Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica per il Management

PROGETTAZIONE DI UN SISTEMA DLT PER L'INTEROPERABILITÁ E LO SCAMBIO DI DATI SANITARI UTILIZZANDO LE TECNOLOGIE IOTA E IPFS

Relatore:
Chiar.mo Prof.
GABRIELE D'ANGELO

Presentata da:
MICHELE MONGARDI

Correlatori:
Chiar.mo Prof.
STEFANO FERRETTI
Chiar.mo Dott.
MIRKO ZICHICHI

Sessione III
Anno Accademico 2018/2019

Indice

Elenco delle figure

Listings

Introduzione

É noto che una trasmissione sicura ed efficiente dei dati sanitari permette un incremento notevole della qualità dell'assistenza sanitaria. A tal fine, negli ultimi dieci anni i sistemi sanitari di tutto il mondo hanno investito considerevolmente in una varietà di procedure e strumenti che mirano a migliorare lo scambio interoperabile di dati. Queste soluzioni hanno raggiunto diversi gradi di successo nel mondo odierno. Tra queste, le Distributed Ledger Technologies, abbreviate in DLTs, sono un nuovo tipo di tecnologia in cui i dati vengono salvati all'interno di un registro distribuito, gestito da nodi in una rete peer-to-peer tramite un protocollo basato sul consenso. Queste offrono un grande potenziale per rendere immutabile e sicuro lo scambio di informazioni sulla salute.

Lo scopo di questa tesi è di esplorare il potenziale di una specifica DLT nel supportare la trasmissione di dati sanitari sviluppando un programma per lo scambio automatizzato, immutabile e sicuro delle informazioni sulla salute dei pazienti. L'obiettivo della struttura da realizzare è quello di ottenere una profilazione accurata dei pazienti raccogliendo periodicamente informazioni sullo stato di salute degli individui, in modo da poter fornire successivamente assistenza sanitaria di alta qualità grazie allo scambio efficiente ed efficace dei dati raccolti.

Il caso d'uso studiato gestisce il monitoraggio relativo alla glicemia, parametro che misura il livello di glucosio nel sangue. In particolare, ci si è concentrati sullo scambio di dati glicemici (BG) da parte del paziente-fornitore, non solo a causa dell'elevato numero di pazienti affetti da diabete in tutto il mondo, ma anche per l'importanza della comunicazione di questi dati al personale sanitario per una migliore gestione del trattamento. Solitamente i pazienti monitorano i livelli di glucosio manualmente o fanno affidamento

al software del fornitore del dispositivo di rilevamento per compilare un report. Questo processo manuale è dispendioso in termini di tempo e soggetto ad errori, evidenziando la necessità di sistemi che supportano lo scambio di dati tra dispositivi BG di automonitoraggio e cartelle cliniche elettroniche (EHR) in modo sicuro, efficace e a prova di manomissione.

Lo sviluppo del progetto si è basato sul rispetto delle seguenti specifiche: controllato dal paziente, completamente digitale, interoperabile e gestito in maniera decentralizzata, garantendo l'anonimato e la sicurezza delle informazioni. Controllato dal paziente significa che gli utenti, oltre a fornire il consenso per lo scambio dei dati, possono anche concederne l'accesso alle parti che ritengono opportune, che si tratti del medico di base, dell'endocrinologo o altri. Digitale specifica che i dati vengono scambiati in un formato digitale, riducendo al minimo eventuali errori di inserimento manuale dei parametri. L'interoperabilità del sistema specifica che le informazioni di carattere personale possono essere lette, se il consenso è fornito dal paziente, da un altro sistema informativo di una differente ala dell'assistenza sanitaria. Infine, la gestione decentralizzata specifica che tutti i dati generati vengono archiviati tramite una tecnologia a prova di manomissione su più nodi distribuiti, anziché su una singola repository di dati centralizzata.

La tesi è strutturata in quattro capitoli nel modo seguente:

- Nel primo capitolo viene enunciato lo Stato dell'Arte, delineando il problema affrontato e le ragioni che hanno movimentato questo studio. Viene introdotto il sistema che permette lo scambio delle informazioni sanitarie e le normative Europee che delineano il trattamento dei dati personali. In seguito si approfondisce il tema delle Distributed Ledger Technologies descrivendone la nascita e i cambiamenti che hanno apportato. Successivamente vengono analizzate le Blockchain, evidenziandone le caratteristiche e le modalità d'uso. Infine si presentano le tecnologie che sono alla base dello sviluppo del progetto, il sistema distribuito IOTA, il protocollo di comunicazione dati Masked Authenticated Messaging (MAM) e IPFS, un sistema distribuito per l'archiviazione e l'accesso a file, siti Web, applicazioni e dati.

- Nel secondo capitolo viene definita la fase di progettazione, nella quale si descrive lo studio preliminare alla realizzazione del progetto. Si analizzano le possibilità applicative delle tecnologie introdotte nel capitolo precedente e, successivamente, viene esaminato il caso d'uso in discussione. Infine si approfondisce la tematica riguardante la gestione dei dati medicali, nonché i rischi ad essi associati.

- Nel terzo capitolo viene descritta la fase di sviluppo del programma, nella quale si introducono gli strumenti tramite cui è stato possibile intraprendere lo sviluppo del sistema. Tra questi viene enunciato Node.js, un ambiente di esecuzione di codice JavaScript, e le librerie API con cui è stato possibile creare un'interazione con i protocolli MAM e IPFS. In questa sezione viene quindi descritto il metodo adottato per lo sviluppo dell'infrastruttura, in modo che sia garantita sicurezza e stabilità, e le tecnologie implementative impiegate al fine di svolgere operazioni di testing, per poi integrarle in un singolo programma funzionante. Dopodiché viene introdotto il sistema realizzato e se ne spiega il funzionamento. Per finire si espongono i principali vantaggi per cui si è deciso di impiegare queste tecnologie e il loro ruolo all'interno del progetto.

- Nel quarto e ultimo capitolo vengono raccolti i risultati ottenuti dallo sviluppo del progetto, nonché descritte le metodologie per accedere ai dati manipolati. Successivamente si procede con un'analisi statistica relativa al calcolo della latenza che intercorre tra l'inizio e la fine di un'operazione di caricamento. Al fine di ottenere valori maggiormente affidabili, viene inoltre misurata la latenza media relativa all'invio dei dati su IPFS e IOTA, nonché i corrispettivi intervalli di confidenza, in modo da garantire la validità dell'esito ottenuto.

Capitolo 1

Stato dell'Arte

Il seguente capitolo introduce il sistema realizzato, delineando l'importanza di una gestione sicura dei dati medicali in un'ottica decentralizzata, permettendo una profilazione migliore degli utenti che prendono parte al servizio, chiamati pazienti, in modo da fornire al medico curante una serie di informazioni dettagliate nel tempo sul soggetto monitorato, consentendo il trattamento delle condizioni cliniche del paziente in maniera efficiente.

A tal fine, si analizzano gli aspetti relativi al trattamento dei dati personali imposti dall'Unione Europea nel regolamento n. 2016/679, in sigla GDPR, e allo standard FHIR che descrive i formati e gli elementi di dati da impiegare per lo scambio elettronico di informazioni sanitarie. In quest'ottica va sottolineata la necessità di garantire l'anonimato dei pazienti, gestendo la manipolazione dei dati di ciascun soggetto in maniera sicura, evitando che entità non autorizzate entrino in possesso di informazioni personali in modo indebito. Viene quindi implementando un sistema per garantire l'immutabilità delle informazioni e la tutela degli individui monitorati.

Pertanto viene introdotto il concetto di Distributed Ledger Technologies e di Blockchain, tecnologie che possiedono caratteristiche affini alla realizzazione dell'elaborato. Infine si fa riferimento al protocollo IOTA e alla sua estensione MAM, impiegati in collaborazione all'architettura IPFS per una gestione affidabile dei dati, permettendo l'interoperabilità con il sistema, sia da parte del personale sanitario, sia lato utente.

Al fine della realizzazione del progetto è stato necessario valutare gli aspetti relati-

vi a data reversal risk, data linkability risks, processing time, file size compatibility e complessità del sistema.

1.1 Sistema per la scambio di Informazioni Medicali

L'obiettivo di questo programma è quello di sviluppare un sistema che permetta lo scambio immutabile e interoperabile di dati glicemici di un paziente, in maniera conforme con quanto stabilito dal regolamento generale sulla protezione dei dati (GDPR). Con questa precisazione si vuole sottolineare la necessità di elaborare i dati raccolti, in quanto appartenenti alla categoria dei dati sensibili. A tal fine, le informazioni medicali devono essere anonimizzate e criptate.

Per garantire immutabilità e interoperabilità dei dati vengono impiegate due tecnologie: l'InterPlanetary File System (IPFS) e IOTA. IPFS viene definito come un file system distribuito per l'archiviazione e l'accesso a file. Esso semplifica la concettualizzazione di come potrebbe funzionare un web distribuito. Simile al modo in cui un BitTorrent sposta i dati su Internet, IPFS elimina la necessità di relazioni client-server centralizzate (ovvero, il web corrente). IPFS implementa una distribuzione dei dati in tutta la rete, in modo da proteggere i documenti da possibili attacchi o smarrimenti. Nel progetto realizzato, questa tecnologia viene integrata per l'archiviazione dei file contenenti i parametri glicemici. Dal caricamento di un documento su questa piattaforma ne deriva una sequenza di caratteri univoca, che prende il nome di *hash*, il quale rappresenta l'identificatore con cui rintracciare il file all'interno del sistema. Non è possibile, essendo a conoscenza dell'hash, risalire al contenuto da cui è stato generato. IPFS implementa così un protocollo di cifratura dei dati medicali.

IOTA rappresenta un registro distribuito che consente ai dispositivi collegati di trasferire dati tra loro a zero commissioni. Esso è in grado di gestire in maniera ottimale grandi volumi di transazioni. Grazie al suo protocollo di comunicazione dati MAM, è possibile creare un flusso di dati su cui memorizzare gli hash ottenuti da IPFS. Il canale implementa un meccanismo di cifratura per mantenere in memoria questi valori. Coloro

che hanno accesso al canale (ad esempio, il medico curante e l'endocrinologo) potranno recuperare gli hash memorizzati al suo interno e utilizzarli per reperire i documenti contenenti i parametri glicemici, attraverso IPFS.

In questa maniera viene a crearsi un archivio dati decentralizzato dei pazienti, con cui è possibile interagire al fine di scambiare informazioni medicali e ottenere un'assistenza sanitaria di alta qualità.

1.1.1 Direttive sul trattamento dei Dati

Il Regolamento Europeo n. 2016/679 relativo al General Data Protection Regulation (GDPR) è il documento redatto dal Parlamento e dal Consiglio Europeo relativo alla protezione delle persone fisiche con riguardo al trattamento dei dati personali, nonché alla libera circolazione di tali dati, e che abroga la direttiva 95/46/CE (regolamento generale sulla protezione dei dati)[2]. In questa normativa vengono inoltre trattate le informazioni protette di carattere sanitario, anche dette Protected Health Information (PHI), stabilendo le modalità di salvaguardia dei dati personali e definendo i principali diritti della persona a cui si riferiscono: diritto di rettifica, diritto di cancellazione, diritto di accesso e diritti relativi al trattamento automatizzato.

Le DLTs trovano nuove applicazioni in materia di interazione con i dati personali e fanno riferimento al regolamento europeo per la loro gestione. Tuttavia vi sono tre macroaree in cui il GDPR non offre ancora abbastanza chiarezza su come dovrebbero essere sviluppate applicazioni DLTs nel mondo reale. Si identificano responsabilità e ruoli, anonimizzazione dei dati personali e conflitto di diritti del GDPR. Per quanto concerne l'anonimizzazione delle informazioni protette, è chiaro che il GDPR non viene applicato ai dati anonimizzati e che questo tipo di informazioni può essere memorizzato su DLT. Tuttavia, ciò che si qualifica come anonimo non è ancora chiaro. L'unica indicazione da perseguire è che deve essere irreversibilmente impossibile identificare un individuo sostenendo costi, in termini di tempo e denaro, per cui non sia vantaggio entrare in possesso di tali informazioni. In ambito sanitario, raggiungere questa identificazione irreversibile è ancora più difficile poiché le PHI includono non solo dati personali generali,

ma anche informazioni sullo stato di salute, dati genetici ed elementi biometrici. Per tali motivi il GDPR stabilisce una nuova direttiva da perseguire per l'anonimizzazione dei dati sanitari rispetto ai metodi di "pseudonimizzazione" attualmente utilizzati nella ricerca clinica, in cui la riservatezza è garantita attraverso una più semplice codifica dei dati.

In questo studio la guida fornita dal GDPR su come trattare i dati personali si riferisce semplicemente alla necessità di ridurre al minimo il reversal risk (ovvero la ricostruzione dei dati originali) e il linkability risk (cioè il rischio di ricollegare i dati anonimi ad un individuo esaminando i modelli di utilizzo o di contesto, o confrontandoli con altre informazioni). Una varietà di tecniche può realizzare l'anonimizzazione delle informazioni, tra cui offuscamento, crittografia, hash e aggregazione. Tuttavia, rimane poco chiaro dal punto di vista legale, quali di esse (individualmente o in combinazione) siano più adeguate per convertire i dati personali in dati anonimi.

In aggiunta al rispetto della normativa 2016/679, si vuole usufruire dello standard Fast Healthcare Interoperability Resources (FHIR), che regola le modalità da perseguire per lo scambio elettronico di informazioni sanitarie[3]. Le cartelle cliniche stanno diventando sempre più digitalizzate e progressivamente allo spostamento dei pazienti all'interno dell'ecosistema sanitario, queste devono presentarsi disponibili, rilevabili e comprensibili. Inoltre, per sostenere l'automatizzazione alle decisione cliniche e altre elaborazioni basate su macchine, i dati devono anche essere strutturati e standardizzati. FHIR mira a semplificare l'implementazione senza sacrificare l'integrità delle informazioni, sfrutta i modelli logici e teorici esistenti per fornire un meccanismo coerente, facile da implementare e rigoroso per lo scambio di dati tra applicazioni sanitarie. FHIR dispone di meccanismi integrati per la tracciabilità delle informazioni e altri importanti modelli di contenuto. Ciò garantisce l'allineamento ai modelli e alle pratiche precedentemente definite del sistema sanitario. In questo maniera viene a crearsi un ecosistema omogeneo, ma deterministico, che permette la manipolazione di dati sanitari in una modalità riconosciuta universalmente.

1.2 Distributed Ledger Technologies

Per comprendere il significato di DLT occorre in prima istanza definire il concetto del termine Ledger. Il Ledger, in italiano "Libro Mastro", delinea la base fondamentale della contabilità. É possibile affermare che esso rappresenta un elemento centrale della comunità odierna e del modo di interpretare e gestire le relazioni tra persone e organizzazioni. Da quando la popolazione umana ha iniziato a lasciare memoria delle proprie azioni, il Ledger è diventato un riferimento per le iniziative di carattere commerciale o di scambio tra due o più parti. Il Libro Mastro assume valore nel momento e nella misura in cui può essere consultato, in quanto esso rappresenta una memoria storica utile per controllare, verificare, gestire le transazioni e gli scambi che sono stati effettuati. Perciò, nel passato, il suo ruolo subì delle evidenti limitazioni dai mezzi che lo componevano. Grazie all'avvento dell'informatizzazione è stato possibile velocizzare alcuni passaggi e semplificare determinati controlli a cui era sottoposto, ma per molto tempo ancora i Ledger sono stati interpretati con la stessa fondamentale logica centralizzata che caratterizzava la carta. Vi era perciò qualcuno che si occupava del data entry delle informazioni, che nascevano come analogiche, altri che gestivano i sistemi e infine qualcuno che, centralmente, gestiva le estrazioni dei dati o la loro elaborazione. Pertanto il dato ha continuato ad essere concepito come analogico e gestito con un paradigma dello stesso tipo, anche se con strumenti digitali.

In seguito alla diffusione della Blockchain, prendono vita i primi Digital Ledger grazie alla contemporanea disponibilità di due fattori abilitanti: la criptografia e lo sviluppo di algoritmi di controllo e verifica dei dati, che aprono le porte a quelle che diventano le DLTs. Con le DLTs ci si addentra nell'ambito dei Database Distribuiti, ovvero dei Ledgers che possono essere aggiornati, gestiti, controllati e coordinati in modo distribuito da parte di tutti gli attori che partecipano al sistema. Le fondamenta delle DLTs sono da ricercarsi nella creazione di grandi network costituiti da una serie di partecipanti, dove ognuno di essi è chiamato a gestire un nodo di questa rete. Ciascun entità è autorizzata ad aggiornare i Distributed Ledgers in modo indipendente dagli altri, ma sotto il con-

trollo consensuale dei restanti elementi. I records non sono più gestiti, come accadeva tradizionalmente, sotto il controllo rigoroso di una autorità centrale, ma vengono creati e caricati da ogni nodo in modo indipendente. Tramite questa metodologia organizzativa, ogni partecipante è in grado di processare e controllare ogni transazione ma, al contempo, qualunque operazione, ancorché gestita in autonomia, deve essere verificata, votata e approvata dalla maggioranza dei partecipanti alla rete. L'autonomia di ciascun nodo è subordinata al raggiungimento di un consenso sulle operazioni che vengono svolte e solo attraverso esso vengono successivamente autorizzate e attivate. Quanto appena descritto definisce il concetto principale che si trova alla base delle DLTs, il consenso.

Il rapporto tra Blockchain e Distributed Ledger è fondamentale. La Blockchain appartiene alla categoria delle DLTs, le quali rappresentano un insieme di sistemi la cui caratteristica comune è quella di fare riferimento a un registro distribuito, gestito in modo da consentire l'accesso e la possibilità di effettuare modifiche da parte di più nodi di una rete. Le varie tipologie di DLTs si distinguono primariamente nelle modalità con cui si gestisce il controllo e la verifica delle azioni di scrittura sul registro, il raggiungimento del consenso necessario per validare le azioni e la struttura del registro distribuito. I Distributed Ledgers sono sottoposti ad aggiornamento solo dopo aver ottenuto il consenso. Dopodiché ogni nodo viene aggiornato con l'ultima versione di ogni singola operazione di ciascun partecipante. Grazie a questo, ogni operazione viene mantenuta in modo indelebile e inalterabile su ogni singolo nodo. É possibile verificare che ciascun partecipante disponga di una copia, immutabile, di ciascuna operazione.

Le DLTs hanno apportato un grosso cambiamento rispetto alle tradizionali logiche centralizzate, dove la verifica e l'autorizzazione erano gestite da un'unica autorità, e quando l'accesso a tutti gli archivi era gestito a livello centrale. Questo modello architetturale permette di interpretare il database in senso più ampio rispetto al passato. Non è più possibile identificare i Ledgers come semplici archivi, bensì è doveroso parlare di DLTs come un rapporto rinnovato tra persone e informazioni.[4]

1.2.1 Blockchain

Consentendo la distribuzione ma non la copia delle informazioni, la tecnologia Blockchain ha definito la struttura portante per un nuovo tipo di Internet. Originariamente, la Blockchain, è stata introdotta come tecnologia per la valuta digitale Bitcoin ma nel tempo sono state trovate altre potenziali implementazioni per questo sistema. Ameer Rosic descrive la Blockchain come una serie di record immutabili, ognuno dei quali contenente dati e un time-stamp che permette di definire il momento esatto, espresso in millisecondi, in cui quel record è stato creato[5]. Questa struttura non appartiene ad una singola entità centralizzata, bensì viene gestita da un insieme di computer connessi tra di loro tramite la rete (cluster). Ogni record è protetto e associato l'un l'altro mediante dei principi crittografici, al fine di rendere i dati non modificabili. Dal momento che essa rappresenta un registro condiviso e immutabile, le informazioni in esso contenute sono aperte a chiunque. Quindi, tutto ciò che è costruito sulla Blockchain è per sua natura trasparente e ogni soggetto coinvolto è responsabile delle proprie azioni.

Questa tecnologia rappresenta un modo semplice ma geniale di trasferire informazioni da "A" a "B", in modo completamente automatizzato e sicuro. Una parte di una transazione avvia il processo creando un blocco. Questo blocco è verificato da migliaia, forse milioni, di computer distribuiti sulla rete. Il blocco verificato viene aggiunto ad una catena, che viene archiviata in rete, creando non solo un record univoco, ma anche una cronologia unica associata ad esso. Falsificare un singolo record significherebbe falsificare l'intera catena composta da milioni di istanze, un processo quasi impossibile da realizzare. Bitcoin utilizza questo modello per transazioni monetarie, ma può essere implementato in molti altri modi.

L'aspetto fondamentale che ha permesso a questa tecnologia di prendere piede è dato dalla sua caratteristica di essere gratis. Non solo la Blockchain può trasferire e archiviare denaro, ma può anche sostituire tutti i processi e i modelli di business che fanno affidamento su una piccola commissione per una transazione, o qualsiasi altra transazione tra due parti. Inoltre, il motivo per cui la Blockchain ha guadagnato così tanta ammirazione, deve ricercarsi nelle sue proprietà intrinseche:

- Non appartiene ad una singola entità, bensì è decentralizzata.
- I dati sono archiviati crittograficamente al suo interno.
- É immutabile, quindi nessuno può manomettere ciò che costituisce il suo contenuto.
- É possibile tenere traccia dei dati, se lo si desidera, in quanto è "trasparente".

L'aspetto relativo all'immutabilità deriva dall'applicazione di una funzione hash crittografica. Applicare una funzione di hashing consiste nel fornire come input una stringa di lunghezza arbitraria e ottenere come output un valore di lunghezza fissa. Una delle funzioni hash più utilizzate in ambito crittografico è la funzione SHA-256 (Secure Hashing Algorithm 256). Come è possibile notare nella Figura 1.1, applicando questa funzione, l'output che si ottiene ha una lunghezza predefinita (256 bit) indipendentemente dalla dimensione dell'input che si fornisce alla funzione.

INPUT	HASH
Hi	3639EFCD08ABB273B1619E82E78C29A7DF02C1051B1820E99FC395DCAA3326B8
Welcome to blockgeeks. Glad to have you here.	53A53FC9E2A03F9B6E66D84BA701574CD9CF5F01FB498C41731881BCDC68A7C8

Figura 1.1: Esempio SHA-256 [5]

Questa parentesi relativa alle funzioni hash serve per comprendere meglio il concetto di puntatore, utilizzato nelle Blockchain. Come descritto precedentemente, quest'ultima assume la struttura di un elenco collegato contenente dei dati. La concezione di collegamento è da attribuire al puntatore hash, anch'esso contenuto all'interno dei record della catena, il cui ruolo è quello di mantenere una referenza al blocco precedente. Esso rappresenta una sorta di contenitore che ha al suo interno l'indirizzo del record antecedente e l'hash dei dati all'interno di quel blocco. Questa modifica è ciò che rende le Blockchain

così affidabili e pionieristiche. Si supponga un attacco da parte di un hacker che tenta di modificare i dati all'interno di una transazione. Ne deriva una conseguente modifica dell'hash relativa al contenuto di quel record. Quest'alterazione dovrebbe ripercuotersi su tutti gli elementi della Blockchain e ciò implicherebbe l'intera variazione della catena, il che non è possibile. In questo modo si garantita l'immutabilità.

La Blockchain viene gestita da una rete peer-to-peer. Il network è composto da una raccolta di nodi interconnessi tra loro, identificati come i singoli computer, chiamati *miners*, che accettano input e svolgono una funzione, fornendo come risultato un output. Essi assumono tale nomenclatura in quanto effettuano un'operazione detta "mining", che consiste nella creazione di un blocco valido della catena attraverso la risoluzione di un calcolo matematico relativamente complesso (Proof-of-Work, o PoW). La PoW rappresenta l'algoritmo di consenso alla base della rete Blockchain. Questo algoritmo, difatti, viene utilizzato per confermare le transazioni e produrre i nuovi blocchi della catena. La PoW incentiva i miners a competere tra loro nell'elaborazione degli scambi, ricevendo in cambio una ricompensa. La Blockchain utilizza questo speciale tipo di rete per suddivide l'intero carico di lavoro tra i nodi partecipanti, i quali sono tutti ugualmente privilegiati, chiamati "peer".

1.3 IOTA

James Brogan, nell'articolo *Authenticating Health Activity Data Using Distributed Ledger Technologies[6]* , definisce IOTA come un protocollo progettato per essere leggero e malleabile, ponendosi come punto fermo per la comunicazione sicura di dati tra dispositivi appartenenti all'Internet-of-Things (IoT). Quest'ultimo si differenzia dai registri distribuiti basati sulle blockchain tradizionali per due principali caratteristiche, ovvero costi e latenza.

IOTA non utilizza i concetti di blocco e di miners. In questo sistema tutte le transazioni che vogliono essere aggiunte al registro distribuito di IOTA, noto come Tangle, devono convalidare due transazioni non ancora confermate sul registro, risolvendo un

"puzle" da costo computazionale relativamente basso. Tramite questo meccanismo è possibile ottenere un'architettura innovativa con un approccio scalabile delle transazioni definito dal tasso di convalida di queste ultime.

Il protocollo IOTA è stato progettato prendendo in esame gli ambienti con ampiezza di banda limitata, e con accortezza relativamente ai computer quantistici, implementando lo schema di firma one-time di Winternitz, il quale fornisce resistenza a questi calcolatori, consentendo un'autenticazione efficiente delle trasmissioni nelle reti di sensori grazie alla bassa potenza di alimentazione richiesta per il calcolo e la comunicazione. Poiché la trasmissione e la persistenza di una transazione nel Tagle sono esenti da commissioni, è possibile utilizzare il protocollo IOTA per garantire l'integrità dei dati nel tempo.

IOTA mette a disposizione un'estensione chiamata MAM, un modulo sperimentale che funge da protocollo di comunicazione dati di secondo livello, il quale estende le funzionalità delle transazioni IOTA.

1.3.1 Descrizione di IOTA

IOTA viene descritto come una DLT open-source che consente ai dispositivi connessi di trasferire dati e token IOTA tra di loro, senza alcun costo di commissione. Con dispositivi connessi si intendono nodi e client che operano sul network di questo protocollo. I nodi possono essere definiti come la colonna portante di una rete IOTA, in quanto sono gli unici dispositivi che hanno accesso in lettura e scrittura al record immutabile delle transazioni chiamato Tangle. I client, d'altro canto, sono identificati come i dispositivi che si connettono ai nodi per effettuare transazioni o archiviare dati sul Tangle.

IOTA implementa una gestione dei dati attraverso una rappresentazione che segue il sistema numerico ternario, che consiste di trits e trytes. Rispetto al binario, quest'ultimo, è considerato più efficiente in quanto può raffigurare i dati in tre stati anziché solo in due. Questo permette di esprimere i concetti in un sistema ternario bilanciato, che consiste in 1, 0 o -1. Tali valori sono chiamati trits, e tre di questi corrispondono ad un tryte, il quale può assumere 27 (3^3) valori possibili. Per facilitarne la lettura, essi sono rappresentati

come uno dei possibili caratteri codificati, che consistono nel numero 9 e nelle lettere maiuscole dalla A alla Z.

Tramite IOTA è possibile sviluppare applicazioni che godono delle seguenti proprietá:

- Autenticitá, tramite cui si può verificare l'identità di colui che ha inviato i dati e controllare se possiede o meno i token IOTA che ha dichiarato.
- Integritá, che consente di dimostrare l'immutabilitá dei dati.
- Confidenzialitá, attraverso la quale si ha il controllo di chi ha accesso ai dati, tramite crittografia.
- Micropagamenti, con cui è possibile inviare piccole quantità di token senza dover pagare alcun costo di commissione.

Come introdotto in precedenza, i token IOTA sono utilizzati come valuta di scambio tra i nodi del network. Nel dettaglio, questi token rappresentano l'unita di valore che può essere trasferita sulla rete IOTA all'interno di un pacchetto chiamato bundle. Essi assumono importanza in quanto il loro numero è finito e integrato nella rete, e non può essere modificato; inoltre rappresentano l'unica metodologia per effettuare un trasferimento di valore sul network.

IOTA implementa due reti pubbliche di nodi, la Mainnet e la Devnet, ognuna delle quali ha il suo Tangle in cui i nodi possono allegare transazioni. Una transazione può essere definita come una singola istruzione di trasferimento che può ritirare i token IOTA da un indirizzo, depositarli in un altro o avere valore zero (contenere dati, un messaggio o una firma). Se si desidera inviare qualcosa attraverso una rete IOTA, è necessario inoltra ad un nodo come transazione. Questa è composta da 2673 caratteri con codifica tryte e, nel momento della sua decodifica, è possibile individuare i seguenti campi: hash, signatureMessageFragment, address, value, obsoleteTag, timestamp, currentIndex, lastIndex, bundle, trunkTransaction, branchTransaction, attachmentTag, attachmentTimestamp, attachmentTimestampLowerBound, attachmentTimestampUpperBound, nonce. Insieme all'hash del bundle, l'hash della transazione fa parte di ciò che rende immutabile il

Tangle. Se uno qualsiasi dei valori nei campi della transazione dovesse cambiare, l'hash della transazione non sarebbe valido, il che invaliderebbe anche i figli della transazione e qualunque transazione ne faccia riferimento direttamente o indirettamente nel Tangle. Una transazione può essere classificata in uno dei seguenti tre tipi:

- Input transaction, la quale contiene le informazioni per prelevare token IOTA da un indirizzo.
- Output transaction, che definisce l'indirizzo su cui depositare token IOTA.
- Zero-value transaction, la quale ha un valore pari a zero nel campo "value". Questo tipo di transazioni sono estremamente utili per inviare messaggi senza utilizzare token IOTA.

Uno dei concetti chiave appartenenti a IOTA è il Tangle, la struttura dei dati immutabile che contiene una cronologia aggiornata delle transazioni. Tutti i nodi di una rete IOTA memorizzano una copia del Tangle e raggiungono un consenso sul suo contenuto. Per rendere immutabile il Tangle, ogni transazione in esso contenuta è collegata a due transazioni precedenti attraverso gli hash di queste ultime, definite nei suoi campi "branch" e "trunk". Questi riferimenti formano una struttura di dati chiamata Directed Acyclic Graph (DAG), in cui le transazioni a sinistra vengono prima rispetto alle transazioni sulla destra (Figura 1.2).

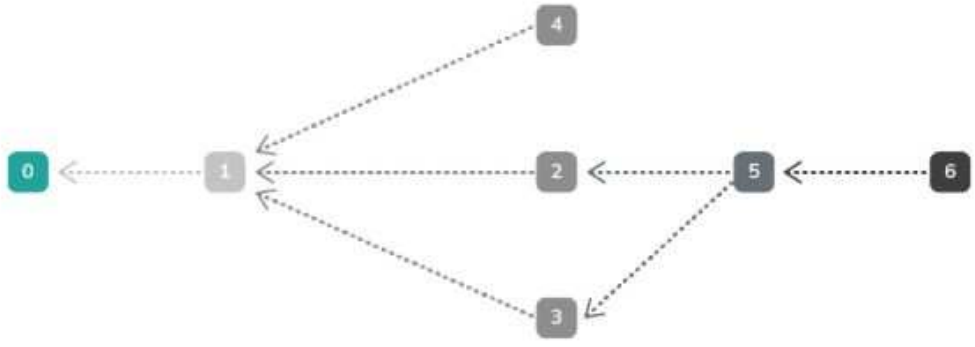

Figura 1.2: Transactions's References [7]

I riferimenti del Tangle possono essere Diretti o Indiretti:

- Diretti, collegano una transazione con quelle definite nei suoi campi banch e trunk. Come si può notare, la transazione numero 5 fa riferimento diretto alle transazioni 2 e 3.

- Indiretti, collegano una transazione con quelle che precedono le transazioni definite nei suoi campi branch e trunk. Per esempio, la transazione 6 è collegata indirettamente alla transazione 3 (tramite la transazione 5).

Questi riferimenti formano la storia di una transazione, per cui se una transazione è figlia, i suoi riferimenti diretti rappresentano i suoi genitori mentre i suoi riferimenti indiretti costituiscono i suoi nonni, e così via. Le transazioni non ancora validate sono quelle transazioni a cui non è associato alcun riferimento. Queste transazioni attendono di essere selezionate dai nodi al fine di essere approvate; a quel punto la loro cronologia viene convalidata e saranno referenziate da una nuova transazione. Se una transazione non è valida, i nodi non la selezioneranno. Di conseguenza, la transazione e la sua cronologia non saranno mai confermate.

É importante evidenziare che in questo ambiente i nodi stessi, essendo interconnessi tra di loro, formano la rete IOTA. Per sincronizzare i loro registri con il resto del network, tutti i nodi inviano e ricevono transazioni reciprocamente. Dopo aver ricevuto una nuova transazione, essi verificano di avere la cronologia della transazione nel loro registro. Se un nodo manca di transazioni, invia richieste di sincronizzazione ai suoi vicini nella rete. In questo modo, tutti i nodi convalidano tutte le transazioni, memorizzandole nella loro copia locale del Tangle.

Per convalidare una transazione, i nodi effettuano una Proof of Work, ovvero un calcolo computazionale che segue un procedimento detto “trial and error” al fine di soddisfare determinati requisiti. Il punto focale della PoW è che essa è difficile da completare, ma semplice da convalidare. Quando viene eseguita la PoW per una transazione, è necessario specificare un Minimum Weight Magnitude (MWM), il quale definisce la quantità di lavoro da eseguire. I nodi accettano un certo weight minimo, a seconda della loro confi-

gurazione. Se viene utilizzato un valore troppo basso, i nodi rifiuteranno la transazione, definendola come non valida.

Enunciata la premessa precedente, è possibile affermare che i nodi costituiscono la Mainnet, il network IOTA che utilizza i token sopra descritti. Gli sviluppatori del protocollo IOTA hanno implementato anche la Devnet, simile alla Mainnet tranne per il fatto che i token sono gratuiti e richiede meno tempo e potenza computazionale per creare e inviare una transazione. Essa è perciò una rete predisposta al testing di applicazioni che si vogliono sviluppare.

IOTA è una piattaforma sicura per la condivisione e l'accesso ai dati, la quale impiega un'unica fonte di verità. In quanto tale, questo protocollo può generare opportunità in molteplici settori, migliorandone l'efficienza, aumentando la produzione e garantendo l'integrità dei dati. Grazie ad esso, IOTA ha trovato applicazione nei campi della mobilitá, del commercio globale, dell'industria IoT, della sanitá e dell'energia.

1.3.2 Masked Authenticated Messaging

IOTA mette a disposizione un'estensione che prende il nome di Masked Authenticated Messaging (MAM), un modulo sperimentale che funge da protocollo di comunicazione dati di secondo livello. Tramite esso è possibile creare canali codificati sul Tangle IOTA. MAM infatti consente di estendere le funzionalità delle transazioni IOTA, sfruttando la crittografia e l'autenticazione dei data stream. Pertanto, gli utenti che utilizzano questa tecnologia possono trasmettere e recuperare flussi di dati crittografati e autenticati che vengono trasmessi attraverso il Tangle come zero-value transaction. Grazie a MAM i messaggi che vengono inseriti in questa rete sono firmati dal proprietario che ha generato l'informazione, permettendone l'autenticazione. IOTA si contraddistingue rispetto agli altri distibuted ledger proprio grazie a MAM che, attraverso le sue proprietà, gli consente di creare flussi di dati e transazioni più economiche, sicure e onnipresenti.

L'attuale implementazione di MAM detiene due peculiarità significative: la post-quantum cryptography e il forward transaction linking. Con la prima si fa riferimento all'implementazione di algoritmi che sono pensati per resistere ad attacchi svolti da com-

puter quantistici. Si ritiene che gli algoritmi crittografici post-quantici siano sicuri contro un attacco elaborato da un computer quantistico sufficientemente potente. Ciò non è vero per molti algoritmi di codifica attualmente utilizzati per cifrare i messaggi che viaggiano su Internet, i quali rivestiranno un'importanza fondamentale per il trasporto di dati sensibili in futuro. Il forward transaction linking, il cui significato in italiano corrisponde letteralmente a "collegamento delle transazioni in avanti", è affine alla struttura dati nota come single-linked list. Sia data una transazione "n", questa conterrà un riferimento (puntatore) alla transazione successiva, n+1. In questo modo le transazioni sono a conoscenza della posizione della transazione successiva, e non di quella precedente (n-1). Grazie a questa caratteristica non è possibile leggere alcun messaggio in un flusso di dati prima del proprio punto di ingresso. Pertanto viene garantita la segretezza in avanti, essendo fornita la possibilità di lettura solo alle transazioni future.

Date queste proprietà, MAM soddisfa un'esigenza importante nei settori in cui integrità e privacy si incontrano.

1.3.3 Canale MAM

MAM rappresenta il cuore di IOTA, nonostante si tratti di un protocollo ancora in fase di sperimentazione. Analogamente a YouTube, MAM utilizza un canale (channel) dove il proprietario pubblica, mentre gli spettatori si iscrivono e visualizzano ciò che viene pubblicato, per ottenere le informazioni disponibili. IOTA permette questa implementazione "owner-subscribers" e ne garantisce la sicurezza tramite l'utilizzo dei seed. Il seed è sostanzialmente una password univoca che permette di dimostrare la proprietà di messaggi e/o dei token IOTA che sono associati ad un indirizzo. Esso è composto da 81 trytes nei quali sono contenute la privacy e tutte le sue proprietà. Se il seed viene comunicato o entra in possesso di qualcun altro, quest'ultimo guadagna la possibilità di pubblicare messaggi sul quel canale. Di conseguenza è buona pratica custodirlo in un luogo sicuro e non esporlo per alcun motivo.

MAM mette a disposizione tre tipi di opzioni che possono essere scelte dagli editori nel momento della pubblicazione di un nuovo messaggio sul proprio canale:

- Pubblica, tramite cui chiunque può vedere quello che viene pubblicato.
- Privata, attraverso cui solo il proprietario può vedere ciò che pubblica.
- Ristretta, in cui è possibile specificare chi può visualizzare quello che viene pubblicato, fornendo loro una chiave. Questa chiave prende il nome di sideKey.

I messaggi MAM vengono inviati nel campo signatureMessageFragment delle zero-value transaction. Quando viene pubblicato un messaggio su qualsiasi canale, indipendentemente dalla modalità utilizzata, si ottiene un ID, il quale funge da identificatore, in modo tale da consentire ad altri di iscriversi e recuperare i messaggi sul Tangle. L'ID del canale rappresenta l'indirizzo della transazione che contiene il messaggio MAM, e prende anche il nome di root.

Nel protocollo IOTA, come per molti altri, gli utenti possono allegare messaggi arbitrari alle transazioni ma, come anticipato precedentemente, questo ambiente è privo di commissioni. Tuttavia, al mittente è consentito allegare un solo messaggio per volta. Non è perciò possibile pubblicare più messaggi correlati consecutivi in un contesto arbitrario. Prendendo in considerazione l'esempio riportato da ABmushi nel suo articolo relativo a MAM [9], viene esaminato il caso d'uso riguardante la pubblicazione di dati inerenti alla temperatura ambientale, con cadenza regolare fissata a 15 minuti, senza l'utilizzo del protocollo MAM. Data questa premessa, sarebbe necessario pubblicare ogni messaggio sullo stesso indirizzo. Siccome un distributed ledger, incluso il Tangle, è pubblicamente accessibile, è facile per un malintenzionato identificare tale indirizzo, in quanto ha una frequenza di aggiornamento fissata a 15 minuti, e interferire con transazioni spam. Una possibile soluzione potrebbe essere quella di cambiare indirizzo ogni volta che si pubblicano nuovi dati, ma insorge il problema di dover tenere traccia di tutti gli addresses utilizzati, dovendo così sostenere un costo relativo all'osservazione e alla memorizzazione di queste informazioni. Ragion per cui si evidenzia l'utilità di MAM, che grazie al suo design "Message Chain", permette di proteggere il canale da ogni tipo di interferenza spam e lascia liberi gli utenti dal dover gestire tutti quanti gli indirizzi. Difatti MAM si occupa di pubblicare ogni messaggio ad un indirizzo differente, con informazioni detta-

gliate che li collegano. In questa catena di messaggi, da una generazione alla prossima, i messaggi più vecchi guidano verso i più recenti, tramite un flusso che fluisce a senso unico.

É importante sottolineare che tutti i canali MAM vengono generati da un Merkle tree (detto anche hash tree), dove tutti i messaggi vengono firmati con una delle chiavi private che risiedono nelle foglie di questa struttura. Questo sistema viene istanziato tramite una funzione che prende in input il seed e altri parametri relativi alle caratteristiche dell'albero. In crittografia, questa struttura rappresenta un albero in cui ogni nodo foglia è contrassegnato con l'hash crittografico di un blocco di dati, e ogni nodo non foglia è definito con l'hash dei suoi nodi figli. I Merkle tree consentono di verificare efficacemente e in maniera sicura il contenuto di grandi strutture di dati. [10] Essi possono essere utilizzati per verificare qualsiasi tipo di dato archiviato, gestito e trasferito all'interno e tra i computer. Principalmente, garantiscono che i blocchi di dati ottenuti da altri peer, in una rete peer-to-peer, siano ricevuti integri e inalterati, e verificano che gli altri nodi non mentano inviando blocchi falsi.

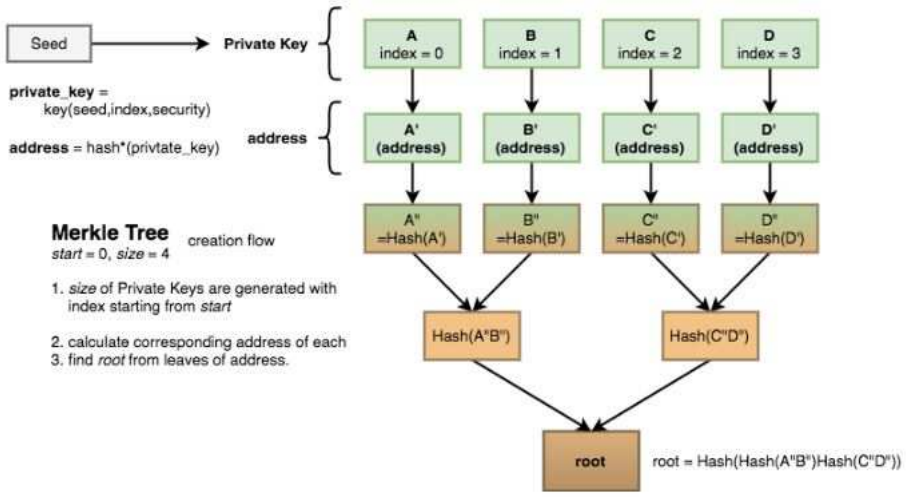

Figura 1.3: Struttura Merkle tree [9]

Successivamente alla creazione del canale è possibile iniziare ad effettuare operazioni

di caricamento di messaggi su di esso. Se in fase di pubblicazione viene utilizzata la modalità pubblica, il messaggio sarà pubblicato all'indirizzo indicato dalla root, ed essa rappresenterà anche la chiave di cifratura e decriptazione per accedere alle informazioni. Se invece si utilizza la modalità privata, si implementa un ulteriore livello di sicurezza per gli utenti non autorizzati, dato dall'hash della root come indirizzo della transazione che contiene il messaggio MAM. Come risultato si otterrà che solo coloro che conoscono la root possono decifrare il contenuto del messaggio, calcolando l'hash di questa come chiave di decifratura. A tal proposito è necessario comunicare in maniera sicura la root alle entità iscritte al canale, in modo da consentire loro l'accesso ai messaggi. Infine, se si fa uso della modalità ristretta, viene aggiunta una chiave di autorizzazione, la sideKey, alla modalità privata. L'indirizzo della transazione che contiene il messaggio MAM è dato dall'hash congiunto della sideKey e della root. Di conseguenza, solo coloro che conoscono questi due valori possono decifrare i messaggi. Tale modalità è utile per poter revocare l'accesso al canale cambiando semplicemente la chiave. In questa circostanza è sufficiente modificare la sideKey e distribuirla alle parti che si decide di autorizzare a seguire il resto del canale.

Indistintamente da come venga effettuata la pubblicazione, i messaggi che vengono trasmessi sullo stesso canale sono collegati secondo un ordine cronologico, che segue un flusso in avanti: è garantita la forward security in quanto non è possibile visionare le informazioni che risalgono cronologicamente ad un tempo precedente rispetto al punto di ingresso in quello stream.

1.3.4 Struttura dei messaggi MAM

Il pacchetto che compone MAM può essere suddiviso in due sezioni principali: la Signature section e la MAM section. La prima ricopre un ruolo di vidimatore sulla seconda e si occupa di registrare i dati come "signatureFragment" delle transazioni, mentre la MAM section gestisce la memorizzazione dell'effettivo messaggio mascherato. Analizzando il contenuto di quest'ultima è possibile identificare un numero intero, chiamato branch_index, che viene utilizzato in fase di validazione del messaggio, la root in cui verrà

pubblicato il messaggio successivo di quel canale, i "fratelli" del branch_index e una stringa di codice ASCII di lunghezza arbitraria che corrisponde al testo del messaggio che l'editore vuole pubblicare. Questa però dovrà essere convertita in trytes prima di essere pubblicata in quanto l'architettura IOTA utilizza il sistema ternario precedentemente descritto.

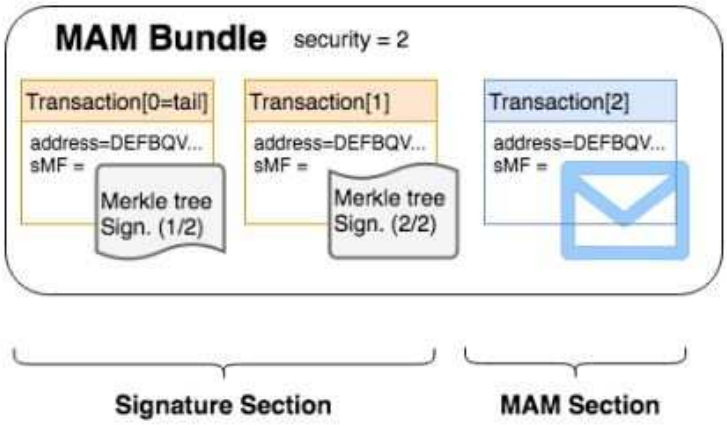

Figura 1.4: MAM Bundle [9]

In fase di pubblicazione la MAM section viene criptata e assume il nome di messageTrytes. Infine viene generata una firma a partire dal seed e dal brach_index utilizzati, che permette di firmare il messageTryte e ottenere la Signature section.

Al momento del recupero dei dati, è necessario avere a disposizione la root e/o la sideKey a seconda della modalità di pubblicazione sul canale. Per prima cosa è necessario individuare l'indirizzo in cui è memorizzato il messaggio. Una volta rintracciato è necessario decifrare il messageTryte, ottenendo così i campi descritti sopra. A questo punto ha luogo la fase di convalida del messaggio smascherato: viene effettuato un processo di vidima del messageTryte che come output restituisce un indirizzo. Questo rappresenta la foglia del Merkle tree che è stata utilizzata per la creazione della root su cui è stato pubblicato il messaggio. Tramite la combinazione di questo valore e dei suoi fratelli, ottenuti con la decrittazione del messageTryte, è possibile calcolare la radice di un Merkle tree, chiamata temp-root. Se questo valore è equivalente al root con cui

sono state recuperate le informazioni, il messaggio non mascherato analizzato è valido per quel canale. In caso contrario significa che il messaggio non è stato pubblicato dal proprietario del channel.

MAM è uno dei moduli più potenti di IOTA e apre un nuovo campo di casi d'uso da costruire su di esso. Essere in grado di proteggere l'integrità dei dati e controllarne la gestione degli accessi è un prerequisito per attività come Data Marketplaces, Fog Analytics, assicurazioni automatizzate, gestione dati medicali e molto altro ancora.

1.4 InterPlanetary File System

L'InterPlanetary File System (IPFS) è un file system distribuito che cerca di connettere tutti i dispositivi informatici con lo stesso sistema di file. Se un file system indica un meccanismo con il quale i file sono posizionati e organizzati su dispositivi informatici utilizzati per l'archiviazione dei dati, un file system distribuito è un particolare file system che permette la memorizzazione di file e risorse in dispositivi di archiviazione distribuiti in una rete informatica, anziché letti o archiviati in maniera centralizzata su di un dispositivo locale. In un certo senso, l'obiettivo perseguito da questa tecnologia è simile a quello originale del Web, ma IPFS è in realtà più simile ad un Bittorrent che scambia degli oggetti. Questo protocollo potrebbe diventare un nuovo importante sottosistema di Internet. Se costruito correttamente, potrebbe anche integrare o sostituire HTTP. [11]

Per comprendere meglio il funzionamento di IPFS si introduce un esempio: si supponga che si voglia effettuare una ricerca su IOTA. In prima istanza si potrebbe visitare la pagina Wikipedia relativa a questa tecnologia. Inserendo l'URL relativo ad essa nella barra degli indirizzi del browser, il computer richiede ad uno dei calcolatori di Wikipedia, che potrebbe risiedere in qualsiasi luogo nel pianeta, la pagina relativa a IOTA. Tuttavia, questa non rappresenta l'unica possibilità di recuperare le informazioni di cui si ha bisogno. Memorizzato su IPFS vi è un mirror di Wikipedia che può essere utilizzato anziché sfruttare il metodo classico. Se si fa uso di IPFS, il computer richiederà la pagina di IOTA in questo modo:

`/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/IOTA.html`

IPFS sa come recuperare le informazioni riguardo a questa tecnologia dal contenuto della pagina, e non dalla sua posizione. Il documento IPFS che possiede le informazioni ricercate è rappresentato dalla stringa di numeri che si trova nel mezzo dell'indirizzo (QmXo...). Usando questo protocollo, diversamente dal richiedere ad uno dei computer di Wikipedia la pagina, l'elaboratore utilizza IPFS per domandare ai computer in giro per il mondo che impiegano questa struttura di condividere la pagina con lui. É possibile recuperare le informazioni riguardo ai IOTA da chiunque le possegga, non solo da Wikipedia. Nel momento in cui si utilizza IPFS non viene semplicemente scaricato il file richiesto da qualcuno: il computer aiuta la distribuzione del documento. Quando un nodo della rete necessita della stessa pagina, potrebbe avere la possibilità di ottenerla dall'elaboratore che in precedenza ne aveva fatto richiesta, ma ha la medesima probabilità di riceverla da un peer limitrofo o da chiunque altro usi IPFS. Tale tecnologia rende possibile quanto descritto non soltanto per pagine web, ma anche per qualsiasi altro tipo di file che un computer potrebbe archiviare, che si tratti di un documento, un'email, o persino un record di un database.

IPFS assume importanza in quanto rendere possibile il download di un file da più posizioni che non siano gestite da un organo centrale:

- Sostiene una rete resistente. Se qualcuno decidesse di attaccare i server di Wikipedia sarebbe comunque possibile recuperare le webpages da altre parti.

- Complica la censura dei contenuti. Siccome i file su IPFS sono decentralizzati, e quindi possono trovarsi in diversi luoghi, è più difficile per Stati, corporazioni o qualcun altro ente, bloccare le informazioni. Nel 2017, la Turchia ha bloccato Wikipedia mentre la Spagna bloccò l'accesso al sito relativo al movimento indipendente Catalano. Si spera che IPFS possa aiutare a fornire un modo per circumnavigare azioni come queste quando avvengono.

- Possono velocizzare la rete quando si è lontani o disconnessi. Nell'eventualità in cui si riesca a recuperare un file da qualcuno che si trova nelle vicinanze, anziché

a centinaia o migliaia di chilometri di distanza, è possibile ottenere ciò di cui si ha bisogno più velocemente. Questo è particolarmente utile se la community è collegata in rete a livello locale, ma non possiede una buona connessione sulla rete estesa di Internet.

Su questo ultimo punto è dove IPFS prende il nome di InterPlanetary File System. L'organizzazione madre si sta adoperando per costruire un sistema che funzioni attraverso luoghi distaccati o distanti "quanto i pianeti". Per comprenderne il funzionamento si introduce il concetto di CID.

L'insieme di caratteri che si trovano dopo 'ipfs', nell'indirizzo riportato sopra, prende il nome di "content identifier", detto anche CID, mediante il quale IPFS può recuperare il contenuto da luoghi differenti. Di norma un URL identifica un file in base a dove si trova, ovvero alla posizione in cui è allocato sul pc. Tale concetto non funziona se il file è presente in più luoghi. IPFS, come già anticipato, non è basato sulla posizione fisica del file, bensì indirizza il documento in base al suo contenuto. Il content identifier è ottenuto da una funzione hash crittografica del contenuto di quell'indirizzo. L'hash è univoco rispetto al contenuto da cui viene generato. Esso permette di verificare che quanto ricevuto sia ciò che è stato effettivamente richiesto. Un singolo indirizzo IPFS può fare riferimento ai metadata di una singola parte di un file, ad un intero documento, ad una directory, ad un sito web completo o a qualsiasi altro tipo di contenuto. Dacché l'indirizzo di un file IPFS è creato dal contenuto stesso del file, i link IPFS non possono essere cambiati. Se, per esempio, il testo di una pagina web venisse modificato, la nuova versione generata otterrebbe un nuovo, diverso, indirizzo IPFS. Inoltre il contenuto non può essere spostato ad un indirizzo differente. Il link generato precedentemente punterà ancora allo scorso oggetto. Un'altra caratteristica che contraddistingue IPFS permette di evitare che il contenuto di un documento sia rimosso dall'architettura finché vi è qualcuno interessato abbastanza da garantirne la disponibilità, che sia esso l'autore originale del file o chiunque altro.

Sebbene sia presente molta tecnologia complessa in IPFS, le idee fondamentali che lo identificano riguardano il modo in cui comunicano le reti di persone e di computer. Il

World Wide Web di oggi è strutturato sulla base di proprietà e di accesso, il che sta a significare che i file sono forniti da chiunque li possieda, se essi determinano opportuno concederne l'accesso. IPFS, d'altro canto, si basa sul concetto di possesso e partecipazione, in cui molti elementi possiedono file condivisi, e tutti partecipano per garantire la disponibilità. Pertanto IPFS funziona in maniera appropriata solo quando i nodi concorrono attivamente sulla rete: se viene utilizzato un dispositivo per condividere file utilizzando IPFS ma successivamente lo si spegne, gli altri nodi non saranno in grado di ottenere file da quel device. Sarebbe quindi ragionevole archiviare le copie dei documenti in un computer che mantiene la sua esecuzione attiva ed esegue IPFS, in modo da rendere quei file disponibili per tutti coloro che li desiderano.

1.4.1 Come lavora IPFS

IPFS è una rete peer-to-peer (p2p) dedicata allo storage di file. Il contenuto dei documenti è accessibile attraverso i peer, i quali possono trasmettere o memorizzare informazioni (o fare entrambe le cose). Tali nodi possono essere localizzati in qualsiasi parte del globo. IPFS riesce a rintracciare ciò che gli si chiede grazie al contenuto dell'indirizzo. Il protocollo analizzato ha tre aspetti fondamentali da conoscere.

- Content addressing and linked data: IPFS utilizza il content addressing per identificare il contenuto di un documento in base a ciò che contiene, piuttosto che al luogo in cui si trova. La ricerca di un elemento in base al contenuto è una pratica che si applica in continuazione. Basti pensare alla ricerca di un libro in biblioteca; solitamente lo si richiede tramite il suo titolo. Quanto descritto corrisponde esattamente al content addressing, in quanto si effettua una richiesta del libro in base a cosa tratta, ovvero al suo contenuto. Se invece si utilizzasse una ricerca basata sulla posizione in cui esso si trova, verrebbe effettuata una richiesta simile alla seguente: "Voglio il libro che si trova al secondo piano, prima fila, terzo scaffale dal basso, quattro libri da sinistra". Nel caso in cui qualcuno spostasse il libro non sarebbe più possibile trovarlo. Attualmente quanto descritto è ciò che accade

su Internet e sui computer. Al contrario, ogni contenuto che utilizza il protocollo IPFS ha un content identifier, detto anche CID, che corrisponde al suo hash. L'hash è univoco al contenuto da cui è generato. Il content addressing attraverso l'hash è diventato un mezzo ampiamente utilizzato per connettere i dati nei sistemi distribuiti. Tuttavia, le strutture di dati sottostanti a questi sistemi non necessariamente garantiscono l'interoperabilità. Il resto dell'ecosistema IPFS si basa su questi concetti fondamentali.

- Conversione dei file in DAG: IPFS e molti altri sistemi distribuiti si avvalgono di una struttura dati chiamata Directed Acyclic Graphs, o DAG. In particolare, esso utilizza i Merkle-DAG, i quali rappresentano DAG in cui ogni nodo ha un identificatore che corrisponde all'hash del suo contenuto. IPFS utilizza questa struttura ottimizzata per la rappresentazione di directory e file, ma è possibile strutturare i Merkle-DAG in molti modi differenti. Git, per esempio, fa uso di questa architettura per avere in memoria diverse versioni del repo su cui si sta lavorando. Al fine di comporre una rappresentazione Merkle-DAG del contenuto di un nodo, IPFS effettua preventivamente una divisione in blocchi di quest'ultimo. Da tale comportamento deriva che diverse parti di un file possano provenire da fonti differenti, ed essere autenticate rapidamente. Va sottolineato quindi che ogni elemento ha un CID: ai file è associato un CID; una cartella contenente diversi documenti possiede un CID ed esso contiene i CID dei file all'interno di essa. A loro volta, quei file potrebbero essere costituiti da diversi blocchi, e ognuna di queste parti avrebbe un CID. Infine, un'altra caratteristica funzionale dei Merkle-DAG e della suddivisione del contenuto in blocchi è data dalla condivisione di parti dell'albero in presenza di file simili; quindi, parti del Merkle-DAG diverse possono fare riferimento agli stessi dati. Ponendo come caso d'uso l'aggiornamento di un sito web, solamente i file che sono stati sottoposti a modifiche otterranno nuovi indirizzi. La versione precedente e quella aggiornata possono fare riferimento agli stessi blocchi per tutto ciò che è rimasto invariato. Questa peculiarità può rendere più efficiente il trasferimento di versioni di grandi insiemi di dati (come la ricerca

genomica) grazie alla necessità di dover trasferire soltanto le sezioni nuove o quelle sottoposte a modifiche, diversamente dal dover creare file completamente nuovi.

- La DHT: per individuare i nodi che ospitano il contenuto che si sta cercando, IPFS utilizza una Distributed Hash Table, detta anche DHT. Una hash table può essere descritta come un database composto da coppie chiave-valore. Una hash table distribuita è una HT in cui la tabella è divisa tra tutti i peer che compongono una rete distribuita. Per individuare il contenuto ricercato è necessario interrogare i peers. Nel momento in cui si identificano i nodi su cui è memorizzato ciò che sto cercando, (ovvero si rintracciano il peer o i peers che hanno registrato i blocchi che compongono il contenuto ricercato), si fa uso della DHT per individuare la posizione corrente di quei peer. A questo punto non rimane che ottenere il contenuto, utilizzando un modulo chiamato Bitswap. Bitswap permette di connettersi ai peers che possiedono le informazioni che si desiderano, inviando loro una wantlist, ovvero una lista dei blocchi di cui si necessita, e aspettare di ricevere gli oggetti richiesti. Al momento della ricezione, è possibile verificarli effettuando l'hashing del loro contenuto, in modo da ottenere i CID ed effettuare una comparazione.

L'ecosistema IPFS è costituito da diverse librerie modulari che gestiscono parti specifiche di qualsiasi sistema distribuito. É possibile utilizzare qualsiasi parte dello stack in modo indipendente, o combinarle tra di loro in maniera innovativa.

Capitolo 2

Fase di Progettazione

In questo capitolo viene effettuata un'analisi dell'ecosistema attuale, relativo alla gestione delle Protected Health Information e delle nuove tecnologie che possono portare ad applicazioni innovative in questo ambiente. Viene quindi evidenziato lo scopo che si persegue e gli strumenti utilizzati per poterlo realizzare. Si descrive il percorso che si è seguito per poter ottenere un'infrastruttura che garantisce sicurezza e stabilità, analizzando lo studio effettuato per riuscire a portare a termine il progetto. Per questo motivo verranno analizzate le librerie JavaScript relative a IOTA, MAM e IPFS, tramite le quali sono state effettuate operazioni di testing, per poi impiegarle nella creazione di in un singolo programma funzionante.

Successivamente alla sessione di test, utile a prendere dimestichezza con le tecnologie e a comprenderne il funzionamento, è stato possibile passare allo sviluppo dell'applicazione.

Grazie a questa fase si è potuto comprendere il funzionamento dei protocolli e implementare alcune delle loro funzionalità utilizzando i componenti messi a disposizione dalle relative API JavaScript. In questo modo si è potuto realizzare un singolo sistema che permette di gestire autonomamente il caricamento dei dati sanitari, immagazzinati in file JSON e conformi alla normativa GDPR/FHIR, sulla piattaforma IPFS, impiegata principalmente per minimizzare il reversal risk e il linkability risk, ottenendo un hash univoco tramite cui è possibile risalire ad essi. Successivamente, si utilizza questo hash come contenuto di un messaggio da caricare su un canale MAM di tipo ristretto. In tal

modo si ottiene una protezione dei dati multipla garantita dall'anonimato, dalle funzioni crittografiche implementate da IPFS e dalle proprietà intrinseche dei canali MAM. L'unica cosa di cui deve occuparsi l'utente-paziente è quella di definire il seed e la sideKey con cui si vuole creare il canale MAM in modo tale da possedere le "Credenziali" per poter risalire alla catena di informazioni salvata in maniera permanente e sicura sul Tangle di IOTA.

Nel momento in cui il soggetto sarà sottoposto ad una visita potrà comunicare al medico curante o, a chi di dovere, le credenziali di accesso al MAM channel in maniera che anch'esso possa accedere al registro che si è creato nel tempo contenente i dati raccolti. Un altro caso d'uso applicabile potrebbe essere quello che vede il medico come l'attore che crea il canale MAM, fornendo poi le credenziali al paziente, il quale potrà utilizzarle per effettuare il caricamento dei dati su di esso, stabilendo un contatto continuo con l'ambiente sanitario.

Questo modulo relativo al caricamento dei dati è parte di un progetto più vasto che potrebbe comprendere l'effettiva raccolta dati da parte di un dispositivo bluetooth che si occupa di monitorare i parametri vitali di un individuo, quali frequenza cardiaca o glicemia, li elabora in maniera da garantirne l'anonimato e, tramite un'apposita app sviluppata per dispositivi mobile, permetta all'utente di definire le credenziali di accesso per la creazione del canale Mam tramite una UI che ne facilita l'utilizzo, in modo da definire i parametri che creeranno il channel su cui i dati verranno caricati, dove entra in gioco il programma sviluppato in questa tesi che permette l'effettivo caricamento delle informazioni sulle varie piattaforme.

Figura 2.1: Scambio di dati glicemici basato su DLTs [1]

2.1 Metodo adottato

Come è stato osservato nel capitolo precedente, lo scopo perseguito è quello di sviluppare un sistema che sia immutabile, che garantisca l'interoperabilità e che sia conforme al regolamento generale sulla protezione dei dati (GDPR) per lo scambio di informazioni di carattere sanitario. Grazie allo sviluppo delle DLTs e al loro modo di gestire le informazioni, è possibile realizzare nuove metodologie per la manipolazione e la gestione di questi dati.

Il caso d'uso sotto esame è quello di un sistema per lo scambio dei dati medicali che sia controllato dal paziente, completamente digitalizzato, interoperabile e gestito in maniera distribuita. Il modulo principale su cui si è lavorato è quello relativo all'archiviazione e alla registrazione dei dati, il quale si occupa di caricare i record anonimizzati FHIR su una DLT, tralasciando l'aspetto relativo alla raccolta dati. L'uso dello standard FHIR per i progetti relativi alla gestione di dati personali di carattere medico dovrebbe consentire una semplice integrazione dei dati presenti nelle DLTs all'interno degli electronic health records (EHR) di un medico. Un esempio relativo a questa tematica è burstIQ [19], una piattaforma conforme ad HIPAA, GDPR ed FHIR che consente la creazione di profili sanitari per persone e luoghi, consentendo le interazioni tra di loro. La variante relativa al progetto realizzato mira ad essere conforme al GDPR e tenta di minimizzare il rischio di inversione e di collegamento, con i suoi vantaggi e svantaggi: essa utilizza il protocollo IOTA e la sua estensione MAM in combinazione al file system distribuito IPFS.

É opportuno premettere che i dati manipolati sono stati creati sulla base del file contenuto nell'articolo [1], quindi già in formato FHIR e privi di ogni informazione sulla salute personale, non generati da un sensore fisico. In questo progetto tutti i record FHIR sono stati archiviati su un file system distribuito (IPFS), come spiegato in seguito, e gli hash risultanti registrati nel Tangle di IOTA. Il protocollo IOTA è stato utilizzato perché offre la possibilità di effettuare transazione a costo zero, e gestisce al meglio grandi volumi di messaggi. Tali proprietà rendono questa tecnologia particolarmente adatta allo scambio di dati tra dispositivi e sistemi sanitari.

Ciò che differenzia IOTA dalle DLTs basate su blockchain è che le transazioni, invece di essere raggruppate in blocchi e archiviate in catene sequenziali, sono collegate insieme nel Tangle. Pertanto, le transazioni possono essere emesse contemporaneamente, in modo sincrono e continuo. Inoltre, per condividere, archiviare e recuperare dati criptati, è stato utilizzato il modulo MAM. Questo protocollo consente di criptare i messaggi, confermare l'origine della fonte e creare un data-stream continuo sul Tangle fino a quando non si decide di terminare la pubblicazione di messaggi. É importante ricordare che in un canale MAM, ogni messaggio contiene dati, un riferimento all'indirizzo del messaggio successivo (che scorre solo in avanti) e una firma che permette di verificare l'identità del creatore di quel messaggio. Siccome vengono creati ID univoci per ogni canale (noti come root), solo le parti autorizzate sono in grado di leggere e ricostruire l'intero flusso di messaggi. Per l'approvazione di una transazione inviata al Tangle tramite MAM, viene effettuata una semplice Proof of Work (PoW). Il calcolo di questo piccolo puzzle computazione richiede l'allocazione di poche risorse, rendendolo ben progettato per dispositivi di piccole dimensioni, come gli smartphone. Sarebbe quindi di facile implementazione un ambiente che si interfacci con uno di questi dispositivi, svolgendo le operazioni necessarie in continuazione, senza dover obbligatoriamente disporre di un pc.

Oltre a queste tecnologie è stato incluso anche IPFS per tenere conto di due importanti considerazioni normative: l'incertezza riguardo a quali tecniche di anonimizzazione sono legalmente sufficienti per trasformare i dati medicali in dati anonimizzati e per ridurre al minimo sia i rischi di inversione che di collegamento ad un individuo. Ricapitolando, IPFS è un file system distribuito peer-to-peer basato su collegamenti ipertestuali indirizzati al contenuto. Come tale, gestisce i file in base al loro contenuto, memorizzandoli e monitorandoli utilizzando un DAG. Questi alberi consentono la verifica sicura dei contenuti di grandi strutture dati, utilizzando funzioni hash crittografiche che mappano informazioni di dimensioni arbitrarie in dati di dimensioni fisse (hash). Il vantaggio principale per cui si è fatto uso di questa tecnologia è che i dati archiviati non vengono automaticamente distribuiti tra tutti i partecipanti, ma vengono condivisi solo in caso di richiesta. Inoltre, implementa un meccanismo di sicurezza per cui è computazionalmente

semplice definire l'hash avendo un determinato input, ma è incredibilmente difficile, se non impossibile, derivare l'input disponendo di un hash.

Per collegare le transazioni IPFS alle transazioni MAM autenticate e non cancellabili, gli hash ottenuti dal caricamento dei file su IPFS sono stati caricati all'interno di un canale MAM utilizzando le librerie IOTA. Come risultato finale si viene a creare un'infrastruttura con cui è possibile gestire i dati medicali in maniera sicura, evitando il rischio di risalire al proprietario dei dati e fornendo una profilazione dettagliata del paziente, in modo da poter consentire in secondo luogo l'accesso alle informazioni al medico curante, il quale potrà avere un quadro clinico dettagliato sull'utente, fornendo assistenza sanitaria di alta qualità.

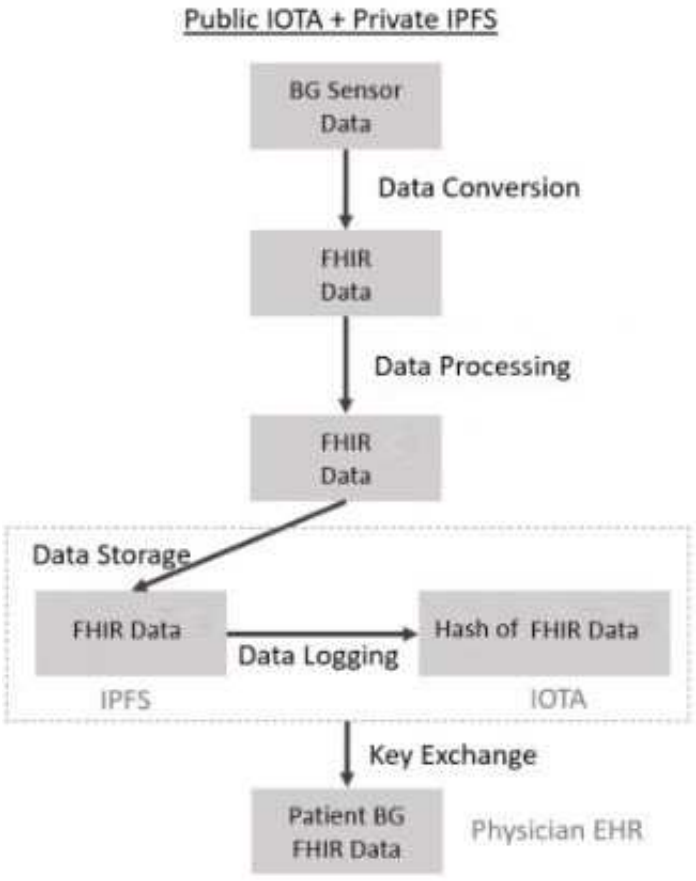

Figura 2.2: Architettura software [1]

2.1.1 Gestione dei dati medicali

Per quanto concerne la gestione delle informazioni personali in ambito sanitario è necessario fare riferimento al GDPR, regolamento europeo che definisce le modalità di gestione e protezione dei dati privati. Esso definisce anche i principali diritti della persona cui i dati fanno riferimento, ma non menziona quali tecniche sono sufficienti per anonimizzare i dati personali per cui l'output risultante può potenzialmente essere archiviato in una DLT. Ciò che si qualifica come anonimo, quindi, non è ancora chiaro. L'unica indicazione di cui si dispone è che deve essere irreversibilmente impossibile identificare un individuo attraverso uno dei mezzi che possono essere utilizzati per il salvataggio dei dati.

In questa fase, la guida fornita dal GDPR su come trattare i dati personali si riferisce semplicemente alla necessità di minimizzare sia il rischio di inversione sia il rischio di collegamento delle informazioni ad un individuo. Volendo entrare nel merito di queste problematiche, la prima identifica il pericolo di poter ricostituire i dati originali dai dati anonimizzati. Il reverse engineering dell'hash (ottenuto da IPFS) al fine di ottenere il file originale è, dal punto di vista computazionale, un compito enorme poiché richiede di testare l'immenso numero di combinazioni possibili di input per poter riuscire ad individuare quella corretta. La seconda, invece, rappresenta il rischio per cui sia possibile collegare le informazioni ad un individuo esaminando i modelli di utilizzo o di contesto. In questo caso ogni hash è univoco, quindi non vi è un metodo ovvio per analizzare in modo incrociato i dati e determinare a chi appartengono.

Lo studio da cui deriva questo progetto prende in considerazione un sensore che raccoglie dati relativi alla glicemia di un paziente. I parametri grezzi raccolti vengono trattati in modo da essere conformi allo standard FHIR, con lo scopo di facilitare l'interoperabilità dei dati relativi all'assistenza sanitaria tra sistemi diversi. Da ogni misurazione del sensore ne deriva un file in formato JSON contenente dati nello standard FHIR, da cui vengono rimosse tutte le informazioni personali per garantire l'anonimato.

In presenza di un sensore che genera dati a cadenza regolare può essere opportuno effettuare un'operazione di aggregazione dei dati, che consiste nel combinare più file JSON sotto un unico file di dimensioni maggiori, sfruttando a pieno le capacità delle reti. In

questo caso tuttavia si sono gestiti i file singolarmente, senza procedere all'aggregazione dei dati, in modo da poter effettuare al operazioni di testing in modo più agevole. Entrambe le varianti mirano a essere conformi al GDPR e tentano di minimizzare il rischio di inversione e di collegamento, con i loro vantaggi e svantaggi. Per concludere si mostra al lettore il formato di dati anonimizzati GDPR/FHIR che si è preso come riferimento per generare campioni di informazioni su cui effettuare i test per lo sviluppo del progetto.


```
{
  "resourceType": "Observation",
  "id": "
     PISZOILIUOYNUHZZH55PKSYWTNEVTEKXGMIIPTP5X5NWNHFWF599APCHSYNJILVGH"
     ,
  "meta": {
    "versionId": "
       PISZOIUUOYNUHNH55PKSYWTNEVTEKXGMIIPTP5X5NWNHFWF599APCHSYWILVGH",
    "lastUpdated": "2019-01-24T12:45:43.7004651+00:00"
  },
  "identifier": [
    {
      "use": "official",
      "system": "http://www.bmc.nl/zorgportal/identifiers/observations"
         ,
      "value": "6323"
    }
  ],
  "status": "final",
  "code": {
    "coding": [
      {
        "system": "http://loinc.org",
        "code": "15074-8",
        "display": "Glucose [Moles/volume] in Blood"
      }
    ]
  },
```

```
  "effectivePeriod": {
    "start": "2019-04-01709:30:10+01:00"
  },
  "issued": "2019-04-01T15:30:10+01:00",
  "valueQuantity": {
    "value": "6.10",
    "unit": "mmo1/1",
    "system": "http://unitsofmeasure.org",
    "code": "mmol/L"
  }
}
```

Listing 2.1: Dati conformi GDPR/FHIR

Capitolo 3

Sviluppo del Programma

Una volta chiari gli obiettivi da raggiungere si è presa familiarità con le tecnologie relative ad IPFS e MAM, per poi passare alla realizzazione di un programma che permetta di gestire autonomamente i due protocolli. In questo capitolo si introducono le tecnologie che sono state testate al fine di poter sviluppare il progetto, nonché la struttura realizzata, che raggruppa in un unico sistema il caricamento dei dati medicali su IPFS e l'upload dei relativi hash sul Tangle di IOTA. Si fa riferimento a JavaScript come linguaggio utilizzato per realizzare il programma e, in via preventiva, per svolgere un'attività di approfondimento sul funzionamento dei vari protocolli. A tal fine si è consultato il codice nelle relative pagine di Github [15, 16, 17].

3.1 Tecnologie Utilizzate

3.1.1 Librerie API JavaScript di IPFS

Per poter accedere a tutte le funzionalità messe a disposizione da IPFS è necessario scaricare sul proprio computer i pacchetti ad esso relativi. Recandosi sulla pagina Web della documentazione ufficiale di IPFS si può individuare la sezione riguardante l'installazione della piattaforma. Una volta che il download sarà completato, è necessario aggiungere alle variabili d'ambiente del sistema operativo (in questo caso Windows) l'e-

seguibile di IPFS, in modo da abilitarne le funzionalità. Completato quanto decritto, sarà necessario istanziare una repository in cui IPFS mantenga tutte le impostazioni e i dati interni come suo contenuto. Quindi, all'interno della cartella che si vuole allocare al protocollo, si impartisce il comando da cmd `ipfs init`. Si otterrà come risposta un valore `peer identity` seguito da una serie di caratteri. Questo rappresenta l'hash che corrisponde all'identificatore del nodo utilizzato. Altri nodi sulla rete IPFS utilizzeranno questo valore per identificarlo e connettersi ad esso. Se si è interessati a recuperare l'id relativo al nodo su cui si sta lavorando è necessario richiamare da linea di comando il codice `ipfs id`. Attualmente la configurazione iniziale è terminata ed è possibile unire il nodo al network pubblico. Pertanto viene eseguito il comando `ipfs daemon` che permette di avviare un "demone", ovvero un programma eseguito in bakground, che consente di mantenere la connessione del nodo sulla rete. A questo punto si dovrebbe essere in grado di ottenere oggetti dal network. É possibile eseguire il comando:

```
ipfs cat /ipfs/QmW2WQi7j6c7UgJTarActp7tDNikE4B2qXtFCfLPdsgaTQ
/cat.jpg >cat.jpg
```

per scaricare l'immagine di un gatto sul proprio nodo, in modo da verificare che tutto stia funzionando. Il comando `cat`, infatti, svolge la funzione di recuperare gli oggetti dalla rete. L'immagine è stata ripescata dal nodo indicato dall'hash riportato tra le voci `ipfs` e `cat`. Non rimane che testare l'invio di oggetti sul network. A tal proposito si effettua un test di caricamento di un sample di dati sanitari. Come si può notare dalla Figura 3.1, l'invio sulla rete del file relativo alle informazioni di carattere personale viene effettuato dal comando `add` che, una volta completato l'upload, restituisce un hash relativo al file, in modo che questo sia individuabile in maniera univoca sulla rete. Per recuperare e visualizzare il documento non resta altro che utilizzare il comando analizzato in precedenza, `cat`.

Quanto svolto può essere così descritto: il gateway ha pubblicato un file dal computer. In fase di recupero del documento, il gateway ha interrogato la tabella hash distribuita (DHT) individuando il computer che ha effettuato la richiesta, ha identificato il file che

si vuole recuperare, la macchina che lo possiede e lo ha inviato al gateway, il quale lo ha restituito per la visualizzazione. Per fornire una User Interface e semplificare alcune procedure, gli sviluppatori hanno creato IPFS Companion, un'estensione browser che semplifica l'accesso alle risorse e aggiunge il supporto al protocollo IPFS.

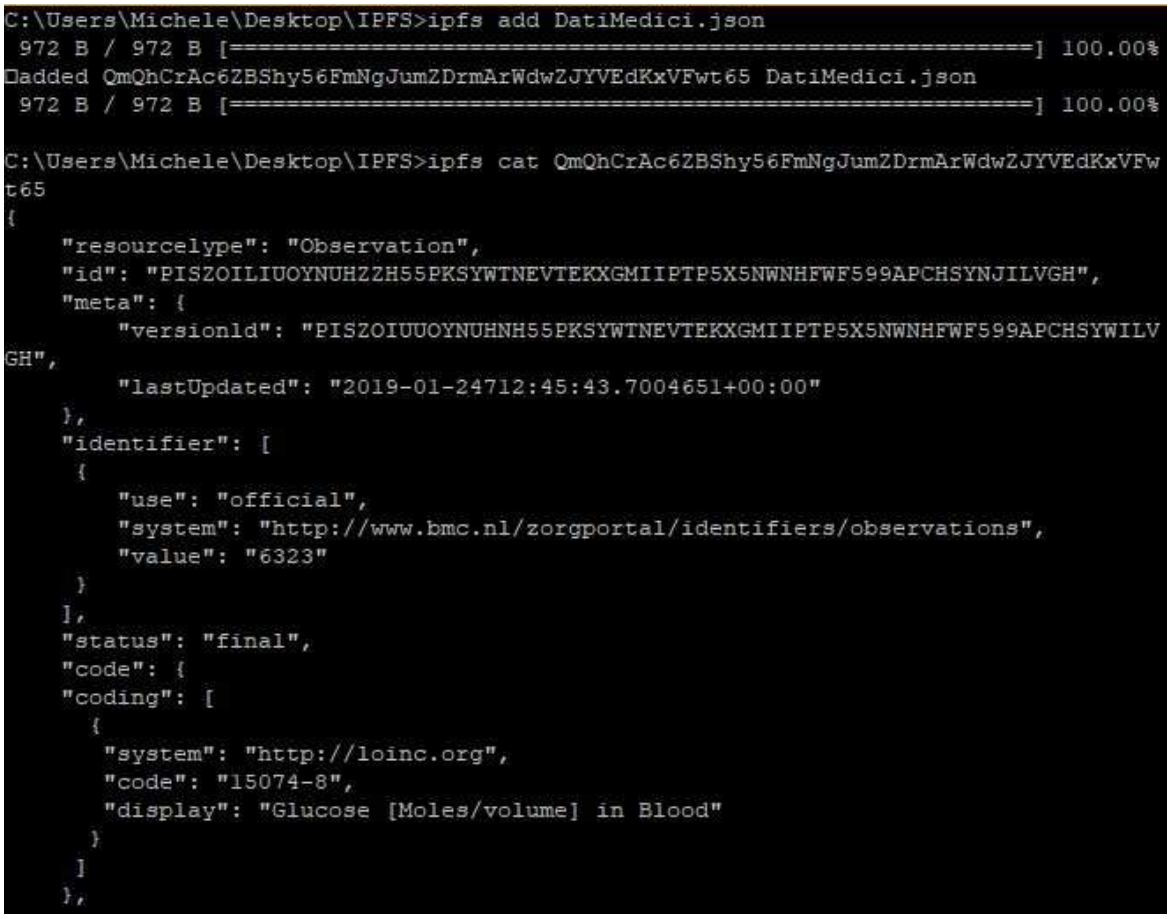

Figura 3.1: Caricamento e visualizzazione dati tramite IPFS

A questo punto si vogliono sfruttare le librerie JavaScript per interfacciarsi con IPFS e automatizzare le procedure descritte. La libreria principale che viene utilizzata per connettersi con IPFS è `js-ipfs`. Per farne uso, è stato necessario installare Node.js, un framework che consente di scrivere applicazioni in JavaScript lato server, e il package manager npm, un gestore di pacchetti per il linguaggio di programmazione JavaScript.

Esso consiste in un client da linea di comando, chiamato anch'esso npm, e un database online di pacchetti pubblici e privati, chiamato npm registry.

Js-ipfs è un'implementazione completa di IPFS che può essere usata come applicazione da riga di comando o come libreria, per avviare un nodo IPFS direttamente da un programma creato. Innanzitutto è necessario istanziare un `package.json` tramite il comando `npm init`, il quale avrà il compito di mantenere al suo interno i pacchetti installati per interfacciarsi con il protocollo. Successivamente si procede con l'installazione del pacchetto necessario, grazie al comando `npm install ipfs`, da digitare nel cmd. A questo punto è possibile iniziare a scrivere un programma che generi un nodo ed effettui il caricamento di un file nella rete IPFS.

Per prima cosa viene istanziato IPFS, dopo aver richiesto il modulo relativo ad esso. Per verificare che la procedura di definizione del nodo sia andata a buon fine, è possibile effettuare una stampa in console relativa alla versione del nodo utilizzato. Dopodiché si può procedere al caricamento di un file usando il comando `node.add`, definendo il percorso e il contenuto di quest'ultimo [15]. In fase di testing si effettua il caricamento di un file contenente dati medicali e si verifica, tramite una stampa nel prompt dei comandi, il completamento dell'operazione, mostrando il nome del file caricato e l'hash da esso generato. L'ultimo passo riguarda il recupero del file tramite il comando `node.cat`, mostrando a video il contenuto di esso nell'interfaccia a riga di comando. Infine, si avvia `ipfs daemon` per abilitare la connessione del nodo tramite IPFS, e si lancia lo script utilizzando il comando `Node IPFSCaricamento.js` che manda in esecuzione il codice. Da notare l'uso della funzione `async` per effettuare la creazione del nodo e il caricamento del documento in formato JSON, in modo tale da non bloccare l'esecuzione del main thread e portare a termine il compito definito in maniera corretta. In conclusione, tramite Node si esegue lo script effettuando il collegamento e il caricamento di un file sulla rete IPFS. Se si è connessi al network, il terminale restituirà la lista degli indirizzi IPFS dei propri peers, la conferma del completamento dell'operazione, l'hash che fa riferimento al file caricato e il contenuto di esso.

```
1 const IPFS = require('ipfs');
```

```
2
3  //istanzio l'oggetto JSON dei dati medici per poi convertire l'oggetto
       in stringa
4  var obj = require('./patient.json');
5
6  //converto l'oggetto in stringa e uso questa come contenuto del
       caricamento del file
7  var str = JSON.stringify(obj);
8
9  //creo un'istanza di IPFS nella variabile node
10 //creo una variabile in cui richiedo la versione di ipfs che gira su
       quel nodo
11 async function main() {
12   const node = await IPFS.create();
13   const version = await node.version();
14
15   //stampa della versione di IPFS che gira su quel nodo
16   console.log('Version:', version.version);
17
18   //codice che mi permette di aggiungere un file ad IPFS
19   const filesAdded = await node.add({
20     path: 'patient1.json',
21     content: str
22   });
23
24   //stampa il percorso del file aggiunto e la sua hash
25   console.log('Added file:', filesAdded[0].path, filesAdded[0].hash);
26
27   //recupera l'hash del file aggiunto e stampa il contenuto di quell'
       hash in console
28   const fileBuffer = await node.cat(filesAdded[0].hash);
29   console.log('Added file contents:', fileBuffer.toString());
30 }
31
```

```
32 main();
```

Listing 3.1: Script per il caricamento su IPFS

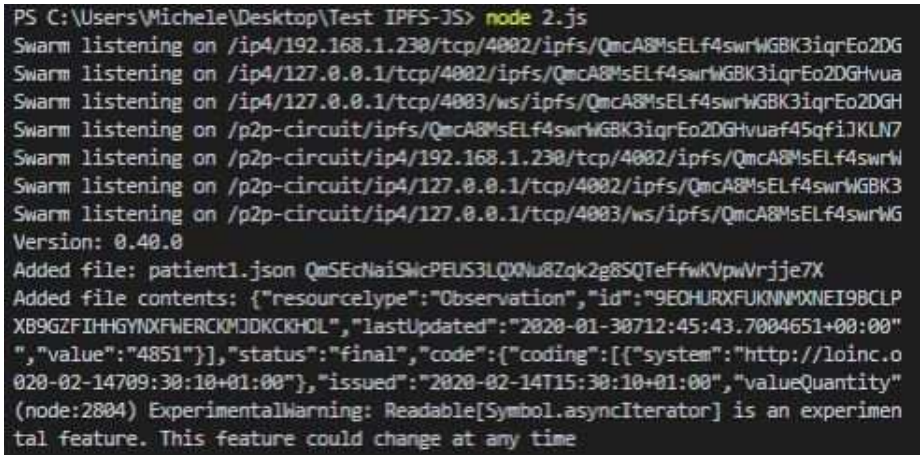

Figura 3.2: Pubblicazione File su IPFS

3.1.2 Librerie API JavaScript di IOTA

Al fine della realizzazione del progetto si è fatto uso delle librerie client JavaScript tramite le quali si è reso possibile interfacciarsi con il protocollo IOTA e la sua estensione MAM [16, 17]. Grazie ad esse si ha la capacità di creare un canale MAM a cui allegare messaggi. Nel caso studiato, verrà utilizzata questa tecnologia per effettuare la profilazione del paziente, inserendo all'interno del channel i dati che fanno riferimento alle informazioni di carattere medicale relativi ad un individuo, in modo da poterle recuperare in un secondo momento, fornendo un sistema a prova di manomissione.

Tramite la MAM Client JS Library è possibile pubblicare sul Tangle zero-value transactions, ovvero transazioni che contengono solo messaggi di testo, senza alcun valore finanziario, senza alcun costo. Ciò introduce molte possibilità per la comunicazione dei dati. La libreria utilizzata, tuttavia, è ancora in fase di sviluppo e potrebbe subire alcuni cambiamenti in futuro. Come per IPFS, essa lavora in combinazione con Node. Per prima cosa è necessario istanziare un `package.json` tramite il comando `npm init`,

il quale conterrà al suo interno tutti i pacchetti che verranno installati e utilizzati per interfacciarsi con i vari protocolli, se non lo si è fatto in precedenza.

Il passo successivo è stato quello di procurarsi la libreria client JavaScript di IOTA e le sue dipendenze. A tal fine si inserisce il codice da linea di comando `npm install @iota/core`. Questo pacchetto gestisce i metodi di basso livello che consentono l'interazione con il Tangle di IOTA. Grazie ad esso è possibile creare e firmare transazioni, generare indirizzi e interfacciarsi con i nodi della rete. Dopodiché si procede scaricando il pacchetto MAM con il comando `npm install @iota/mam` grazie al quale è possibile connettersi a nodi, creare data stream e recuperare transazioni. Come ultimo pacchetto viene installato il converter, il quale implementa gli algoritmi utili alla conversione del messaggio da codice ASCII in tryte: `npm install @iota/converter`. Nel momento in cui i pacchetti sono stati installati, è sufficiente creare un file JavaScript in cui vengono richiamati per usufruire delle funzioni descritte antecedentemente. A tal fine si fa uso del codice:

```
const MAM = require('@iota/mam');
const converter = require('@iota/converter');
```

per utilizzare i seguenti pacchetti.

Successivamente viene istanziata la classe con relativo costruttore in modo tale da poter creare il MAMChannel nel momento in cui viene essa viene utilizzata. I parametri che vengono presi in input per creare l'oggetto sono mode, provider, seed e sideKey che definisco le caratteristiche con cui il canale deve essere creato. Dopodiché è necessario effettuare l'inizializzazione di MAM, definendo l'indirizzo di rete su cui si lavora (in questo caso un nodo della Devnet) e il seed che si vuole utilizzare per la creazione degli indirizzi. In questo modo si ottiene come risultato un oggetto "stato" che tiene traccia dei progressi del proprio canale e dei canali che si seguono. Così facendo è possibile ottenere l'accesso alle funzionalità extra di registrazione, fetching e aggiunta di allegati.

Nel progetto realizzato, il caso studiato prevedeva l'utilizzo di canali MAM del tipo restricted. Per questa motivazione si è gestito un metodo che permetta di cambiare la sideKey che protegge l'accesso ai dati, nell'eventualità in cui il proprietario del canale

la voglia modificare o voglia rimuovere qualcuno dalle persone abilitate a visualizzare i suoi dati.

```
1  //importo librerie
2  const MAM = require('@iota/mam');
3  const converter = require('@iota/converter');
4  const seedrandom = require('seedrandom');
5
6  class MAMChannel {
7    /**
8     * Constructor
9     * @param {String} mode - 'public','private','restricted'
10    * @param {String} provider - IOTA provider
11    * @param {String} seed - The IOTA seed or a key used to generate it
12    * @param {String} sideKey - The secret key for 'restricted' mode
13    */
14   constructor(mode, provider, seed = null, sideKey = null) {
15     this.mode = mode;
16     this.provider = provider;
17     this.seed =
18       seed && seed.length === 81 && /[A-Z9]/.test(seed)
19         ? seed
20         : MAMChannel.utils.iotaSeedGen(seed);
21     this.sideKey = sideKey;
22     this.mamState = null;
23     console.log(seed);
24   }
25
26   /**
27    * Open or create and set channel mode
28    */
29   openChannel() {
30     this.mamState = MAM.init(this.provider, this.seed);
31     this.mamState = MAM.changeMode(
32         this.mamState, this.mode, this.sideKey
33     );
```

```
34   }
35
36   /**
37    * Change the secret key to use from now on
38    * @param {String} key - The new key
39    */
40   changeKey(key) {
41     this.sideKey = key;
42     this.mamState = MAM.changeMode(
43         this.mamState, this.mode, this.sideKey
44     );
45   }
```

Listing 3.2: Classe MAMChannel, creazione canale

Si procede poi con la creazione della funzione che permette di pubblicare il messaggio sul canale. Essa elabora il messaggio che gli viene passato in formato JSON e lo trasforma secondo il sistema ternario su cui è basato IOTA. Fatto ciò, crea il payload del messaggio MAM, pubblicandolo nel Tangle all'indirizzo che si sta seguendo grazie al metodo `attach`, stampando successivamente nella console di comando il contenuto del messaggio e l'indirizzo su cui sono memorizzati i dati, se l'operazione è andata a buon fine.

```
1  /**
2    * Publish a message to tangle
3    * @param {Object} packet - The message (a JSON object)
4    * @returns {String} The message's root
5    */
6   async publish(packet) {
7     //Create MAM Payload - STRING OF TRYTES
8     let trytes = converter.asciiToTrytes(JSON.stringify(packet));
9     let message = null;
10    let limit = 20;
11    //Search for messages already present in the tangle with that root
12    do {
13      message = MAM.create(this.mamState, trytes);
```

```
14        this.mamState = message.state;
15      } while (
16        typeof (await MAM.fetchSingle(message.root, this.mode, this.
              sideKey))
17          .payload !== 'undefined' &&
18        limit++ > 0
19      );
20
21      //Attach the payload to the channel
22      //9 is the minimum weight magnitude (for the PoW)
23      //3 is the Depth of the Merkel Tree
24      try {
25        await MAM.attach(message.payload, message.address, 3, 9);
26        console.log(
27          'Published on MAM channel:\n',
28          packet,
29          '\nRoot:',
30          message.root,
31          '\n'
32        );
33      } catch (e) {
34        console.log(e);
35      }
36      return message.root;
37    }
```

Listing 3.3: Classe MAMChannel, pubblicazione messaggio

In fase di testing si è istanziato un oggetto MAM per verificare il funzionamento del file. Viene testata la creazione di un canale restricted con seed e sideKey definiti in maniera conforme all'utilizzo del servizio da parte di un qualunque utente, inserendo come contenuto del messaggio informazioni anonimizzate relative ad un paziente X. Lo script viene eseguito richiamando Node da linea di comando in questo modo: `Node MAMChannel.js`, eseguendo quanto definito nel codice 3.4.

```
1 //recupero il file e lo trasformo in stringa
```

```
var str = JSON.stringify(patient);

//istanzio il canale mam
const mam = new MAMChannel(
  'restricted',
  'https://nodes.devnet.iota.org:443',
  'TGZPSNZBPVSIGWIHQYCQSKHRESSNWNKKUFQCGUERHONBPQRBU9BMPGFK
  GKWCEVBCFRKFDFATJVLDDFIBC'
  'PSW9W9S9P9');

mam.openChannel();

//contiene la root attuale
var test = mam.getRoot();

//pubblico il messaggio e stampo la lista di messaggi del canale
mam.publish({
      message: str,
      timestamp: (new Date()).toLocaleString()
    }).then(result =>{ console.log(result);
    mam.fetchFrom(test).then(temp => console.log(temp));
    })
```

Listing 3.4: Testing MAM

In questo modo sono stati utilizzati Node e la libreria JavaScript per inviare messaggi mascherati e autenticati sul Tangle di IOTA. Se l'operazione sarà portata a termine con successo, in console potrà essere visualizzata la conferma della pubblicazione con la root su cui risiede il messaggio e il contenuto che è stato memorizzato. Inoltre viene indicata la nextroot, ovvero l'indirizzo successivo in cui risiederà il prossimo messaggio di quel canale. Se all'interno del MAM risultano essere presenti altri dati, verrà stampata l'intera catena dei messaggi che appartengono a quel channel. La fase di pubblicazione avviene in maniera asincrona tramite la funzione `async` in quanto è possibile che la richiesta sulla rete impieghi molto tempo, al fine di evitare il blocco del thread principale.

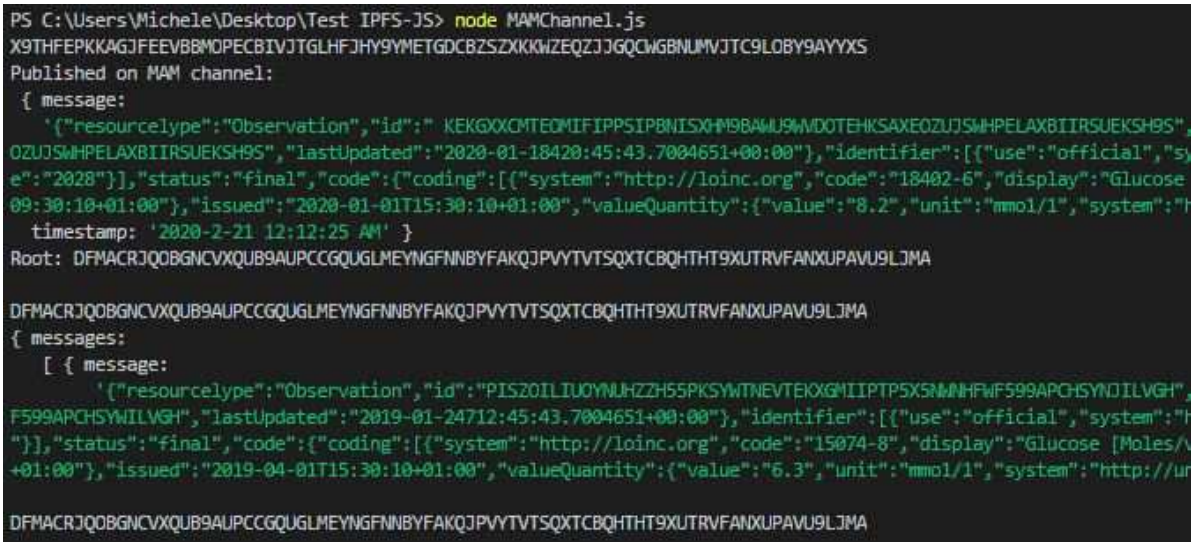

Figura 3.3: Pubblicazione dati sul Tangle

É inoltre possibile verificare l'avvenuta pubblicazione del messaggio nel tool "MAM EXPLORER" messo a disposizione da IOTA, inserendo la root su cui il messaggio è pubblicato, la modalità di pubblicazione utilizzata e, nel caso si sia fatto uso della restricted, la sideKey per poter decifrare e accedere al contenuto.

3.2 Creazione dell'Infrastruttura

In seguito all'analisi delle tecnologie sotto esame, IPFS e IOTA, è stato possibile realizzare il modello argomentato in questa tesi. Si è prodotto un sistema che consente di gestire la memorizzazione di parametri glicemici all'interno della rete IPFS, e la pubblicazione dei relativi hash risultanti sul Tangle di IOTA, grazie al protocollo MAM. In tal modo viene a crearsi un ambiente interoperabile per la consultazione e lo scambio di dati medicali.

Analizzando il file che implementa tale modulo, è possibile individuare il codice `const MAMChannel = require('./MAMChannel.js');` Questo fa riferimento al programma sviluppato nella sezione precedente, il quale è stato modificato al fine di svolgere la semplice operazione di creazione di un canale e la successiva pubblicazione dei dati in esso.

Inoltre si importa la libreria che permette di interfacciarsi con IPFS: `const IPFS = require('ipfs');` A questo punto si definisce una variabile oggetto che consenta di gestire il file JSON, contenente i dati sanitari raccolti, in modo da poterlo convertire in stringa e fornirlo come input alla funzione che ne effettuerà il caricamento sulla rete IPFS, siccome quest'ultima non accetta come attributo documenti in questo formato.

```
let obj = require('./patient.json');
let str = JSON.stringify(obj);
```

Si procede poi con istanziare la funzione `inst`, che si occupa di definire il nodo, grazie al pacchetto IPFS, e il canale MAM. Questi avranno i compiti di effettuare i rispettivi caricamenti sulle due piattaforme. Come è possibile dedurre dal codice 3.5, il channel viene creato in modalità restricted, con sideKey e seed definite a piacimento dell'utente.

```
1  let node;
2  let mam;
3
4  //creo funzione che mi istanzia nodo IPFS e mam channel
5  async function inst() {
6    node = await IPFS.create();
7    //faccio una stampa della versione di IPFS che gira su quel nodo
8    console.log('Version:', await node.version());
9
10   mam = new MAMChannel(
11     'restricted',
12     'https://nodes.devnet.iota.org:443',
13     'PSW9hj',
14     'PASSWORD9'
15   );
16   mam.openChannel();
17   console.log(mam.getRoot());
18 }
```

Listing 3.5: Definizione delle tecnologie

A questo punto non resta che definire una funzione che consenta di inserire sulla rete di IPFS i dati anonimizzati che sono stati creati, e recuperare l'hash che ne deriva. Una volta che questo valore è salvato in una variabile è possibile utilizzarlo come corpo del messaggio MAM che verrà pubblicato sul canale definito in precedenza. Questo metodo viene implementato all'interno di un meccanismo try&catch in modo da stampare a video l'errore nel caso in cui qualcosa vada storto.

Sempre grazie a Node, è possibile eseguire il programma tramite `Node IPFS_MAM.js` che manderà in esecuzione la funzione main(), la quale si occuperà di lanciare i metodi async `inst` e `publishAll`, i quali svolgeranno i compiti riportati sopra, in sequenza. Il modulo adibito ad interfacciarsi con le tecnologie IPFS e IOTA è quindi completato e funzionante.

```
1  //codice che mi permette di aggiungere un file ad IPFS
2  async function publishAll() {
3    try {
4      const filesAdded = await node.add({
5        path: 'patient1.json',
6        content: str
7      });
8
9      //codice per la pubblicazione dell'hash sul Tangle
10     mam.publish({ messageHash: filesAdded[0].hash });
11   }catch (e) {
12     console.log(e);
13   }
14 }
15
16 async function main(){
17   await inst();
18   await publishAll();
19 }
20
```

```
21 main();
```

Listing 3.6: Upload dei dati nei due protocolli

3.2.1 Gestione IPFS

Ricordando che l'InterPlanetary File System (IPFS) è un distributed file system utilizzato per soddisfare due importanti considerazioni, l'incertezza riguardo a quali tecniche di anonimizzazione sono legalmente sufficienti per trasformare PHI in dati anonimizzati, e per ridurre al minimo i rischi relativi al trattamento dei dati personali, tutti i record FHIR sono stati archiviati in esso. La gestione dei file che gli vengono affidati avviene in base al loro contenuto, salvandoli in una struttura dati Merkle-DAG, simile al Merkle tree che utilizza MAM per la generazione degli indirizzi.

Questa tecnologia è stata impiegata, in combinazione a IOTA, in quanto implementa dei meccanismi di sicurezza affidabili e permette di mappare file di dimensioni arbitrarie in dati di dimensione fissa (hash). Difatti è computazionalmente impossibile per un individuo risalire al contenuto di un documento essendo a conoscenza dell'hash che questo ha generato. Un altro vantaggio che ha favorito il suo impiego in questo progetto è dato dalla sua caratteristica di condivisione dei file solo su richiesta, e della loro cancellazione in un qualsiasi momento. Poiché la struttura da cui si è preso ispirazione considera lo scambio di dati in un ambiente in cui i soggetti partecipanti sono noti, ovvero paziente e medico, è possibile effettuare la rimozione dei documenti memorizzati su IPFS anche nei nodi su cui risiedono le copie di backup dei file. Per questo motivo, l'utente-fornitore possiede il diritto e le facoltà di cancellare una qualsiasi file da IPFS.

In conclusione questo protocollo è stato impiegato per effettuare il caricamento dei file, contenenti i dati personali anonimizzati dei pazienti, nel suo network, ottenendo un hash che verrà successivamente memorizzato sul Tangle di IOTA, fornendo una struttura sicura e affidabile, utile per lo scambio di informazioni di carattere medico che permettono al soggetto paziente di ottenere un'assistenza sanitaria di alta qualità garantita da una profilazione dei suoi parametri nel tempo.

3.2.2 Gestione MAM

In questo progetto tutti i record FHIR sono stati archiviati su un file system distribuito, IPFS, e l'hash risultante è stato registrato su IOTA. Il modulo MAM consente di crittografare i messaggi, confermare l'origine della fonte e creare un flusso di messaggi continuo sul Tangle. La caratteristica fondamentale di questo protocollo è che vengono creati Id univoci per ciascun canale e solo le parti autorizzate sono in grado di leggere e ricostruire l'intero flusso di messaggi.

Per creare il channel si utilizza la funzione descritta in precedenza, che prende in input modalità, provider, seed e sideKey. Una volta che il canale viene generato è possibile pubblicare messaggi su di esso e verificarne il contenuto, per controllare il registro di informazioni che sono state mappate al suo interno. Per svolgere questa operazione è necessario visitare la pagina relativa al MAM Explorer, che consente di accedere ai messaggi di un canale inserendo root e sideKey per decifrarne il contenuto, nel caso in cui venga utilizzata la modalità restricted. D'altro canto si può far uso della libreria JavaScript di IOTA per implementare un'operazione di recupero e visualizzazione delle informazioni:

```
1  async fetchFrom(root) {
2      let result = await MAM.fetch(root, this.mode, this.sideKey);
3      if (typeof result.messages !== 'undefined'
4      && result.messages.length > 0) {
5        for (var i = result.messages.length - 1; i >= 0; i--) {
6          result.messages[i] = JSON.parse(converter.trytesToAscii(result.
               messages[i])
7          );
8        }
9      }
10     return result;
11 }
```

Listing 3.7: Fetch degli hash dal Tangle

Per permettere al personale sanitario di accedere ai dati del canale è necessario comunicare loro le credenziali, fornendogli seed e sideKey, in modo tale da poter risalire alla catena di messaggi. L'utente, per giunta, può decidere di rimuovere qualcuno dalle persone autorizzate a visionare i dati del MAM Channel: si prenda in esempio un paziente che aveva comunicato le credenziali di accesso alle sue informazioni ad un medico e che, passato un certo ammontare di tempo, egli venga preso in cura da un altro dottore. Ricordando che in MAM è implementato un meccanismo di forward transaction linking, grazie al quale dato un indirizzo su cui è memorizzato un messaggio, è possibile risalire a tutte le transazioni successive ad esso, ma non alla precedenti, l'utente può decidere di modificare la chiave di accesso al canale; così facendo coloro che si è deciso di escludere dalla consultazione dei propri dati non avranno più la possibilità di visionarli: in questo caso, il primo medico.

É possibile applicare questo strumento anche nel caso d'uso in cui sia il medico stesso a generare il canale MAM e che, successivamente, fornisca le credenziali al paziente per effettuare il caricamento dei dati su di esso. Un principio importante da applicare è di mantenere al sicuro i dati relativi alle credenziali e comunicarli solo a coloro che si ritengono indispensabili per la consultazione delle informazioni, in maniera tale che non si verifichi una manomissione del canale da parte di un soggetto che, entrando in possesso di seed e sideKey, effettua operazioni di spam su di esso, o più semplicemente, visualizza le informazioni necessarie per impossessarsi dei dati sanitari personali di quell'individuo.

Lo scambio della sideKey e del seed può essere effettuato seguendo diversi metodi. Il più comune, e forse quello più sicuro, è lo scambio delle credenziali tra le parti effettuato di persona. Semplicemente, nel momento dell'incontro, è possibile effettuarne la comunicazione a voce o tramite il chip NFC dei loro smartphone. Ancora, è possibile utilizzare un sistema che consenta di condividere informazioni di un qualsiasi tipo su un dispositivo tramite scannerizzazione QR. In questo caso basterebbe disporre di un file contenente seed e sideKey e selezionare l'opzione di creazione del codice, in modo tale che l'altra parte debba semplicemente inquadrare l'immagine con la fotocamera di un device per entrare in possesso del file.

In quest'ottica, la fase di definizione delle credenziali potrebbe essere gestita tramite una UI che consenta all'utente, in maniera intuitiva, di inserire i valori che si vogliono utilizzare per la creazione del canale. Una volta che i dati sono stati inseriti, questi potrebbero essere direttamente aggiungi al file che gestisce il caricamento dei dati sulle varie piattaforme, oppure potrebbero essere immagazzinati in un documento separato e recuperati in un secondo momento da quel programma. Così facendo potrebbe essere implementato un meccanismo che generi in maniera randomica questi valori, nel caso in cui l'utente-fornitore preferisca lasciar decidere al sistema quali dati impiegare per la creazione del canale, e utilizzare una seconda UI che permetta di recuperare e mostrare a schermo i parametri inseriti in questo documento. Il salvataggio in un file differente, come descritto, potrebbe avere anche una seconda applicazione, utile per la condivisione. In tal modo, difatti, potrebbero essere applicate diverse metodologie per lo scambio delle credenziali, che vanno oltre alla comunicazione effettuata di persona.

Una possibile variante è quella di applicare un sistema crittografico, nello specifico un ambiente NTRU (Nth degree-truncated polynomial ring units) che è costituito da una raccolta di algoritmi matematici, con cui è possibile effettuare lo scambio delle credenziali tra paziente e medico tramite mezzo telematico e in maniera sicura. Il funzionamento di tale sistema è da ricercarsi in uno scambio reciproco di richieste crittografate di condivisione dati, utilizzando le chiavi pubbliche e private in possesso delle parti per decifrarle ed effettuare una comunicazione sicura tra i due utenti, medico e paziente.

Capitolo 4

Risultati

Questo capitolo introduce al lettore i risultati ottenuti dal programma realizzato. Viene mostrato all'utente l'esito del caricamento dei file sul network di IPFS e la memorizzazione del relativo digest sul Tangle di IOTA tramite l'utilizzo di tools che, attraverso un'interfaccia grafica, consentono una consultazione e un'interazione semplificata con il sistema realizzato, nonostante l'implementazione sviluppata nel progetto, di facile comprensione per i soli utenti consapevoli. Nella parte conclusiva, si introduce l'analisi dei tempi di latenza relativi al caricamento dei dati sulle due piattaforme, effettuando anche il calcolo relativo al tempo medio di invio grazie a ripetute operazioni di upload dati, al fine di ottenere informazioni più affidabili.

4.1 Verifiche sui Dati

4.1.1 Visualizzazione IPFS

Una volta che si dispone dei dati in formato FHIR relativi ad un individuo, è possibile effettuarne il caricamento sulla rete IPFS, sfruttando l'apposita funzione descritta nei capitoli precedenti. Nel momento in cui il file è stato memorizzato con successo, è possibile recuperarlo tramite la rete IPFS, grazie all'hash che ne deriva. Per fare ciò è necessario aver installato nel proprio elaboratore l'eseguibile di IPFS, che consente

al PC di interfacciarsi con esso, e inserire nella barra di ricerca del proprio browser `https://ipfs.io/ipfs/hash_ottenuto` in modo tale che il gateway recuperi il file.

Un altro sistema, ancora migliore, è quello di utilizzare un tool implementato dagli stessi sviluppatori di IPFS che tramite un interfaccia utente consente, in maniera intuitiva, di accedere a tutti i documenti e alle directory che quel nodo ha caricato sulla rete. L'utente-paziente, quindi, nel momento in cui voglia accedere ad IPFS per visualizzare tutti i file che automaticamente sono stato caricati al suo interno, non dovrà fare altro che assicurarsi di avere il daemon in funzione e recarsi nella pagina di questo strumento dal proprio dispositivo (`http://localhost:5001/ipfs/hash_del_proprio_nodo`), accedere alla sezione "Files", recarsi nella voce pins e selezionare l'hash corrispondente al file che vuole visionare. Quando ciò accade, verrà aperta una sezione che gli consentirà di leggere i proprio dati in maniera comprensibile. Inoltre è possibile effettuare una ricerca di un qualsiasi documento sulla rete IPFS inserendo il suo hash nella barra di ricerca posta all'inizio della pagina.

Un'altra operazione che l'utente può compiere è la cancellazione di uno di questi elementi. Per eliminare il file egli non deve fare altro che selezionare l'Html-button raffigurante i tre puntini e cliccare sulla voce "unpin". In questo modo il documento è stato rimosso dal nodo IPFS. Nel caso in cui quel file sia stato condiviso con altri nodi, siccome è stato implementato un sistema privato per lo scambio delle informazioni e i nodi sono noti, è possibile richiedere agli altri elementi della rete che possiedono una copia di quei parametri di effettuare a loro volta l'eliminazione, in modo che non vi sia più alcuna traccia di quei dati su nessun nodo.

Questo tool è quindi utile per una gestione semplificata dei propri file, ma non è indispensabile grazie alle implementazioni introdotte nel programma realizzato, di difficile comprensione per utenti non consapevoli.

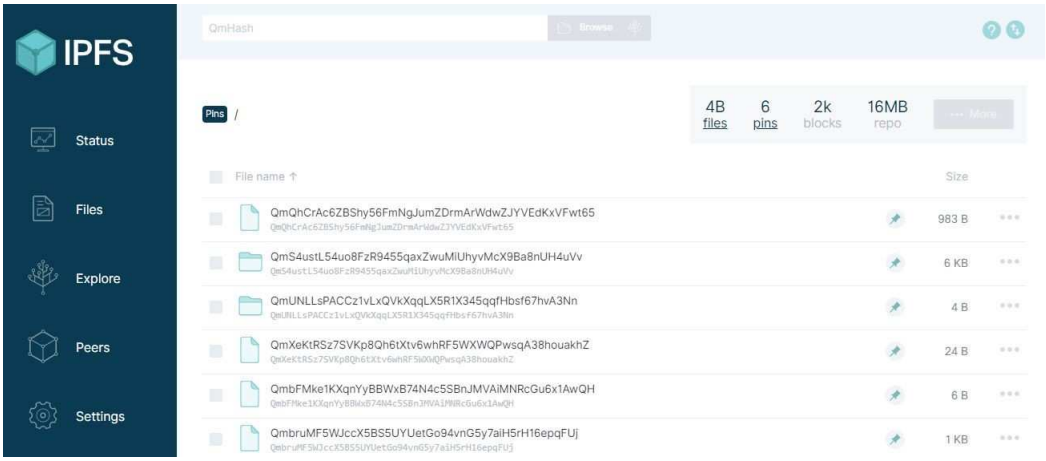

Figura 4.1: IPFS Web Console

4.1.2 Visualizzazione MAM

Per quanto concerne IOTA e la sua versione rivista della Blockchain, il Tangle, è possibile recuperare i messaggi contenenti gli hash dei file caricati su IPFS direttamente da linea di comando, sfruttando la funzione di fetch di cui si è discusso in precedenza.

Per semplificare la vita agli utenti che devono ottenere le informazioni ricercate sul canale MAM, è possibile utilizzare un'applicazione che prende il nome di MAM Explorer, la quale consente di visualizzare il contenuto dei messaggi se si è a conoscenza dell'indirizzo in cui esso risiede, e la chiave di decifratura nel caso in cui l'upload della transazione sia avvenuto con opzione restricted.

É importante tenere a mente il forward linking, tramite cui dato un indirizzo sul canale è possibile risalire a tutti gli indirizzi successivi ad esso, e quindi alla catena di messaggi che sono stati inseriti dopo quel punto di ingresso. Avendo quindi una root che si vuole visitare e una sideKey con cui decifrarla è possibile effettuare una ricerca, cliccando sul bottone "fetch", il quale restituirà dopo una relativamente breve ricerca il messaggio o la lista di messaggi che da quella root e con quella chiave sono stati caricati sul canale. Si entrerà in possesso di una serie di hash che permetteranno all'utente interessato di effettuare il recupero dei dati in chiaro grazie alla procedura descritta nella

sezione relativa ad IPFS. Durante la ricerca è necessario, ovviamente, inserire anche il provider, ovvero la rete IOTA su cui vogliamo effettuare la ricerca.

Nel caso sotto esame, siccome si tratta di un'applicazione in fase di sviluppo, il caricamento delle informazioni è stato effettuato sulla rete per sviluppatori, la Devnet. Per questo motivo è necessario indicare questo network come provider di ricerca, inserendo `https://nodes.devnet.iota.org:443` nell'apposito campo.

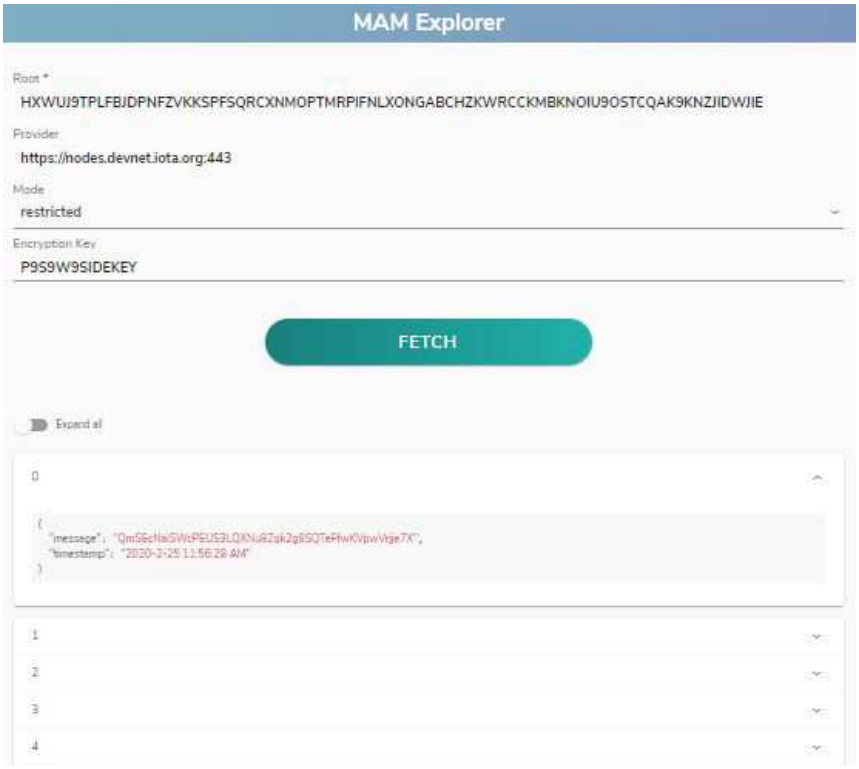

Figura 4.2: MAM Explorer [18]

Come si può vedere dalla Figura 4.2, sono stati recuperati i messaggi del MAM Channel a quell'indirizzo e ai successivi, utilizzando come chiave di decriptazione la sideKey che stata impiegata in modalità restricted.

4.2 Analisi tempi di latenza

Per effettuare delle analisi statistiche sulle tecnologie utilizzate si è deciso di testare le performance dell'attuale progetto in termini di latenza nel caricamento dei file. Siccome questi modelli vantano un'usabilità anche in dispositivi con risorse limitate e in presenza di banda ristretta, i dati sono stati raccolti da un computer LENOVO ESSENTIAL E51 con processore Intel Core i5-6200U @2.30GHz, connesso alla rete tramite modulo WiFi con larghezza di banda così disponibile: 2.16Mbps in download, 0.33Mbps in upload e PING di 67ms.

I dati di carattere sanitario sono stati trasmessi utilizzando il runtime Node.js e il programma sviluppato. Inoltre, va specificato che è stata utilizzata la Devnet di IOTA per effettuare i test relativi a MAM.

Si è proceduto con la modifica delle funzioni che gestiscono il caricamento dei documenti sulle due piattaforme, inserendo delle variabili che mi permettono di mantenere in memoria l'esatto momento, in millisecondi, in cui un'operazione inizia o termina. In particolare, nella funzione `publishAll` del sistema sviluppato, vengono inserite due variabili, `startTimeIPFS` e `finishTimeIPFS`, le quali rispettivamente mantengono in memoria il momento in cui inizia e finisce l'operazione di caricamento sulla rete IPFS, in maniera tale da poter calcolare il tempo impiegato per portare a termine l'operazione, eseguendo una sottrazione tra questi due parametri.

A questo punto è stata corretta la funzione `publish` all'interno del file MAMChannel, a cui sono state aggiunte due variabili con il medesimo scopo di quelle precedentemente descritte. La differenza sostanziale è che, siccome l'operazione di pubblicazione sul Tangle si trova in un altro file, è stato necessario restituire da esso il valore relativo alla latenza (`finishTimeMAM`) al file principale. Una volta memorizzati i tempi di latenza che si sono verificati per le due tecnologie è stato calcolato il tempo di latenza complessivo dato dalla somma di questi valori.

```
mam.publish({ messageHash: filesAdded[0].hash }).then(result =>
{ var latTot = (+result + +latenzaIPFS);
```

```
console.log("Latenza totale: "+latTot+"ms")
});
```

Il sistema viene eseguito, come al solito, attraverso l'utilizzo di Node, dove però in console si visualizzeranno anche le informazioni relative ai tempi di caricamento. I risultati che sono stati ottenuti sono relativi alla macchina e al luogo da cui i test sono stati svolti, nonché dai file utilizzati per il caricamento. Per tale motivo, modificare uno di questo parametri potrebbe comportare risultati differenti.

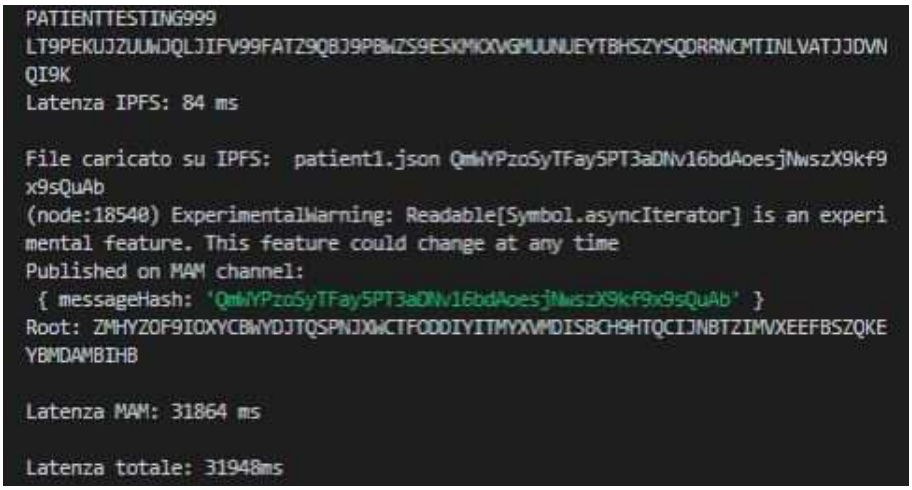

Figura 4.3: Latenze dei caricamenti

4.2.1 Calcolo latenze medie

Per concludere la sezione relativa all'analisi statistica si introduce il calcolo della latenze medie. Tramite le singole latenze relative al caricamento dei dati sulle piattaforme di IPFS e MAM è possibile effettuare un calcolo più approfondito di quella che è la latenza presente in media quando si effettuano operazioni sui dati. Per ottenere questo valore è stato implementato il caricamento su IPFS di 20 file contenenti parametri medicali, e il conseguente caricamento degli hash risultanti sul Tangle di IOTA.

I valori relativi ai tempi di upload su IPFS di ciascun documento sono stati salvati all'interno di un Array, `latsIPFS`. Gli hash ottenuti sono stati caricati su MAM e si è

proceduto con la memorizzazione delle relative latenze nell'Array `latsMAM`. Terminato il ciclo che autonomamente gestiva i vari file, si sono sommate le 20 tempistiche all'interno delle due liste, in modo da ottenere un elenco (`latsTotali`) contenente le singole latenze totali che si sono presentate nello svolgere queste operazioni. Come ultimo step, sono stati sommati i parametri all'interno di questo Array, al fine di ottenere la latenza complessiva che si è verificata durante l'intero processo. Infine, per ottenere il parametro che si sta ricercando, il valore derivante dalla sommatoria descritta in precedenza è stato diviso per il numero di upload effettuati (20).

```
1  //istanzio tre array che conterranno le latenze
2  async function publishAll() {
3    latsIPFS = new Array();
4    latsMAM = new Array();
5    latsTotali = new Array();
6
7  //for per il caricamento dei file
8    for (var i = 0; i < 20; i++) {
9      const startTimeIPFS = new Date().getTime();
10     try {
11       const filesAdded = await node.add({
12         path: 'patient1.json',
13         content: str + i
14       });
15       //calcolo la latenza per IPFS
16       const finishTimeIPFS = new Date().getTime();
17       const latenzaIPFS = finishTimeIPFS - startTimeIPFS;
18
19       //inserisco la latenza nell'Array
20       latsIPFS.push(latenzaIPFS);
21
22       mam.publish({
23         messageHash: filesAdded[0].hash
24       }).then(result => {
25       //inserisco nell'Array la latenza del caricamento sul Tangle
```

```
26          latsMAM.push(result);
27
28          var sommaLatenze;
29          var latenzaMedia;
30
31          //quando dispongo di tutti i dati calcolo le latenze medie
32          if (latsMAM.length === 20) {
33
34            console.log("Latenze di IPFS: "+latsIPFS);
35            console.log("Latenze di MAM: "+latsMAM);
36
37          //sommo le latenze di IPFS e MAM per ottenere le latenze totali
38            var latsTotali = latsMAM.map(function (num, idx) {
39              return num + latsIPFS[idx];
40            });
41
42            console.log("Somma delle latenze tra IPFS e MAM: "+latsTotali
                 );
43          //sommo le latenza totali per ottenere la latenza complessiva
44            sommaLatenze = latsTotali.reduce((a, b) => a + b, 0);
45            console.log("Somma latenze totali: " + sommaLatenze);
46          //calcolo la latenza media
47            latenzaMedia = sommaLatenze/20;
48            console.log("La latenza media e': " + latenzaMedia + " ms");
49          }
50        });
51      } catch (e) {
52        console.log(e);
53      }
54    }
55  }
```

Listing 4.1: Script per il calcolo delle latenze

Così facendo si ottiene la stima puntuale relativa al tempo medio della latenza di invio, in questo caso corrispondete a 19.13 secondi, che identifica il tempo medio che

occorre al dispositivo per effettuare l'upload dei file su IPFS e IOTA. In aggiunta a ciò, sono state calcolate separatemene la media relativa al caricamento dei file su IPFS e la media relativa alla memorizzazione degli hash sul Tangle, ottenendo rispettivamente 0,42 e 18,71 secondi.

Latenze in millisecondi		
Latenze IPFS	Latenze MAM	Latenze Totali
81	13497	13578
78	14282	14360
190	10589	10779
172	12576	12748
214	19567	19781
189	9461	9650
168	22373	22541
1013	14957	15970
614	20673	21287
1003	12279	13282
127	16196	16323
1298	14658	15956
414	14344	14758
1532	14563	16095
352	25116	25468
249	29773	30022
173	27626	27799
77	24518	24595
71	24845	24916
366	32357	32723
Media in millisecondi		
419,05	18712,5	19131,55

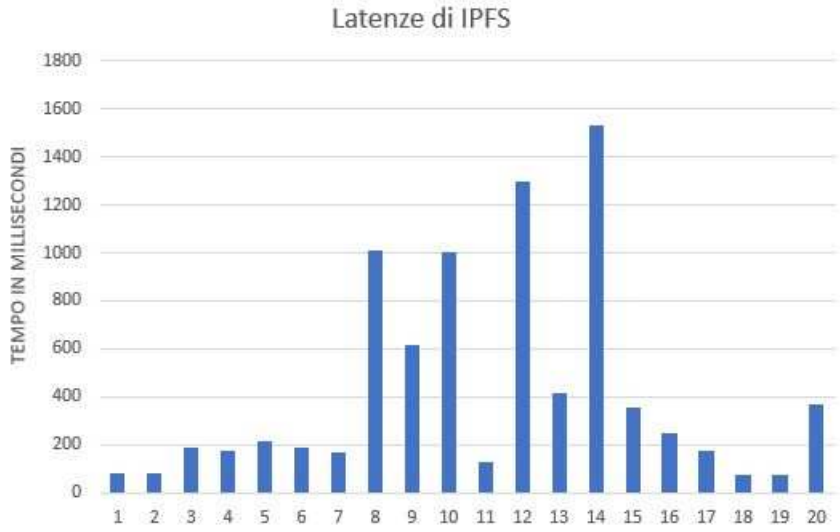

Figura 4.4: Grafico Latenze IPFS

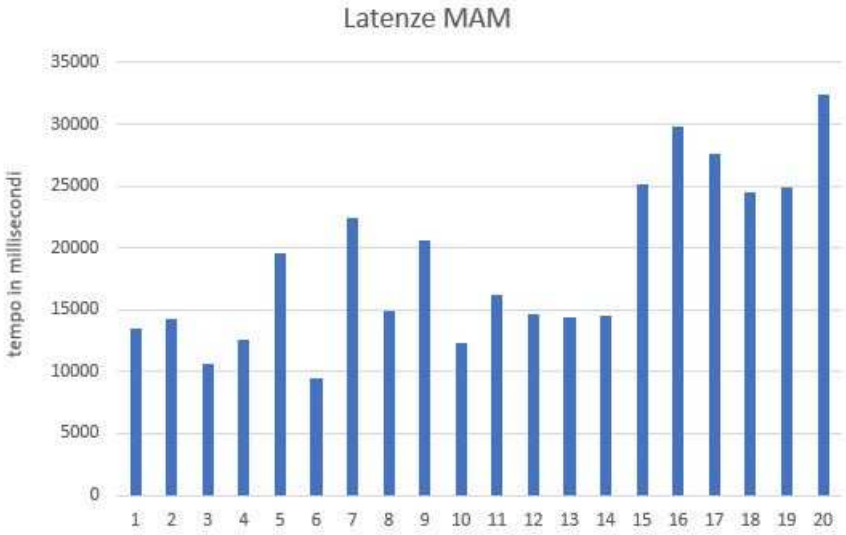

Figura 4.5: Grafico Latenze MAM

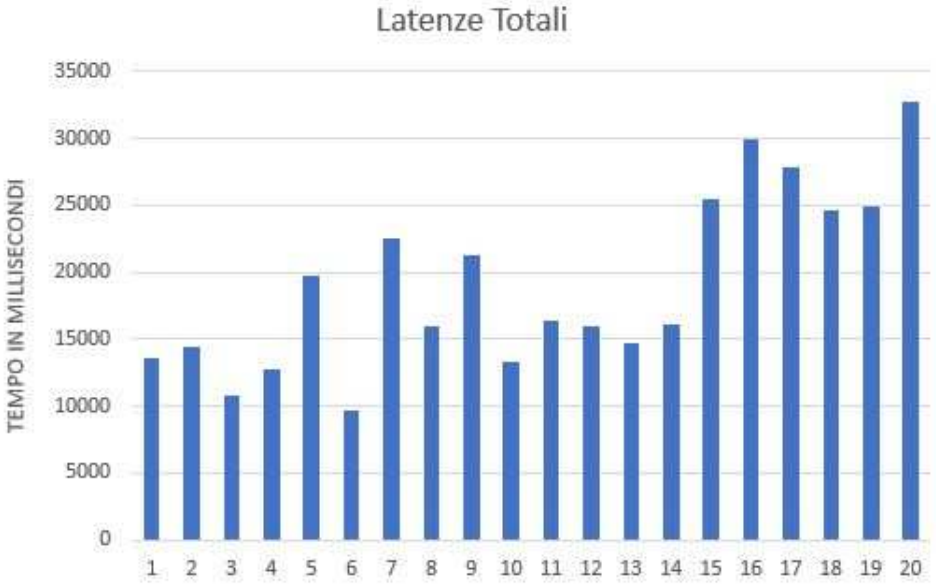

Figura 4.6: Grafico Latenze Totali

4.2.2 Calcolo Intervallo di Confidenza

La stima puntuale fornisce un singolo valore. Tuttavia questo parametro non coincide quasi mai con la misura reale relativa alla latenza; Inoltre, campioni diversi forniscono stime puntuali differenti. La stima intervallare fornisce un intervallo che ha una predeterminata probabilità di contenere il valore effettivo relativo alla latenza. Pertanto gli intervalli ottenuti da campioni diversi, in genere, si sovrappongono. Per intervallo di confidenza di un parametro θ della latenza, si intende un intervallo delimitato da due limiti, inferiore e superiore, che abbia una definita probabilità (1 - α) di contenere il vero parametro della latenza [20], dove 1 - α rappresenta il grado di confidenza, e α identifica la probabilità di errore. L'intervallo di confidenza diminuisce se:

- Diminuisce il livello di confidenza (1 - α).
- Aumenta la numerosità del campione.
- Diminuisce la variabilità nella latenza.

Al fine di calcolare l'intervallo di confidenza è necessario disporre della media puntuale $\bar{x}$ dei 20 (n) dati osservati, e della loro deviazione standard s. Nonostante si sia già a conoscenza della media aritmetica, calcolata nella sezione precedente, viene introdotta la formula matematica per la corrispettiva computazione:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

A questo punto viene esplicitato il modulo per la deviazione standard s, al fine di poter comporre l'espressione dell'intervallo di confidenza:

$$s = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2}$$

Per ottenere la formula necessaria per il calcolo desiderato non resta che stabilire il livello di confidenza. Generalmente questo parametro viene fissato dal ricercatore, e il valore scelto più frequentemente corrisponde al 95%. Per tali ragioni, in questa analisi si impiega il medesimo dato. L'intervallo di confidenza al 95% è definito da:

$$\bar{x} \pm 1,96 \frac{s}{\sqrt{n}}$$

Se la numerosità campionaria è relativamente piccola (n<60, nel caso sotto esame n = 20), è necessario utilizzare un'altra distribuzione di probabilità, la distribuzione t di Student, caratterizzata da una maggiore dispersione. Pertanto, in funzione del parametro n e del valore di α, si individua il quantile 2,093 per il calcolo ricercato. Viene riformulata l'espressione in questo modo:

$$\bar{x} \pm 2,093 \frac{s}{\sqrt{n}}$$

Ora è possibile determinare la deviazione standard per le due tecnologie, e impiegare la formula riportata sopra per individuare i corrispettivi intervalli di confidenza. I valori calcolati vengono riportati nelle apposite sezioni della tabella sottostante.

Latenza media in ms ($\bar{x}$)		
IPFS	MAM	Totali
419,05	18712,5	19131,55
Deviazione Standard in ms (s)		
439.22	6762.45	6661.61
Intervallo di Confidenza in ms		
419,05±205,66	18712,5±3166,40	19131,55±3119,18

Conclusioni

I sistemi basati su DLT possiedono un considerevole potenziale di miglioramento per ciò che concerne lo scambio di informazioni di carattere sanitario. Il loro design, tuttavia, deve essere conforme alle recenti normative riguardanti il trattamento dei dati personali, al fine di minimizzare il rischio di elaborazione e/o conservazione impropria delle informazioni.

In questo studio, si è valutato il potenziale di una specifica DLT, IOTA, impiegata congiuntamente al file system distribuito IPFS, per lo scambio di dati sanitari, progettando e sviluppando un programma che rispetti le significative considerazioni normative, garantendo sicurezza e immutabilità. Tali sistemi devono mirare a soddisfare prestazioni esaurienti relativamente all'usabilità, dal punto di vista dell'utente. Pertanto, sono state effettuate delle analisi prestazionali sul modello realizzato in presenza di determinate condizioni.

In questo elaborato vengono affrontati alcuni dei punti di tensione tra GDPR e DLT, proponendo un progetto che consente lo scambio di PHI utilizzando questa tecnologia. Tuttavia, un possibile sviluppo futuro deve affrontare in modo più dettagliato i restanti punti di conflitto, come le metodologie di occultamento dei dati, siccome non vi è chiarezza su quando è possibile considerare i dati anonimizzati, e la gestione del consenso espresso da un individuo per il trattamento delle sue informazioni.

Infine è importante delineare alcune limitazioni relative al progetto proposto in questa tesi. In primo luogo, lo scambio delle chiavi tramite chip non è pratico in quanto richiede che gli studi medici siano forniti di un lettore pertinente collegato all'EHR della struttura, se si vuole garantire piena autonomia nel sistema. Una possibile alternativa riguarda la

realizzazione di un protocollo per la comunicazione delle chiavi tramite una rete wireless locale e sicura. In secondo luogo, l'impiego di un dispositivo di monitoraggio bluetooth per la generazione dei record FHIR a cadenza regolare comporta un significativo utilizzo della batteria del device. Queste implicazioni possono costituire una limitazione rilevante che richiede ulteriori approfondimenti.

Per concludere è importante sottolineare che gli esperimenti condotti in questo studio sono stati effettuati su un elaboratore con CPU Intel Core i5-6200U @2.30GHz, 8.00GB di RAM DDR3. Pertanto, è possibile che dispositivi con risorse disponibili differenti possano richiedere tempi diversi per generare e collegare una risorsa FHIR ai protocolli IPFS e MAM.

Al giorno d'oggi è necessario che tutto il sistema sanitario disponga di una rete affidabile e interconnessa, che garantisca una gestione ottimale del paziente, fornendo assistenza sanitaria di alta qualità grazie alla sua profilazione all'interno dei registri della rete. In quest'ottica, l'impiego del file system distribuito IPFS, come archivio crittografico dei documenti contenenti i parametri glicemici, in combinazione a MAM, utilizzato per costruire una cronologia indelebile, immutabile e sicura con cui risalire a questi valori, rappresentano una valida applicazione conforme alle recenti normative sulla privacy dei dati, la quale riduce al minimo i rischi di elaborazione inappropriata delle informazioni e restituisce prestazioni e usabilità soddisfacenti. Ogni progetto che mira al raggiungimento di questo obiettivo è fondamentale per riuscire ad individuare gradualmente i mezzi con cui poter realizzare il sistema desiderato.

Bibliografia

[1] J Med Internet Res, Designing a Distributed Ledger Technology System for Interoperable and General Data Protection Regulation–Compliant Health Data Exchange: A Use Case in Blood Glucose Data,
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6595943/

[2] EUR-Lex, Access to European Union Law,
https://eur-lex.europa.eu/legal-content/IT/TXT/?uri=uriserv
:OJ.L_.2016.119.01.0001.01.ITA&toc=OJ:L:2016:119:TOC

[3] FHIR Overview,
https://www.hl7.org/fhir/overview.html

[4] Mauro Bellini, Che cosa sono e come funzionano le Blockchain Distributed Ledgers Technology - DLT,
https://www.blockchain4innovation.it/esperti/cosa-funzionano-le-blockchain
-distributed-ledgers-technology-dlt/

[5] Ameer Rosic, What is Blockchain Technology?
https://blockgeeks.com/guides/what-is-blockchain-technology/

[6] Computational and Structural Biotechnology Journal (volume 16, 2018), Authenticating Health Activity Data Using Distributed Ledger Technologies,
https://www.sciencedirect.com/science/article/pii/S2001037018300345

[7] IOTA Documentation: What is IOTA? IOTA overview,
https://docs.iota.org/docs/getting-started/0.1/introduction/overview

[8] IOTA Documentation: Masked Authenticated Messaging,
https://docs.iota.org/docs/client-libraries/0.1/mam/introduction/overview

[9] ABMushi, IOTA: MAM Eloquently Explained,
https://medium.com/coinmonks/iota-mam-eloquently-explained-d7505863b413

[10] Wikipedia, Merkle tree,
https://en.wikipedia.org/wiki/Merkle_tree

[11] Kyle Drake, HTTP is obsolete. It's time for the distributed, permanent web
https://ipfs.io/ipfs/QmNhFJjGcMPqpuYfxL62VVB9528NXqDNMFXiqN5bgF
YiZ1/its-time-for-the-permanent-web.html

[12] IPFS Documentation, What is IPFS?
https://docs.ipfs.io/introduction/overview/

[13] IPFS Documentation, How IPFS Works
https://docs.ipfs.io/introduction/how-ipfs-works/

[14] IPFS Documentation - Getting Started,
https://docs.ipfs.io/introduction/usage/

[15] Git, IPFS - spawn a node and add a file to IPFS network
https://github.com/ipfs/js-ipfs/tree/master/examples/ipfs-101

[16] MAM Client JS Library,
https://github.com/iotaledger/mam.client.js

[17] Mirko Zichichi, middleMAM
https://github.com/miker83z/middleMAM/blob/master/src/MAMChannel.js

[18] MAM Explorer, https://mam-explorer.firebaseapp.com/

[19] burstIQ, https://www.burstiq.com/

[20] Giuseppe Verlato - Roberto de Marco, Intervallo di Confidenza
http://biometria.univr.it/sesm/files/lezione_8.pdf

Ringraziamenti

Giunto alla conclusione di questo lavoro, volevo ringraziare innanzitutto il professor Stefano Ferretti, e il dottor Mirko Zichichi. Essi mi hanno dato la possibilità di lavorare su una tematica interessante e stimolante, facendomi conoscere e apprezzare il mondo della ricerca universitaria. Ciò per cui li ringrazio maggiormente, però, è di avermi permesso di lavorare in un clima amichevole e informale, e di essersi sempre dimostrati disponibili a fornirmi utili consigli e spiegazioni per superare i problemi che man mano ho incontrato nello sviluppo del presente lavoro. Tutto questo mi ha permesso di imparare moltissimo e, in generale, di trarre il meglio da questa esperienza.

Non posso fare a meno di ringraziare anche i miei genitori che, nonostante le inevitabili incomprensioni, mi hanno sempre sostenuto, non mi hanno mai fatto mancare nulla, nemmeno le cose più superflue, e mi hanno permesso di arrivare dove sono. Così come non posso non ringraziare mio fratello e Sara, che con i loro due bimbi piccoli mi hanno regalato dei bellissimi momenti da zio.

Ci tengo a ringraziare i miei nonni, che nei momenti di sconforto mi hanno risollevato il morale con un bel piatto di tagliatelle al ragù e un bicchiere di vino.

Inoltre un ringraziamento speciale va ai miei colleghi e amici Kev, Pablo, Giovi, Fede, Inge, Andre e Albert, con cui ho condiviso momenti indimenticabili, sia dentro che fuori le mura dell'Università.

Infine desidero ringraziare Chiara, che con il suo sorriso è sempre riuscita a mettermi di buon umore e che, nel bene e nel male, è la persona con cui ho condiviso tutta la vita. In lei ho trovato non solo una compagna, ma anche un'amica, una spalla, una complice. Averla al mio fianco mi rende la persona più fortunata del mondo.

Per concludere e per essere sicuro di non dimenticare nessuno, voglio ringraziare tutti i parenti e gli amici che ho avuto vicino in questi anni: chi più chi meno, hanno tutti contribuito a rendermi la persona che sono oggi.