anche aperto la strada a nuove innovazioni come gli smart contract[19].

Il problema principale che Nakamoto ha affrontato con la blockchain era quello di creare fiducia in un sistema distribuito: l'idea era quella di creare un registro distribuito di documenti con marcature temporali, in cui nessuno potesse alterare i dati o le tali marcature senza essere scoperto. Questo problema è diverso dai problemi di autenticazione, integrità e non ripudio, risolti dalle firme digitali, in quanto pur legando in modo verificabile una persona ad un documento, dimostrandone l'intenzionalità della firma, non fornisce informazioni sul momento in cui il documento è stato firmato. La blockchain risolve questo problema fornendo un meccanismo di fiducia distribuita: più parti mantengono un registro delle transazioni e ciascuna parte può verificare che l'ordine e le marcature temporali delle transazioni non siano state alterate[19].

Negli ultimi anni, la tecnologia blockchain ha guadagnato notevole popolarità in vari settori, tra cui quello della salute. Questo interesse è stato alimentato dalle sue caratteristiche uniche e dal fatto che offre un modo innovativo per gestire e condividere in modo sicuro e trasparente i dati sanitari, migliorando l'efficienza e la sicurezza delle operazioni nel settore sanitario[3].

3.1 Un'analisi iniziale della blockchain

3.1.1 Tecnologia dei registri distribuiti (DLT)

La **Tecnologia dei registri distribuiti (DLT)** è un sistema digitale che consente la registrazione sicura e trasparente delle transazioni attraverso una rete di computer[20].
I principi fondamentali nella progettazione di una tecnologia blockchain includono l'utilizzo della crittografia per garantire la sicurezza delle comunicazioni e l'archiviazione dei dati delle transazioni, nonché l'implementazione di protocolli di consenso distribuito che permetta ai nodi della rete di concordare sulla validità e sull'ordine delle transazioni.
La tecnologia blockchain rappresenta un notevole avanzamento rispetto alla tecnologia dei database distribuiti, studiata sin dagli anni settanta. Mentre i database distribuiti coinvolgono un database di transazioni condiviso da diversi utenti, le DLT in generale, compresa la blockchain, sono progettate per gestire dati condivisi in modo distribuito[21]. La blockchain consente di condividere un

registro di transazioni che vengono lette, convalidate e archiviate in una catena di blocchi, come illustrato nella figura3.1.

Figura 3.1: Evoluzione della DLT: dal registro tradizionale alla blockchain[21].

La DLT rappresenta un'innovazione chiave che ha reso possibile un cambiamento significativo del modo in cui si archiviano e si gestiscono i dati digitali; è una tecnologia che consente di archiviare e aggiornare un registro distribuito in modo decentralizzato, dove per registro distribuito si intende una struttura di dati digitali che è replicata e distribuita su più dispositivi informatici. La blockchain e le sue varianti sono un tipo di DLT; comprendono tre componenti fondamentali: un modello di dati che cattura lo stato attuale del registro, un linguaggio di comunicazione definito da transazioni che modificano lo stato del registro e un protocollo utilizzato per ottenere il consenso dei partecipanti su quali transazioni accettare e in quale ordine[21].

3.1.2 Scopo primario della Blockchain

Una **blockchain** è una DLT strutturata come una catena di blocchi, che può essere combinata con un modello di dati e un linguaggio di comunicazione per abilitare smart contract e altre tecnologie. Le blockchain sono caratterizzate dalla crittografia, che consente una trasmissione sicura dei dati e assicura l'immutabilità dei record. In una blockchain, nuovi blocchi vengono aggiunti alla fine del registro, garantendo che i record siano immutabili e verificabili nel tempo[21].

Queste funzionalità consentono la creazione di una nuova generazione di applicazioni transazionali che pongono al centro fiducia, responsabilità e trasparenza, rivoluzionando numerosi settori[21]. Lo scopo primario di una piattaforma blockchain è quello di favorire relazioni digitali peer−to−peer, agevolando gli scambi digitali e l'automazione delle attività commerciali. Infatti, la blockchain consente lo scambio digitale di risorse e la condivisione di dati preziosi tra le parti senza limitazioni geografiche o temporali. Tali risorse possono variare da dati sensibili che richiedono una gestione con restrizioni di accesso fino a risorse che possono essere condivise liberamente; questo influenza la scelta tra blockchain con piena autorizzazione e blockchain senza autorizzazione.

Tipi di blockchain Esistono tre principali tipologie di blockchain: pubbliche, private e del consorzio[22].

- **Blockchain pubbliche** (o senza autorizzazione): in una blockchain pubblica, chiunque può partecipare alla rete come nodo, verificare, visualizzare ed effettuare transazioni. Anche l'attività di mining è aperta a chiunque desideri parteciparvi, contribuendo volontariamente con la propria potenza di calcolo al meccanismo di consenso. Questo tipo di blockchain è caratterizzato da un alto livello di decentralizzazione e sicurezza, con un complesso meccanismo di consenso, criptografia a chiave pubblica-privata e governance distribuita.
- **Blockchain private** (o autorizzate): una blockchain privata è gestita centralmente per quanto riguarda le autorizzazioni di scrittura e modifica dei blocchi. Le autorizzazioni di lettura possono essere pubbliche o limitate a un numero ristretto di utenti. Questo tipo di blockchain è più adatto ai modelli di business tradizionali in cui la privacy e il controllo dei dati sono fondamentali. Sebbene meno decentralizzata rispetto alle blockchain pubbliche, le blockchain private offrono maggiore controllo e sicurezza per le aziende.
- **Consortium Blockchain:** le blockchain del consorzio sono gestite da un insieme preselezionato di nodi che formano un consorzio. Il meccanismo di consenso è controllato da questi nodi, che possono essere, ad esempio, istituzioni finanziarie. L'attività di mining può essere gestita tramite un accordo congiunto tra i partecipanti e il consenso può essere raggiunto attraverso un sistema di votazione a soglie di maggioranza prestabilite. Le autorizzazioni di lettura possono essere pubbliche o limitate ai partecipanti del consorzio. Questa tipologia di blockchain combina caratteristiche delle blockchain pubbliche e private, offrendo un maggiore controllo rispetto alle pubbliche e una maggiore decentralizzazione rispetto alle private.

In sintesi, le blockchain pubbliche offrono massima decentralizzazione e trasparenza, le blockchain private offrono maggiore controllo e privacy, mentre le consortium blockchain rappresentano un compromesso tra le due, adatte a casi d'uso in cui è necessario un certo grado di decentralizzazione ma anche controllo e fiducia tra i partecipanti.

3.2 Caratteristiche della blockchain

Si possono schematizzare come segue le principali caratteristiche di una blockchain[21]:

1. **Decentralizzazione**: permette la condivisione e l'archiviazione dei dati peer−to−peer (P2P) senza affidare la gestione del registro a un'autorità centrale. Ciò non significa eliminare completamente gli intermediari che convalidano le transazioni, ma decentralizzarli insieme ai loro ruoli.
2. **Immutabilità**: i registri distribuiti proteggono i dati da qualsiasi tipo di manipolazione e falsificazione, rendendoli immutabili. Anche se possono verificarsi situazioni in cui il registro viene modificato, l'immutabilità dei dati rende i dati accessibili e gestibili da diverse entità che non si fidano l'una dell'altra.
3. **Integrità**, **autenticità** e **non ripudio**: l'hashing dei dati garantisce che i dati non vengano modificati durante la trasmissione, assicurando l'integrità. Inoltre, l'origine di una transazione può essere accertata attraverso la diffusione della chiave pubblica dei mittenti, mentre l'autenticità e il non ripudio sono garantiti dalla procedura di firma dei dati che coinvolge la chiave privata.

4. **Verificabilità**: ogni trasferimento di dati nei sistemi blockchain deve essere convalidato e verificato, rendendo ogni transazione visibile a tutti i partecipanti alla blockchain e assicurando che tutte le operazioni siano tracciabili. Anche se possono esistere canali privati per isolare e mantenere più registri, i dati dei registri rimangono visibili a tutti i partecipanti del canale.

Queste caratteristiche qualificano la tecnologia blockchain ad un livello di affidabilità piuttosto elevato, differenziandola dal classico database distribuito. Inoltre, le caratteristiche della blockchain sono strettamente correlate al meccanismo di consenso in uso.[21]

I registri distribuiti e i sistemi centralizzati possiedono alcune proprietà rilevanti e, a volte, discordanti, quali[23]:

- **Verificabilità pubblica**, in quanto nei registri distribuiti, ogni transizione di stato è confermata da verificatori, permettendo a chiunque di verificare la correttezza dello stato del sistema mentre, nei sistemi centralizzati, gli osservatori devono avere fiducia nell'entità centrale per fornire loro lo stato corretto.
- **Trasparenza dei dati**, requisito di verificabilità pubblica poiché la quantità di informazioni trasparenti per un osservatore può variare e non è necessario che tutti i partecipanti abbiano accesso a ogni informazione.
- **Privacy**, la quale è più facile da raggiungere in un sistema centralizzato, dove la trasparenza e la verificabilità pubblica non sono richieste.
- **Integrità delle informazioni**, nei sistemi con verificabilità pubblica chiunque può verificare l'integrità dei dati mentre, nei sistemi centralizzati l'integrità può essere garantita solo se l'entità centrale non viene compromessa.
- **Ridondanza dei dati**: Nei sistemi blockchain, la ridondanza è intrinsecamente fornita attraverso la replica tra gli scrittori. Nei sistemi centralizzati, la ridondanza è ottenuta tramite replica su diversi server fisici e backup.
- **Trust Anchor**, il quale rappresenta l'autorità massima di un sistema che ha il potere di concedere e revocare l'accesso in lettura e scrittura a un sistema.

3.2.1 Struttura, caratteristiche e parole chiavi della blockchain

La blockchain è caratterizzata da una struttura a blocchi, come illustrato nella figura 3.3, all'interno dei quali sono presenti elementi che possono variare a seconda della tipologia di blockchain con cui si lavora. Alcuni, però, sono comuni a tutte le blockchain:

- **Header esterno**: Questo header identifica la blockchain e specifica la dimensione del blocco. Contiene informazioni generali sulla blockchain stessa.
- **Header del blocco**: Questo header contiene informazioni specifiche relative alla validazione del blocco e al suo blocco genitore. Include dettagli come il numero di versione del blocco, l'hash del blocco genitore, il nonce e il timestamp.
- **Blocco corpo**: Questo componente consiste in un elenco di transazioni e un contatore di transazioni. Le transazioni sono i dati effettivi che vengono registrati nella blockchain.

– **Numero di versione del blocco**: Indica il protocollo blockchain e l'algoritmo di consenso utilizzato.

- **Hash del blocco genitore**: È l'output di una funzione di hashing applicata all'intestazione del blocco precedente e serve a collegare i blocchi in modo sequenziale.
- **Nonce**: È una stringa di lunghezza fissa utilizzata nel processo di validazione del blocco, che spesso viene modificata iterativamente fino a trovare un valore valido.
- **Timestamp**: Indica il tempo trascorso dall'istante predefinito, consentendo di tenere traccia dell'ordine cronologico delle transazioni.
- **Radice dell'albero di Merkle**: È il risultato di una procedura di hash tree[1] applicata alle transazioni nel corpo del blocco. Le transazioni vengono aggregare e sottoposte a hashing in modo iterativo per garantire l'integrità dei dati nel blocco.

Complessivamente, la struttura dati dei blocchi in una blockchain è progettata per garantire l'integrità, la coerenza e la sicurezza delle transazioni registrate. Ogni componente svolge un ruolo fondamentale nel funzionamento complessivo della blockchain e nel mantenimento della sua affidabilità.

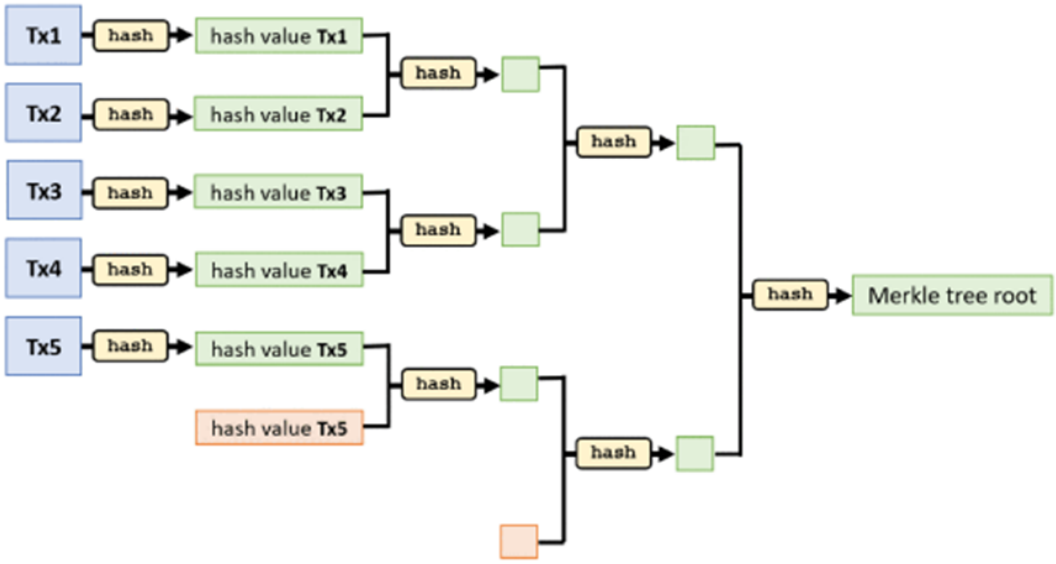

Figura 3.2: Esempio di procedura Merkle Hash Tree: le transazioni duplicate (hash) sono contrassegnate in arancione[19].

La hash dell'intestazione del blocco funge da "impronta digitale" univoca che identifica in modo univoco quel blocco all'interno della blockchain. Questo hash viene calcolato utilizzando algoritmi crittografici e include informazioni cruciali come il nonce, il timestamp e l'hash del blocco genitore. Serve da collegamento ai futuri nuovi blocchi, poiché ogni nuovo blocco conterrà, nella hash del suo header, il valore hash del blocco precedente, creando così una catena di blocchi connessi in modo sequenziale.

Il corpo del blocco contiene tutte le transazioni coinvolte nel calcolo della radice di Merkle, ovvero un hash che rappresenta tutte le transazioni incluse nel blocco. Tale radice, viene calcolata

[1]In crittografia e informatica , un hash tree o Merkle tree è un albero in cui ogni "foglia" (nodo) è etichettata con l'hash crittografico di un blocco di dati, e ogni nodo che non sia una foglia (chiamato ramo o nodo interno) è etichettato con l'hash crittografico delle etichette dei suoi nodi figli.[24]

utilizzando un albero di hash, in cui le transazioni sono raggruppate e sottoposte a hashing iterativo fino a che non viene ottenuta una singola radice che verrà poi inclusa nell'intestazione del blocco. Inoltre, il corpo del blocco contiene un contatore di transazioni che tiene traccia del numero totale di transazioni incluse al suo interno. Questo è importante perché ogni blocco ha un limite di dimensione, imposto dalle regole del protocollo della blockchain, che influenza direttamente il numero massimo di transazioni che possono essere incluse al suo interno: quando la capacità massima viene raggiunta, non è possibile aggiungere ulteriori transazioni fino a quando non viene generato il blocco successivo. Questo limite è progettato per garantire che la blockchain rimanga gestibile e scalabile nel tempo.

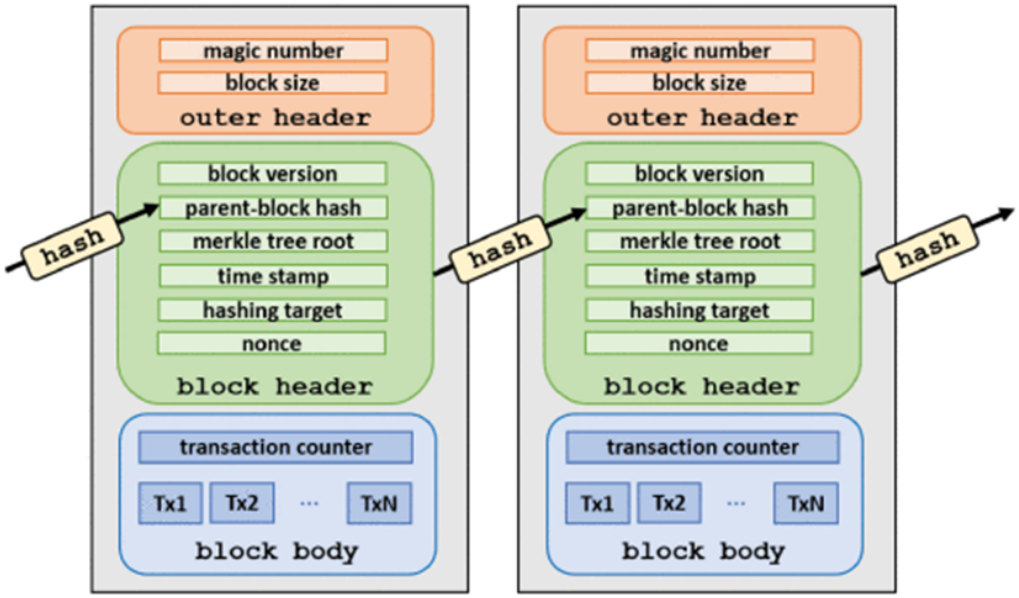

Figura 3.3: Rappresentazione di una struttura blockchain[19].

Altri elementi che spesso si presentano quando si parla di blockchain sono i **wallet** e i **token**, strumenti che aiutano a sfruttare al meglio i vantaggi di una rete di questo tipo.
I **wallet** sono un software o un hardware dedicato che permettono di fare azioni su una blockchain e hanno la caratteristica di conservare tutti i dati delle transazioni. A seconda della piattaforma utilizzata o del loro grado di autonomia, possono esserci diversi tipi di wallet:

- Desktop wallet: software che si installano sul computer, la memorizzazione delle chiavi può avvenire o in locale o sul server del fornitore del servizio;
- Mobile wallet: si installa sugli smartphone e sono molto leggeri e facili da utilizzare, la memorizzazione delle chiavi può avvenire o in locale o sul server del fornitore del servizio;
- Web wallet: sono accessibili solo tramite un browser web e memorizzano le informazioni sulle chiavi delle transazioni su un server di proprietà di terzi;
- Hardware wallet: dispositivi mobili che si collegano al PC mediante la porta USB e sono hardware progettati appositamente per gestire i wallet;
- Paper wallet: indirizzi e chiavi segrete vengono stampate per essere memorizzati a lungo termine.

I **token** invece, possono rappresentare una vasta gamma di asset, sia tangibili che intangibili, consentendo una maggiore liquidità, transazioni più veloci ed economiche e una maggiore trasparenza e provabilità nelle operazioni finanziarie. La tokenizzazione è un processo fondamentale in quanto consente di convertire qualsiasi risorsa di valore in un token digitale, utilizzabile su una blockchain e presenta diversi vantaggi[25]:

- Più liquidità: Una volta tokenizzati, gli asset possono essere resi disponibili a un pubblico più ampio, aumentando la liquidità del mercato. Questo elimina il "premio di liquidità" associato agli asset tradizionalmente meno liquidi, consentendo agli investitori di acquisire la proprietà frazionaria di asset come opere d'arte o immobili.
- Transazioni più veloci ed economiche: I token crittografici consentono di aggirare gli intermediari di mercato coinvolti nei tradizionali processi di gestione patrimoniale. Ciò riduce i costi di transazione e i tempi di elaborazione degli scambi, consentendo trasferimenti di valore più efficienti ed economici.
- Trasparenza e provabilità: Poiché i token crittografici esistono sulla blockchain, è possibile tracciare facilmente la loro provenienza e la cronologia delle transazioni in modo crittograficamente verificabile. Le transazioni sono registrate automaticamente sulla blockchain, garantendo un'alta affidabilità e trasparenza.

Si approfondisce adesso, brevemente, alcuni concetti chiave che sono già stati citati nel testo e sul quale è utile soffermarsi:

Funzioni Hash Le funzioni hash svolgono un ruolo cruciale nel funzionamento delle blockchain. Queste funzioni prendono in input una stringa o un dato di qualsiasi tipo e producono un output, chiamato hash, che è una rappresentazione sicura e crittografata dei dati di input. Gli hash sono di lunghezza fissa, indipendentemente dalla dimensione dei dati di input. Inoltre, anche una piccola variazione nei dati di input produrrà un hash completamente diverso.

Le caratteristiche principali delle funzioni hash sono[19]:

1. **Unicità dell'output:** Due input diversi devono produrre output diversi. Questo significa che anche una piccola modifica nei dati di input dovrebbe generare un hash completamente diverso.
2. **Invertibilità inattuabile:** Dato l'hash di un dato, dovrebbe essere praticamente impossibile risalire ai dati originali. In altre parole, non è possibile recuperare i dati di input.
3. **Efficienza computazionale:** Il calcolo dell'hash deve essere efficiente, ovvero non deve richiedere molto tempo o risorse computazionali.
4. **Resistenza alle collisioni:** È molto difficile trovare due input diversi che producono lo stesso hash. Anche se matematicamente possibile, è computazionalmente impraticabile e antieconomico.

Un'applicazione comune delle funzioni hash è l'archiviazione sicura delle password: invece di memorizzare le password in chiaro nei database, i siti web memorizzano l'hash delle password, così facendo, quando un utente tenta di accedere, la password inserita viene nuovamente "hashata" e confrontata con l'hash presente nel database. Se le due funzioni corrispondono, l'accesso viene concesso. Questo metodo migliora notevolmente la sicurezza dei dati degli utenti, poiché anche se un database viene compromesso, gli hacker non possono facilmente risalire alle password originali.

Gli algoritmi di hashing più comuni includono Secure Hash Algorithms (SHA) e sono ampiamente utilizzati nelle blockchain.[19].

Firma digitale La crittografia svolge un ruolo fondamentale nel garantire la sicurezza e l'integrità dei dati nelle transazioni blockchain. In particolare, la crittografia asimmetrica, o crittografia a chiave pubblica, viene combinata con schemi di hashing e firme digitali per fornire garanzie di sicurezza.

Ecco come funziona uno schema di firma digitale[19]:

1. **Generazione della coppia di chiavi**: Ogni utente blockchain genera una coppia di chiavi: una chiave privata e una chiave pubblica. La chiave privata è utilizzata per firmare le transazioni, mentre la chiave pubblica viene utilizzata dal destinatario per verificare l'autenticità della firma.

2. **Fase di firma:** Il mittente della transazione esegue l'hashing dei dati della transazione e genera una firma digitale utilizzando la propria chiave privata. La firma digitale, insieme ai dati originali codificati, viene inviata al destinatario. L'hashing dei dati non solo comprime i dati rendendoli più efficienti, ma assicura anche che i dati trasferiti mantengano la loro integrità.

3. **Fase di verifica:** Il destinatario della transazione decodifica i dati firmati utilizzando la chiave pubblica del mittente e confronta l'hash dei dati originali con l'hash calcolato dai dati decodificati. Se i due hash corrispondono, ciò significa che i dati non sono stati modificati durante il trasferimento e che la firma è autentica.

La crittografia asimmetrica e gli schemi di firma digitale consentono ai partecipanti alla blockchain di autenticare le transazioni e garantire l'integrità dei dati senza la necessità di un'autorità centrale di fiducia. Ciò contribuisce alla creazione di un ambiente in cui le transazioni possono essere effettuate in modo sicuro e verificabile senza dover fare affidamento su intermediari.

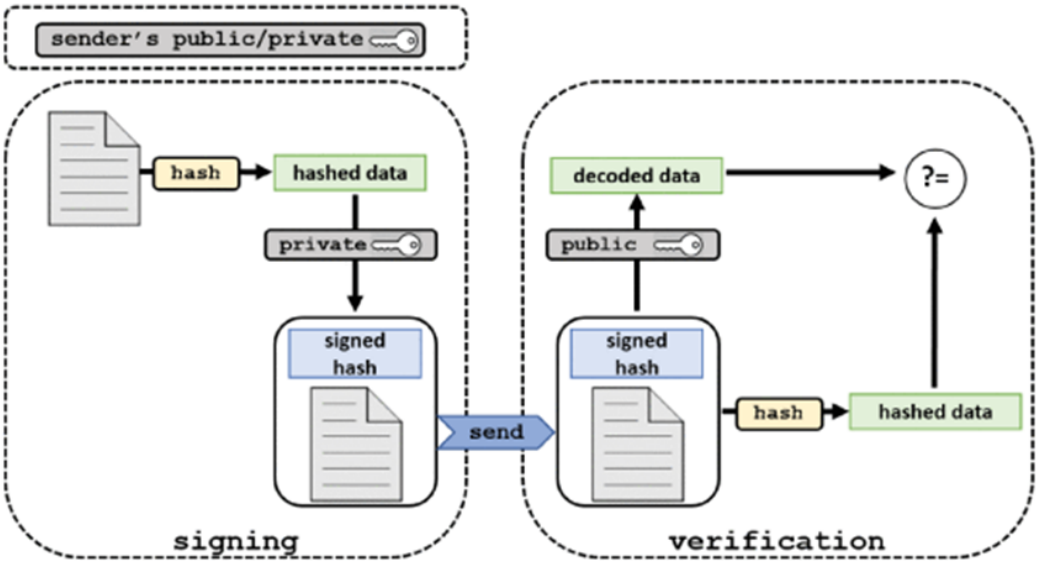

Figura 3.4: Fasi del protocollo di firma digitale[19].

3.2.2 Vantaggi e svantaggi della Blockchain

È necessario considerare attentamente sia i vantaggi che gli svantaggi della blockchain prima di decidere di adottarla.

Vantaggi della blockchain[26]:

1. **Repository condiviso gestito dai peer**: La blockchain offre un repository condiviso accessibile a tutti i partecipanti, consentendo loro di visualizzare le transazioni e accedere ai dati; la distribuzione delle informazioni tra i nodi previene la perdita di dati in caso di guasti o eventi imprevisti.
2. **Fornisce fiducia tra le parti**: Grazie alla firma digitale e alla validazione delle transazioni, la blockchain garantisce un alto livello di fiducia tra i diversi nodi e utenti, eliminando la necessità di intermediari per garantire l'integrità delle operazioni.
3. **Possibile archivio di dati mondiale**: La blockchain ha il potenziale per diventare un archivio globale a cui possono accedere diversi attori, consentendo a tutti di leggere e scrivere dati sulla blockchain, potenzialmente facilitando la condivisione e l'accesso alle informazioni a livello globale.
4. **Trasparenza**: La blockchain garantisce trasparenza, consentendo a tutti di visualizzare non solo lo stato attuale delle transazioni, ma anche la storia completa degli stati precedenti, offrendo un elevato livello di visibilità e accountability all'interno del sistema.
5. **Immutabilità dei dati**: I dati archiviati sulla blockchain non possono essere cancellati o modificati una volta inseriti, assicurando l'integrità e l'affidabilità delle informazioni conservate sulla blockchain nel tempo.
6. **Decentralizzazione**: La blockchain può operare senza la necessità di un'autorità centrale di controllo, perciò non può essere facilmente manipolata, censurata o spenta da un singolo ente, garantendo una maggiore resistenza e sicurezza del sistema nel suo complesso.
7. **Automazione tramite contratti intelligenti**: Grazie agli smart contract, la blockchain consente l'automazione di varie attività e processi, questi contratti possono facilitare la stipula e l'esecuzione automatica di accordi senza la necessità di intervento umano, migliorando l'efficienza e riducendo i costi operativi.

Svantaggi della blockchain[26]:

1. **Elevato consumo energetico**: Le operazioni di mining e validazione delle transazioni richiedono una quantità significativa di energia, contribuendo a un elevato consumo complessivo di energia. Ad esempio, una singola transazione Bitcoin può costare fino a 6 dollari in termini di energia consumata dai nodi della rete. Anche se è importante fare presente che le reti blockchain tendono a utilizzare meno energia rispetto alla maggior parte dei sistemi finanziari tradizionali, e c'è una crescente tendenza all'utilizzo di fonti di energia rinnovabile per il mining[27].
2. **Mining e spreco di risorse**: Il mining richiede hardware costoso e la maggior parte della potenza di calcolo viene sprecata. I nodi competono per risolvere complessi problemi matematici, ma solo il nodo più veloce vince, mentre gli altri sprecano risorse.
3. **Spazio richiesto per la replica dei dati**: Ogni nodo della rete deve archiviare una copia completa della blockchain, il che richiede notevole spazio di archiviazione. Questo rende le prestazioni della blockchain meno efficienti rispetto ai tradizionali database.

4. **Lentezza nell'aggiunta di informazioni**: La creazione di un nuovo blocco sulla blockchain richiede tempo, con Bitcoin che impiega dai 10 ai 60 minuti e Ethereum 15 secondi. Questa lentezza può limitare l'efficienza delle transazioni sulla blockchain.

5. **Rischio per la privacy e la reputazione degli utenti**: L'immutabilità e la trasparenza della blockchain possono compromettere la privacy e la reputazione degli utenti, in quanto ogni nodo della rete può accedere al contenuto della blockchain.

6. **Limitazioni degli smart contract**: Gli smart contract non possono fare affidamento su API esterne e richiedono che tutte le informazioni siano immutabili sulla blockchain. Gli oracoli possono consentire l'iniezione di dati esterni, ma richiedono un sistema di reputazione robusto.

7. **Possibilità di difetti negli smart contract**: Gli smart contract sono soggetti a difetti nel codice, che una volta implementati sulla blockchain diventano immutabili. Questo li rende vulnerabili agli attacchi informatici e può causare problemi di sicurezza.

Gli svantaggi, sono già al centro delle attenzioni e degli sforzi dei principali esperti di blockchain in tutto il mondo. Nel lungo periodo, i benefici della tecnologia blockchain supereranno senz'altro i contro[26].

3.3 Attori della blockchain

3.3.1 Nodi

I **nodi** svolgono un ruolo fondamentale nella determinazione del consenso all'interno di una blockchain. Questo consenso può essere raggiunto attraverso l'elezione di un nodo leader temporaneo che agisce come una sorta di "dittatore". Il **leader** ha il compito di selezionare quale blocco proporre come candidato per l'inclusione nel registro blockchain e di verificare la correttezza di tale proposta. Una volta convalidato il blocco proposto, il leader rinuncia immediatamente al suo potere decisionale. Durante il suo mandato (ovvero, il periodo in cui il leader è in carica), gli altri nodi non possono essere certi che il loro blocco venga accettato. Alla fine di ogni mandato, si avvia il processo di elezione di un nuovo leader.

La procedura di elezione dei leader è un elemento intrinseco al meccanismo di consenso adottato dalle diverse blockchain. Le blockchain, sia con che senza autorizzazione, utilizzano metodi distinti per selezionare il peer incaricato di proporre blocchi ai validatori.

Esistono due tipologie di nodi:

- **Full node:** memorizza l'intera blockchain e gestisce tutti gli aspetti del protocollo;
- **Light node:** non memorizza l'intera blockchain ma solo gli header di tutti i blocchi; possono comunque inviare transazioni alla rete e verificarne la correttezza.

Il nodo che inizia il trasferimento dei dati attraverso una transazione è detto **mittente dei dati**. Non è necessariamente l'autore della transazione né il nodo con il diritto di avviare il trasferimento di dati. In alcuni casi, come nei contratti intelligenti, potrebbe essere un nodo autorizzato esterno alla rete. Tuttavia, il mittente dei dati è responsabile della firma delle transazioni con la propria chiave privata per autenticare l'origine del trasferimento dei dati[21].
Qualsiasi utente che riceve una transazione firmata è, invece, il **destinatario dei dati**. Ha la capacità di recuperare la chiave pubblica del mittente dal messaggio e verificare l'autenticità della

transazione. Questo processo consente trasferimenti di dati sicuri e a prova di manomissione nella rete blockchain, poiché qualsiasi nodo blockchain può recuperare e verificare la firma.

La *convalida dei nodi* è un passaggio cruciale all'interno del processo blockchain. Gli attori validatori, come già accennato, gestiscono l'algoritmo di consenso e sono responsabili di raggiungere un accordo sulle proposte avanzate dai nodi principali. La validazione di un blocco rappresenta il consenso tra i nodi validanti su quale blocco pubblicare e in quale ordine.

Riflettendo sul percorso della transazione fin qui delineato, emergono chiaramente i ruoli che gli attori assumono in questo contesto. Nella prima fase, la transazione coinvolge le parti contraenti, ossia il mittente e il destinatario dei dati. Successivamente, la transazione viene trasmessa ai nodi principali, i quali sono incaricati di verificare la correttezza delle transazioni, raccoglierle in blocchi e proporre il blocco come candidato per la validazione. Nella fase finale, i peer validatori determinano la validità del blocco.

Nei contesti autorizzati, ogni attore svolge un ruolo specifico senza sovrapposizioni nelle fasi di proposta e convalida del blocco. Questa distinzione deriva dalla procedura di consenso scalabile basata sul voto adottata nelle blockchain autorizzate. Al contrario, nelle blockchain ad accesso aperto, i miners ricoprono ruoli sovrapposti. In pratica, il processo di mining può essere considerato una sorta di simulazione dell'elezione del leader nei tradizionali protocolli di consenso.

È importante notare che una transazione blockchain coinvolge non solo questi tre attori principali, ma anche altri soggetti. Alcune blockchain autorizzate migliorano la loro scalabilità assegnando a diversi peer compiti aggiuntivi, come i controlli di verifica dell'esecuzione, l'elezione e l'ordinamento dei leader.

3.3.2 Protocolli di consenso

Il consenso all'interno di una rete blockchain garantisce che i nodi concordino su una singola richiesta o su una sequenza di richieste, che spesso corrispondono a un valore numerico. Ogni protocollo di consenso comporta due eventi fondamentali: la proposta e la decisione. Durante la proposta, i nodi propongono un valore, mentre durante la decisione i nodi concordano su tale valore. Le proprietà fondamentali del consenso includono[21]:

- **Accordo:** Ogni nodo corretto o onesto deve concordare sullo stesso valore proposto.
- **Validità:** Se tutti i nodi propongono lo stesso valore, allora tutti i nodi corretti devono decidere su quel valore.
- **Terminazione:** Ogni nodo corretto deve prendere una decisione su un valore proposto.

Le applicazioni blockchain possono richiedere proprietà aggiuntive, che possono variare a seconda del contesto e delle esigenze specifiche del progettista. Ad esempio, alcuni autori confrontano i protocolli in termini di gestione dell'identità di rete, consumo energetico e resistenza agli attacchi dannosi. Altri confronti si concentrano su aspetti come sicurezza, prestazioni, throughput, scalabilità e latenza.

Un **protocollo di consenso**, quindi, consiste in un insieme di regole che ogni transazione deve seguire, garantendo così proprietà come resilienza e sicurezza, fondamentali per il funzionamento affidabile della rete blockchain[21]. I meccanismi di consenso si riferiscono alla capacità di far convergere agenti multipli verso un interesse comune. In un sistema distribuito, come quelli che caratterizzano le blockchain, è essenziale che gli agenti raggiungano un accordo riguardo a un determinato interesse, che può essere un valore, un'azione o altro, in base al loro stato. Nei sistemi distribuiti e nelle blockchain, i protocolli di consenso devono gestire agenti dinamici e possibili

guasti durante il processo di accordo; la tolleranza agli errori è cruciale per garantire che il sistema possa continuare a funzionare correttamente nonostante guasti dei nodi, sia causati da errori di processo sia da comportamenti dannosi. Tra i primi tentativi di sviluppare meccanismi di consenso nei database distribuiti si trova la tecnica di replicazione della macchina a stati (SMR), la quale consente la costruzione di protocolli di consenso resilienti ai guasti, protocolli robusti contro i guasti da crash in ambienti affidabili. I protocolli **Byzantine Fault Tolerant** (BFT) sono un esempio popolare di questo tipo di protocolli, che consentono alle macchine a stati di concordare una sequenza comune di richieste per mantenere repliche coerenti.

Le prime blockchain, come Bitcoin ed Ethereum, implementano il consenso tra milioni di utenti in modo probabilistico, dove la resilienza ai guasti è una caratteristica fondamentale finché i nodi dannosi rimangono una minoranza nella rete *peer−to−peer* (P2P). Per garantire la sicurezza e prevenire comportamenti dannosi, sono stati introdotti costi computazionali, come la **Proof of Work** (PoW) utilizzata da Bitcoin, che richiede ai peer di fornire una prova di lavoro per convalidare un blocco di transazioni. Sono state proposte diverse alternative che mirano a sostituire i calcoli inutili tipici del PoW con prove alternative di sforzo nella convalida delle transazioni. Queste alternative includono una varietà di meccanismi di consenso che formano la classe degli algoritmi di consenso **Proof of X** (PoX)[21].

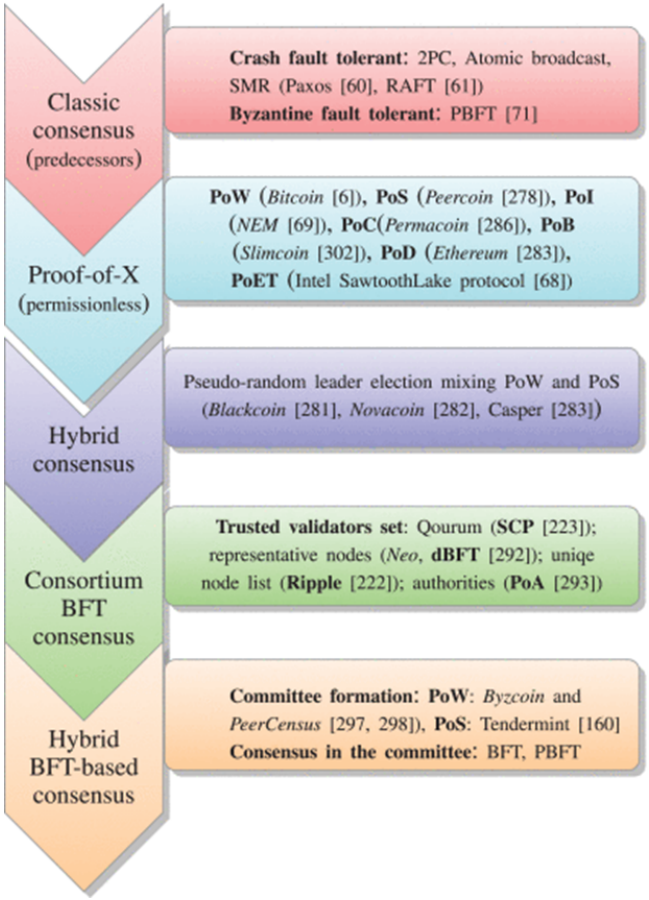

Figura 3.5: Percorso evolutivo dei protocolli di consenso in cinque classi dai protocolli pre blockchain a quelli post blockchain.[21]

Nella figura 3.5 viene schematizzato il percorso evolutivo dei protocolli di consenso. Avendo appena discusso dei protocolli classici, si andranno ora ad approfondire i vari cambiamenti.

Consenso Proof of X

Gli algoritmi PoX includono approcci come il Proof of Stake (PoS), il Proof of Authority (PoA), il Proof of Burn (PoB), e altri. Ogni algoritmo PoX introduce nuove modalità per selezionare il validatore di un blocco di transazioni e per determinare il processo di convalida, cercando di migliorare la scalabilità, ridurre il consumo energetico e affrontare altre limitazioni associate al PoW. Questi approcci rappresentano una direzione importante nell'evoluzione delle tecnologie blockchain, poiché cercano di equilibrare la sicurezza, la scalabilità e l'efficienza in modi innovativi. I protocolli PoX sono progettati per blockchain senza autorizzazione e si basano su un processo probabilistico di elezione del leader. In ambienti privi di autorizzazione ogni nodo ha la possibilità di diventare leader semplicemente dimostrando di aver fatto qualche "sforzo". Quest'ultimo può avere natura computazionale, monetaria o di archiviazione oppure può essere uno sforzo per affermarsi sulla rete blockchain. Il leader eletto mantiene il suo ruolo di voto fino a quando non saranno disponibili i risultati delle nuove elezioni. Di seguito elenchiamo e presentiamo brevemente gli algoritmi basati su PoX più utilizzati[21].

Proof of Work La PoW implica che i nodi devono trovare un valore hash per il blocco che soddisfi un certo requisito di difficoltà: il nodo che riesce per primo a trovare questo valore hash valido può convalidare il blocco e ricevere una ricompensa. Quindi, i miner vincenti svolgono il ruolo sia di nodi principali che di nodi di convalida.

L'obiettivo principale del protocollo PoW è rendere il processo di convalida difficile da eseguire ma facile da verificare. Utilizzando algoritmi come il *Secure Hash Algorithm* (SHA), che garantiscono una procedura costosa per trovare un hash valido, si rende il processo di convalida computazionalmente oneroso.

Tuttavia, la PoW non garantisce un consenso definitivo, poiché possono verificarsi *fork* nella blockchain. La blockchain più lunga, definita come quella con il maggior numero di blocchi, è considerata valida. Questo processo può portare a una coerenza probabilistica della blockchain, in cui le conferme delle transazioni sono basate sulla catena più lunga[21].

Le ***fork*** nelle reti blockchain si verificano quando il software utilizzato dai diversi miners o nodi della rete non è allineato. Questo può portare alla creazione di due versioni della blockchain, ognuna con le proprie regole e transazioni. Esistono due tipi principali di fork[28]:

- **Hard Fork**: è un cambiamento radicale del software che richiede che tutti gli utenti della rete aggiornino il loro software alla nuova versione per rimanere sulla stessa blockchain; i nodi che non si aggiornano saranno incompatibili con la nuova versione del software e continueranno a utilizzare la versione precedente della blockchain. É quindi una divergenza permanente dalla versione precedente della blockchain, il che significa che le due versioni non possono interagire tra loro.

- **Soft Fork**: è retro compatibile, il che significa che i nodi che non si aggiornano possono ancora essere in grado di interagire con la nuova versione della blockchain. Tuttavia, la blockchain aggiornata è responsabile dell'approvazione delle transazioni, e i nodi che non sono stati aggiornati non riconosceranno più i blocchi nuovi come validi. Funziona solo in una direzione: la blockchain aggiornata non riconosce i nodi non aggiornati e, quindi, affinché una soft fork funzioni correttamente, la maggior parte dei miners deve accettare le nuove regole.

Alternative Proof of Stake e mining virtuale

La Proof of Stake (PoS) rappresenta un'alternativa alla PoW, trasformando il mining reale in un mining virtuale, senza il consumo energetico associato. In questo meccanismo, l'elezione del leader si basa sul possesso di quote di criptovaluta da parte degli utenti della rete, che determinano il loro potere decisionale nel consenso. L'idea di fondo è che gli utenti con maggiori investimenti nella criptovaluta sono meno inclini ad attaccare la blockchain, poiché ciò danneggerebbe il valore delle loro stesse quote.

Esistono diverse varianti della PoS per affrontare problemi come la centralizzazione del potere decisionale e l'attacco noto come "niente in gioco", il quale si verifica quando i validatori tentano di convalidare più blocchi possibile, poiché il costo computazionale per farlo è basso. Per mitigare questo problema, le varianti della PoS richiedono ai validatori di ponderare le loro quote o di impegnare risorse durante la fase di convalida.

Alcune implementazioni alternative come la **Proof of Importance** (PoI) cercano di contrastare la centralizzazione incoraggiando comportamenti desiderabili nella rete, incentivando i leader a aumentare il flusso e il volume delle transazioni nella rete. Altre alternative sono la **Proof of Autorithy** (PoA), dove ciò che si mette in gioco non è la valuta bensì la reputazione e quindi i validatori devono rendere pubblica la loro identità; la **Proof of Burn** (PoB), dove la criptomonete viene spesa (o bruciata) in quanto, maggiore è il numero di monete bruciate da un utente in favore del sistema, maggiore sarà la sua potenza di mining e più è alta la probabilità che tale utente sia scelto come prossimo validatore. La **Proof of Activity** combina, invece, la PoW e la PoS in modo da ottenere un protocollo ancora più sicuro; infatti, i miners affrontano la PoW per proporre i blocchi e il blocco proposto dal vincitore viene validato da chi viene sorteggiato in base della quantità di criptomoneta posseduta.

Per migliorare l'efficienza, il meccanismo può essere implementato con elezioni ristrette, come nel caso di **Delegated Proof of Stake** (DPoS), dove alcuni utenti vengono eletti come delegati per partecipare al processo di consenso, rendendo il sistema più scalabile e efficiente[21].

Algoritmi basati su BFT e ibridi basati su BFT

I protocolli basati su BFT (**Byzantine Fault Tolerance**) sono efficaci nelle blockchain con un numero limitato di partecipanti, rendendoli adatti per sistemi chiusi. Gli algoritmi BFT sono progettati per garantire sia la vitalità che la sicurezza di una rete, richiedendo che almeno due terzi dei partecipanti siano onesti affinché la rete funzioni correttamente.

Tuttavia, per estendere l'applicabilità dei protocolli BFT, sono stati sviluppati algoritmi ibridi. Questi algoritmi combinano elementi della PoX con la BFT, utilizzando la PoX per formare comitati di nodi e la BFT per raggiungere un accordo sul consenso. Questa combinazione permette una maggiore flessibilità e scalabilità, disaccoppiando la generazione dei blocchi dalla loro validazione. In altre parole, i processi di creazione e di convalida dei blocchi sono gestiti da attori separati, anche se possono essere gli stessi nodi che ricoprono ruoli diversi all'interno del sistema.

La combinazione di PoX e BFT sfrutta i vantaggi di entrambi gli approcci, garantendo al contempo un elevato livello di affidabilità e resistenza alla manipolazione della rete[21].

3.4 Dove ha senso una Blockchain?

Il diagramma di flusso nella figura 3.6 fornisce un'utile guida decisionale per determinare se l'utilizzo di una blockchain aperta o autorizzata è appropriato in un determinato contesto. In sintesi, l'uso

di una blockchain ha senso quando più entità, reciprocamente diffidenti desiderano interagire e cambiare lo stato di un sistema senza dover fare affidamento su una terza parte fidata online.

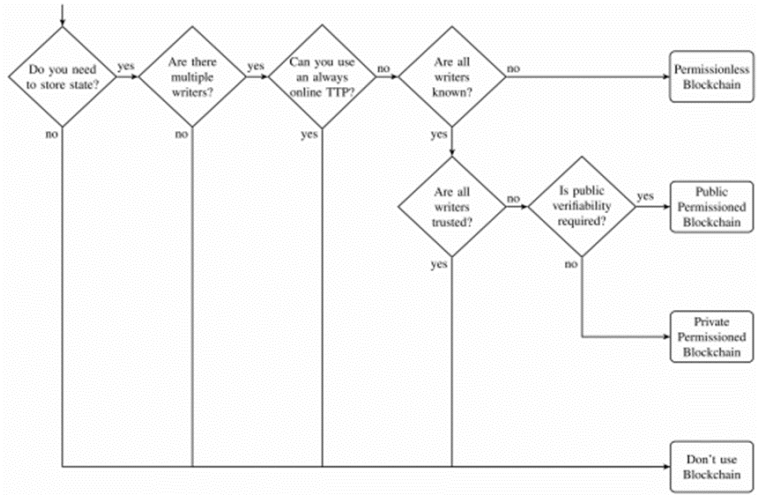

Figura 3.6: Un diagramma di flusso per determinare se una blockchain è la soluzione tecnica adeguata per risolvere un problema[23].

Dal diagramma si possono estrapolare diverse informazioni chiave, derivanti dal confronto di alcune proprietà delle blockchain autorizzate, senza autorizzazione e di un database centralizzato. Tra queste si trovano ad esempio[23]:

- Necessità di archiviare dati: Se non è necessario archiviare dati, una blockchain non è utile come forma di database.
- Numero di writer: Se esiste un solo writer o un gruppo limitato di writer fidati, un database tradizionale è più adatto e offre prestazioni migliori in termini di throughput e latenza rispetto a una blockchain.
- Presenza di una terza parte fidata (TTP): Se è disponibile una terza parte affidabile (TTP), ci sono due opzioni:
  - Se il TTP è sempre online, può essere delegato con le operazioni di scrittura e funzionare come verificatore delle transizioni di stato.
  - Se il TTP è solitamente offline, può fungere da autorità di certificazione in una blockchain autorizzata, dove tutti gli autori del sistema sono conosciuti.

  –
- Livello di fiducia reciproca tra writer: Se tutti gli scrittori si fidano reciprocamente, la soluzione migliore è un database con accesso in scrittura condiviso; altrimenti ha senso utilizzare una blockchain autorizzata.
- Verificabilità pubblica: Se è richiesta la verificabilità pubblica, si può optare per una blockchain con autorizzazione pubblica, altrimenti l'insieme di lettori può essere limitato in una blockchain con autorizzazione privata.

- Flessibilità degli scrittori: Se l'insieme degli scrittori non è fisso e noto ai partecipanti, come nel caso di molte criptovalute come Bitcoin, una blockchain senza autorizzazione è una soluzione adatta.

3.5 Principali piattaforme Blockchain disponibili

Le piattaforme blockchain in circolazione sono numerose e si differenziano per la loro affidabilità, sicurezza, protocolli di consenso ed altro.
In figura 3.7 sono riportate altre blockchain che non verranno esaminate in questa tesi.

Platform	Bitcoin	Ethereum	Hyperledger platforms	Corda	Tendermint	Chain Core	Quorum
Common Features							
Data encription and hashing ⇒ data confidentiality and integrity Digitial signature ⇒ data authenticity and non-repudiation Auditability, immutability, open sourced code							
General Features							
Identity and membership	✗	✗	✓	✓	✓	✓	✓
Major usage	Public payment system	Generic blockchain platform	Modular blockchain platforms	Specialized distributed ledger platform for financial industry (digital assets)	blockchain consensus engine	multi-assets ledger designed for assets trading	general application platform
Cryptocurrency	Bitcoin	Ether cryptocurrency Tokens via smart contract	Currency and tokens possible via chaincode	✗	At first, but now ✗	✗	✗
Governance	N/A	Ethereum developers	Linux Foundation	R3	Tendermint developers	Chain, Microsoft, IC3	JPMorgan
Architectural Features							
Data model	UTXO	Account based	Key-value	UTXO+	various	UTXO+	Account based
Smart contracts execution	native	EVM	Fabric: docker, Sawtooth: native	JVM	various	Chain Virtual Machine (CVM), TxVM	EVM
Smart contract language	not Turing-complete	Solidity, Serpent, LLL	Fabric: GO & Javascript, Sawtooth: Java, Go, JavaScript, Rust or Solidity	Kotlin, Java	depends on software choice	written in bytecode instructions for the CVM	Go
Modularity	✗	✗	✓(consensus, membership services)	✓(consensus)	✗	✗	✗
Consensus protocol	Mining, PoW ledger level	PoW, POS transaction level	Various	RAFT (centralize), BFT via BFT-SMaRt toolkit	BFT	BFT - The Federated Concensus	QuorumChain, RAFT-based
Adversary model	50%	50%	BFT: 33%, PoET: Trusted Hardware	RAFT: 50%, BFT: 33%	33%	33%	RAFT based: 50%, Quorum chain 33%
Execution	sequentially on all peers	sequentially on all peers	parallel	sequentially on all peers	sequentially on all peers	sequentially on all peers	sequentially on all peers
Architecture	order-execute	order-execute	execute - order-validate	order-execute	order-execute	order-execute	order-execute
Node isolation	✗	✗	Fabric via channels	✗	✗	✗	✗
Dissemination	flooding	gossip (D∃Vp2p)	gossip	gossip	gossip	gossip	gossip-v.1.x HTTPS–v.2.x
Throughput	7 tps	15-40 tps; in private setup ~ thousand tps	dozen of thousands tps [49, 131]	120-1000 tps [154]	tens of thousands tps within single data-center [160]	N/A	dozens to hundreds of tps
Latency	600 sec	~ 15 sec	< 1 sec	N/A	< 1 sec	N/A	N/A
Source Code	[123].	[129].	Sawtooth [138], Fabric [167], Indy [168], Iroha [169], Burrow [141], Grid [170]	[155].	[161].	[162].	[166].

Figura 3.7: Confronto delle piattaforme Blockchain[21]

Tra quelle disponibili ve ne sono alcune alcune che presentano aspetti consolidati; di seguito ne sono riportate alcune che ricoprono un ruolo di spicco nella comunità blockchain odierna.

BITCOIN La Blockchain Bitcoin è una rete pubblica e decentralizzata basata su Proof of Work, che offre un registro aperto delle transazioni.
Essendo la prima rete blockchain pubblica utilizzata ampiamente, Bitcoin è considerato un predecessore rigido rispetto ai moderni framework più avanzati che hanno superato alcuni dei suoi limiti. È importante notare che mentre Bitcoin potrebbe essere riutilizzato per scopi diversi dal-

la criptovaluta, come l'archiviazione dei dati, è stato progettato principalmente come sistema di pagamento pubblico decentralizzato e non è adatto per sistemi privati autorizzati.

I partecipanti nella rete Bitcoin possono essere nodi completi, clienti o MINERS. I clienti (utenti) possono ricevere e inviare transazioni, mentre i MINERS sono responsabili del mining tramite PoW.

La rete Bitcoin funziona attraverso quattro processi principali:

- scoperta di rete
- creazione di transazioni
- convalida dei blocchi
- mining.

Il protocollo peer-to-peer di Bitcoin prevede che un peer mittente trasmetta una transazione firmata ai suoi vicini, che la inoltreranno solo se risulta valida. I MINERS, insieme agli altri nodi della rete, ricevono e verificano queste transazioni, archiviandole in un pool non verificato. Una volta estratto, un nuovo blocco viene trasmesso attraverso la rete P2P e ogni nodo completo verifica la sua validità prima di aggiungerlo al registro.

Bitcoin utilizza due operazioni di base per la rete P2P: una strategia di collegamento, che determina come i client si connettono ai loro vicini, e una strategia di comunicazione, che definisce come i nodi comunicano tra loro[21].

ETHEREUM Ethereum è una piattaforma aperta progettata per sviluppare e utilizzare **applicazioni decentralizzate** (DApps) che eseguono smart contract, ovvero programmi che si attivano automaticamente quando vengono soddisfatte certe condizioni. Questa flessibilità è resa possibile grazie a un linguaggio di programmazione completo di Turing integrato chiamato **Solidity**.

Uno smart contract su Ethereum permette agli utenti di progettare le proprie applicazioni scrivendo la logica in poche righe di codice. Tuttavia, a differenza di Bitcoin, Ethereum utilizza un approccio basato sui conti, in cui gli attori possono creare transazioni, inviare messaggi e creare contratti.
Anche Ethereum utilizza un algoritmo di consenso basato su Proof of Work chiamato **Ethash**, progettato per enfatizzare la durezza della memoria, rendendo il mining computazionalmente intensivo e richiedendo un'ampia larghezza di banda di accesso alla memoria.

La rete Ethereum supporta tre tipologie di conti:

- Contract Account (CA), possono eseguire transazioni interne ad altri contratti o inviare fondi,
- External Owned Account (EOA), possono trasferire ether e interagire con i contratti
- Miner, sono responsabili del mining di Ether e della convalida delle transazioni.

Anche se Ethereum è principalmente una rete pubblica, il software è open source e può essere configurato per creare reti private locali utilizzando il meccanismo di consenso Proof of Authority. In una configurazione pubblica, Ethereum può gestire circa 15-40 transazioni al secondo (tps) con una latenza di circa 15 secondi per blocco, mentre in una configurazione privata può gestire circa mille tps[21].

CORDA Corda è un framework blockchain autorizzato sviluppato dalla società di software R3, che guida un consorzio di oltre duecento istituzioni finanziarie globali. A differenza di altre soluzioni blockchain, offre accesso alle informazioni e alle funzioni di validazione solo alle parti coinvolte direttamente in una specifica transazione, consentendo il consenso a livello di singoli accordi anziché a livello di sistema.

Le identità dei nodi in Corda sono autenticate da un certificato X.509 emesso da un servizio di autorizzazione chiamato "Doorman", presente in ogni rete. A differenza di molte altre piattaforme blockchain autorizzate, Corda non ordina tutte le transazioni in un'unica esecuzione virtuale per formare la blockchain, piuttosto definisce stati e transazioni, dove ogni transazione consuma uno o più stati e ne produce uno nuovo. Questo modello produce un grafico aciclico diretto con hash. Sperimentalmente, Corda ha raggiunto migliaia di transazioni al secondo utilizzando il consenso RAFT con un cluster di 3 membri e un log distribuito Kafka, sebbene nominalmente si preveda un throughput di circa 120 transazioni al secondo[21].

HYPERLEDGER Hyperledger è un sistema open source progettato per promuovere le tecnologie blockchain interindustriali. Attualmente, Hyperledger comprende quattordici progetti, sei dei quali sono registri distribuiti e gli altri otto sono moduli di supporto. La comunità ufficiale di Hyperledger conta più di 270 organizzazioni partecipanti.

I framework Hyperledger sono progettati principalmente per applicazioni blockchain autorizzate e la sua architettura include diversi componenti fondamentali:

- Strato di Consenso: Responsabile della verifica dei blocchi di transazioni e di concordare il loro ordine.
- Smart Contract Layer: Si occupa dell'elaborazione delle transazioni, inclusa la loro acquisizione, esecuzione e convalida.
- Livello di Comunicazione: Gestisce il trasporto peer-to-peer.
- Data Store Abstraction: Fornisce un'astrazione per diversi archivi dati utilizzati da altri moduli.
- Crypto Abstraction: Gestisce gli algoritmi crittografici utilizzati nella blockchain.
- Servizio Identità: Consente la creazione di una radice di fiducia, l'iscrizione e la registrazione delle identità, nonché l'autenticazione e l'autorizzazione.
- Servizio Politiche: Si occupa della gestione delle politiche all'interno della rete.
- API per le Applicazioni: Fornisce un'interfaccia per le interazioni con le applicazioni.
- Servizio di Interoperabilità: Supporta l'interoperabilità tra diverse istanze blockchain.

L'obiettivo di Hyperledger è fornire una distribuzione affidabile delle applicazioni tramite smart contract, tuttavia, i framework Hyperledger vanno oltre le prestazioni dei sistemi blockchain pubblici basati su Proof-of-Work, poiché non richiedono un PoW computazionalmente impegnativo.
I framework Hyperledger sono stati sviluppati con l'idea che nessuna piattaforma sia adatta per tutte le esigenze. Pertanto, ogni framework è stato progettato per essere modulare e adattabile alle esigenze specifiche dell'utente[21].
Sarà proprio quest'ultimo la blockchain di interesse in questa tesi. In particolare, si entrerà nel dettaglio portando in luce tutti gli aspetti fondamentali di *Hyperledger Fabric*.

Capitolo 4

Hyperledger Fabric

Hyperledger Fabric è una piattaforma di tecnologia di registro distribuito (DLT) autorizzata di livello aziendale open source, progettata per l'uso in contesti aziendali, differente da altre piattaforme.[29]

Hyperledger è stata fondata dalla Linux Foundation ed è utile per portare avanti progetti open source; è governata da un comitato direttivo tecnico diversificato ed è stata progettata da un insieme formato da manutentori provenienti da più organizzazioni. La piattaforma si presenta come il supporto per protocolli di consenso collegabili che permettono di personalizzarne in modo più efficace l'aspetto, al fine di adattarsi a particolari casi d'uso e a modelli di fiducia. Si differenzia da altri sistemi blockchain in quanto è privato e autorizzato.[29]

4.1 Caratteristiche di Hyperledger Fabric

Di seguito vengono riportate le caratteristiche principali della piattaforma Hyperledger Fabric.

Modularità

Hyperledger Fabric ha un'architettura modulare; la piattaforma è stata progettata e configurata per soddisfare la diversità dei requisiti dei casi d'uso aziendali. Ad alto livello, Fabric è composto dai seguenti componenti modulari:

- Un **servizio di ordinazione** collegabile che stabilisce il consenso sull'ordine delle transazioni e quindi trasmette i blocchi ai peer.
- Un **fornitore di servizi di abbonamento** collegabile che è responsabile dell'associazione delle entità nella rete con identità crittografiche.
- Un **servizio gossip peer-to-peer** opzionale che diffonde l'output dei blocchi ordinando il servizio ad altri peer.
- Gli **smart contract** ("chaincode") vengono eseguiti all'interno di un ambiente isolato (ad esempio Docker).
- Il **ledger** può essere configurato per supportare una varietà di DBMS[1].
- Un'**applicazione collegabile** di politica di approvazione e convalida che può essere configurata in modo indipendente per ogni applicazione.

[1]Un sistema di gestione di basi dati, in inglese Data base management system (Dbms) è un software che permette di creare, gestire e interrogare database, ovvero un insieme di dati organizzati secondo principi specifici[30].

Blockchain autorizzata

In una blockchain autorizzata, come nel caso di Hyperledger Fabric, l'insieme dei partecipanti è noto, identificato e controllato, il che genera fiducia al suo interno. In questo modo, si possono utilizzare protocolli di consenso che non richiedono un *mining* oneroso ed è più difficile che un partecipante introduca intenzionalmente un codice dannoso attraverso uno smart contract[29].

Smart Contract

Uno **smart contract**, o ciò che in Fabric è anche detto **chaincode**, funziona come un'applicazione distribuita affidabile che ottiene la sua sicurezza/fiducia dalla blockchain e dal consenso sottostante tra i peer.[29] Molti smart contract vengono eseguiti contemporaneamente nella rete e possono essere distribuiti dinamicamente, mentre il codice dell'applicazione deve essere considerato non attendibile e potenzialmente dannoso. Il protocollo di consenso convalida ed ordina le transazioni propagandole poi a tutti i nodi peer, i quali, eseguono, in sequenza, le transazioni.

Execute-Order-Validate

In Hyperledger Fabric l'architettura per le transazioni è chiamata **Execute-Order-Validate**; essa la esegue prima di raggiungere un accordo finale sull'ordine. Essa, per gestire le sfide di resilienza, flessibilità, scalabilità, prestazioni e riservatezza, separa il flusso di dette transazioni in tre fasi:

- *eseguire* una transazione e verificarne la correttezza,
- *transazioni di ordini* definite tramite un protocollo di consenso,
- *convalidare* le transazioni rispetto a una politica di approvazione specifica dell'applicazione prima di inserirle nel registro.

Una politica di approvazione deve garantire la corretta esecuzione di uno smart contract e, a tal fine, ogni transazione deve essere eseguita (e quindi approvata) solo dal sottoinsieme dei nodi peer necessari per soddisfare la politica di approvazione della transazione. In questo modo i risultati incoerenti possono essere filtrati prima dell'ordinazione, eliminando la componente non deterministica e permettendo l'uso di linguaggi di programmazione standard.

Privacy e Riservatezza

In un contesto autorizzato come Hyperledger Fabric, si possono sfruttare forme di consenso alternative, come ad esempio approcci che limitano la distribuzione di informazioni riservate solo ai nodi autorizzati. Nel caso specifico di Hyperledger, la riservatezza è garantita dalla sua struttura di canali e dalla funzionalità dei dati privati. All'interno di questi canali, i partecipanti della rete Fabric istituiscono una sottorete in cui ciascun membro ha visibilità limitata ad un insieme specifico di transazioni. Di conseguenza, solamente i nodi coinvolti in un canale possono accedere agli smart contract e ai dati scambiati, preservando così la riservatezza e la privacy delle informazioni.

L'utilizzo dei dati privati facilita la condivisione di informazioni tra i membri di un canale, offrendo una protezione simile a quella dei canali separati senza l'onere aggiuntivo derivante dalla creazione e dalla gestione di canali distinti.

Consenso modulare

In Hyperledger Fabric, ad occuparsi dell'ordinamento delle transazioni è il **Servizio di Ordinazione**, il quale è logicamente disaccoppiato dai peer che eseguono le stesse e mantengono il ledger. Essendo modulare, l'utilizzo del consenso può essere adattata al presupposto di fiducia di una particolare implementazione o soluzione.

Prestazioni e scalabilità

Le prestazioni di una piattaforma blockchain possono essere influenzate da molte variabili come la dimensione della transazione, la dimensione del blocco, la dimensione della rete, nonché i limiti dell'hardware.

4.2 Componenti di una rete blockchain su Hyperledger Fabric

Dopo aver descritto il funzionamento e le caratteristiche della struttura di una rete blockchain su Hyperledger Fabric, in questa sezione ci si soffermerà sulle componenti della rete (per capirne meglio le definizioni e l'utilizzo) e, in maniera più approfondita, si tratteranno gli attori principali della rete, quali le **Autorità di certificazione (CA)**, i **Peer**, il **Ledger**, gli **Smart contract**, il **Servizio di ordinazione** e i **Canali**.

Prima di entrare nel dettaglio, è utile però citare e illustrare brevemente alcune componenti della rete.

Identità: per identità si intende il processo di identificazione degli utenti che accedono alla rete. Ognuno degli attori presenti in una rete blockchain, possiede un'**identità digitale**, rappresentata da un **certificato digitale** *X.509*. L'importanza delle identità risiede nel fatto che esse determinano autorizzazioni per accedere alle risorse e alle informazioni all'interno della rete.

L'unione di un'identità con gli attributi che la caratterizzano, è nota come **principal**, che può essere paragonata ad un userID con maggiore flessibilità in quanto possono includere una vasta gamma di proprietà dell'identità di un attore, come l'organizzazione di appartenenza, l'unità organizzativa, il ruolo e persino l'identità specifica dell'attore. Queste proprietà determinano le autorizzazioni associate all'entità.
Affinché un'identità sia considerata autenticata, deve derivare da un'autorità affidabile: in Fabric, il ruolo è ricoperto dal **Membership Service Provider** (*MSP*).

Membership Services Provider (MSP): Come si è già detto, Fabric è una rete autorizzata e, per questo motivo, c'è bisogno di un modo per dimostrare la propria identità prima di poter fare transazioni. Le *CA* rilasciano identità creando una coppia di chiavi pubblica-privata, e il Membership Services Provider (*MSP*) serve proprio in questo contesto. Infatti, l'*MSP*, sul servizio di ordinazione, contiene la chiave pubblica e viene usato per verificare che la firma. creata con la chiave privata e allegata alla transazione, sia valida.
Questo consente ad organizzazioni, canali e peer di avere degli *MSP* che sono a conoscenza di tutte le autorizzazioni degli attori presenti sulla rete. L'*MSP* oltre ad essere un elenco di chi partecipa alla rete, di chi è membro di un canale e ad aiutare nel creare l'elenco di identità revocate, identifica proprio i privilegi che un attore possiede, specificandone il ruolo: amministratore, peer, client, ordinante o membro.
Esistono due tipi di *MSP*: quelli che agiscono localmente sul nodo di un attore (**MSP locale**) e quelli che invece agiscono nella configurazione del canale (**canale MSP**), che si differenziano proprio per l'ambito in cui si collocano. I primi, quindi gli *MSP* locali, sono definiti per i client e per i nodi (peer e ordinanti), definendone le autorizzazioni e consentendo all'utente di autenticarsi nelle sue transazioni. I secondi invece, definiscono i diritti amministrativi e partecipativi a livello di canale identificando i partecipanti allo stesso: le organizzazioni che vorranno unirsi, dovranno includere un *MSP* contenente la catena di fiducia per far si che le transazioni che partono da loro non verranno rifiutate.

Gli **MSP locali** sono specifici per un'organizzazione, mentre gli **MSP globali** sono per un canale e sono condivisi tra tutti i partecipanti dello stesso. In figura 4.1, il canale di sistema di rete è gestito da *ORG1*, il peer viene gestito da *ORG2*, mentre l'ordinatore è gestito da *ORG1*. *ORG1* riconosce e accetta le identità di *RCA1*, mentre *ORG2* fa la stessa cosa per le identità di *RCA2*: entrambi le identità sono relative all'amministrazione e indicano chi ha il potere di gestire questi componenti.

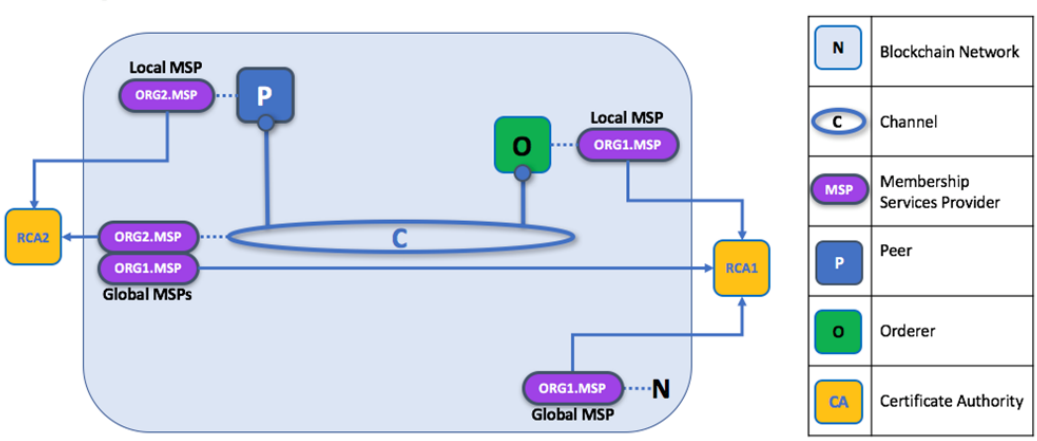

Figura 4.1: Membership Services Provider

Infrastruttura a chiave pubblica: Un'**infrastruttura a chiave pubblica** (*PKI*) è una raccolta di tecnologie Internet che fornisce comunicazioni sicure in una rete.[29]

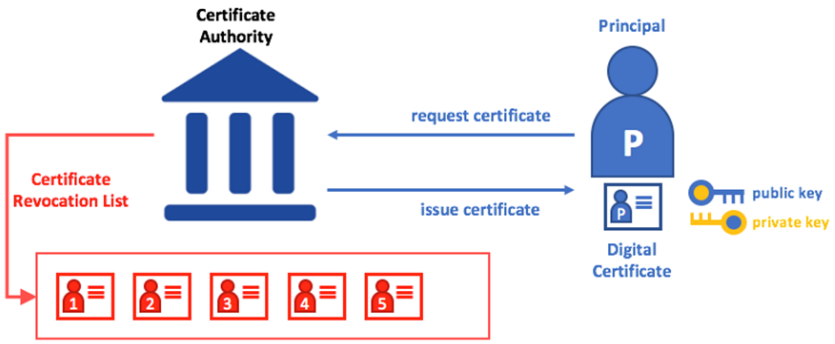

Figura 4.2: Infrastruttura a chiave pubblica

Per avere una comunicazione sicura tra tutti i partecipanti e garantire l'autenticità dei messaggi, la rete blockchain utilizza lo standard **PKI**. Gli elementi fondamentali in un'infrastruttura del genere sono i seguenti:

- **Certificati digitali:** Un certificato digitale è un documento nel quale sono presenti le informazioni riguardanti il titolare dello stesso. La tipologia più diffusa è quella conforma allo standard *X.509*.
- **Chiavi pubbliche e private:** I meccanismi di autenticazione si basano spesso su firme digitali, utilissime per garantire l'integrità del messaggio. Queste firme, richiedono che i diretti interessati posseggano ciascuno una chiave pubblica ed una privata: la prima da rendere disponibile per l'autenticazione, la seconda per la produzione stessa della firma.
- **Autorità di certificazione:** I certificati citati in precedenza sono prodotti dalle autorità di certificazione (CA) e si trovano nei protocolli di sicurezza.

- **Elenchi di revoche di certificati:** Con elenchi di revoche di certificati, si intende un elenco di riferimenti ai certificati che un'autorità di certificazione sa che sono stati revocati. Infatti, una CA per verificare le identità deve prima controllare tale elenco per assicurarsi che il certificato sia ancora valido.

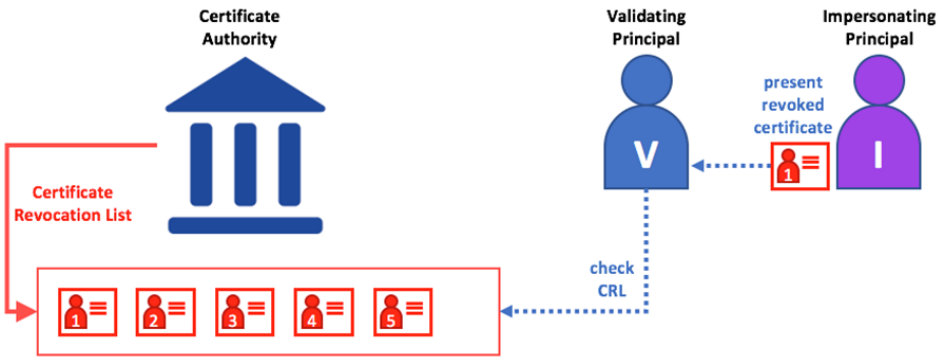

Figura 4.3: Elenchi di revoche di certificati

Politiche: Una politica è un insieme di regole che definiscono la struttura su cui vengono prese le decisioni e vengono raggiunti risultati specifici[29]. Identificano chi ha l'accesso ad una risorsa o quali diritti si hanno su di essa; non sono altro che il meccanismo per la gestione dell'infrastruttura. Le politiche stabiliscono come i membri di una rete concordano sull'accettazione o il rifiuto di modifiche alla rete, a un canale o a uno smart contract e sono definite all'inizio dalla configurazione originale della rete da parte dei membri del consorzio: questo non vieta il loro aggiornamento nel tempo man mano che la rete evolve. Si può pensare, ad esempio, ai criteri per l'aggiunta o la rimozione di membri da un canale, alla modifica della procedura di formazione dei blocchi o ancora al numero di organizzazioni richieste per approvare uno smart contract. In breve, ogni operazione sulla rete Fabric è soggetta a una politica. Sono proprio loro a rendere Hyperledger Fabric differente dalle altre blockchain, in quanto in Ethereum o Bitcoin, ad esempio, esse non possono essere modificate in corso d'opera, ma sono fisse in quanto ogni nodo della rete può validare le transazioni. In Fabric invece, essendo una blockchain autorizzata, vi è fiducia tra i componenti e non c'è la necessità di risalire al processo che governa il codice per poter apportare modifiche. Ovviamente, prima di poter cambiare la definizione di una politica, le organizzazioni devono concordare: vanno quindi raccolte le firme allegate alla proposta e alle transazioni e, se la governance che la rete ha in precedenza approvata è soddisfatta, si può procedere con le modifiche.

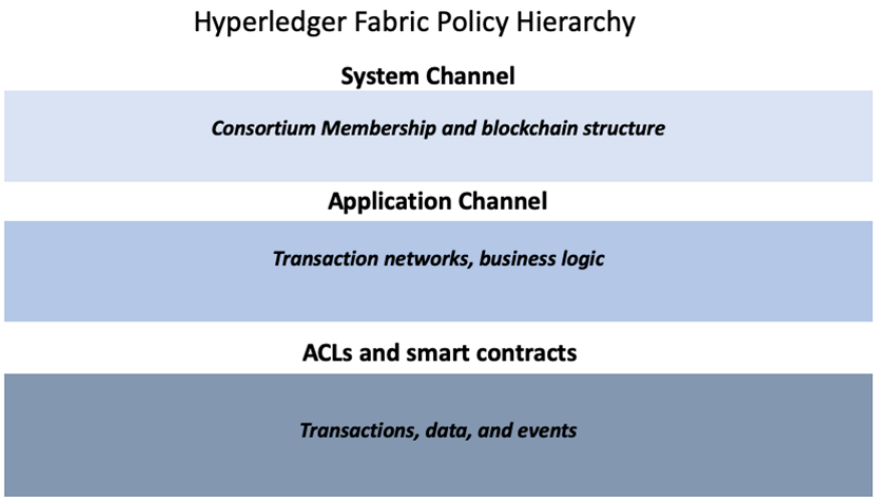

Figura 4.4: Gerarchia delle politiche in Fabric

In figura 4.4 è illustrata una gerarchia di una politica, in quanto esse vengono implementate su diversi livelli: ogni ambito, infatti, si occupa di diversi aspetti del funzionamento della stessa rete. Le politiche in Fabric sono adattabili e possono essere personalizzate per soddisfare le specifiche esigenze di una rete, ma, nonostante ciò, la loro struttura tende comunque a creare una divisione tra i domini controllati dal servizio di ordinazione e quelli dei membri del consorzio.

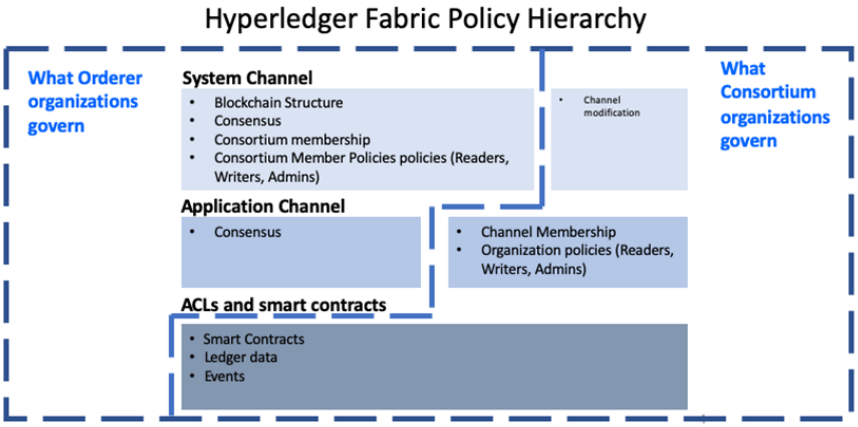

Figura 4.5: Gerarchia delle politiche in Fabric

In figura 4.5 viene mostrato come le policy predefinite gestiscano i diversi domini delle policy Fabric: questi domini consentono di assegnare privilegi e ruoli differenti a ciascuna organizzazione. Così facendo, si permette ai fondatori del servizio di ordinazione di stabilire le regole iniziali e la composizione del consorzio e, al contempo, si permette alle organizzazioni appartenenti allo stesso di creare canali applicativi privati, gestire la propria logica aziendale e controllare l'accesso ai dati presenti in rete.
É già stato fatto presente che le politiche, all'interno di una rete Fabric, possono essere modificate in corso d'opera, ma è importante comprendere che il cambiamento delle politiche è gestito da una politica all'interno della stessa politica: la **mod policy** o **politica di modifica**. Funziona come qualsiasi altra politica all'interno di una configurazione di rete o canale, definendo un insieme di organizzazioni autorizzate a modificarla.

Avendo un quadro più chiaro delle componenti di una rete Fabric, si può passare a descrivere gli attori principali che si trovano al suo interno.

4.2.1 Autorità di certificazione

Le **autorità di certificazione** (*CA*) rilasciano i certificati digitali conformi allo standard *X.509*. Tra le *CA* più comuni si trovano *Symantec*, *GeoTrust*, *DigiCert*, *GoDaddy* e *Comodo*.

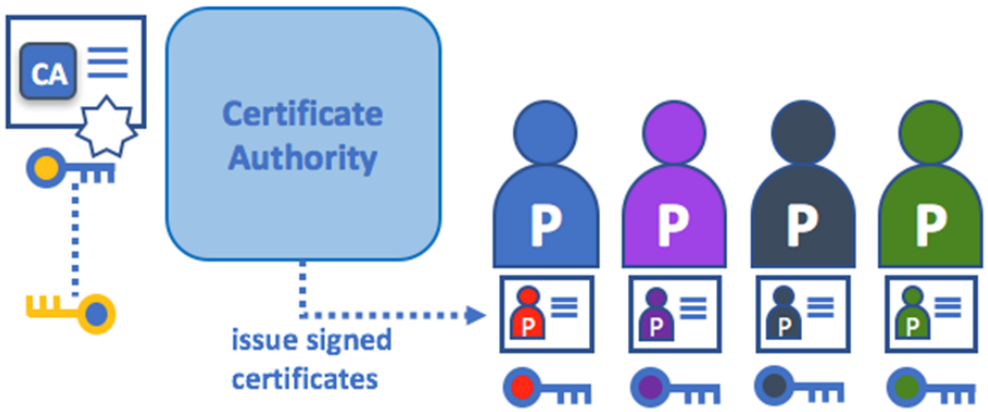

Figura 4.6: Autorità di certificazione

Le *CA* firmano digitalmente i certificati e collegano gli attori della rete con la propria chiave pubblica. Questo genera fiducia negli attori, dando per scontato che ci si fida delle *CA*.
Le *CA* hanno anch'esse un proprio certificato, reso accessibile, in modo da permettere ai chi utilizza le identità emesse da una specifica CA di verificarle.

Esistono due tipologie di CA: **CA radice** e **CA intermedie**. Le prime devono distribuire in modo sicuro i certificati agli utenti, distribuendoli alle seconde in modo da creare una *catena di fiducia*; in questo modo si garantisce maggiore sicurezza scalando la loro funzione e limitando l'esposizione della CA radice. Infatti, se quest'ultima venisse compromessa, metterebbe in pericolo l'intera struttura a catena, mentre il rischio è minore se ad essere compromessa è una intermedia. L'utilizzo dei due tipi di CA dipende dalle esigenze della rete.

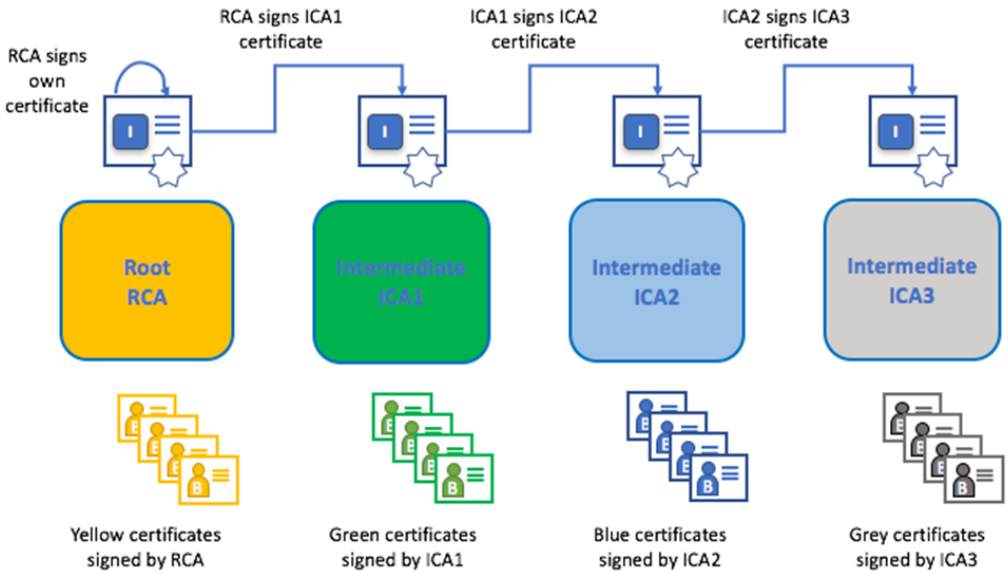

Figura 4.7: Autorità di certificazione

Fabric include una componente integrata che consente di creare *CA* all'interno delle reti blockchain chiamato **Fabric CA**; questo agisce come una *CA* radice privata progettata proprio per gestire le identità digitali dei partecipanti sotto forma di certificati *X.509*.

4.2.2 Peer

I nodi **peer** sono una parte essenziale all'interno di una rete blockchain, in quanto ospitano ledger e smart contract. I ledger registrano in modo immutabile tutte le transazioni generate dagli smart contract e sono utilizzati per incapsulare rispettivamente i processi condivisi e le informazioni condivise all'interno di una rete. Si possono inoltre creare, arrestare, riconfigurare e anche eliminare.

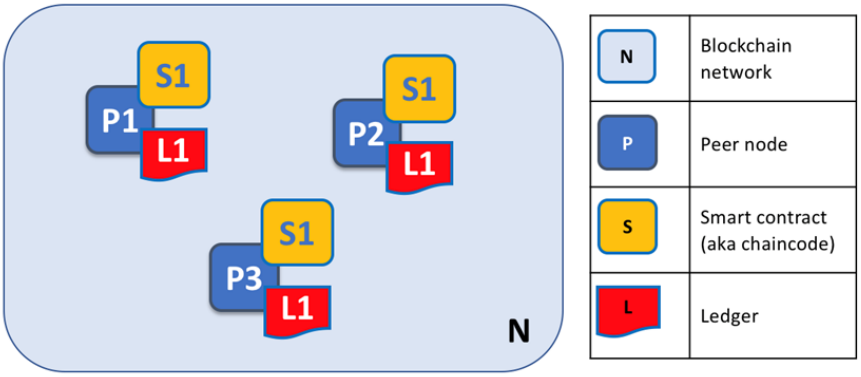

Figura 4.8: Rete blockchain composta da nodi peer

Nella figura 4.8 i tre peer *P1*, *P2* e *P3* mantengono il ledger *L1* ed utilizzano lo stesso chaincode *S1*.
É importante sapere che un peer può ospitare più di un ledger.

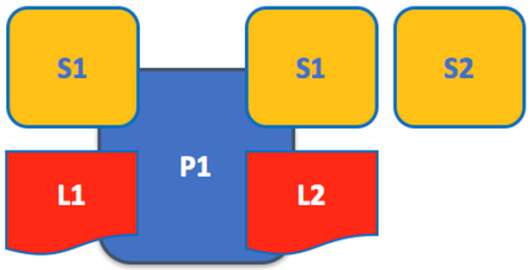

Figura 4.9: Ledger multipli

Ciascuno ledger può non avere o avere un numero qualsiasi di chaincode; nella figura 4.9 il peer *P1* ospita sia *L1* che *L2* che, a loro volta, permettono l'accesso solo tramite il chaincode *L1* (per quanto riguarda *P1*) o tramite *S1* ed *S2* (per quanto riguarda invece *L2*). Logicamente è possibile ospitare un ledger sul peer che non ha alcun chaincode che vi acceda, ma è raro imbattersi in una situazione simile in quanto, oltre i chaincode installati sui peer per permettere l'utilizzo degli stessi da parte di applicazioni esterne, vi sono sempre dei peer di sistema che sono presenti sul peer. Oltre a ledger multipli, uno stesso peer può ospitare anche chaincode multipli, in quanto non è presente nessuna relazione tra il numero di ledger e quello di chaincode che vi hanno accesso.

Nella figura 4.10 *P1* presenta *L1* e *L2*, e si accede al primo tramite gli smart contract *S1* ed *S2*, al secondo tramite *S1* ed *S3*.

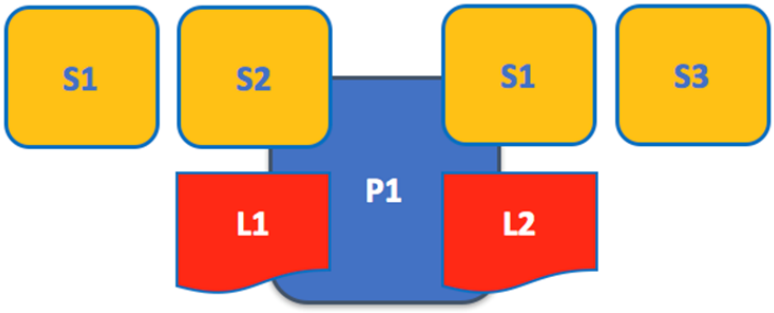

Figura 4.10: Chaincode multipli

I peer possono assumere diversi ruoli in base alla configurazione di rete:

- **peer committente:** riceve blocchi di transazioni generate, i quali vengono validati prima di essere registrati nella copia del ledger tramite un'operazione di aggiunta. Ogni nodo peer in un canale è un peer committente.
- **peer di approvazione:** lo smart contract sul peer è usato da un'applicazione client nel generare una risposta di transazione con firma digitale. Ogni peer può essere un peer di approvazione se ha installato un contratto intelligente.
- **Leader peer:** è colui che distribuisce le transazioni dall'ordinante agli altri peer committenti. Possono essere presenti più leader e ciò può migliorare la resilienza e la scalabilità nelle reti di grandi dimensioni
- **peer di ancoraggio:** possono essere utili per favorire la comunicazione con un peer facente parte di un'altra organizzazione. Possono essere presenti più peer di questa tipologia, oppure possono essere assenti.

É utile sapere che uno stesso peer può essere un peer di ognuno di queste 4 categorie contemporaneamente e che, di norma, per facilitare l'organizzazione all'interno di una rete, i primi tre tipi sono sempre presenti.
Per quanto riguarda le applicazioni, esse devono sempre passare per i peer per avere accesso a ledger e chaincode, ed è grazie alla loro connessione ai peer che possono eseguire un chaincode ed quindi interrogare o aggiornare un ledger.

Nella figura 4.11, l'applicazione *A* per prima cosa si connette al peer *P1* ed invoca una proposta. *P1* la riceve ed invoca, con la proposta, lo smart contract *S1*. Quest'ultimo genera una risposta che torna ad *A*. A questo punto, *A* crea una transazione e la invia al servizio di ordinazione *O1* il quale la invia, sotto forma di blocco, al peer. *P1* deve convalidarla e aggiornare *L1* che, infine, genera un evento che invia ad *A* che segna la fine del processo.

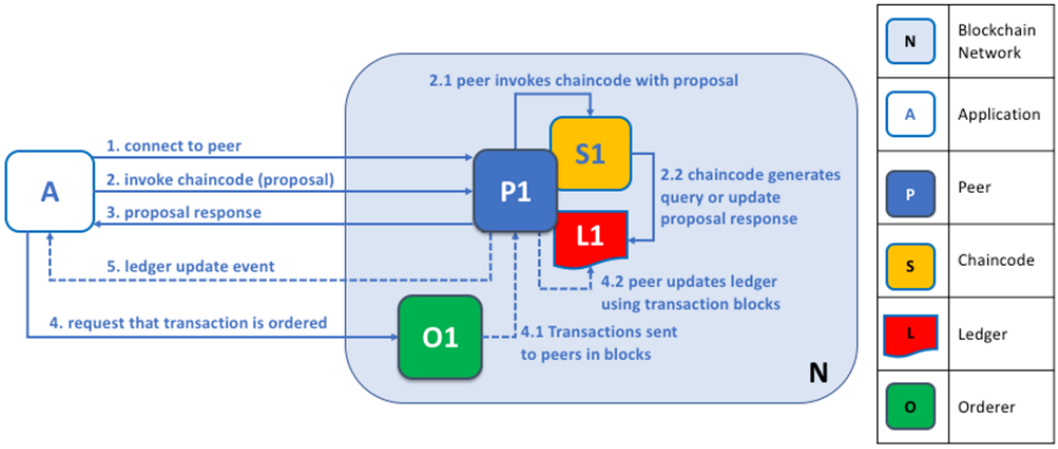

Figura 4.11: Come le applicazioni si connettono ai peer

Ad ogni peer viene assegnata un'identità mediante un certificato digitale; l'assegnazione è gestita da un amministratore delle organizzazioni che fanno parte del consorzio.

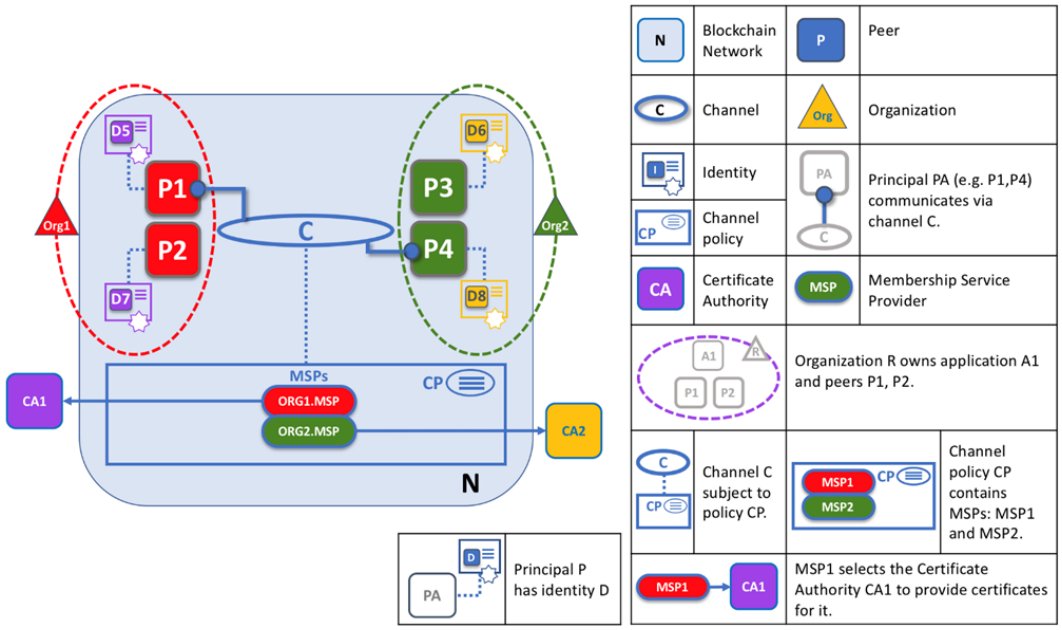

Figura 4.12: Peer e identità

Il certificato digitale identifica l'organizzazione di appartenenza del peer attraverso un MSP specifico per quel canale. Nella figura 4.12 si considerano *P1* e *P2*: essi hanno identità emesse da *CA1* e sono associati all'organizzazione *Org1* utilizzando *ORG1.MSP*; allo stesso modo, *P3* e *P4* sono identificati da *ORG2.MSP* come parte di *Org2*. Inoltre, una politica nella configurazione del canale utilizza l'identità del peer per stabilirne i diritti, il tutto è gestito dall'*MSP*, il quale definisce

il ruolo del peer all'interno dell'organizzazione e quindi il suo accesso alle risorse blockchain. Ogni peer può appartenere ad una sola organizzazione e quindi essere associato ad un unico *MSP*.

Infine, la posizione fisica di un peer non è cruciale: ciò che conta è il certificato digitale associato al peer, che lo identifica come appartenente a una specifica organizzazione. Ad esempio, *P3* potrebbe essere ospitato nel data center di *Org1*, ma se il suo certificato digitale è emesso da *CA2*, allora è considerato di proprietà di *Org2*.

4.2.3 Servizio di Ordinazione

In Hyperledger Fabric, c'è un nodo dedicato chiamato **ordinante** o **servizio di ordinazione**, il quale gestisce l'ordinamento delle transazioni in modo deterministico. Questo assicura che i blocchi validati dai peer siano definitivi e corretti, eliminando la possibilità di ledger divergenti o fork. Inoltre, separare l'ordinamento dall'esecuzione del chaincode fornisce vantaggi in termini di prestazioni e di scalabilità. Il servizio di ordinazione non solo ordina le transazioni, ma anche gestisce l'elenco delle organizzazioni autorizzate a creare canali sulla blockchain, ovvero il **consorzio**. Gli ordinanti applicano anche controlli di accesso sui canali, limitando chi può leggere, scrivere e configurare i dati, verificando che le modifiche soddisfino le politiche definite nel canale.

Anche il servizio di ordinazione ha una propria identità, grazie ad un certificato digitale e all'*MSP* e, così come i peer, appartiene ad un'organizzazione ed utilizza un'autorità di certificazione separata per garantire la sicurezza e l'affidabilità del sistema.

Il servizio di ordinazione può sembrare una componente *centralizzata*, poiché viene utilizzato per creare la rete e si connette a tutti i canali della rete. In realtà, può essere completamente *decentralizzato* in quanto può essere composto da molti nodi individuali, di proprietà di diverse organizzazioni.

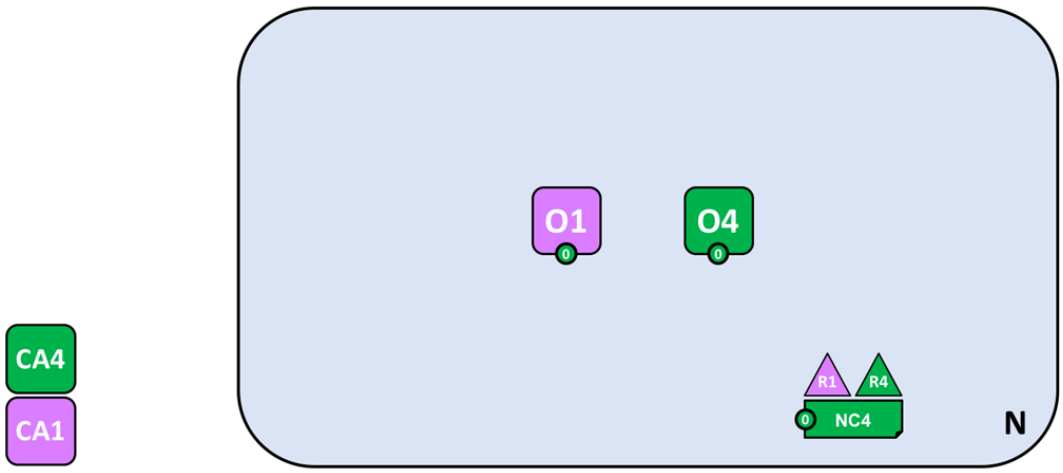

Figura 4.13: Servizio di ordinazione decentralizzato

Nella figura 4.13, il servizio di ordinazione comprende i nodi del servizio di ordinazione *O1* e *O4*, dove *O1* è fornito dall'organizzazione *R1* e *O4* è fornito dall'organizzazione*R4*. La configurazione di rete *NC4* definisce i permessi delle risorse di rete per gli attori di entrambe le organizzazioni *R1* e *R4*.

Come si è già detto in precedenza, le applicazioni coinvolte nell'aggiornamento dei ledger seguono un processo che garantisce che tutti i peer nella rete blockchain mantengano ledger coerenti. Il processo si divide in tre fasi:

Fase uno: PROPOSTA. Un'applicazione client invia una proposta di transazione a un gruppo di peer che restituiscono una risposta di proposta all'applicazione client.

Fase due: ORDINAMENTO E CREAZIONE DI BLOCCHI. É qui che entra in gioco il servizio di ordinazione: le transazioni contenenti risposte di proposte approvate vengono inviate dall'applicazione ad un nodo del servizio di ordinazione, il quale crea blocchi di transazioni che saranno distribuiti a tutti i peer. I nodi del servizio di ordinazione, lavorando insieme, organizzano le transazioni inviate in sequenza e le impacchettano nei blocchi, che diventano i blocchi della blockchain. La dimensione e la durata dei blocchi dipendono dalla configurazione del canale. Questi blocchi vengono salvati nel ledger e distribuiti a tutti i peer.

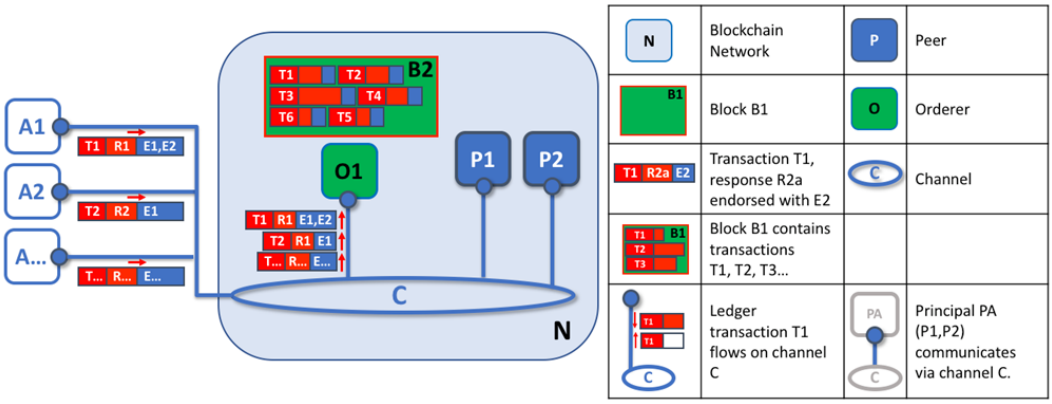

Figura 4.14: Ordinamento e creazione dei blocchi

Nella figura 4.14, *A1* invia una transazione *T1* arrivata dagli eventi *E1* ed *E2* ad *O1*. In contemporanea, *A2* invia *T2* ricevuta da *E1* ad *O1* il quale impacchetta *T1* da *A1* e *T2* da *A2*, insieme ad altro, nel blocco *B2*. Possiamo vedere che la sequenza con cui le transazioni vengono salvate nel blocco non è per forza la stessa con cui esse sono arrivate al servizio di ordinazione, in quanto più nodi possono aver ricevuto più o meno nello stesso istante delle transazioni. L'importante è che il servizio definisca un ordine e che i peer utilizzino questo durante la convalida e il commit.

Inoltre, si può notare come gli ordinanti non eseguano smart contract o elaborano transazioni, ma sono comunque responsabili di questi processi fondamentali.

Fase tre: CONVALIDA E COMMIT. L'ordinante distribuisce i blocchi a tutti i peer connessi i quali, convalidano i blocchi in modo indipendente ma deterministico, verificando che le transazioni siano state approvate dagli altri peer, che le approvazioni corrispondano e che non siano state invalidate da altre transazioni recenti. Le transazioni convalidate vengono aggiunte al ledger, assicurando coerenza tra tutti i peer nella rete blockchain.

In sintesi, il processo di ordinamento e gestione delle transazioni in una rete blockchain segue un flusso in tre fasi: proposta, ordinamento e creazione di blocchi, convalida e commit. Questo assicura che tutti i peer mantengano registri coerenti e aggiornati, garantendo l'integrità e l'affidabilità della rete.

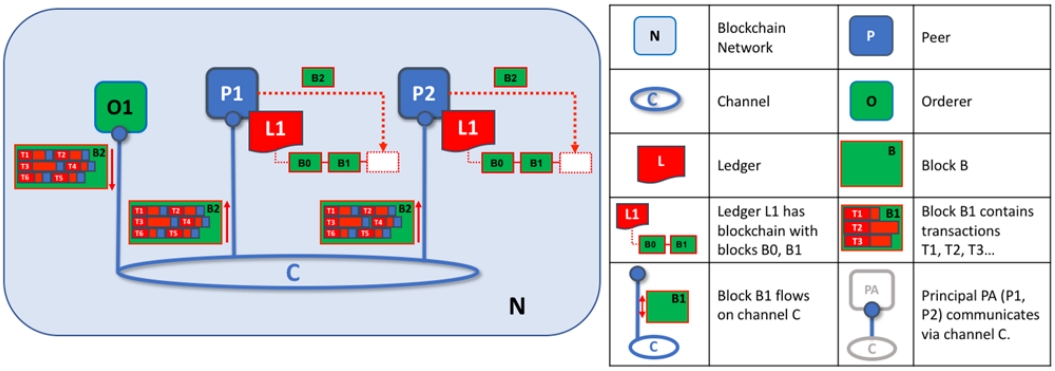

Figura 4.15: Convalida e commit

Nella figura 4.15 *O1* distribuisce il blocco *B2* a *P1* ed a *P2*: il primo elabora il blocco, determinando l'aggiunta di un nuovo blocco ad *L1* su *P1*. Il secondo elabora lo stesso blocco, determinando l'aggiunta di un nuovo blocco ad *L1* su *P2*. In questo modo, *L1* è stato costantemente aggiornato sui peer e i due possono informare le applicazioni connesse che la transazione è stata elaborata.

Questo processo è chiamato **consenso** in quanto tutti i peer hanno raggiunto un accordo sull'ordine e sul contenuto delle transazioni, in un processo in cui il servizio di ordinazione funge da mediatore.

4.2.4 Ledger

Il **ledger** in Hyperledger Fabric svolge un ruolo cruciale nella registrazione e nell'assicurazione dell'integrità delle transizioni di stato. Ogni transazione, derivante da invocazioni di chaincode effettuate dai partecipanti, genera una serie di coppie *chiave-valore* che vengono registrate nel ledger durante operazioni come creazione, aggiornamento o eliminazione di asset.

Hyperledger Fabric organizza il suo sistema di registro attraverso due componenti principali: il **registro delle transazioni** e lo **stato mondiale**. Ogni partecipante all'interno di una rete possiede una copia del ledger relativo.

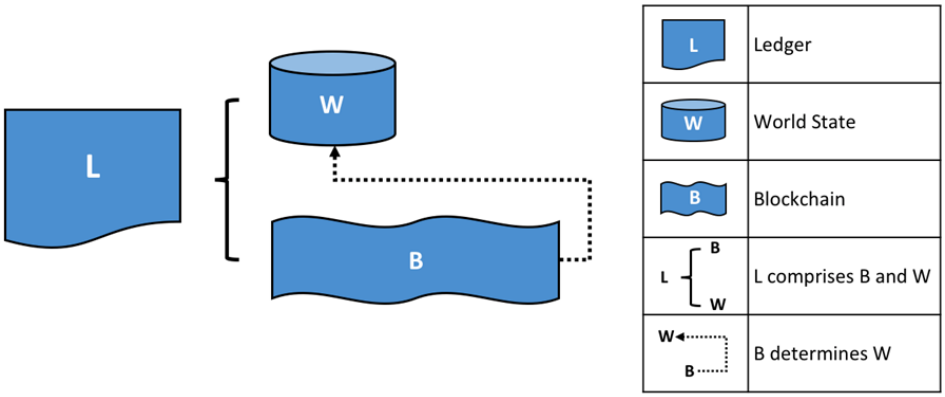

Figura 4.16: Struttura del ledger

Il registro delle transazioni tiene traccia di tutte le transazioni che hanno contribuito a determinare lo stato attuale del ledger, fornendo una cronologia degli aggiornamenti. La struttura dei dati del registro è differente da quella dello stato mondiale in quanto è immutabile.
Lo stato mondiale è un database che conserva i valori attuali di un insieme di stati contabili. Questo aspetto semplifica l'accesso diretto da parte di un programma al valore corrente di uno stato, evitando la necessità di calcolarlo attraverso l'intero log delle transazioni. É soggetto a frequenti cambiamenti poiché gli stati possono essere creati, aggiornati e cancellati nel corso del tempo: rappresenta lo stato attuale del ledger in un determinato momento.
Quando il ledger viene creato per la prima volta, lo stato mondiale è vuoto, in quanto non sono ancora stati registrati né asset né transazioni; ogni transazione valida che avviene sulla blockchain viene però registrata, contribuendo a modificarlo. Quindi le transazioni, sia valide che non valide, vengono aggiunte a un registro distribuito, ma solo le transazioni valide aggiornano lo stato mondiale. Così facendo il ledger può essere ricostruito dalla blockchain stessa, recuperando tutte le transazioni passate e gli stati associati ad esse.

Questa caratteristica può tornare utile ad esempio quando un nuovo peer viene aggiunto alla rete, in quanto lo stato mondiale viene generato automaticamente in base alle transazioni passate. Inoltre, se un peer dovesse fallire in modo anomalo, il suo stato mondiale può essere ripristinato al riavvio, garantendo che tutte le transazioni precedenti siano ancora disponibili e che il peer sia sincronizzato con il resto della rete.

Dal punto di vista dell'implementazione, lo stato mondiale è fisicamente realizzato attraverso un database. Due delle opzioni più comuni per questo database sono *LevelDB* e *CouchDB*, ognuno dei quali ha le proprie caratteristiche e vantaggi.

LevelDB è spesso la scelta predefinita ed è particolarmente adatto quando gli stati contabili sono semplici coppie chiave-valore. Questo tipo di database è integrato direttamente nel nodo peer, il che significa che è parte integrante del processo del sistema operativo. LevelDB eccelle nell'archiviazione e nel recupero efficienti di dati semplici.

D'altra parte, CouchDB diventa una scelta più appropriata quando gli stati del ledger sono strutturati come documenti `json`. Questo database supporta query avanzate e l'aggiornamento di tipi di dati più complessi.

4.2.5 Smart contract

Uno **smart contract** definisce la logica eseguibile che genera nuove informazioni da aggiungere al ledger e stabilisce le regole tra le diverse organizzazioni. Le applicazioni richiamano uno smart contract per generare transazioni registrate nel ledger. Uno smart contract può implementare le regole di governance per qualsiasi tipo di asset aziendale, applicandole automaticamente durante la sua esecuzione.

I termini *smart contract* e *chaincode* vengono usati nel parlato per indicare la stessa cosa, ma in realtà uno smart contract definisce la logica delle transazioni che controlla il ciclo di vita di un oggetto nel ledger e viene poi incapsulato in un chaincode. É quest'ultimo che viene distribuito sulla rete blockchain. Quindi uno smart contract è una parte del chaincode ed è infatti possibile definire più smart contract all'interno dello stesso chaincode. Uno smart contract ha accesso alle due parti del ledger inserendo, recuperando ed eliminando stati nello stato mondiale e interrogando il registro immutabile delle transazioni sulla blockchain. Alcune operazioni di base sono:

- `get` : query per recuperare informazioni sullo stato corrente di un oggetto.
- `put` : crea un nuovo oggetto o ne modifica uno esistente nello stato mondiale del ledger.
- `delete` : rimuove un oggetto dallo stato corrente del ledger, ma non dalla sua cronologia .

Ogni chaincode ha una politica di approvazione associata, che si applica a tutti gli smart contract definiti al suo interno. Questa politica è fondamentale, poiché indica quali organizzazioni nella rete blockchain devono firmare una transazione affinché questa sia considerata valida. Ad esempio, una politica di approvazione potrebbe richiedere che tre delle quattro organizzazioni partecipanti alla rete blockchain firmino una transazione affinché sia valida. Le politiche di approvazione sono solo un esempio di politica in Hyperledger Fabric, poiché è possibile definirne altre per governare l'accesso e le modifiche al registro, così come per l'aggiunta o la rimozione dei partecipanti dalla rete. Queste politiche dovrebbero essere concordate preventivamente dal consorzio di organizzazioni partecipanti, anche se possono essere soggette a modifiche.

Una transazione viene convalidata in due fasi da ciascun peer della rete:

1. si controlla se la transazione è stata firmata da un numero sufficiente di organizzazioni secondo la politica di approvazione;
2. si verifica che il valore corrente dello stato mondiale corrisponda al set di letture della transazione quando è stata firmata.

Solo se la transazione supera entrambi questi test viene considerata valida.

Chaincode: Il chaincode è un programma scritto in Go, Node.js o Java che implementa un'interfaccia predefinita. Viene eseguito in un contenitore Docker isolato dal processo di approvazione del peer; inoltre inizializza e gestisce lo stato del ledger attraverso le transazioni inviate dalle applicazioni.

Un chaincode gestisce generalmente la logica aziendale concordata dai membri della rete e può essere considerato uno smart contract. Gli aggiornamenti del ledger creati da un chaincode sono limitati a quel chaincode ed altri non vi hanno accesso direttamente. Tuttavia, all'interno della stessa rete e con le autorizzazioni appropriate, un chaincode può invocarne un altro per accedere al suo stato. Per quanto riguarda la sua distribuzione, è necessario avere il consenso delle organizzazioni sulla gestione del chaincode prima che possa essere utilizzato su un canale. Per installare un chaincode bisogna impacchettarlo in un file *tar* e quindi installarlo su ogni peer che eseguirà e approverà le transazioni. Dopodiché verrà restituito un identificatore del pacchetto chaincode, ovvero l'etichetta dello stesso combinata con un *hash*. Questo identificatore viene utilizzato per associare il pacchetto installato con la definizione del chaincode approvata. Per approvare una definizione, i membri del canale devono concordare su diversi parametri, tra cui:

- `Nome:` ciò che le applicazioni usano per invocare il chaincode;
- `Versione:` valore associato a un determinato pacchetto di chaincode;
- `Politica di approvazione:` come eseguire e convalidare l'output della transazione, e quindi quali e quante organizzazioni sono coinvolte;
- `Sequenza:` numero di volte in cui è stato definito il chaincode, tenendo traccia degli aggiornamenti del chaincode;
- `Configurazione della raccolta:` percorso di un file di definizione della raccolta dati privati;
- `Plugin ESCC/VSCC:` nome di un plugin di approvazione o convalida personalizzato da utilizzare dal chaincode.
- `Inizializzazione:` funzione utilizzata per inizializzare il chaincode.

Questa approvazione è fondamentale affinché il chaincode possa essere utilizzato nel canale.

Una volta che un numero sufficiente di organizzazioni ha approvato la definizione del chaincode, un'amministrazione dell'organizzazione può impegnare la definizione nel canale. Questo processo coinvolge l'invio di una transazione di `commit` ai peer del canale per la conferma della definizione. Dopo il `commit`, il chaincode viene attivato su tutti i peer del canale e può essere utilizzato per eseguire le transazioni.

Infine, dopo che il chaincode è stato definito sul canale, le organizzazioni Org1 e Org2 possono iniziare ad utilizzarlo. Una volta attivo, il chaincode può essere invocato per eseguire le operazioni desiderate e può essere aggiornato in qualsiasi momento.

Per poterlo fare, bisogna seguire i seguenti passaggi:

1. **Riconfezionare il chaincode**;
2. **Installare il nuovo pacchetto sui peer**: questo genererà un nuovo ID pacchetto. É importante anche modificare la versione per verificare gli aggiornamenti;
3. **Approvare una nuova definizione** : aggiornando la versione l'ID del pacchetto, ogni organizzazione deve approvare la nuova definizione.
4. **Inviare la definizione al canale**: passaggio che può essere portato avanti solo una volta che un numero sufficiente di membri del canale ha approvato la nuova definizione; la nuova definizione sovrascrive quella precedente.

Dopo aver confermato la definizione del chaincode, verrà avviato un nuovo docker del chaincode con i binari aggiornati.
È anche possibile aggiornare una politica di approvazione senza riconfezionare o reinstallare il chaincode ma approvando una nuova definizione di chaincode con la nuova politica e inviarla al canale. Le organizzazioni possono utilizzare pacchetti chaincode diversi per creare più istanze di chaincode su un canale, ciascuna con politiche di approvazione diverse. Ciò consente una maggiore flessibilità nell'implementazione di logiche aziendali specifiche.

4.2.6 Canali

I **canali** in Hyperledger Fabric fungono da meccanismo attraverso il quale un insieme specifico di peer e applicazioni può comunicare all'interno di una rete blockchain. Ogni canale ha il suo registro completamente separato ma è comunque possibile stabilire comunicazioni tra i vari canali in modo che applicazioni e smart contract possano accedere e scambiare informazioni. Questo è particolarmente utile quando ci sono partecipanti concorrenti all'interno delle reti e non si desidera che ogni transazione sia visibile a tutti. Creando un canale, due o più partecipanti possono assicurarsi che solo loro abbiano accesso alle copie del ledger relative a quel canale, mantenendo così la privacy delle loro transazioni.

Hyperledger Fabric consente a un'organizzazione di partecipare simultaneamente a più reti blockchain separate tramite l'uso dei canali. Unendosi a più canali, un'organizzazione può creare una sorta di "rete di reti", permettendo una condivisione efficiente delle risorse e mantenendo al contempo la privacy dei dati e delle comunicazioni.

I membri del canale possono eseguire un smart contract solo dopo che il chaincode è stato definito all'interno del canale stesso: ogni membro del canale accetta i parametri del chaincode approvando la sua definizione per la propria organizzazione cosicché questa possa essere impegnata nel canale. Gli smart contract all'interno del chaincode possono quindi essere eseguiti dai membri del canale, rispettando la politica di approvazione specificata nella definizione.

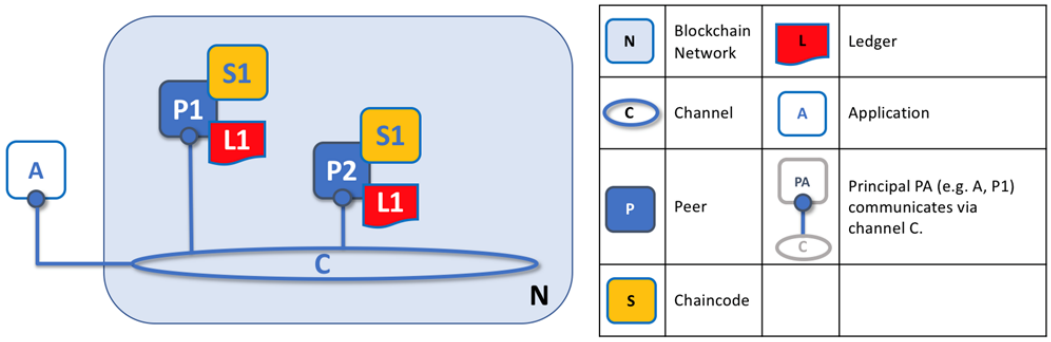

Figura 4.17: Canali

Nella figura 4.17 si può notare come, per comunicare, l'applicazione *A* e i peer *P1* e *P2* devono servirsi del canale *C*.

Si può quindi dire che i canali possono essere visti come gruppi di peer fisici che collaborano per condividere e gestire copie del ledger associate a quel canale.

4.3 Costruzione di una rete blockchain in Hyperledger Fabric

Una rete blockchain campione che si può costruire in Hyperledger Fabric può essere rappresentata come rappresentato nella figura 4.18.

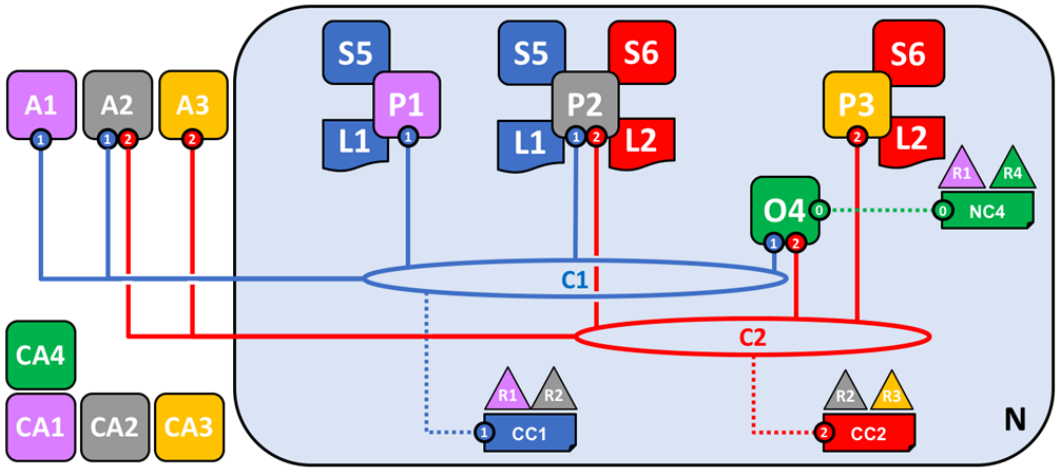

Figura 4.18: Rete Blockchain

Nella figura, sono indicate con *R1*, *R2*, *R3* ed *R4* le **organizzazioni** che hanno creato la rete. *R4* è stato scelto come iniziatore della rete e, come tale, imposta la versione iniziale della stessa, senza la possibilità di eseguirvi transazioni di tipo commerciale. *R1* ed *R2* sono capaci di eseguire

transazioni all'interno del canale *C1* mediante l'utilizzo di un'**applicazione client**, così come *R2* ed *R3* per il canale *C2*.

Con *P1*, *P2* e *P3* si identificano i **peer** della rete, i quali mantengono una copia dei *ledger L1* ed *L2* associati ai rispettivi canali. Quindi, *P1*, che è connesso solo al canale *C1*, conserva una copia del ledger *L1*, mentre *P2*, connesso ad entrambi i canali, conserva una copia di *L1* ed una copia di *L2*. Allo stesso modo *P3* conserva una copia solo di *L2*.

NC4 indica la configurazione di rete nella quale sono specificate le politiche di gestione e, come si nota dalla 4.18, la rete è sotto il controllo di *R1* ed *R4*. Per quanto riguarda i canali, le politiche di regolazione di *C1* sono indicate nella configurazione del canale *CC1* ed è sotto il controllo di *R1* ed *R2*, mentre le politiche di regolazione di *C2* sono indicate nella configurazione del canale *CC2* ed è sotto il controllo di *R2* ed *R3*. E' presente un **servizio di ordinazione** che si occupa di questioni amministrative per la rete che utilizza il canale di sistema *NC4*, ma supporta anche i due canali applicativi C1 e C2 per gestire l'ordinamento delle transazioni. Infine, ognuna delle 4 organizzazioni, ha un'**autorità di certificazione** preferita CA1, CA2, *CA3* ed *CA4*.

4.3.1 Costruzione di una rete blockchain

Per comprendere meglio la struttura e il funzionamento dell'intera rete, si parte da zero seguendo dall'inizio la sua costruzione come rappresentato in figura 4.19.

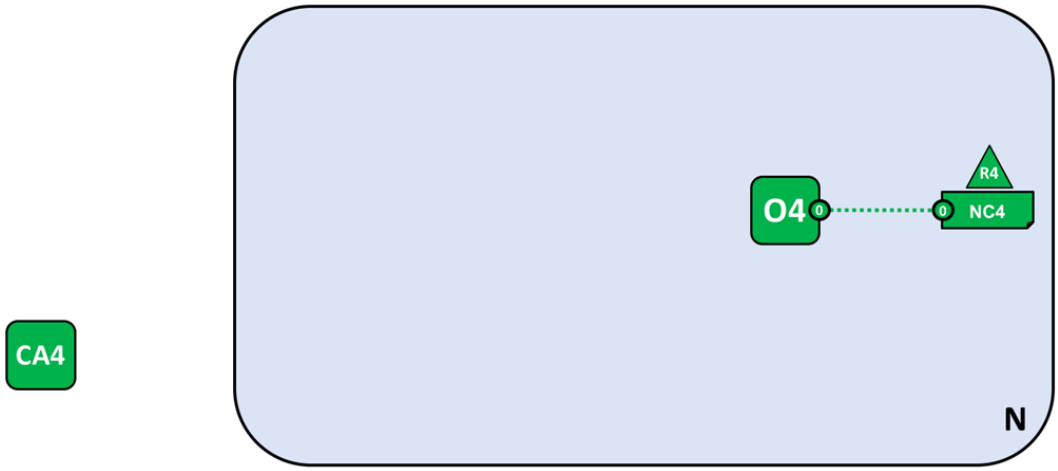

Figura 4.19: Rete Blockchain

La rete si forma una volta avviato il servizio di ordinazione che, nel caso specifico dell'esempio, è composto da un solo nodo *O4*, configurato seguendo una configurazione *NC4*. All'organizzazione *R4* è stato affidato il ruolo di iniziatore della rete conferendole tutti i diritti di amministrazione; l'**autorità di certificazione** *CA4* distribuisce identità agli amministratori e ai nodi di rete di *R4*. I certificati emessi dalle *CA* possono essere usati per firmare transazioni e avere quindi conferma dell'approvazione di queste da parte delle organizzazioni e, quindi, poter essere accettate nel ledger.

I certificati vengono usati per rendere nota la propria partecipazione ad una specifica organizzazione, infatti generalmente si trovano più *CA* all'interno di una stessa rete blockchain. Hyperledger Fabric fornisce una *CA* integrata chiamata **Fabric-CA**.

La loro mappatura alle organizzazioni viene ottenuta tramite il MSP. *NC4* può utilizzare un *MSP* nelle politiche, in modo da permettere agli attori di *R4* particolari diritti, come ad esempio la possibilità di aggiungere nuove organizzazioni alla rete.

Si può ampliare la rete consentendo anche agli utenti di *R1* la sua amministrazione, avendo così pari diritti con *R4*, nonostante sia *R4* a gestire *O4*. Si può quindi pensare ad *O4* come ad un punto di amministrazione che un accesso controllato alla rete a più organizzazioni.

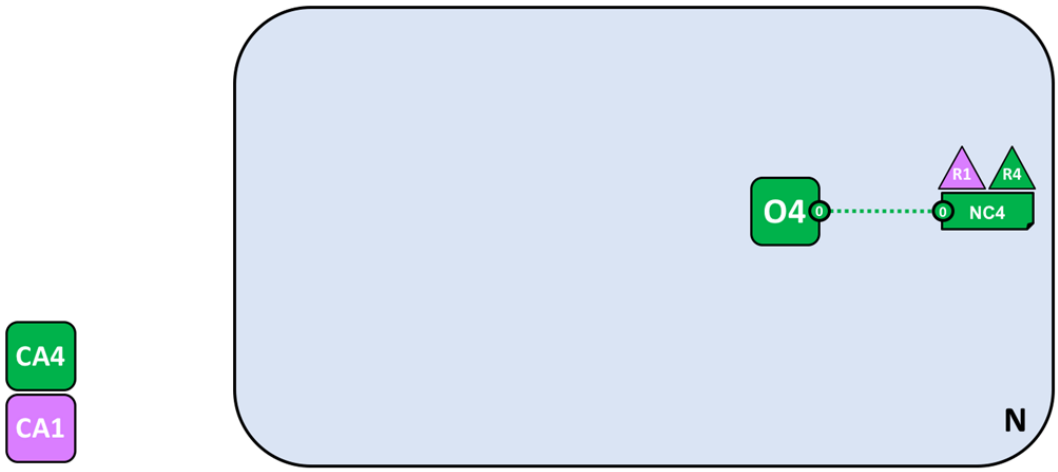

Figura 4.20: Rete Blockchain

Come si può notare in figura 4.20, parallelamente ad *R1* è stata aggiunta anche l'autorità di certificazione *CA1* per identificare gli utenti dell'organizzazione.

Ma manca ancora da definire un consorzio *X1*, letteralmente *un gruppo con un destino condiviso*, che può contenere un numero qualsiasi di membri organizzativi. Nel caso in questione, *X1* contiene *R1* ed *R2*, mentre la sua definizione sarà archiviata all'interno di *NC4*. Anche in questo caso,come si evince dalla figura 4.21, verrà aggiunta un'autorità di certificazione *CA2* per l'organizzazione *R2*.

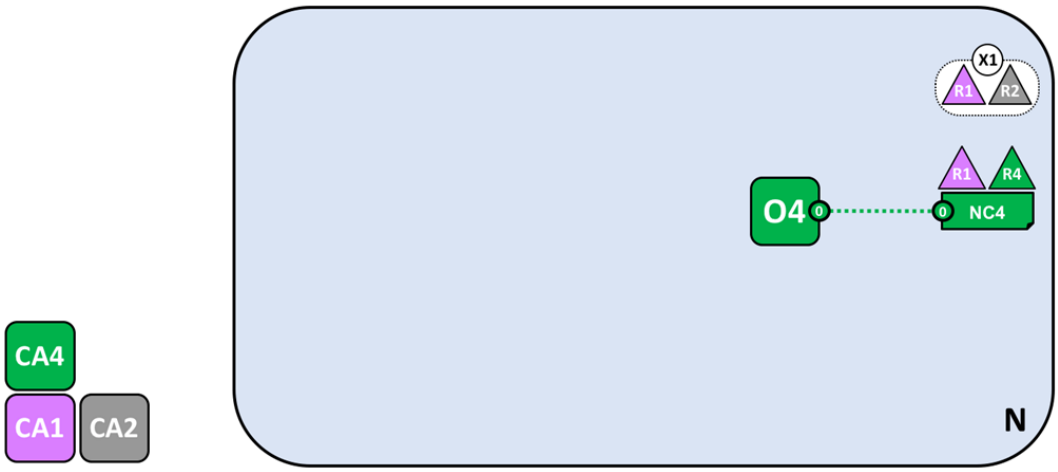

Figura 4.21: Rete Blockchain

La configurazione usata nella definizione della rete, permette solo ad *R1* ed *R4* di creare nuovi consorzi. L'importanza di questo nuovo elemento, risiede nel fatto che definisce l'insieme di organizzazioni nella rete che condividono la necessità di effettuare transazioni tra loro[29].
É proprio grazie ad *X1*, che si può creare il **canale** *C1*, attraverso il quale i membri di un consorzio possono comunicare tra loro.

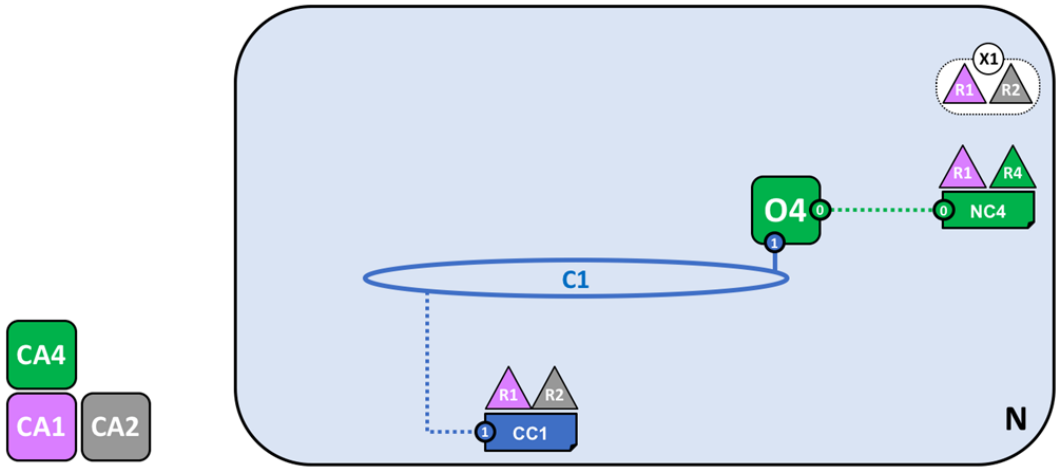

Figura 4.22: Rete Blockchain

La configurazione del canale in figura 4.22 è *CC1*, è separata da quella di rete *NC4* e al suo interno si possono trovare tutte le politiche riguardanti i diritti che *R1* ed *R2*, i quali gestiscono il canale, hanno su di esso. *C1* permette di comunicare privatamente su *X1*, in quanto le altre organizzazioni non sono presenti su di esso e quindi non partecipano allo scambio di transazioni mediante lo stesso: solo le due già presenti possono autorizzarne l'aggiunta.

I canali garantiscono la privacy da altri canali e dalla rete: Hyperledger Fabric consente alle organizzazioni di condividere l'infrastruttura e mantenerla privata allo stesso tempo.[29]

Il prossimo passo è connettere la rete blockchain e le componenti organizzative: a tal fine, aggiungiamo un nodo **peer** *P1* e un **ledger** *L1*.

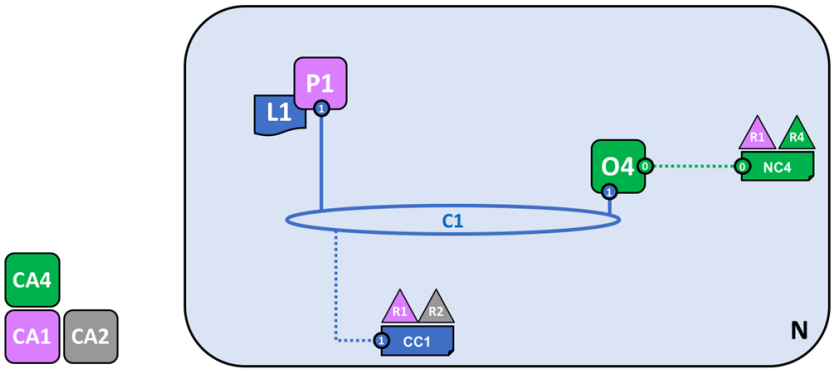

Figura 4.23: Rete Blockchain

Tramite il canale *C1*, ora *P1* e *O4* possono comunicare.
L1 è logicamente ospitato sul canale, ma fisicamente ospitato sul peer, il quale esiste proprio per questo scopo, come indicato in figura 4.23. Per la configurazione di *P1*, *CA1* emette un'identità *X.509* che lo associa all'organizzazione *R1*. I peer sono uniti al canale dalle organizzazioni che li possiedono; quando ciò accade, il peer inizia ad estrarre blocchi dall'ordinante il quale utilizza la configurazione del canale per determinare le autorizzazioni di P sul canale.
Perciò *R1* unisce *P1* a *C1*; *P1* estrae blocchi da *O4* che, tramite *CC1* determine le autorizzazioni di *P1* su *C1*.

4.3.2 Introduzioni delle applicazioni client e del chaincode

La parte successiva è connettere al ledger le applicazioni client per utilizzare alcuni dei servizi forniti dai peer. Uno **smart contract** *S5* viene installato quindi su *P1* e l'**applicazione client** *A1* lo può utilizzare per accedere ad *L1* tramite *P1*. Essendo tutti collegati al canale, possono comunicare tra loro e con *O4*, collegato anch'esso a *C1*.

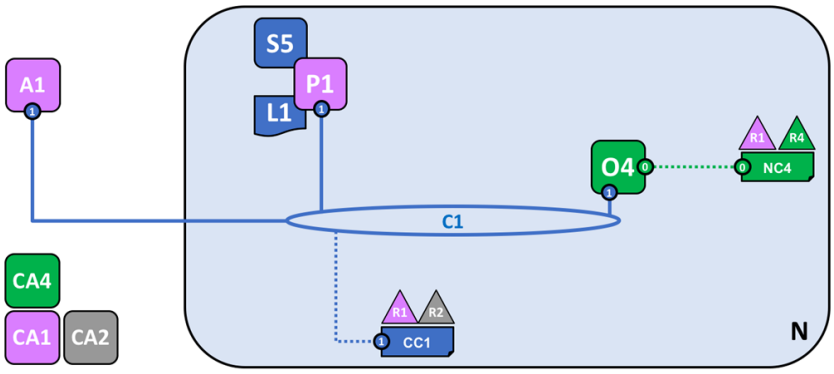

Figura 4.24: Rete Blockchain

Anche le applicazioni client hanno un'identità che le associano ad un'organizzazione, come in questo caso che *A1* è associata ad *R1* sempre tramite il canale *C1*. Gli accessi ad *L1* sono gestiti non dal peer, ma da un **chaincode** dello smart contract *S5*, nel quale troviamo diverse modalità per interrogare o aggiornare il ledger. Gli smart contract sono utili per generare transazioni che verranno poi distribuite ad ogni nodo della rete e devono essere installati sui peer e definiti su un canale. Per chiarezza, gli smart contract definiscono la logica della transazione, viene quindi confezionato il chaincode che viene poi distribuito sulla rete, ma i due termini vengono utilizzati in maniera interscambiabile.

Nel momento in cui sono presenti più canali all'interno della rete, ovviamente sarà presente uno smart contract per ogni canale, ma non sarà necessario installarli tutti su ogni nodo peer presente. Ad esempio, nella figura 4.18 il peer *P1* ha solo lo smart contract *S5*, così come *P3* ha solo *S6*, mentre su *P2* sono installati entrambi gli smart contract.

Il chaincode è gestito e governato dalle organizzazioni, che devono approvare una definizione dello stesso prima di poter utilizzare lo smart contract. Affinché l'approvazione del chaincode sia valida, un numero pari alla maggioranza delle organizzazioni deve dare conferma, cosicché lo smart contract possa essere utilizzato e possa quindi interagire con il ledger del canale. La logica dello smart contract rimane privata per il peer che ne possiede l'installazione, nonostante ogni componente del canale possa avere accesso ad *S5*.
Una volta approvata la definizione di *S5* in questo caso, l'applicazione *A1* può richiamarlo e sfruttare le sue funzionalità. I client inviano proposte di transazioni ai peer innescando così un input per lo smart contract, che a sua volta genera una risposta che lo stesso peer restituisce all'applicazione.

A questo punto, aggiungendo un'altra organizzazione *R2* al canale *C1*, la quale aggiunge un altro peer *P2* con lo smart contract *S5* e il ledger *L1*, la rete assume un aspetto come quello illustrato nella figura 4.25:

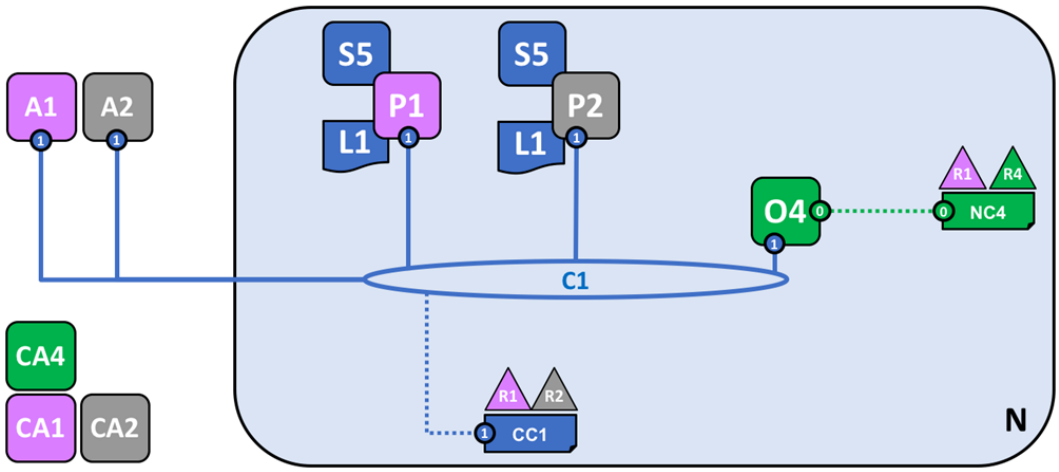

Figura 4.25: Rete Blockchain

Anche un'altra autorità di certificazione *CA2* si è aggiunta alla rete i cui certificati identificano *A2* e *P2*, perciò adesso sia *A1* che *A2* possono invocare *S5* sul canale utilizzando *P1* o *P2*.

4.3.3 Ampliamento della rete

Prima di proseguire con l'aggiunta sulla rete di altre *CA*, viene semplificata la notazione in modo da rendere più comprensibili le ulteriori aggiunte. In questo caso, partendo dalla figura 4.25 la rete assumerà l'aspetto rappresentato nella figura 4.26

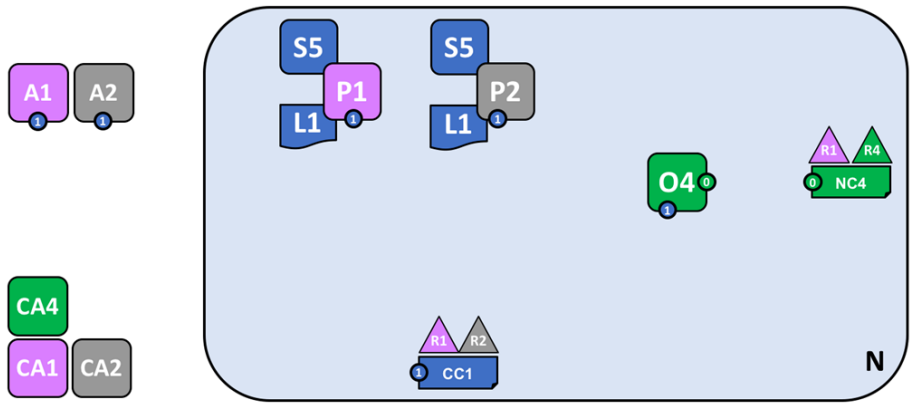

Figura 4.26: Rete Blockchain

Una volta rese più chiare le aggiunte, è necessario definire un nuovo **consorzio** *X2*, di cui fanno parte *R2* e una nuova organizzazione *R3*, come illustrato nella figura 4.27.

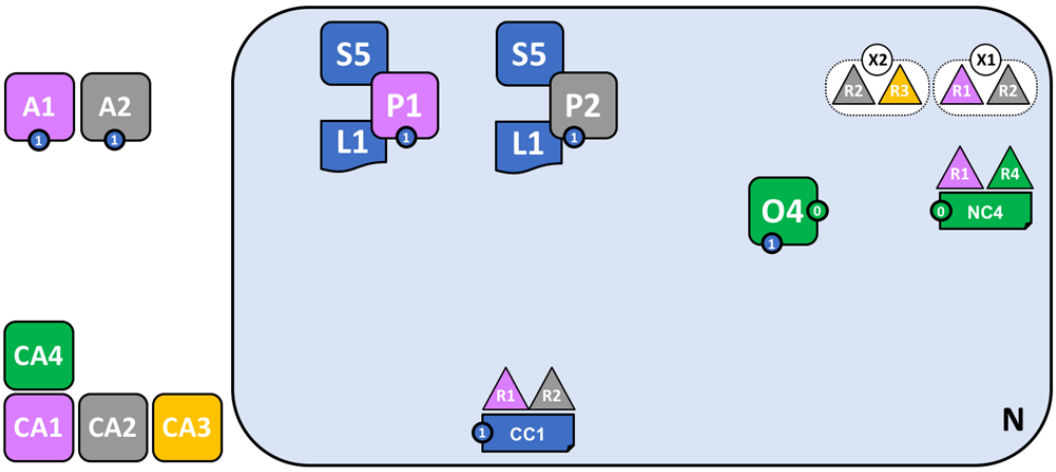

Figura 4.27: Rete Blockchain

L'aggiunta di *X2* è stata decisa da un amministratore di rete di *R1* o di *R4*, in quanto sono gli unici ad averne il potere. Il consorzio *X2* servirà ad aggiungere un altro canale *C2* alla rete, anch'esso può essere creato solo da chi ne ha diritto, ovvero le organizzazioni del consorzio *X1*. Per rendere visivamente comprensibile le interazioni nella rete, si è indicato con il colore blu tutto ciò che riguarda il canale *C1*, mentre con il colore rosso tutto ciò che riguarda il canale *C2*.

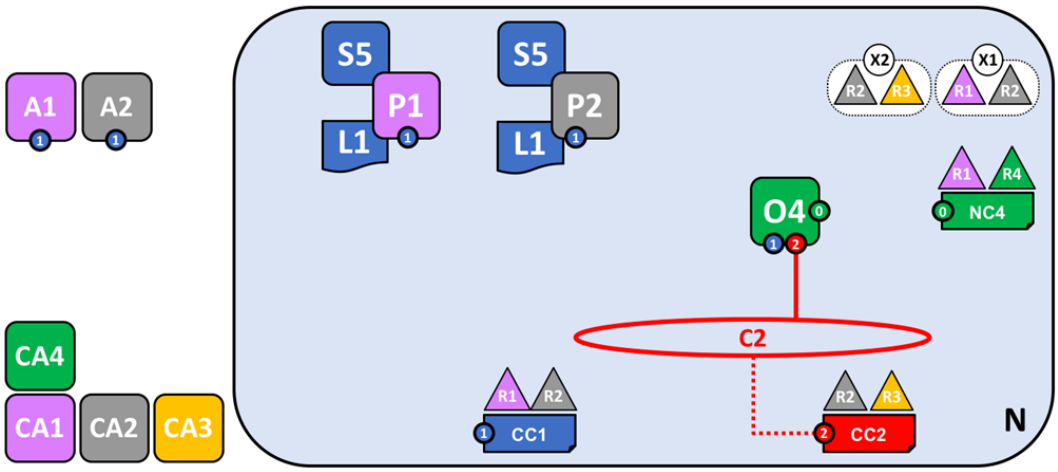

Figura 4.28: Rete Blockchain

La configurazione di *C2* è *CC2* ed anche in questo caso è separata dalla configurazione di rete. Le organizzazioni che gestiscono *C2* sono *R2* ed *R3*, mentre le altre due non hanno alcun diritto su questo canale, come si evince dalla figura 4.28.

Tramite *C2*, le organizzazioni del consorzio *X2* hanno un canale di comunicazione privato e sono le uniche a poter modificare la configurazione *CC2*, aggiungendo ad esempio nuove organizzazioni. Le due configurazioni *CC1* e *CC2* sono indipendenti l'una dall'altra, così come le politiche dei canali: questo mette ancora più in risalto la decentralizzazione che Hyperledger Fabric offre per le sue reti.

Le configurazioni assumono un ruolo di primaria rilevanza poiché in esse si trovano le direttive concordate tra i partecipanti della rete, fornendo un punto di riferimento comune per regolare l'accesso alle risorse di rete. Esse includono anche dettagli sulla composizione della rete e dei canali, come i nomi dei consorzi e delle relative entità organizzative. Ad esempio, durante la fase iniziale di creazione della rete attraverso il servizio d'ordinazione *O4*, le sue azioni sono governate dalle impostazioni di rete *NC4*. Inizialmente, la configurazione *NC4* contiene solo politiche che consentono all'organizzazione *R4* di esercitare il controllo sulle risorse di rete, ma, in seguito, viene aggiornato per estendere tali privilegi anche all'organizzazione *R1*. Con tale modifica, ogni amministratore di *R1* o di *R4* che si connette a *O4* acquisisce i diritti di gestione della rete.
P1 e *P2* interagiscono con le applicazioni client *A1* o *A2*, seguendo le politiche definite nella configurazione del canale *CC1* per regolare l'accesso alle risorse del canale *C1* e lo stesso schema viene applicato agli attori e alla configurazione del canale *CC2*.

É come se i nodi del servizio d'ordinazione gestiscano una sorta di *mini-blockchain*, collegata attraverso un canale di sistema, tramite il quale i nodi di *O4* distribuiscono le transazioni di configurazione della rete. In tal modo, si mantiene una copia coerente della configurazione di rete su ciascun nodo del servizio d'ordinazione. Analogamente, i nodi peer all'interno di un canale di applicazione distribuiscono le transazioni di configurazione del canale, garantendo una coerenza nella configurazione del canale su ciascun nodo peer.

A questo punto, si possono aggiungere un peer *P3*, la sua copia locale del ledger *L2* ed uno smart contract *S6* in modo che la rete si aggiorni e diventi ancora più grande, come rappresentato in figura 4.29.

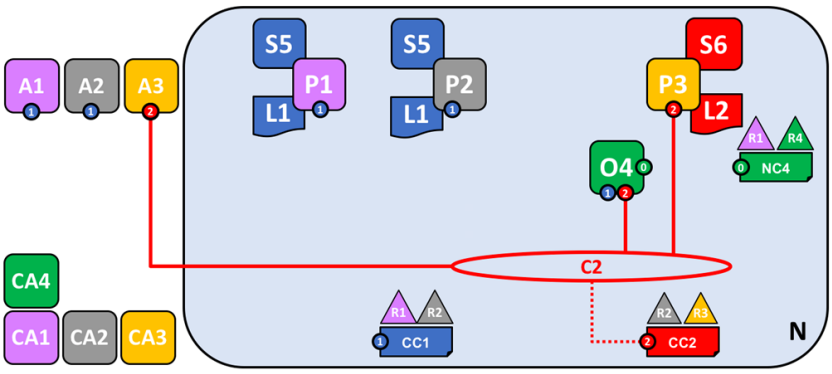

Figura 4.29: Rete Blockchain

Essendo connesso al canale *C2*, *P3* ha un ledger *L6* diverso rispetto ai peer che usano *C1*, completamente separato da *L1* in quanto il ledger di *C2* è un registro per le transazioni private tra le organizzazioni appartenenti al consorzio *X2*, che sono differenti da quelle che si trovano in *X1*. Per la stessa logica, anche lo smart contract installato su *P3* sarà differente poiché offre un accesso controllato ad *L2*.

4.3.4 Un peer e due canali

Fino a questo momento, un peer interagiva con un solo canale per volta. Questo non significa che uno stesso nodo non possa però avere interazioni con più di un canale.

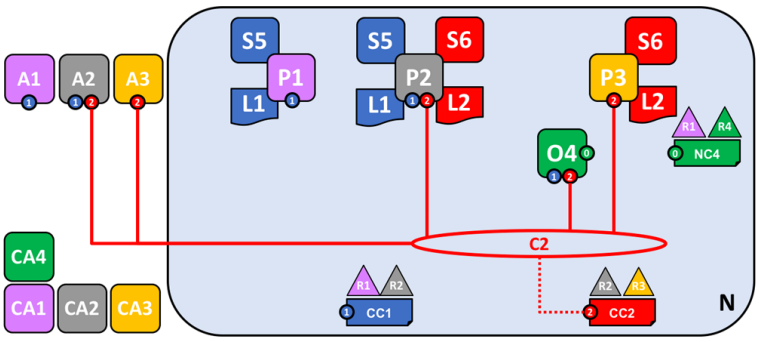

Figura 4.30: Rete Blockchain

Nella figura 4.30 il peer *P2* può comunicare sia mediante il canale *C1* che mediante il canale *C2*. Perciò *A1* può utilizzare solo *C1* per comunicare con i peer *P1* e *P2*, così come *A3* può utilizzare solo *C2* per le interazioni con *P2* e *P3*, mentre l'applicazione client *A2* può comunicare con tutti i peer presenti sulla rete, tramite *C1* per interfacciarsi con i primi due peer e tramite *C2* per interfacciarsi con gli ultimi due. Tutte e tre le applicazioni hanno la possibilità di comunicare con il servizio di ordinazione sfruttando la propria connessione con i canali.

R1 è l'unica organizzazione sulla rete che possiede entrambi gli smart contract installati sul proprio peer, e quindi è l'unica a poter effettuare transazioni con tutte le organizzazioni presenti sulla rete: con *P1* tramite *C1* e con *P3* tramite *C2*, sottostando sempre alle politiche di configurazione specifiche del canale.

A questo punto non resta altro che visualizzare la rete che si è formata, utilizzando notazioni grafiche più semplici, come in figura 4.31

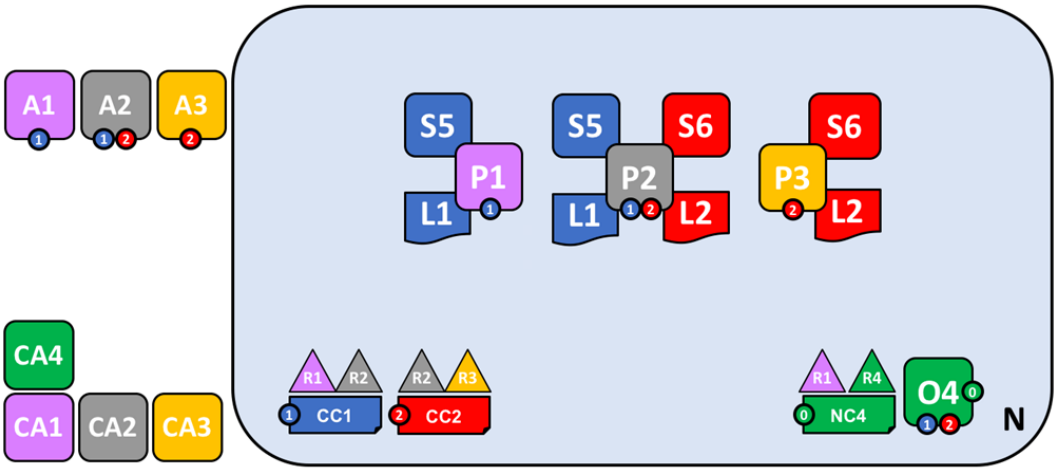

Figura 4.31: Rete Blockchain

Ricapitolando, ciò che si è ottenuto è una rete composta da 4 organizzazioni, un servizio di ordinazione, 2 canali e 3 nodi peer, sul quale sono istallati 2 smart contract. Il tutto è supportato da 4 autorità di certificazione e da 3 applicazioni client che interagiscono con gli smart contract mediante l'ausilio dei canali.

Capitolo 5

Rete blockchain per lo scambio di EHR

Il lavoro portato avanti è l'implementazione di una rete blockchain per scambio di Electronic Health Record. Il tutto è stato eseguito utilizzando come sistema operativo *Ubuntu* in quanto compatibile con Hyperledger Fabric. La rete infatti, è stata costruita proprio mediante l'utilizzo di Fabric, uno tra i software idonei per l'obiettivo prefissato.

Prerequisiti: Prima di poter implementare una blockchain con Hyperledger Fabric, è necessario avere installato sul PC l'ultima versione **git** e l'ultima di **cURL**, in quanto consigliato dalla guida[29] stessa. Inoltre, è necessario anche l'installazione di **Docker** e **Docker Compose** per garantire che un'applicazione funzioni in modo uniforme in diversi ambienti e non solo sul computer con cui si sta lavorando. Inoltre, è necessario scaricare i file binari di Hyperledger Fabric, cioè le immagini e gli esempi Docker.
Si è scelto, inoltre, di lavorare in *javascript*, perciò anche la sua installazione è un prerequisito fondamentale. In ultimo, è consigliabile anche installare **jq** sul proprio computer, in quanto questo servirà successivamente per aggiungere organizzazioni alla rete blockchain.

La rete blockchain implementata si basa su una configurazione costituita nel seguente modo:

- Due organizzazioni paritarie e un'organizzazione ordinatrice.
- Un servizio di ordinazione Raft[1] a nodo singolo.
- Non viene distribuita un'autorità di certificazione TLS ma tutti i certificati vengono emessi dalle CA radice.
- Distribuzione di una rete Fabric con Docker Compose: i nodi sono isolati all'interno di una rete Docker Compose, perciò la rete non è configurata per connettersi ad altri nodi Fabric in esecuzione.

[1]Raft è un algoritmo di consenso distribuito che si occupa di mantenere la coerenza e la sicurezza della rete assicurando che tutti i nodi di ordinamento concordino sugli aggiornamenti da apportare alla blockchain. Assicura che ogni nodo del cluster concordi sulla stessa serie di transizioni di stato anche di fronte a guasti[31].

5.1 Operazioni base su una rete

Per gestire una rete Fabric mediante l'uso di immagini Docker, la prima cosa da fare è aprire la directory in cui si trova il file `network.sh`. In questo file si trovano i comandi per gestire la rete, tra i quali:

- `up`: permette di portare su ordinatori e peer di una rete Fabric
- `createChannel`: permette di creare ed unire un canale alla rete
- `up createChannel`: permette di portare su la rete con un canale
- `deployCC`: permette di distribuire il chaincode al canale
- `down`: permette di portare giù la rete

Per essere sicuri di non aver contenitori docker già presenti sulla rete, si utilizza il comando `network.sh down` così da iniziare a lavorare su un ambiente pulito, per poi attivare la rete mediante l'utilizzo del comando `up`. In questo modo, si è dato vita ad una rete Fabric che comprende due nodi peer e un servizio di ordinazione. Le organizzazioni presenti sono due ospedali, *Hosp1* ed *Hosp2*, che gestiscono un peer ciascuno: `peer0.Hosp1.com` e `peer0.Hosp2.com`. Il servizio di ordinazione, invece. è gestito da un'unica organizzazione.
A questo punto, vi è la necessità di creare un canale per poter permettere le comunicazioni tra i due ospedali, perciò mediante l'utilizzo della funzione `createChannel`, *Hosp1* e *Hosp2* avranno un canale sul quale unire i loro peer. Successivamente si avvia il chaincode sul canale utilizzando `deployCC`: essendo la prima volta che si distribuisce, verranno installate anche le dipendenze dal chaincode; in particolare questo comando installerà il chaincode di trasferimento per le risorse. La politica di approvazione scelta, richiede che la maggioranza dei peer firmi la transazione, è necessario quindi, esportare le variabili di ambiente in modo da poter operare come *Hosp1*: così facendo si ha accesso al materiale crittografico di *Hosp1* e si può richiamare il chaincode per inserire un elenco di risorse iniziali nel ledger. Allo stesso modo è necessario interrogare il chaincode sul peer di *Hosp2* e quindi di operare come esso.

Vi è anche la possibilità di attivare la rete mediante le autorità di certificazione utilizzando il flag `-ca`; così facendo si possono registrare le identità client con gli *SDK Fabric* e si possono creare certificati e chiavi private per le applicazioni.
Le funzioni principali sono:

- `create`: `./network.sh` crea i certificati e le chiavi per due organizzazioni peer e il servizio di ordinazione. Si può osservare che di default, lo script utilizza lo strumento Cryptogen. Se si utilizza il flag `-ca` per creare autorità di certificazione, lo script utilizza i file di configurazione del server Fabric CA; in entrambi i modi si crea il materiale crittografico e le cartelle *MSP* per tutte le organizzazioni presenti.
- `createChannel`: `./network.sh` esegue lo script `createChannel.sh` per creare un canale. Dopo aver creato il canale, lo script utilizza la CLI del peer `peer0.Hosp1.com` per far unire al canale `peer0.Hosp2.com` e garantire che entrambi i peer siano ancorati al canale.
- `deployCC`: `./network.sh` esegue lo script `deployCC.sh` per installare il chaincode di trasferimento delle risorse (di base) su entrambi i peer e quindi definire il chaincode sul canale. Una volta che la definizione del chaincode è stata confermata sul canale, la CLI del peer inizializza il chaincode e lo richiama per inserire i dati iniziali nel ledger.

5.1.1 Chaincode

Prima di poter installare il chaincode sui peer, è necessario installare le sue dipendenze. Si è scelto di operare in javascript, perciò si utilizza la versione javascript del chaincode delle risorse dopo aver installato tutte le dipendenze. Al fine di installarle tutte, si esegue il comando `nmp install` cosicché, se l'esito del comando è positivo, tutti i pacchetti di javascript verranno installati all'interno di una stessa cartella. Solo a questo punto è possibile creare il pacchetto chaincode e lo si fa utilizzando una `peerCLi`. Bisogna poi aggiungere i file binari al percorso CLI, specificando il linguaggio di programmazione usato e ancora la posizione del codice dello smart contract.
Successivamente, è necessario installare il chaincode sui peer della rete. Esso va installato su tutti i peer che approveranno una transazione e, poiché la politica di approvazione scelta richiede che entrambi gli ospedali siano d'accordo, è necessario installare il chaincode su entrambi i peer `peer0.Hosp1.com` e `peer0.Hosp2.com`.

Partendo con il primo peer, è necessario agire come utente amministratore di *Hosp1* impostando le variabili d'ambiente opportune, si installa il chaincode sul peer e ciò restituirà l'identificatore del pacchetto, che servirà in seguito per l'approvazione. Allo stesso modo si deve agire come utente amministratore di *Hosp2*, sempre impostando le variabili di ambiente e installando il chaincode.
La definizione del chaincode deve essere approvata prima di poter essere distribuita e, poiché la maggioranza di 2 è 2, è necessario che sia *Hosp1* che *Hosp2* siano favorevoli.
Si approva quindi la definizione, lavorando come *Hosp2* e si reimpostano le variabili di ambiente per poter nuovamente lavorare come amministratori di *Hosp1* e poter approvare la definizione in maniera analoga.
Una volta che un'organizzazione ha installato il chaincode sul proprio peer, l'ID del pacchetto deve essere incluso nella definizione del chaincode approvata, cosicché il chaincode risulti approvato dall'organizzazione in questione. É presente un parametro `- sequence` che altro non è che un numero intero che tiene traccia delle volte in cui il chaincode è stato utilizzato o aggiornato: essendo la prima volta che viene utilizzato, tale numero sarà pari ad 1. Avendo ora la maggioranza di cui si necessitava, si può implementare il trasferimento di risorse del chaincode sul canale.

Si può verificare se i membri del canale hanno approvato la stessa definizione del chaincode, in questo modo di otterrà una mappa `json` di questo tipo:

```
{
        "Approvals": {
                "Hosp1MSP": true,
                "Hosp2MSP": true
        }
}
```

Si ha quindi la certezza che entrambe le organizzazioni hanno approvato gli stessi parametri: **la definizione del chaincode è pronta per essere impegnata nel canale**. Eseguendo il comando `commit`, la transazione viene inviata ai peer del canale per interrogare il chaincode. Intanto, le approvazioni della definizione, vengono inviate al servizio di ordinazione il quale aggiunge un blocco e le distribuisce al canale: i peer possono quindi verificare che la politica di approvazione è stata soddisfatta e possono invocare il chaincode.

Il chaincode è finalmente pronto per essere richiamato dalle applicazioni client e, attraverso l'uso del comando `invoke`, si può creare un set iniziale di pazienti nel libro mastro.

Utilizzando una funzione di query è possibile leggere l'insieme appena creato:


```
[{"Key":"asset1","Record":{"ID":"asset1","nome paziente":"Mario Rossi",
"data di nascita":"22/08/1987","luogo di nascita":"Potenza (IT)","sesso":"maschio",
"colore occhi":"blu","peso in kg":92}},

{"Key":"asset2","Record":{"ID":"asset2","nome paziente":"Daniele Bianchi",
"data di nascita":"30/06/1998","luogo di nascita":"Roma (IT)","sesso":"maschio",
"colore occhi":"verde","peso in kg":87}},

{"Key":"asset3","Record":{"ID":"asset3","nome paziente":"Antonella Pastrame",
"data di nascita":"05/12/2000","luogo di nascita":"Trieste (IT)","sesso":"femmina",
"colore occhi":"marrone","peso in kg":59}},

{"Key":"asset4","Record":{"ID":"asset4","nome paziente":"Luisa Ignolo",
"data di nascita":"17/02/1965","luogo di nascita":"Torino (IT)","sesso":"femmina",
"colore occhi":"verde","peso in kg":64}},

{"Key":"asset5","Record":{"ID":"asset5","nome paziente":"Maria Pastore",
"data di nascita":"09/07/1992","luogo di nascita":"Palermo (IT)","sesso":"femmina",
"colore occhi":"nero","peso in kg":93}},

{"Key":"asset6","Record":{"ID":"asset6","nome paziente":"Pasquale Treni",
"data di nascita":"25/05/1070","luogo di nascita":"Bergamo (IT)","sesso":"maschio",
"colore occhi":"marrone","peso in kg":98}}]
```


É anche possibile aggiornare uno smart contract utilizzando un processo come quello illustrato finora, anche se il chaincode è già stato distribuito, in quanto i membri del canale possono aggiornarlo installando un nuovo pacchetto e approvando una sua nuova definizione con il nuovo ID, con una nuova versione e con il numero di sequenza incrementato di uno. Anche in questo caso, il nuovo chaincode può essere utilizzato solo dopo che la definizione viene inserita nel canale in modo che i membri possano aggiornarsi al cambiamento e garantire che un numero sufficiente di membri sia disponibile all'utilizzo. Ad esempio, si può voler cambiare il linguaggio di programmazione in cui è stato scritto il chaincode.

5.1.2 Esecuzione di un'applicazione Fabric

Per eseguire un'applicazione su Hyperledger Fabric, ci sono alcune informazioni fondamentali da comprendere sul trasferimento delle risorse. Utilizzando l'**applicazione**, che effettua le chiamate alla rete blockchain invocando le transazioni implementate nello smart contract, e lo **smart contract**, che implementa le transazioni, si possono creare, aggiornare ed eseguire query sulle risorse. Inizialmente, bisogna **configurare la rete blockchain** con l'applicazione con cui interagirà; in seguito bisognerà **eseguire l'applicazione** come in figura 5.1.

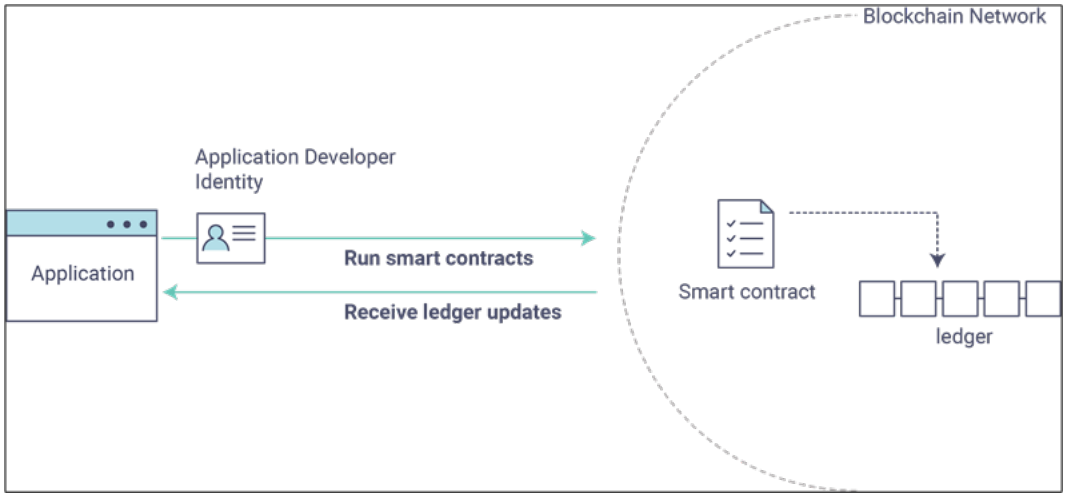

Figura 5.1: Esecuzione di un'applicazione Fabric

Per configurare la rete blockchain, costituita da un servizio di ordinazione, due peer e 3 autorità di certificazione, ovvero *Orderer*, *Hosp1* ed *Hosp2*, bisogna attivare le autorità di certificazione. Successivamente è necessario distribuire lo smart contract assicurandosi che il pacchetto chaincode sia eseguito correttamente.
Per poter creare l'applicazione, è necessario installare tutte le dipendenze dell'applicazione sempre mediante il comando `npm install`, in quanto si lavora in javascript. Il più importante tra questi è il pacchetto *Node.js* in quanto fornisce l'*API client Fabric Gateway*: quest'ultima consente di connettere un *Fabric Gateway*[2] e inviare e valutare transazioni.

Innanzitutto, l'applicazione deve stabilire una connessione gRPC[3] al servizio Fabric Gateway che utilizzerà per accedere ai canali accessibili al Fabric Gateway e distribuiti in modo intelligente su tali reti.
La connessione Gateway richiede:

- Connessione gRPC al Fabric Gateway.
- Identità del client utilizzata per le transazioni.
- Implementazione della firma utilizzata per generare firme digitali per l'identità del client.

Dopo aver creato tale connessione, l'applicazione la utilizzerà per ottenere un riferimento all'interno di un chaincode distribuito sulla rete: immediatamente dopo la distribuzione iniziale del pacchetto chaincode, il ledger è vuoto. É l'applicazione a richiamare una funzione di transazione per popolare il ledger con alcune risorse di esempio. Infine, l'applicazione sarà pronta per eseguire query, creare risorse aggiuntive e modificare le risorse nel ledger invocando le funzioni di transazione sullo smart contract.

[2]Il Fabric Gateway è una componente di Hyperledger Fabric che fornisce un'interfaccia per l'accesso a una rete blockchain Fabric; punto di ingresso per le applicazioni esterne che desiderano interagire con la rete Fabric.[29]
[3]Il gRPC è l'abbreviazione di **gRPC Remote Procedure Call**, è un moderno framework RPC (Remote Procedure Call) open source ad alte prestazioni che può essere eseguito in qualsiasi ambiente. Può connettere in modo efficiente i servizi all'interno e tra data center con supporto collegabile per bilanciamento del carico, tracciabilità, controllo dello stato e autenticazione.[32]

In figura 5.2 è rappresentato in maniera semplificata il funzionamento di una query.

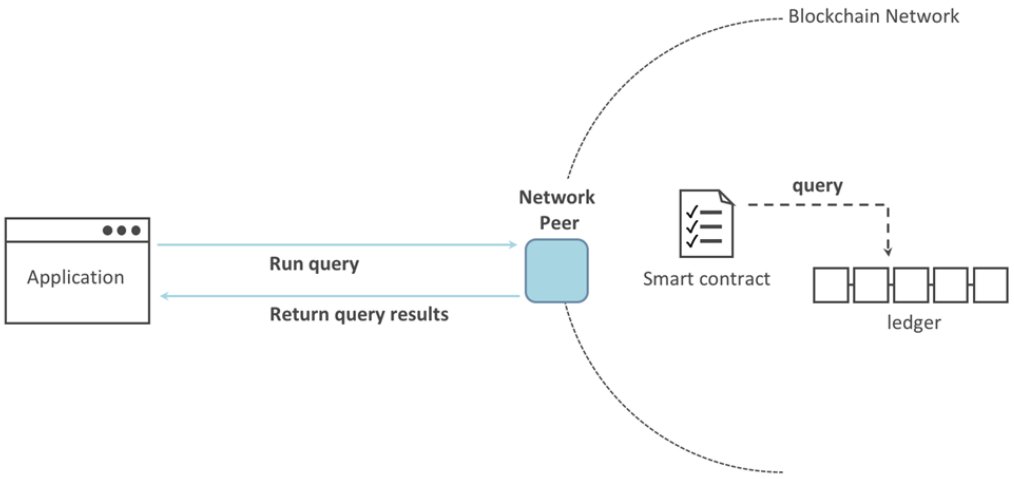

Figura 5.2: Funzionamento di una query

L'applicazione di esempio sta semplicemente ottenendo tutte le risorse create nel passaggio precedente quando si è popolato il ledger. Tali risorse, si possono visualizzare, osservando che i risultati delle funzioni di transazione vengono sempre restituiti come byte.

L'output che si trova risulta essere:

```
{
   ID: 'asset1',
   nome paziente: 'Mario Rossi',
   data di nascita: '22/08/1987',
   luogo di nascita: 'Potenza (IT)',
   sesso: 'maschio',
   colore occhi: 'blu',
   peso in kg: 92,
   docType: 'asset'
 },
 {
   ID: 'asset2',
   nome paziente: 'Daniele Bianchi',
   data di nascita: '30/06/1998',
   luogo di nascita: 'Roma (IT)',
   sesso: 'maschio',
   colore occhi: 'verde',
   peso in kg: 87
 },
```

```
{
  ID: 'asset3',
  nome paziente: 'Antonella Pastrame',
  data di nascita: '05/12/2000',
  luogo di nascita: 'Trieste (IT)',
  sesso: 'femmina',
  colore occhi: 'marrone',
  peso in kg: 59
},
{
  ID: 'asset4',
  nome paziente: 'Luisa Ignolo',
  data di nascita: '17/02/1965',
  luogo di nascita: 'Torino (IT)',
  sesso: 'femmina',
  colore occhi: 'verde',
  peso in kg: 64
},
{
  ID: 'asset5',
  nome paziente: 'Maria Pastore',
  data di nascita: '09/07/1992',
  luogo di nascita: 'Palermo (IT)',
  sesso: 'femmina',
  colore occhi: 'nero',
  peso in kg: 93
},
{
  ID: 'asset6',
  nome paziente: 'Pasquale Treni',
  data di nascita: '25/05/1070',
  luogo di nascita: 'Bergamo (IT)',
  sesso: 'maschio',
  colore occhi: 'marrone',
  peso in kg: 98
},
```

Successivamente, l'applicazione invia una transazione per creare una nuova risorsa:

```
  const assetId = `asset${Date.now()}`;
  await contract.submitTransaction(
      `CreateAsset`,
      `asset7`,
      `Gianluca Mipio`,
      `05/09/2002`,
      `Isernia (IT)`,
      `maschio`,
      `nero`,
      `78`    );
```

Gli argomenti previsti dal chaincode, devono essere ovviamente tutti inseriti nell'ordine corretto, ovvero: *ID, nome paziente, data di nascita, luogo di nascita, sesso, colore occhi, colore capelli, peso in kg.*

Aggiornare il ledger, come si può vedere in figura 5.3 non è complesso in quanto un'applicazione riceve una notifica di convalida e conferma dopo aver inviato la transazione alla rete.

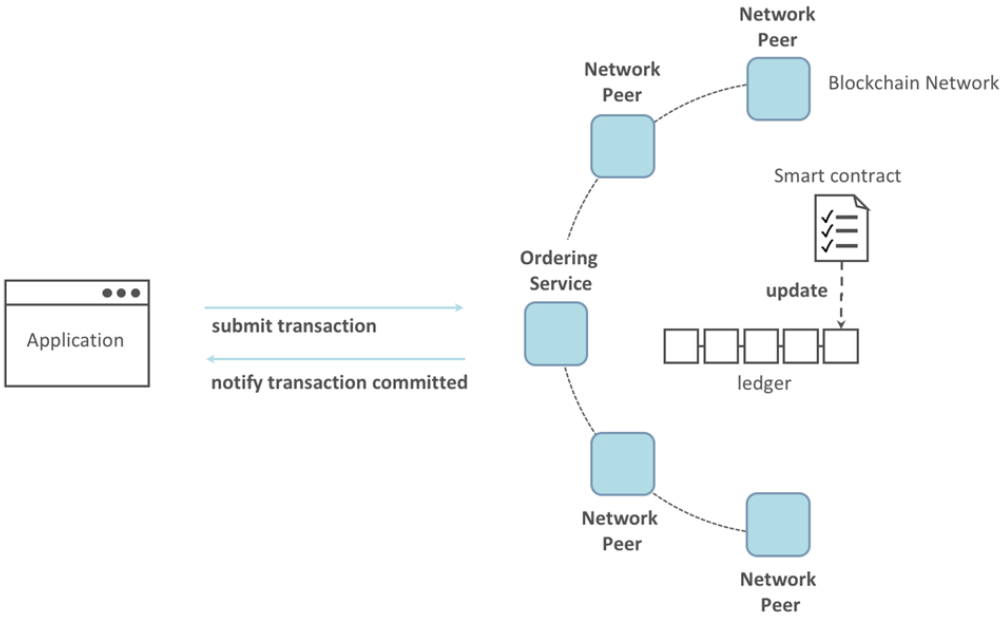

Figura 5.3: Aggiornamento ledger

Anche una risorsa può essere aggiornata; ad esempio, si può inviare una transazione per modificare il peso di un paziente.

Interrogando la query a seguito delle modifiche richieste, si ottiene un output del tipo:

```
Result:
{
ID: 'asset7',
nome paziente: 'Gianluca Mipio',
data di nascita: '05/09/2002',
luogo di nascita: 'Isernia (IT)',
sesso: 'maschio',
colore occhi: 'nero',
peso in kg: 69
}
```

5.2 Scambio di EHR su una rete blockchain

5.2.1 Ciclo di vita di un EHR

Una EHR è una Electronic Health Record che contiene informazioni sulla salute di un paziente, come la sua storia clinica. Si presenta con una parte iniziale contenente i dati per identificare la cartella stessa. Per esempio, Mario Rossi , un paziente con problemi di salute, si reca presso *Hosp1* per dei problemi di salute. Non ha una sua EHR, perciò la prima cosa da fare è crearla, quindi un operatore sanitario dipendente di *Hosp1* **crea** la Electronic Health Record 001 che risulterà così composta:

```
CREATORE = Hosp1
ID_EHR = 001
PROPRIETARIO = Hosp1
PAZIENTE = Mario Rossi
DATA REGISTRAZIONE = 15/01/2021
STATO CORRENTE = Creata
```

Si suppone che ad un certo punto della sua vita Mario Rossi debba trasferirsi in un'altra città e, per tale motivo, ha necessità di cambiare ospedale. Nella nuova città, sarà proprio *Hosp2* l'ospedale di riferimento, appartenendo alla rete organizzativa posta sullo stesso canale di *Hosp1*, può **acquisire** l'EHR da *Hosp1*, in tal caso la cartella si presenterà come segue:

```
CREATORE = Hosp1
ID_EHR = 001
PROPRIETARIO = Hosp2
PAZIENTE = Mario Rossi
DATA REGISTRAZIONE = 15/01/2021
STATO CORRENTE = Acquisita
```

Si può notare come, oltre allo stato della EHR, sia cambiato anche il proprietario: Mario Rossi, infatti, da adesso in poi sarà paziente di *Hosp2*.
Nel momento in cui Mario Rossi perderà la vita, *Hosp2* avrà il compito di **archiviare** la EHR del paziente, non prima di averla trasmessa ad *Hosp1*, che ne era il proprietario originario.

La cartella in questo caso si presenterà nel seguente modo:

```
CREATORE = Hosp1
ID_EHR = 001
PROPRIETARIO = Hosp1
PAZIENTE = Mario Rossi
DATA REGISTRAZIONE = 15/01/2021
STATO CORRENTE = Archiviata
```

Questo conclude il ciclo di vita di una EHR, che può essere schematizzato come indicato in figura 5.4. Può essere utile tenere un registro delle cartelle cliniche rilasciate e lo stato `Archiviata` consente un'identificazione immediata. L'insieme degli stati della EHR costituisce lo stato mondiale del ledger.

Un modo per identificare uno stato univocamente in un determinato contesto è `Hosp1001`, ovvero una combinazione nota come **chiave** di chi ha creato l'EHR e dell'ID ad essa associato. Una chiave di stato consente di identificare un documento in modo univoco: viene creata durante l'operazione di emissione e poi aggiornata attraverso l'acquisto e il riscatto. Hyperledger Fabric richiede che ogni stato all'interno di un ledger abbia una chiave univoca.

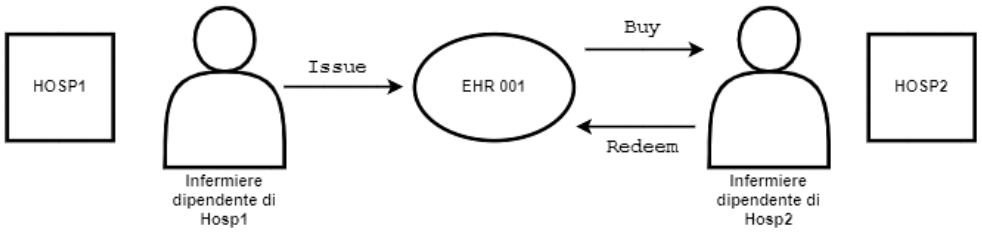

Figura 5.4: Ciclo di vita di un EHR

5.2.2 Distribuzione di uno smart contract sulla rete Fabric

Si sta lavorando sempre con la stessa rete Fabric composta da due organizzazioni peer, *Hosp1* e *Hosp2*, e un servizio di ordinazione. Ogni organizzazione gestisce la propria autorità di certificazione (*CA*) e i due peer, i database di stato , il nodo del servizio di ordinazione e ciascuna *CA* dell'organizzazione viene eseguita nel proprio contenitore *Docker*. Ecco come appare la rete 5.5:

Figura 5.5: Struttura Rete

Una volta avviata la rete e creato il canale sul quale interagiranno i due ospedali, mediante il comando `docker ps` è possibile vedere i nodi Fabric in esecuzione.
Sono riportati solo i due contenitori Docker dei peer a scopo illustrativo:

```
CONTAINER ID        a86f50ca1907                        77d0fcaee61b
IMAGE               hyperledger/fabric-peer:latest      hyperledger/fabric-peer:latest
COMMAND             "peer node start"                   "peer node start"
CREATED             About a minute ago                  About a minute ago
STATUS              Up About a minute                   Up About a minute
PORTS               7051/tcp, 0.0.0.0:9051->9051/tcp    0.0.0.0:7051->7051/tcp
NAMES               peer0.Hosp2.com                     peer0.Hosp1.com
```

Ciò significa che il peer *Hosp1*, `peer0.Hosp1.com` è in esecuzione nel contenitore `a86f50ca1907`, mentre il peer *Hosp2*, `peer0.Hosp2.com` è in esecuzione nel contenitore `77d0fcaee61b`. Questi contenitori, insieme a quelli relativi alle *CA* e al servizio di ordinazione, formano tutti una rete Docker chiamata *fabric_hosp*, che appare come segue:


```
[
    {
        "Name": "fabric_hosp",
        "Id": "f4c9712139311004b8f7acc14e9f9017",
        "Created": "2020-04-28T22:45:38.525016Z",
        "Containers": {
            "03373d116c5abf2ca94f6f00df98bb74f8903": {
                "Name": "orderer.com",
                "EndpointID": "0eed871a2aaf9a5dbcf7896aa3",
                "MacAddress": "02:42:c0:a8:70:05",
                "IPv4Address": "192.168.112.5/20",
                "IPv6Address": ""
            },
            "2438df719f57a597de592cfc76db30013adf": {
                "Name": "couchdb1",
                "EndpointID": "52527fb450a7c80ea509cb571d1",
                "MacAddress": "02:42:c0:a8:70:06",
                "IPv4Address": "192.168.112.6/20",
                "IPv6Address": ""
            },
            "6b4d87f65909afd335d7acfe6d79308d": {
                "Name": "ca_Hosp2",
                "EndpointID": "1cc322a995880d76e1dd1f3",
                "MacAddress": "02:42:c0:a8:70:04",
                "IPv4Address": "192.168.112.4/20",
                "IPv6Address": ""
            },
            "77d0fcaee61b8fff43d33331073ab9ce36": {
                "Name": "peer0.Hosp1.com",
                "EndpointID": "05d0d34569eee412e28313ba7ee06875",
                "MacAddress": "02:42:c0:a8:70:08",
                "IPv4Address": "192.168.112.8/20",
                "IPv6Address": ""
            },
            "7b01f5454832984fcd9650f05b4affce9731": {
                "Name": "ca_orderer",
                "EndpointID": "057390288a424f49d6e9d6f788049b",
                "MacAddress": "02:42:c0:a8:70:02",
                "IPv4Address": "192.168.112.2/20",
                "IPv6Address": ""
            },
            "7eb5f64bfe5f20701aae8a6660815c4e3a8": {
                "Name": "couchdb0",
                "EndpointID": "bfe740be15ec9dab7baf3806964e6b1",
                "MacAddress": "02:42:c0:a8:70:07",
```

```
                "IPv4Address": "192.168.112.7/20",
                "IPv6Address": ""
            },
            "87aef6062f2324889074cda80fec8fe014d844": {
                "Name": "ca_Hosp1",
                "EndpointID": "a740090d33ca94dd7c6aaf14a79e1cb3",
                "MacAddress": "02:42:c0:a8:70:03",
                "IPv4Address": "192.168.112.3/20",
                "IPv6Address": ""
            },
            "a86f50ca19079f59552e8674932edd02f7f9": {
                "Name": "peer0.Hosp2.com",
                "EndpointID": "6e56772b4783b1879a06f86901786fe",
                "MacAddress": "02:42:c0:a8:70:09",
                "IPv4Address": "192.168.112.9/20",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
```

Si lavora con due cartelle separate, una per *Hosp1* e un'altra per *Hosp2*, nelle quali sono contenuti gli smart contract e i file delle rispettive applicazioni, in modo da avere la possibilità di operare come entrambe le organizzazioni.

Nello smart contract sono presenti le tre funzioni `issue`, `buy` e `redeem` che vengono utilizzate dalle applicazioni per inviare transazioni che, rispettivamente, *creano*, *acquisiscono* e *archiviano* EHR nel ledger. Inizialmente, si agisce come amministratore di Hosp1; di seguito sono riportati i punti chiave dello script:

- `const { Contract, Context } = require('fabric-EHR-api');` : due classi chiave di Hyperledger Fabric che vengono usate nello smart contract, ovvero `Contracte` e `Context`;
- `class EHRContract extends Contract {` : all'interno di questa classe sono definiti i metodi che implementano le operazioni chiave delle funzioni;
- `async issue(CREATORE, ID_EHR, PROPRIETARIO, PAZIENTE...);` : definisce la creazione di una EHR;
- `let EHR = EHR.createInstance(CREATORE, ID_EHR, PROPRIETARIO...);`: crea una nuova EHR in memoria con gli input di transazione forniti;
- `await ctx.EHRList.addEHR(EHR);` : aggiunge la nuova EHR al ledger;
- `return EHR;` : restituisce un buffer binario come risposta dalla transazione per l'elaborazione da parte del chiamante dello smart contract.

Prima che l'`EHRContract` possa essere utilizzato dalle applicazioni, deve essere installato sui nodi peer appropriati della rete e definito sul canale utilizzando il ciclo di vita del chaincode di Hyperledger Fabric: infatti più organizzazioni devono concordare i parametri di un chaincode prima

della sua distribuzione su un canale. Pertanto, è necessario installare e approvare il chaincode come amministratori sia di Hosp1 che di Hosp2.

Per installare e approvare lo smart contract come le due organizzazioni, non bisogna fare nient'altro che ripetere i passaggi che sono già stati illustrati in precedenza, ovvero impostare le variabili di ambiente corrette in modo che il pacchetto chaincode venga creato, installato, e quindi approvato e confermato per l'utilizzo sul canale.

Solo una volta che entrambe le organizzazioni avranno approvato il chaincode, la sua definizione verrà confermata sul canale, permettendo l'avvio del container del chaincode su entrambi i peer. Una volta distribuito sul canale, il chaincode può essere utilizzato dalle applicazioni client per interagire con la rete blockchain.

In figura 5.6 è riportato uno schema riassuntivo di come agisce la funzione `issue` nell'applicazione.

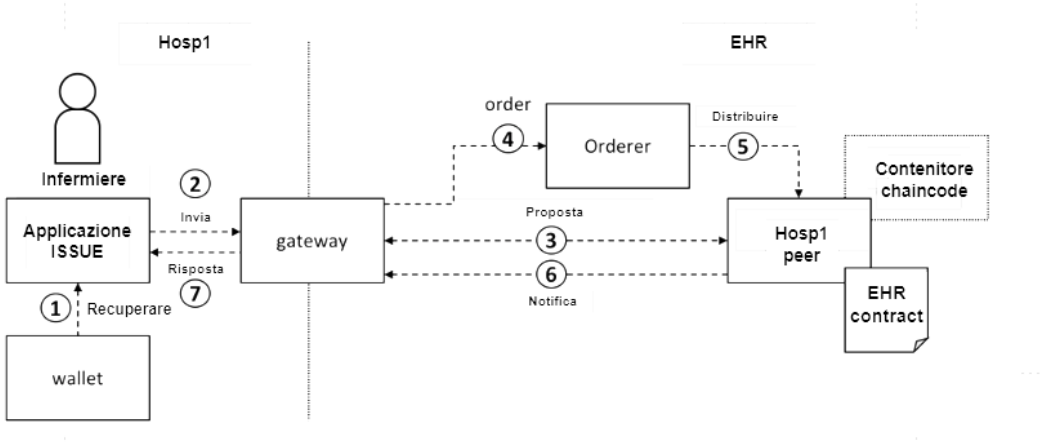

Figura 5.6: Gateway

Nel caso specifico dell'applicazione `issue` il gateway è utilizzato per inviare le transazioni sulla rete. Esso svolge un ruolo fondamentale nell'interazione tra un'applicazione e la rete blockchain, in quanto permette all'applicazione di gestire l'invio e la ricezione delle transazioni, coordinandone il processo di elaborazione delle proposte e degli ordini tra i diversi componenti della rete; in pratica, il gateway semplifica la comunicazione dell'applicazione con la blockchain.

Prima di poter inviare le transazioni, l'applicazione recupera il certificato X.509 dell'operatore di *Hosp1* dal suo wallet. Questo wallet può essere archiviato localmente nel file system o in un modulo di sicurezza hardware HSM (Hardware Security Module).

Il **wallet**, nel contesto di Hyperledger Fabric è un componente critico che memorizza in modo sicuro i certificati e le chiavi crittografiche necessari per interagire con la rete blockchain. Può essere considerato come un contenitore sicuro in cui vengono archiviati i materiali crittografici degli utenti e delle identità che partecipano alla rete.

All'interno del wallet, vengono generalmente memorizzati elementi come i certificati, le chiavi private ed altri materiali crittografici. Esso offre un livello di sicurezza maggiore alle applicazioni blockchain.

L'uso del gateway insieme al wallet semplifica notevolmente lo sviluppo delle applicazioni Hyperledger Fabric, in quanto fornisce un'interfaccia semplificata per l'interazione con la rete blockchain, consentendo alle applicazioni di concentrarsi sulla loro logica di business principale.

L'operatore di Hosp1 utilizzerà il file `issue.js` per caricare la sua identità nel suo wallet e utilizzerà questa identità per creare l'EHR 001 per conto di Hosp1, invocando `EHRcontract`. Tale file, contiene la logica dell'applicazione di creazione di un EHR e tutte le sue dipendenze. Vi sono alcune righe importanti al suo interno:

- `const { Wallets, Gateway } = require('fabric-network');` importa le classi chiave dell'SDK Hyperledger Fabric, cioè Wallets e Gateway, necessarie per l'interazione con la rete.
- `const wallet = await Wallets.newFileSystemWallet('../identity/user/nurse1/wallet');` indica che l'applicazione utilizzerà il wallet dell'operatore di *Hosp1* quando si connette alla rete blockchain.
- `await gateway.connect(connectionProfile, connectionOptions);` stabilisce la connessione alla rete utilizzando il gateway e le opzioni di connessione specificate.
- `const network = await gateway.getNetwork('channel');` collega l'applicazione al canale di rete specificato, in questo caso `channel`.
- `const contract = await network.getContract('EHRcontract');` ottiene l'accesso allo smart contract `EHRContract` sulla rete. Una volta ottenuto il contratto, l'applicazione può eseguire qualsiasi transazione implementata all'interno del chaincode.
- `const issueResponse = await contract.submitTransaction('issue', 'Hosp1', '001', ...);` invia una transazione alla rete utilizzando la transazione di emissione definita nel contratto intelligente. *Hosp1* e 001 sono i parametri necessari per creare una nuova EHR.
- `let EHR = EHR.fromBuffer(issueResponse);` elabora la risposta dalla transazione.

A questo punto si può procedere all'installazione delle dipendenze necessarie, al termine della quale si generano le credenziali X.509 appropriate per il wallet dell'operatore. Per fare ciò, vengono generate una coppia di chiavi pubblica e privata (utilizzata per firmare le transazioni) e viene richiesto un certificato (contenente la chiave pubblica e altri attributi *X.509*) alla *CA*. Una volta ottenuto il certificato, viene memorizzato nel wallet dell'utente insieme alla chiave privata e, infine, si esegue `node issue.js` per inviare la transazione di emissione dell'EHR 001.

In ogni momento, il Fabric SDK sottostante gestisce il processo di approvazione, ordinazione e notifica della transazione, semplificando la logica dell'applicazione. L'SDK utilizza un gateway per astrarre i dettagli della rete e le opzioni di connessione, permettendo di dichiarare strategie di elaborazione più avanzate come i nuovi tentativi di transazione.

A questo punto, l'operatore di *Hosp1* ha creato un'EHR per un paziente.
Se tale paziente un giorno volesse cambiare e andare da *Hosp2*, un altro operatore dipendente di *Hosp2* dovrebbe acquisire il paziente. Utilizzerà in tal caso la funzione `buy` per inviare una transazione al ledger e trasferire la proprietà dell'EHR 001 da *Hosp1* ad *Hosp2*.
Il `EHRContract` è lo stesso utilizzato da *Hosp1*, ma cambia la transizione in quanto essa è appunto `buy` invece che `issue`.

L'applicazione di *Hosp2* è molto simile a quella di *Hosp1*:

- `const { Wallets, Gateway } = require('fabric-network');`
- `const wallet = await Wallets.newFileSystemWallet('../identity/user/nurse2/wallet');` indica che l'applicazione utilizzerà il wallet dell'operatore di Hosp2 quando si connette alla rete blockchain.

- `await gateway.connect(connectionProfile, connectionOptions);`
- `const network = await gateway.getNetwork('channel');`
- `const contract = await network.getContract('EHRcontract');`
- `const issueResponse = await contract.submitTransaction('buy', 'Hosp2', '001', ...);` invia una transazione alla rete utilizzando la transazione di emissione definita nel contratto intelligente. *Hosp2* e 001 sono i parametri necessari per acquisire l'EHR da *Hosp1*.
- `let EHR = EHR.fromBuffer(issueResponse);`

Anche in questo caso è necessario installare le dipendenze e configurare il wallet dell'operatore di *Hosp2* per utilizzare le applicazioni di *Hosp2* e quindi acquisire ed eventualmente archiviare un'EHR.

Come nel caso precedente, l'operatore di *Hosp2* può memorizzare più identità nel suo wallet, ma in questo caso ne memorizza una sola.

L'operatore di *Hosp2* ha utilizzato `buy.js` per inviare una transazione che ha trasferito la proprietà dell'EHR 001 al proprio ospedale ed esegue `node buy.js` per avere conferma che l'EHR 001 è stata acquisita con successo. Ha invocato la transazione `buy` definita nello smart contract `EHRContract`, che ha aggiornato l'EHR 001 nello stato mondiale.

In ultimo, la transazione finale nel ciclo di vita dell'EHR 001 prevede che *Hosp2* l'archivi trasferendola nuovamente ad *Hosp1*. Perciò, l'operatore del secondo ospedale utilizza `redeem.js` per inviare una transazione per eseguire la logica di archiviazione all'interno dello smart contract. L'output di `node redeem.js` lo informa che l'archiviazione è avvenuta con successo e, ancora una volta, è stata aggiornata l'EHR 001 nello stato mondiale per riflettere che la proprietà è tornata ad *Hosp1*, il creatore dell'EHR.

5.3 Aggiunta di un nuovo canale alla rete

La creazione di un canale in Hyperledger Fabric è un passaggio fondamentale per consentire la comunicazione e il trasferimento di risorse tra organizzazioni specifiche su una rete. I canali agiscono come un livello privato di comunicazione, invisibile agli altri membri della rete, e sono costituiti da un ledger separato accessibile solo ai membri autorizzati del canale.

Per creare un nuovo canale e trasferire le risorse, è necessario seguire una serie di passaggi operativi utilizzando gli strumenti forniti da Hyperledger Fabric. Infatti, è necessario **configurare lo strumento** `configtxgen`, in quanto viene utilizzato per generare una transazione di creazione del canale. Tale strumento, legge un file contenente le informazioni necessarie per creare la configurazione del canale, le quali includono le organizzazioni che possono diventare membri del canale, il servizio di ordinazione che formerà il servizio di ordinazione della rete, le politiche del canale e i profili del canale. Si dovrà poi proseguire unendo i peer al canale applicativo creato e impostando i peer di ancoraggio.

I canali vengono generati attraverso una transazione di creazione del canale inviata al servizio di ordinazione, che specifica la configurazione iniziale e viene utilizzata per scrivere il blocco di genesi dello stesso.

Si impostano le variabili di ambiente, passaggio essenziale per definire la configurazione dei nuovi canali in Hyperledger Fabric. Infatti, contiene tutte le informazioni necessarie per creare la configurazione del canale in un formato facilmente leggibile e modificabile. Lo strumento `configtxgen` utilizza i profili di canale definiti in questo file per generare la configurazione del canale e renderla nel formato `protobuf` comprensibile da Fabric.

All'interno del file sono presenti:

- **Organizzazioni:** specifica le organizzazioni che possono diventare membri del canale, ognuna delle quali è associata al proprio materiale crittografico per la costruzione dell'*MSP* del canale.
- **Servizio di ordinazione:** definisce i nodi di ordinazione che formeranno il servizio di ordinazione della rete e il metodo di consenso che utilizzeranno per concordare un ordine comune delle transazioni. In questa sezione sono presenti anche le organizzazioni che agiranno come amministratori del servizio di ordinazione.
- **Politiche del canale:** definiscono le regole che governano le interazioni delle organizzazioni con il canale, inclusi i criteri per l'approvazione degli aggiornamenti del canale. In questo caso, si sono utilizzate le politiche predefinite fornite da Fabric.
- **Profili del canale:** ogni profilo del canale fa riferimento alle informazioni di altre sezioni del file stesso per creare una configurazione del canale. I profili vengono utilizzati per creare il blocco genesi del canale di sistema e dei canali applicativi. Il profilo del canale di sistema deve includere la configurazione completa dello stesso, mentre il profilo del canale applicativo deve contenere solo le informazioni aggiuntive necessarie per crearne uno.

Per avviare la rete Hyperledger Fabric, è necessario seguire dei passaggi che possono essere schematizzati come segue:

- **Arresto della rete esistente**: Avendo già lavorato su questa rete, per poter operare da uno stato iniziale noto, il comando `./network.sh down` terminerà tutti i contenitori attivi e rimuoverà tutti gli artefatti generati in precedenza.
- **Avvio della rete di test**: Successivamente, si esegue il comando `./network.sh up` dalla stessa directory. In questo modo, si avvia una nuova istanza della rete Fabric sempre con due organizzazioni peer e un singolo servizio di ordinazione. La rete di test viene avviata senza creare un canale applicativo, ma è lo script a creare il canale di sistema automaticamente quando si esegue il comando. Lo script, inoltre, costruisce il blocco genesi del canale di sistema.
- **Creazione del canale di sistema di ordinazione**: Il primo canale creato è il canale di sistema, che definisce il servizio di ordinazione e le organizzazioni che fungono da amministratori di quest'ultimo, ed è creato automaticamente quando si avvia la rete. Comprende anche le organizzazioni che sono membri del consorzio blockchain. Il blocco genesi del canale di sistema è necessario per distribuire un nuovo servizio di ordinazione.
- **Creazione di un nuovo canale applicativo**: Ora si inizia il processo di creazione di un nuovo canale per le organizzazioni peer. Si utilizza lo strumento `configtxgen` per generare una transazione di creazione del canale, la quale definirà la configurazione iniziale del nuovo canale.
- **Invio della transazione di creazione del canale**: Dopo aver generato la transazione di creazione del canale, la si invia al servizio di ordinazione utilizzando la CLI peer. Questo processo richiede l'autenticazione come amministratore delle organizzazioni coinvolte nel canale. Una volta inviata con successo, il servizio di ordinazione crea il nuovo canale e genera il blocco genesi.

- **Verifica della creazione del canale**: Infine, si verifica che il nuovo canale sia stato creato correttamente. La CLI peer restituirà il blocco genesi del nuovo canale, confermando che la creazione del canale è avvenuta con successo.

In questo modo è possibile creare un nuovo canale per consentire la comunicazione e il trasferimento di risorse tra le varie organizzazioni peer.

Dopo aver creato il canale, si può procedere ad associare ad esso i peer. Le organizzazioni che sono membri del canale possono ottenere il blocco di genesi del canale dal servizio di ordinazione e, successivamente, l'organizzazione può utilizzare il blocco di genesi per unire il peer al canale. Una volta unito al canale, il peer inizierà a costruire il ledger recuperando gli altri blocchi presenti sul canale dal servizio di ordinazione.

Usando la CLI peer come amministratore *Hosp1*, uniamo prima il peer *Hosp1* al canale e si continua, poi, seguendo gli stessi passaggi per unire anche il peer *Hosp2* del canale, impostando le variabili di ambiente per utilizzare la CLI peer come amministratore *Hosp2*. Si estrae il blocco di genesi del canale e gli si assegna il nome `channel_Hosp2.block` per distinguerlo dal blocco estratto da *Hosp1*, in modo da usare tale blocco per unire il peer *Hosp2* al canale.

Dopo la creazione del canale, è importante impostare almeno uno dei peer come peer di ancoraggio, peer essenziali per sfruttare funzionalità come i dati privati e il rilevamento dei servizi. Ogni organizzazione dovrebbe avere più peer di ancoraggio sul canale per garantire la ridondanza e la resilienza. Le informazioni sugli endpoint dei peer di ancoraggio di ciascuna organizzazione sono incluse nella configurazione del canale. Ogni membro del canale può specificare i propri peer di ancoraggio aggiornando la configurazione del canale.

Per impostare un peer di ancoraggio per *Hosp1* e *Hosp2* è necessario aggiornare la configurazione del canale. Per prima cosa si seleziona *Hosp1* come peer di ancoraggio e si estrae il blocco di configurazione più recente del canale; si converte il blocco di configurazione del canale in un formato *json* in modo che risulti leggibile e modificabile. Quindi, si aggiunge il peer di ancoraggio *Hosp1* alla configurazione del canale utilizzando `jq`. Dopo aver aggiornato la configurazione del canale, si converte la configurazione originale e quella modificata nel formato *protobuf* e si calcola la differenza tra di esse.

Una volta ottenuto l'artefatto finale, si utilizza per aggiornare il canale. Poiché si sta aggiornando solo la configurazione del canale relativa a *Hosp1*, non è necessaria l'approvazione degli altri membri del canale. Per impostare i peer di ancoraggio per *Hosp2*, eseguiamo nuovamente il processo: si estrae il blocco di configurazione più recente, si aggiunge il peer *Hosp2* come peer di ancoraggio e si converte e si confrontano le configurazioni come prima.

Completato l'aggiornamento, il canale avrà aggiunto due blocchi di configurazione al blocco di genesi, quindi l'altezza del canale aumenterà da 1 a 3.

Per distribuire un chaincode sul nuovo canale e confermare che sia stato creato correttamente, si può utilizzare lo script `network.sh`, in quanto consente di distribuire il chaincode di trasferimento delle risorse di base su qualsiasi canale di rete di test. Si utilizza tale script per la distribuzione sul nuovo canale e può essere utile controllare i log per verificare che il chaincode sia stato distribuito correttamente sul canale. Il chaincode sarà chiamato per aggiungere dati al ledger del canale. Una volta distribuito, si può confermare che i dati siano stati aggiunti al ledger del canale e, in caso di esito positivo, si dovrebbe ottenere un risultato tipo:

```
  {
  ID: 'asset1',
  nome paziente: 'Mario Rossi',
  data di nascita: '22/08/1987',
  luogo di nascita: 'Potenza (IT)',
  sesso: 'maschio',
  colore occhi: 'blu',
  peso in kg: 92,
  docType: 'asset'
},
{
  ID: 'asset2',
  nome paziente: 'Daniele Bianchi',
  data di nascita: '30/06/1998',
  luogo di nascita: 'Roma (IT)',
  sesso: 'maschio',
  colore occhi: 'verde',
  peso in kg: 87
},
{
  ID: 'asset3',
  nome paziente: 'Antonella Pastrame',
  data di nascita: '05/12/2000',
  luogo di nascita: 'Trieste (IT)',
  sesso: 'femmina',
  colore occhi: 'marrone',
  peso in kg: 59
},
```

La configurazione iniziale del canale, immagazzinata nel cosiddetto *blocco di genesi del canale*, può essere modificata tramite specifici aggiornamenti della configurazione del canale. Una volta che un numero sufficiente di organizzazioni approva un tale aggiornamento, un nuovo blocco di configurazione del canale entrerà in vigore dopo il suo completamento sul canale.

5.3.1 Aggiornamento di una configurazione di canale

Una configurazione di canale in Hyperledger Fabric è un insieme di informazioni che definiscono la struttura e i processi di interazione all'interno di una rete blockchain. Questa configurazione comprende utenti, organizzazioni, peer, nodi di ordinazione, autorità di certificazione (CA), ledger e applicazioni. È fondamentale per stabilire le regole e le politiche che determinano le azioni che gli utenti possono compiere e le condizioni in cui possono farlo.

Le informazioni sulla struttura della rete e sui processi di interazione sono contenute nelle configurazioni dei canali, che vengono concordate e create dai membri del canale stesso. Queste configurazioni sono registrate come blocchi all'interno del ledger del canale, con il primo blocco noto come blocco genesi e l'ultimo rappresentante la configurazione corrente del canale.

Per aggiornare una configurazione di canale, come ad esempio aggiungere nuovi membri o modificare le politiche del canale, viene eseguita una transazione di aggiornamento della configurazione. Questo processo coinvolge tipicamente un singolo amministratore del canale, che propone la modifica dopo una discussione e un accordo tra i membri del canale. La configurazione iniziale del

canale viene stabilita allo stesso modo, con i membri iniziali che prendono decisioni fuori dalla rete.

I canali sono altamente configurabili, ma ci sono limiti alle modifiche che possono essere apportate una volta che alcune specifiche sono state fissate, come ad esempio il nome del canale. Qualsiasi modifica ai parametri richiede il rispetto delle politiche specificate nella configurazione del canale. La configurazione del canale dell'applicazione è simile, ma non identica, alla configurazione del canale del sistema ordinante. Tuttavia, seguono le stesse regole e struttura di base. Una volta estratta e resa leggibile, la configurazione può sembrare complicata, ma presenta una struttura logica.

Ci sono alcuni parametri che si ripetono in più parti della configurazione:

- **Politiche**: definiscono le circostanze in cui tutti i parametri possono essere modificati e non sono solo un valore di configurazione che può essere aggiornato, come specificato in mod_policy.
- **Capacità**: assicura che le reti e i canali elaborino le cose in modo coerente, garantendo risultati deterministici per operazioni come gli aggiornamenti della configurazione del canale e le invocazioni del chaincode. Senza risultati deterministici, un peer su un canale potrebbe invalidare una transazione mentre un altro peer la convaliderebbe.
- Il raggruppamento **Canale/Applicazione** gestisce i parametri di configurazione univoci dei canali dell'applicazione, come ad esempio l'aggiunta o la rimozione di membri del canale. Per impostazione predefinita, la modifica di questi parametri richiede la firma della maggior parte degli amministratori dell'organizzazione dell'applicazione.
- **Aggiunta di organizzazioni a un canale**: per aggiungere un'organizzazione a un canale, è necessario generare e aggiungere qui il relativo MSP e altri parametri dell'organizzazione.
- **Parametri relativi all'organizzazione**: è possibile modificare qualsiasi parametro specifico di un'organizzazione, come ad esempio l'identificazione di un peer di ancoraggio o i certificati degli amministratori dell'organizzazione. È importante notare che la modifica di questi valori per impostazione predefinita richiede solo la firma di un amministratore dell'organizzazione stessa, non della maggior parte degli amministratori dell'organizzazione dell'applicazione.
- Il raggruppamento **Canale/Orderer** gestisce i parametri di configurazione univoci del servizio di ordinazione (o del canale del sistema di ordinazione) e richiede la firma della maggior parte degli amministratori delle organizzazioni di ordinazione. Per impostazione predefinita, esiste una sola organizzazione di ordinazione, ma è possibile aggiungerne altre, ad esempio quando più organizzazioni contribuiscono con nodi al servizio di ordinazione.

Alcuni dei parametri di configurazione gestiti nel primo raggruppamento includono:

- **Dimensione del lotto:** determina il numero e la dimensione delle transazioni in un blocco. Questi parametri assicurano che i blocchi non superino una determinata dimensione massima e vengano tagliati prematuramente se necessario.
- **Timeout del batch:** stabilisce la quantità di tempo da attendere dopo l'arrivo della prima transazione prima di tagliare un blocco. Questo influisce sulla latenza e sul throughput della rete.
- **Convalida del blocco:** definisce i requisiti di firma affinché un blocco sia considerato valido.

- **Tipo di consenso:** permette la migrazione dei servizi di ordinazione basati su Kafka ai servizi di ordinazione basati su Raft.

- **Parametri del servizio di ordinazione Raft:** specifici per un servizio di ordinazione Raft.

Il raggruppamento Channel regola i parametri di configurazione a cui devono dare il consenso sia le organizzazioni peer che le organizzazioni del servizio di ordinazione. Richiede il consenso della maggior parte degli amministratori dell'organizzazione dell'applicazione e degli amministratori dell'organizzazione dell'ordinante.

Alcuni dei parametri di configurazione gestiti nell'ultimo raggruppamento, invece, includono:

- **Indirizzi dell'ordinante**: elenco di indirizzi in cui i client possono richiamare le funzioni dell'ordinatore. Il peer sceglie casualmente tra questi indirizzi per recuperare i blocchi.

- **Struttura dell'hashing**: determina la larghezza dell'albero Merkle utilizzato per calcolare l'hash dei dati del blocco.

- **Algoritmo di hashing**: specifica l'algoritmo utilizzato per calcolare i valori hash nei blocchi della blockchain.

Altri valori di configurazione sono univoci per il canale del sistema ordinante, come ad esempio la politica di creazione del canale e le restrizioni sul canale.

Il processo di modifica della configurazione di un canale coinvolge tre passaggi: ottenere l'ultima configurazione del canale, creare una configurazione del canale modificata e creare una transazione di aggiornamento della configurazione. Questo processo può essere eseguito manualmente o utilizzando strumenti come *jq*.

Prima di procedere con l'aggiornamento della configurazione, è fondamentale impostare alcune variabili di ambiente che dipenderanno dalla struttura della distribuzione, come ad esempio:

- `CH_NAME`: il nome del canale che si sta aggiornando.

- `TLS_ROOT_CA`: il percorso del certificato CA radice della CA TLS dell'organizzazione che propone l'aggiornamento.

- `CORE_PEER_LOCALMSPID`: il nome del MSP che si sta usando.

- `CORE_PEER_MSPCONFIGPATH`: il percorso assoluto verso l'MSP dell'organizzazione in questione.

- `ORDERER_CONTAINER`: il nome di un contenitore del nodo di ordinamento.

Una volta impostate queste variabili, si può continuare a lavorare sul file di configurazione:

Estrarre e tradurre il file config:

- Eseguire il seguente comando per ottenere l'ultimo blocco di configurazione del canale in formato *protobuf*;

- Decodificare il file *protobuf* in formato *json* utilizzando `configtxlator`;

- Estrarre la parte rilevante della configurazione *json* e salvarla come `config.json`.

Modifica la configurazione:

- É possibile modificare manualmente la configurazione, utilizzando un editor di testo, o utilizzare `jq` per applicare le modifiche in modo programmabile;
- Selezionare il metodo più adatto al caso d'uso e apportare le modifiche necessarie.

Ricodifica e invia la configurazione:

- Ricodificare la configurazione e creare una transazione di aggiornamento;
- Codifica nuovamente i file *json* in formato *protobuf*;
- Calcolare la differenza tra la configurazione originale e quella modificata;
- Decodificare la differenza in formato *json* e incapsularla in un'apertura di configurazione;
- Codificare l'apertura di configurazione in formato *protobuf*;
- Inviare la transazione di aggiornamento della configurazione.

Completata la transazione di aggiornamento, bisogna assicurarsi di ottenere le firme necessarie per garantire l'approvazione della configurazione. Questo può richiedere la raccolta di firme da altri amministratori del canale o delle organizzazioni coinvolte.

Infine, è necessario verificare che la nuova configurazione sia stata correttamente aggiunta al ledger e, se necessario, bisogna estrarla e convertila in formato *json* per una visualizzazione più comprensibile e come backup dell'ultima configurazione.

5.4 Aggiunta di una nuova organizzazione alla rete

Per allargare la rete, è possibile aggiungere una nuova organizzazione, *Hosp3*, a un canale esistente di Hyperledger Fabric.

La prima cosa da fare è assicurarsi di lavorare nella directory corretta e assicurarsi di aver ripulito lo script `network.sh` utilizzando il comando `down`, in modo da arrestare tutti i contenitori Docker in esecuzione fino a questo momento. É quindi d'obbligo riavviare la rete con il comando `up createChannel` per avviare la rete Fabric con il canale `channel1`. A questo punto o si sceglie di aggiungere *Hosp3* utilizzando uno script preesistente, fornito dalla documentazione di esempio[29] di Hyperledger Fabric, o si sceglie di aggiungerlo manualmente. A puro scopo illustrativo, sono state intraprese entrambe le strade.

Aggiunta di *Hosp3* con lo script: Si sfrutta lo script `addHosp3.sh` per aggiungere rapidamente *Hosp3* al canale. All'interno di `addHosp3.sh` sono presenti molte funzioni simili a quelle presenti nel file `network.sh`, tra cui la possibilità di aggiungere la nuova organizzazione al canale `channel1`:

```
./addHosp3.sh up -c channel1
```

Sarà lo stesso script a gestire il processo di generazione del materiale crittografico per *Hosp3*, la creazione della definizione dell'organizzazione e l'aggiornamento della configurazione del canale per l'inclusione dell'organizzazione.

Aggiunta di *Hosp3* manualmente: La prima cosa da fare è generare il materiale crittografico per *Hosp3*. All'interno della directory di `addHosp3.sh`, si può usare lo strumento `cryptogen` per generare il materiale crittografico per *Hosp3*. Si stampa successivamente la definizione dell'organizzazione per *Hosp3* salvandola come file `json`; tale file contiene la definizione dell'organizzazione *Hosp3*, inclusi criteri, e i certificati importanti.

Una volta seguiti questi passaggi, *Hosp3* sarà pronto per essere aggiunto al canale Fabric.

Creazione dei peer: Dopo aver generato il materiale crittografico per *Hosp3*, è possibile visualizzare il peer *Hosp3*: se il comando ha successo, si vedrà la creazione del peer *Hosp3*. Tale file Docker Compose è stato configurato per integrarsi nella rete esistente, consentendo al peer *Hosp3* di interagire con i peer e il nodo di ordinazione della rete con cui si sta lavorando.

Una volta che il peer *Hosp3* è stato creato con successo, è possibile procedere al recupero della configurazione del canale e aggiungere il materiale crittografico *Hosp3*.

Recupero configurazione del canale: Si deve ottenere l'ultima versione della configurazione per il canale `channel1` per garantire coerenza e prevenire duplicazioni indesiderate. Poiché l'organizzazione *Hosp3* non fa ancora parte del canale, è necessario agire come amministratori di un'altra organizzazione per recuperare questa configurazione. Visto che *Hosp1* è già membro del canale, è possibile utilizzare l'amministratore di tale organizzazione per accedere alla configurazione tramite il servizio di ordinazione. In questo modo si può recuperare l'ultimo blocco di configurazione, il quale è archiviato in una cartella separata per mantenere la gestione degli aggiornamenti separata da altri tipi di file. Una volta nella cartella corretta, si decodifica il blocco di configurazione in `json`, eliminando le informazioni superflue come intestazioni, metadati e firme dei creatori che non sono rilevanti per queste modifiche. Il risultato sarà un file che verrà usato per apportare le modifiche necessarie alla configurazione del canale.

Aggiunta del materiale crittografico: Si può quindi procedere con l'aggiunta del materiale crittografico per l'organizzazione *Hosp3* alla configurazione del canale. Si utilizza nuovamente lo strumento `jq` per integrare la definizione di configurazione *Hosp3* nel campo dei gruppi di applicazioni del canale, generando un nuovo file.

Ora si hanno due file `json` di interesse: il primo, `config.json`, contiene solo i dettagli delle organizzazioni *Hosp1* e *Hosp2*, e il secondo, `modified_config.json`, include anche l'organizzazione *Hosp3*.

Successivamente, si procede con la codifica di entrambi i file `json` in formato *protobuf* e si calcola il **delta**[4] tra di essi. Dopo la decodifica, si incapsula il file in un messaggio di tipo *busta* per ripristinare il campo dell'intestazione precedentemente rimosso.

Infine, si utilizza lo strumento `configtxlator` per convertirlo nell'oggetto *protobuf* completo richiesto da Fabric, che sarà l'aggiornamento finale della configurazione del canale.

Firma e invio degli aggiornamenti di configurazione: Dopo aver preparato il file binario *protobuf* , si devono ottenere le firme dagli utenti amministratori necessari prima che l'aggiornamento della configurazione possa essere registrato nel ledger. La politica di modifica per il gruppo di applicazioni del canale è impostata sulla maggioranza, il che significa che, essendo coinvolte solo *Hosp1* e *Hosp2*, entrambe le firme sono necessarie: senza entrambe, il servizio di ordinazione rifiuterà la transazione.

[4]rappresenta la differenza tra la configurazione attuale del canale e la configurazione desiderata dopo l'aggiornamento: è l'insieme di modifiche che devono essere apportate alla configurazione corrente del canale per renderla conforme alla nuova configurazione desiderata.

Questi passaggi coinvolgono il processo di firma e di invio dell'aggiornamento della configurazione del canale, garantendo che l'operazione sia conforme alle politiche di modifica stabilite e interessi gli amministratori delle organizzazioni coinvolte.

Invio della chiamata di aggiornamento: Dopo aver emesso con successo la chiamata di aggiornamento del canale, verrà restituito un nuovo blocco, denominato blocco 3, a tutti i peer presenti sul canale. È importante ricordare che i blocchi $0 - 2$ rappresentano le configurazioni iniziali del canale, mentre il Blocco 3 funge da configurazione più recente del canale, includendo ora anche *Hosp3*.

Unire *Hosp3* al canale: Avendo completato correttamente l'aggiornamento della configurazione del canale per includere la nuova organizzazione, è ora possibile unire i peer di *Hosp3* al canale.

Per fare ciò, è necessario esportare le variabili di ambiente specifiche per operare come amministratore di *Hosp3*. Una volta fatto ciò, il servizio di ordinazione verificherà se *Hosp3* può estrarre il blocco di genesi e unirsi al canale. Se l'aggiunta di *Hosp3* alla configurazione del canale è stata eseguita correttamente, il servizio di ordinazione accetterà la richiesta di unione di *Hosp3* al canale.

Una volta recuperato, il blocco di genesi può essere utilizzato per unire il peer di *Hosp3* al canale, consentendo così alla nuova organizzazione di partecipare alle attività sul di esso. In breve, è necessario esportare le variabili di ambiente per operare come amministratore di *Hosp3*, per poi recuperare il blocco genesi del canale e, infine, una volta recuperato, utilizzarlo per unire il peer di *Hosp3* al canale, consentendo così all'organizzazione di partecipare alle operazioni sul canale.

5.4.1 Configurazione dell'elezione del leader:

Quando nuove organizzazioni entrano a far parte della rete, i loro peer vengono avviati con il blocco genesi, che non contiene informazioni specifiche sull'organizzazione aggiunta. Di conseguenza, i nuovi peer non possono utilizzare il meccanismo di *gossip*[5] per ricevere blocchi dagli altri peer della stessa organizzazione fino a quando non ricevono la transazione di configurazione che aggiunge l'organizzazione al canale.

Per consentire ai nuovi peer di ricevere blocchi dal servizio di ordinazione, occorre eseguire una delle due seguenti configurazioni:

1. *Configurazione del peer come leader dell'organizzazione:* garantisce che i peer ricevano sempre i blocchi direttamente dal servizio di ordinazione, senza fare affidamento sul gossip per la ricezione dei blocchi.

2. *Configurazione del peer per utilizzare l'elezione del leader:* consente l'elezione dinamica del leader all'interno dell'organizzazione.

In entrambi i casi, è importante considerare che i nuovi peer possono iniziare a proclamarsi leader fintanto che non ottengono la transazione di configurazione completa che li definisce chiaramente come parte del canale. Una volta che la transazione di configurazione viene applicata con successo, solo uno dei peer sarà attivo come leader dell'organizzazione.

[5]Il termine gossip si riferisce al protocollo di diffusione dei dati sicuro, affidabile e scalabile che garantisce l'integrità e la coerenza dei dati. Il termine deriva dall'idea di diffondere informazioni tra i membri della rete.[29]

5.4.2 Installare, definire e invocare il chaincode:

Si può verificare l'appartenenza di *Hosp3* a `channel1` installando e invocando un chaincode sul canale. Se i membri esistenti del canale hanno già definito un chaincode, *Hosp3* può iniziare a utilizzarlo approvando la sua definizione.

Prima di installare il chaincode come *Hosp3*, si può utilizzare lo script `./network.sh` per distribuire il chaincode di base sul canale. Questo script installerà il chaincode sui peer *Hosp1* e *Hosp2*, ne approverà la definizione e quindi la applicherà al canale.

Dopo aver distribuito il chaincode, e dopo aver necessariamente impostato le variabili di ambiente nel terminale per interagire con la rete come amministratore *Hosp3*, per utilizzare il chaincode come *Hosp3* è necessario:

1. Creare un pacchetto chaincode che verrà installato sul peer *Hosp3*.
2. Approvare la definizione del chaincode di base come *Hosp3*, assicurandosi che sia la stessa approvata e applicata al canale da *Hosp1* e *Hosp2*.
3. Verificare se la definizione del chaincode approvata è stata già applicata al canale.
4. Utilizzare il chaincode di base dopo aver approvato la definizione del chaincode. La definizione del chaincode utilizza la politica di approvazione predefinita, che richiede l'approvazione della maggior parte delle organizzazioni sul canale. Poiché ora abbiamo aggiunto *Hosp3*, la politica richiede l'approvazione di due organizzazioni su tre (*Hosp1*, *Hosp2*, e *Hosp3*).
5. Compilare il ledger con alcune risorse di esempio, ottenendo l'approvazione sia dal peer *Hosp2* che dal nuovo peer *Hosp3* per soddisfare la politica di approvazione.
6. Interrogare il chaincode per verificare che il peer *Hosp3* abbia effettivamente eseguito il commit dei dati[29].

A questo punto, si è conclusa anche l'aggiunta di una nuova organizzazione alla rete Fabric, la quale adesso avrà una struttura di questo tipo:

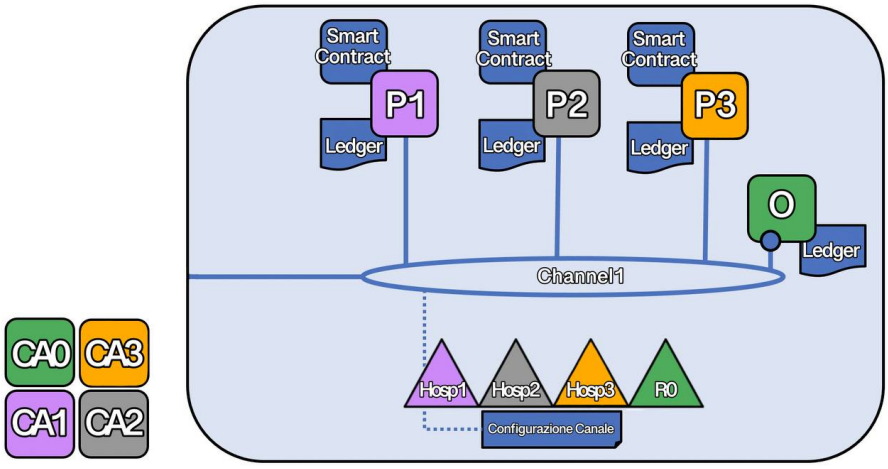

Figura 5.7: Rete con Hosp1, Hosp2 ed Hosp3

Conclusione e sviluppi futuri

Il progetto portato avanti e di cui si è discusso, si è evoluto nel migliore dei modi. Ogni operazione portata avanti, dallo scambio di EHR all'aggiunta di un terzo ospedale, è proceduto senza intoppi, riuscendo ad ottenere, alla fine, la rete desiderata. Il tutto però, è ancora in fase di espansione in quanto è possibile non solo allargare ancora la rete coinvolgendo altre strutture mediche o introducendo nuove operazioni con le EHR, ma anche allargare i campi di utilizzo ed interesse.

Un aspetto cruciale che deve essere affrontato è l'interoperabilità tra i diversi sistemi blockchain. Attualmente, mancano standard aperti che consentano ai diversi prodotti blockchain di comunicare tra loro in modo fluido. Ciò potrebbe ostacolare l'adozione su larga scala della tecnologia blockchain nel settore sanitario. Pertanto, è essenziale impegnarsi nella definizione di standard aperti che favoriscano l'interoperabilità e semplifichino l'integrazione dei sistemi blockchain nei sistemi sanitari esistenti.

Inoltre, vi sono sfide importanti legate alla sicurezza dei dati e alla privacy, alla scalabilità e alla velocità delle transazioni. Queste sono considerazioni cruciali quando si tratta di implementare la tecnologia blockchain nell'ambito sanitario. Migliorare la sicurezza dei dati e garantire la privacy dei pazienti sono priorità assolute, insieme alla necessità di assicurare che la tecnologia possa gestire un volume elevato di transazioni in modo efficiente e rapido[4].

Negli ultimi anni, l'evoluzione della tecnologia indossabile e dell'Internet of Things ha rivoluzionato il settore sanitario, aprendo nuove opportunità e sfide. Grazie al cloud computing e all'analisi dei big data, le informazioni raccolte dai dispositivi indossabili e dagli oggetti connessi contribuiscono alla formazione di enormi quantità di dati sulla salute che, a loro volta, costituiscono un bagaglio informativo prezioso sulla salute e sul benessere dei singoli individui.

Ospedali e istituzioni mediche hanno la possibilità di servirsi di tali sistemi informativi e, a loro volta arricchire le Elettronic Health Records con informazioni dettagliate e in tempo reale sullo stato di salute dei pazienti. Integrando i dati provenienti proprio da tali tecnologie con le informazioni cliniche tradizionali, è possibile migliorare il monitoraggio sanitario, agevolare le diagnosi e personalizzare i trattamenti in modo più efficace.

Oltre gli ospedali, anche le compagnie di assicurazione sanitaria possono trarre vantaggio da questa vasta quantità di dati per offrire polizze più mirate e flessibili. Utilizzando i dati raccolti dalle tecnologie indossabili, insieme alle informazioni dalle EHR, le compagnie assicurative possono valutare in modo più preciso il rischio individuale dei clienti e offrire piani assicurativi su misura per le loro esigenze specifiche.[33].

L'adozione della tecnologia blockchain nel settore sanitario è promettente ma presenta ancora numerose sfide da affrontare. Queste sfide spaziano dalle questioni tecniche e organizzative alla necessità di standardizzazione e alla gestione dei dati. Prima che le applicazioni sanitarie basate su blockchain possano essere pienamente adottate, è fondamentale risolvere questi problemi.

Uno degli ostacoli principali è l'incertezza che circonda l'implementazione della blockchain nel settore sanitario. É necessario garantire la scalabilità delle soluzioni blockchain per gestire grandi quantità di dati in modo efficiente. La standardizzazione dei protocolli e delle interfacce è un'altra

sfida importante, per garantire l'interoperabilità tra diverse piattaforme blockchain e sistemi sanitari. Questa è essenziale affinché i dati possano essere condivisi in modo sicuro e affidabile tra le diverse organizzazioni. Diventa conseguentemente necessaria l'adozione di protocolli comuni che facilitino lo scambio di informazioni tra le parti coinvolte nel settore sanitario[1].

Un'altra questione cruciale è la portata dei dati e la gestione dei costi operativi associati alla raccolta, alla gestione e alla condivisione delle informazioni sanitarie tramite blockchain. Così come è importante l'adeguamento delle norme poste a tutela della privacy e della sicurezza dei dati con leggi e i regolamenti vigenti nel settore sanitario, al fine di garantirne coerenza e conformità[1].

In conclusione, diventa evidente come la ricerca e lo sviluppo della tecnologia blockchain in campo sanitario siano fondamentali per affrontare le sfide ancora aperte, e sfruttare appieno le opportunità offerte da questa tecnologia, il cui consolidamento richiederà collaborazione, innovazione e impegno a lungo termine da parte di ricercatori, professionisti sanitari e sviluppatori tecnologici.

Bibliografia

[1] Thomas K. Dasaklis, Fran Casino, and Constantinos Patsakis. Blockchain meets smart health: Towards next generation healthcare services. In *2018 9th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pages 1–8, 2018.

[2] Tareq Ahram, Arman Sargolzaei, Saman Sargolzaei, Jeff Daniels, and Ben Amaba. Blockchain technology innovations. In *2017 IEEE Technology & Engineering Management Conference (TEMSCON)*, pages 137–141, 2017.

[3] Proposing new blockchain challenges in ehealth. 43(3):64, February 2019.

[4] Cornelius C. Agbo, Qusay H. Mahmoud, and J. Mikael Eklund. Blockchain technology in healthcare: A systematic review. *Healthcare*, 7(2), 2019.

[5] Zainab Alhadhrami, Salma Alghfeli, Mariam Alghfeli, Juhar Ahmed Abedlla, and Khaled Shuaib. Introducing blockchains for healthcare. In *2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, pages 1–4, 2017.

[6] Alevtina Dubovitskaya, Zhigang Xu, Samuel Ryu, Michael Schumacher, and Fusheng Wang. Secure and trustable electronic medical records sharing using blockchain. August 2017.

[7] Data silos. https://www.hpe.com/emea_europe/en/what-is/data-silos.html, 2024. Accessed: 19/02/2024.

[8] Andrè Henrique Mayer, Cristiano Andrè da Costa, and Rodrigo da Rosa Righi. Electronic health records in a blockchain: A systematic review. *Health Informatics J.*, 26(2):1273–1288, June 2020.

[9] Cartella clinica elettronica, emr, ehr, fse: quali differenze? https://www.healthtech360.it/salute-digitale/cartella-clinica-elettronica-emr-ehr-fse-differenze/, 2023. Accessed: 15/02/2024.

[10] Gartner. https://it.wikipedia.org/wiki/Gartner, 2024. Accessed: 19/02/2024.

[11] Massimo Mangia. Cartella clinica elettronica, come renderla utile a medici e pazienti. https://www.agendadigitale.eu/sanita/cartella-clinica-elettronica-serve-una-riprogettazione/, 2021. Accessed: 17/02/2024.

[12] Dimiter V Dimitrov. Blockchain applications for healthcare data management. *Healthc. Inform. Res.*, 25(1):51–56, January 2019.

[13] Maria Karampela, Sofia Ouhbi, and Minna Isomursu. Personal health data: A systematic mapping study. *International Journal of Medical Informatics*, 118:86–98, 2018.

[14] Shan Jiang, Jiannong Cao, Hanqing Wu, Yanni Yang, Mingyu Ma, and Jianfei He. Blochie: A blockchain-based platform for healthcare information exchange. In *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 49–56, 2018.

[15] Fhir, lo standard per condividere i dati sanitari. https://www.healthtech360.it/salute-digitale/fhir-dati-sanitari/, 2023. Accessed: 19/02/2024.

[16] Elisabetta Perra e Andrea Moi. Perché è importante rispettare gli standard hipaa in italia? http://www.andreamoi.it/blog/post13-perche-e-importante-rispettare-gli-standard-HIPAA-in-Italia.html, 2021. Accessed: 19/02/2024.

[17] Ken Ebert e Keela Rose Shatzkin. Cardea, un ecosistema open source che preserva la privacy per la verifica dei dati sanitari, si unisce a hyperledger labs. https://www.hyperledger.org/blog/2023/05/10/cardea-a-privacy-preserving-open-source-ecosystem-for-verifying-health-data-joins-hyperledger-labs, 2023. Accessed: 19/02/2024.

[18] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf. Accessed: 19/02/2024.

[19] Massimo Di Pierro. What is the blockchain? *Computing in Science & Engineering*, 19(5):92–95, 2017.

[20] Introduzione alla tecnologia di contabilità distribuita (dlt). https://fastercapital.com/it/tema/introduzione-alla-tecnologia-di-contabilit%C3%A0-distribuita-(dlt).html. Accessed: 19/02/2024.

[21] Marianna Belotti, Nikola Božić, Guy Pujolle, and Stefano Secci. A vademecum on blockchain technologies: When, which, and how. *IEEE Communications Surveys & Tutorials*, 21(4):3796–3838, 2019.

[22] Informatica e ingegneria online. https://vitolavecchia.altervista.org/differenza-tra-blockchain-pubblica-blockchain-privata-e-consortium-blockchain/. Accessed: 19/02/2024.

[23] Karl Wüst and Arthur Gervais. Do you need a blockchain? In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 45–54, 2018.

[24] Albero merkle. https://en.wikipedia.org/wiki/Merkle$_t ree$, 2024. *Accessed* : 27/02/2024.

[25] Cryptopedia Staff. What is tokenization in blockchain? https://www.gemini.com/it-IT/cryptopedia/what-is-tokenization-definition-crypto-token#section-what-does-crypto-tokenization-look-like, 2023. Accessed: 19/02/2024.

[26] Valentina Gatteschi, Fabrizio Lamberti, Claudio Demartini, Chiara Pranteda, and Víctor Santamaría. To blockchain or not to blockchain: That is the question. *IT Professional*, 20(2):62–74, 2018.

[27] Sfatiamo i 15 miti su bitcoin più diffusi. https://academy.binance.com/it/articles/debunking-the-top-15-bitcoin-myths, 2023. Accessed: 28/02/2024.

[28] Cosa sono le blockchain fork? https://www.cmcmarkets.com/it-it/impara-come-operare-con-criptovalute/cosa-e-una-blockchain-fork, 2024. Accessed: 19/02/2024.

[29] Hyperledger fabric. `https://hyperledger-fabric.readthedocs.io/en/release-2.2/whatis.html`, 2020. Accessed: 05/02/2024.

[30] Giuditta Mosca. Data base management system (dbms), cosa sono e come sceglierne uno. https://www.bigdata4innovation.it/big-data/data-base-management-system-dbms-cosa-sono-e-come-sceglierne-uno/, 2022. Accessed: 27/02/2024.

[31] Morena. Algoritmo di consenso raft. https://comeaprire.com/blog/2021/10/15/algoritmo-di-consenso-raft/, 2021. Accessed: 19/02/2024.

[32] Un framework rpc universale open source ad alte prestazioni. https://grpc.io/, 2024. Accessed: 19/02/2024.

[33] Xueping Liang, Juan Zhao, Sachin Shetty, Jihong Liu, and Danyi Li. Integrating blockchain for data sharing and collaboration in mobile healthcare applications. In *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1–5, 2017.