POLITECNICO DI TORINO

POLITECNICO DI TORINO
Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale

Mobility Data Marketplace: un'applicazione basata sulla tecnologia IOTA

Relatori
Prof. Giovanni Malnati
Prof. Guido Perboli
Ing. Edoardo Calia

Candidato
Jure Rosso

Anno Accademico 2017-2018

Ringraziamenti

Desidero ringraziare il Professor Giovanni Malnati ed Edoardo Calia per avermi messo a disposizione la loro competenza e avermi aiutato nella realizzazione di questa tesi.

Ringrazio l'Istituto Superiore Mario Boella per aver messo me e i miei colleghi nelle migliori condizioni per lavorare e per avermi dato la possibilità di prendere parte alla Blockathon EU.

Ringrazio i miei compagni di tesi che hanno lavorato con me e tutte le persone del SiTI con cui abbiamo condiviso l'ufficio in questi mesi per la loro simpatia e disponibilità.

Ringrazio tutti gli amici che mi hanno accompagnato durante il percorso universitario per tutti i bei momenti passati insieme.

Ringrazio la mia ragazza Francesca che mi è stata vicina in tutti questi anni.

Ringrazio ovviamente la mia famiglia per il supporto che mi ha sempre dato permettendomi di concentrarmi al massimo sui miei obiettivi.

Una dedica particolare va a mio nonno, una delle persone più speciali con cui abbia avuto il piacere di condividere questa parte della mia vita.

Sommario

Negli ultimi anni si è riscontrato un interesse sempre maggiore verso le *Distibuted Ledger Tecnology* (DLT), architetture distribuite che sfruttano protocolli di tipo Peer-to-Peer per dare vita a sistemi sicuri anche in completa assenza di fiducia (*trust*) tra gli attori coinvolti. Tra le applicazioni più note delle DLT figura la gestione di criptovalute e token associati, e il loro scopo primario in questo caso è di garantire il corretto scambio di questi asset in modo del tutto sicuro e trasparente, senza dover sottostare alla supervisione di un ente centrale. Tutti questi sistemi hanno infatti come motivazione fondamentale la completa decentralizzazione della gestione. Per questo motivo le macchine che compongono la rete vengono chiamate *nodi* e hanno tutte pari diritto nel decidere riguardo alle sorti della DLT.

Dopo l'avvento della prima di queste tecnologie, denominata Blockchain e introdotta con la nascita di Bitcoin, ne sono state create diverse, ognuna con specifici elementi caratterizzanti. Nel corso della tesi è stata presa in esame una particolare DLT, denominata IOTA. La struttura del *ledger* associato a IOTA è pensata per applicazioni nel contesto dell'Internet-of-Things (IOT), e per aprire la strada a paradigmi quali *Machine-to-Machine* e in particolare alla *Machine Economy*.

L'insieme delle scelte fatte dagli ideatori del progetto IOTA è l'unione di soluzioni ai problemi riscontrati nelle DLT precedenti e di innovazioni per rispondere alle criticità esposte dal mondo IOT. Tra le innovazioni più significative figura proprio la struttura del ledger, chiamata *Tangle* e costituita da un grafo diretto aciclico (DAG) che contiene tutte le transazioni effettuate; l'inserimento di una transazione comporta lo svolgimento di un lavoro computazionale chiamato *Proof of Work* (anche detta PoW, un calcolo di forza bruta che consiste nel trovare un hash con certe caratteristiche e che richiede risorse e tempo proporzionali alla sua difficoltà) di entità minore rispetto a quello tipico delle blockchain, il quale viene svolto dal nodo che intende inserire una nuova transazione. Ed è proprio questa una seconda caratteristica innovativa che differenzia IOTA dalla maggior parte delle DLT precedenti, nelle quali le transazioni sono inserite all'interno di una struttura più complessa (detta *blocco*) e vengono registrate grazie a utenti con ruoli particolari denominati *miner*, i quali competono tra loro per riuscire a svolgere per primi la PoW per la generazione di blocchi che si susseguono in una struttura lineare. Un ulteriore punto di forza di IOTA è la totale assenza di *fee* per l' emissione di una transazione. Le fee costituiscono parte del meccanismo di incentivazione per i miner nel momento in cui debbano scegliere una transazione da includere in un blocco; la natura fee-less di IOTA si presta molto bene all'utilizzo del sistema per effettuare micro-transazioni o addirittura per inviare messaggi o per la semplice memorizzazione di dati nel Tangle a costo zero utilizzando transazioni che non comportano lo scambio di token. Infine un altro tema sul quale IOTA risulta vincente è quello della scalabilità: con l'aumentare della frequenza delle transazioni il sistema guadagna in sicurezza e stabilità. Non si può dire lo stesso delle architetture blockchain classiche, le quali hanno un numero massimo di transazioni al secondo accettate dovuto alla dimensione dei blocchi e alla frequenza fissa della loro generazione.

Durante il lavoro è stata analizzata nel dettaglio l'architettura: partendo dalla documentazione ufficiale disponibile online, la comprensione dei meccanismi operativi ha richiesto l' osservazione e analisi del codice sorgente del software dei nodi e delle librerie disponibili sui repository ufficiali. Per reperire alcune delle informazioni più complesse in merito a ultimi sviluppi, algoritmi e funzionalità avanzate è stato necessario stabilire contatti diretti con gli sviluppatori coinvolti nello sviluppo della architettura IOTA.

In particolare la partecipazione ad alcuni webinar organizzati dalla IOTA Foundation ha consentito al gruppo di tesisti di accedere come partecipanti attivi ad uno dei primi progetti lanciati dal team nel Dicembre 2017: la sperimentazione di un Data Marketplace al quale i più svariati dispositivi potessero conferire dati per essere successivamente remunerati (in token IOTA) qualora un compratore ne avesse richiesto l'acquisto. Questo meccanismo si è rivelato molto interessante, come anche la modalità in cui veniva utilizzato il sistema IOTA per raggiungere questo obiettivo. IOTA supporta l'aggiunta di moduli di estensione conosciuti come *Iota eXtension Interface* (IXI) grazie ai quali è possibile aggiungere funzionalità particolari al sistema. Un modulo di estensione chiamato *Masked Authenticated Messaging* (MAM) aggiunge la possibilità di creare flussi (o *stream*) di dati all'interno del Tangle concatenando tra loro le transazioni appartenenti allo stesso flusso logico mediante puntatori contenuti nel payload del messaggio. Il meccanismo base del MAM offre la possibilità di creare canali virtuali sui quali è possibile "sintonizzarsi" per seguire lo stream dei dati dal punto di accesso in poi: è sufficiente conoscere l'indirizzo di una transazione appartenente flusso per poter leggere tutte le successive. Il MAM offre varie modalità di funzionamento, corrispondenti a diversi livelli di sicurezza, dalla visibilità pubblica dei dati fino a quella totalmente oscurata tramite tecniche crittografiche.

Questo modulo di estensione è stato analizzato e utilizzato durante lo svolgimento della tesi per la progettazione di un sistema battezzato *Mobility Data Marketplace* (MDM). Alla base vi è l'idea che ogni utente possa ricavare un guadagno condividendo i dati della propria mobilità. Si tratta di dati che già oggi vengono acquisiti dai nostri dispositivi mobili da operatori di TLC o altre società di servizi, le quali li utilizzano per costruire un mercato e un business sui quali l'utente ha solo una limitata visibilità. MDM è un sistema che si propone di dare la possibilità all'utente di condividere in modo del tutto anonimo e automatico i dati della propria mobilità, traendone un guadagno quando questi verranno venduti. Il tutto è pensato per lasciare all'utente il controllo dei propri dati e il possesso di tutte le informazioni sensibili senza che il server di gestione del servizio MDM abbia la possibilità di intervenire. Questo meccanismo è pensato per rispettare la filosofia delle DLT, dove il controllo è il più possibile distribuito e non vi è nessuna entità centrale su cui fare affidamento. Per questo l'uso di una DLT come IOTA e del modulo di estensione sono stati fondamentali: i dati completi vengono firmati e cifrati direttamente sul client sfruttando le funzionalità del MAM, le chiavi rimangono all'utente e sarà suo compito proteggerle; i dati passano dal server già criptati, lasciando visibili solamente le parti utili per la presentazione e la vendita. Ogni utente da' origine ad un flusso di dati distinto che può essere letto in seguito alla comunicazione della chiave di cifratura dello stream dopo l'eventuale acquisto da parte del cliente. Grazie al MAM la granularità dei pacchetti di dati cifrati con la stessa chiave può essere cambiata in base alle necessità. La permanenza dei dati sul Tangle garantisce la loro immutabilità e li rende disponibili ad essere recuperati

senza doversi rivolgere a intermediari: un applicativo software di tipo client può utilizzare le funzionalità del MAM per leggere i dati una volta ricevute le informazioni relative agli stream di interesse.

Sono stati sviluppati i moduli per l'invio e la ricezione dei dati e sono stati effettuati alcuni test sull'effettivo funzionamento e sulle prestazioni. Il resto del sistema è stato solamente progettato e potrebbe essere interessante da studiare e sviluppare in lavori futuri. L'obiettivo di rendere operativi i moduli è stato raggiunto ed è stato possibile fare alcune considerazioni. IOTA in quanto tecnologia giovane ha ancora molte limitazioni sia in termini di architettura che di sviluppo. L'attuale assenza di una funzionalità come quella degli *Smart Contract* (porzioni di codice salvate nella DLT che vengono eseguite sotto il controllo dei nodi della rete al verificarsi di alcune condizione) ha comportato delle scelte come quella di aggiungere un server intermedio e quella di uno scambio di chiavi diretto tra utente e acquirente. Le librerie ancora acerbe del MAM non hanno permesso di gestire liberamente le chiavi con cui vengono firmati i messaggi o di attivare funzionalità come quella del *subscribe* per permettere di leggere i dati in tempo reale, mano a mano che vengono pubblicati.

Parlando di prestazioni le operazioni di lettura e invio sono risultate costose in termini di tempo. I continui processi di cifratura e decodifica dei messaggi sono inefficienti poiché è necessario farlo per ogni dato inviato o ricevuto. Durante la visualizzazione dei dati questo impatta molto sui tempi poiché decifrare centinaia di migliaia di dati richiederebbe molte ore. Due accorgimenti come il raggruppamento di più informazioni in un singolo messaggio e le scansioni di stream diversi in parallelo diminuiscono di gran lunga i tempi di lettura.

In generale però la PoW di IOTA è l'operazione più limitante in quanto anche su PC desktop di medie prestazioni richiede un tempo nell'ordine delle decine di secondi, che rende impensabile l'invio di transazioni con una frequenza molto alta sfruttando un singolo nodo. Per vedere miglioramenti bisogna o utilizzare nodi più potenti o aumentarne il numero per distribuire il carico di lavoro oppure attendere l'acceleratore hardware ancora in fase di sviluppo da parte della IOTA Foundation.

Tralasciando questi problemi il MAM si è dimostrato un modulo con enorme potenziale, capace di esaltare la natura fee-less di IOTA e abilitando alla possibilità di utilizzare il Tangle per lo scambio e la memorizzazione di informazioni.

La flessibilità dei moduli sviluppati ha inoltre permesso di riadattarli per un'altra applicazione progettata durante la competizione Blockathon EU. Il concept consisteva in un sistema per la tracciabilità e la certificazione di un prodotto autentico, il quale viene seguito durante il suo percorso dal produttore al cliente finale e i dati generati mediante scansione di un tag legato all'oggetto vengono memorizzati nel Tangle.

In conclusione IOTA si è rivelata una tecnologia innovativa con molti pregi e alcuni difetti dovuti alla sua gioventù, sarà molto interessante seguire il suo sviluppo nei prossimi anni e osservare come verrà applicata.

Indice

1 Introduzione 1
- 1.1 Distributed Ledger Technology . . . 1
- 1.2 Blockchain . . . 1
  - 1.2.1 Tipi di Nodo . . . 3
  - 1.2.2 Mining e Fee . . . 4
  - 1.2.3 Sicurezza . . . 5
  - 1.2.4 Problemi Principali . . . 6
- 1.3 Tangle . . . 8

2 IOTA 9
- 2.1 Introduzione a IOTA . . . 9
- 2.2 Caratteristiche Principali . . . 10
  - 2.2.1 Decentralizzazione Reale . . . 10
  - 2.2.2 Scalabilità . . . 11
  - 2.2.3 Zero Fee . . . 12
  - 2.2.4 Quantum-Immunity . . . 12
- 2.3 Il Tangle . . . 12
- 2.4 I Nodi . . . 13
  - 2.4.1 Full Node . . . 13
  - 2.4.2 Light Node . . . 15
- 2.5 Transazioni . . . 15
  - 2.5.1 Generazione degli Indirizzi . . . 16
  - 2.5.2 Consolidamento del Bundle . . . 18
- 2.6 Consenso . . . 22
- 2.7 Sicurezza . . . 30
- 2.8 Coordinator . . . 30
- 2.9 Snapshot e Permanode . . . 32
- 2.10 Moduli di Estensione . . . 33

3 Masked Authenticated Messaging 35
- 3.1 Introduzione al MAM . . . 35
- 3.2 Considerazioni sul MAM . . . 36
- 3.3 Modalità di Funzionamento . . . 37
  - 3.3.1 Public . . . 37

3.3.2 Private . . . 37
3.3.3 Restricted . . . 37
3.4 Proprietà . . . 38
3.4.1 Forward Secrecy . . . 38
3.4.2 Channel Splitting . . . 38
3.5 Merkle Tree . . . 39
3.6 MAM Object . . . 41
3.7 MAM Payload . . . 42
3.7.1 Generazione del Merkle Tree . . . 42
3.7.2 Inserimento del Messaggio . . . 44
3.7.3 Calcolo del Nonce . . . 44
3.7.4 Calcolo della Firma . . . 45
3.7.5 Calcolo dei Figli . . . 46
3.8 Lettura del Flusso . . . 47

4 Mobility Data Marketplace 49
4.1 Introduzione . . . 49
4.2 Concept . . . 50
4.3 Vantaggi del Mobility Data Marketplace . . . 50
4.4 Problemi da Affrontare . . . 51
4.5 Architettura del Sistema . . . 52
4.5.1 Utente . . . 52
4.5.2 Server . . . 55
4.5.3 Acquirente . . . 56
4.6 Risultati . . . 57
4.6.1 Prestazioni . . . 58
4.6.2 Discussione delle Problematiche . . . 59
4.7 Sviluppi Futuri . . . 60

5 Blockathon EU 63
5.1 Introduzione . . . 63
5.2 Le Sfide . . . 63
5.3 Il Progetto . . . 64
5.4 Utilizzo di IOTA . . . 66
5.5 Considerazioni . . . 67
5.6 Risultati . . . 68

6 Conclusioni 69

Bibliografia 71

Capitolo 1

Introduzione

1.1 Distributed Ledger Technology

Con il termine Distributed Ledger Technology (DLT) si fa riferimento ad una famiglia di tecnologie dedicate alla gestione di insiemi di dati e transazioni con elevate caratteristiche di robustezza e sicurezza grazie all'uso di tecniche crittografiche e alla replica (con sincronizzazione) su numerosi nodi che operano come una rete Peer-to-Peer.

Le DLT svolgono la funzione di memorizzazione dei dati all'interno di una struttura distribuita e replicata in modo che nessuno sia in grado di modificarne il contenuto. La caratteristica di non-modificabilità implica quindi che i dati possano essere inseriti all'interno del registro senza poter essere successivamente modificati o eliminati.

I nodi che fanno parte di questa architettura non fanno riferimento a nessuna entità centrale, non vi è quindi nessun attore che svolga un ruolo di gestore del sistema. Non essendoci nessuna autorità volta a supervisionare i nodi, come è possibile garantire che non ci siano attori che agiscano in modo malevolo? Il corretto funzionamento è raggiunto attraverso un algoritmo di consenso. Questo meccanismo permette di decentralizzare completamente la gestione del registro, evitando di fare affidamento su di un singolo nodo, il quale potrebbe rappresentare un single-point-of-failure. Mentre in un normale sistema di gestione di database distribuiti si ha una distribuzione e replica dei dati con una gestione centralizzata, in questo scenario si ha anche una divisione del potere decisionale. É quindi questa la caratteristica principale offerta dalle architetture DLT, ovvero la perfetta operabilità anche in assenza di fiducia tra gli attori.

1.2 Blockchain

La prima *Blockchain* distribuita di cui si ha notizia risale al 2008, sviluppata da una persona o da un gruppo di persone, sotto lo pseudonimo di Satoshi Nakamoto. Si parla spesso di Blockchain legata alla criptovaluta Bitcoin poiché essa rappresenta la prima DLT operativa nella storia, e per questo motivo tutte quelle che sono venute dopo prendono questa come punto cardine. Anche in questo documento ogni esempio specifico sarà fatto riferendosi alla blockchain di Bitcoin.

La parola che da il nome a questa tecnologia si riferisce alla struttura dati utilizzata, ovvero una "catena di blocchi", una sorta di lista in cui ogni blocco è collegato a quello successivo. Nel sistema Bitcoin l'operazione base è rappresentata dallo scambio di criptovaluta tra due portafogli tramite una transazione. Un blocco non è altro che un raccoglitore di transazioni, le quali vengono registrate al suo interno per poi essere appese in testa alla blockchain. Una volta effettuata questa operazione non sarà più possibile cancellare o modificare l'informazione scritta, la quale rimarrà per sempre immutata all'interno del sistema.

Un portafoglio è comunemente chiamato *wallet* e non è altro che un sistema di chiavi asimmetriche in cui la chiave privata permette di firmare le transazioni certificando la loro provenienza, mentre la chiave pubblica o un suo derivato funge da indirizzo per l'invio o la ricezione. Il possesso di chiavi opportunamente formate sancisce il controllo su di un wallet che potrà essere utilizzato all'interno del sistema per lo scambio del bene virtuale ad esso associato.

In questo scenario distribuito non si ha una copia di riferimento del registro, questo viene mantenuto dai nodi che costituiscono la rete, ognuno di loro avrà una copia di tutta la catena, dal primo all'ultimo blocco. Qualsiasi nodo (in questo caso chiamato "miner") che partecipa alla rete può immettere blocchi nel sistema.

L'architettura sfrutta una serie di funzioni crittografiche per garantire che tutto proceda correttamente e che nessuno riesca nel tentativo di utilizzare il sistema in modo non consentito.

La Blockchain può essere visualizzata come una lista linkata, in cui ogni blocco punta al precedente. Il collegamento logico si ha poiché nell'header di ogni blocco viene inserito l'hash (SHA256) di quello precedente, questa operazione prende il nome di *conferma*, poiché continuando la catena di blocchi si assume come "verificata".

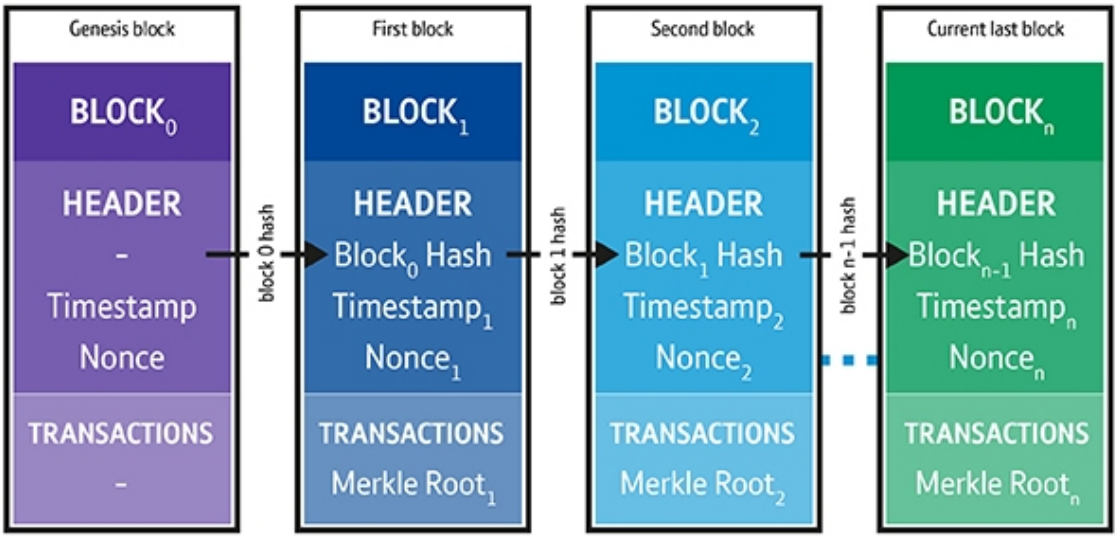

Figura 1.1: Ogni blocco contiene l'hash del precedente

In questa sequenza ordinata i blocchi sono numerati a partire dal cosiddetto *genesis block*, ovvero quello che ha dato inizio alla catena, che nel caso della blockchain di Bitcoin è stato originato nel Gennaio 2009. Il termine *height* indica proprio il numero di un blocco dall'inizio della storia della catena.

In un dato istante è possibile che ci siano due nuovi blocchi che referenziano lo stesso padre, la catena non appare più lineare ma presenta una biforcazione. In questo scenario ci si trova di fronte al fenomeno del *fork*, il quale si verifica quando due blocchi vengono emessi quasi simultaneamente. Questo accade a causa della concorrenza del sistema, dove più attori sono impegnati a completare l'operazione più velocemente degli altri. I sistemi di questo tipo hanno una proprietà intrinseca detta *eventual consistency*: è possibile che due nodi della rete vedano nello stesso tempo due immagini della blockchain differenti, l'unione di queste è una catena che presenta uno o più fork. La risoluzione di un fork si raggiunge quando uno dei due rami cresce più di un altro, sarà quindi riconosciuto come parte della catena principale e soppianterà completamente l'altro, questa regola è conosciuta come *longest chain rule*.

L'eventual consistency è, insieme a *availability* e *partition tolerance*, oggetto del noto *CAP Theorem* [6] il quale afferma che in un sistema distribuito è impossibile avere tutte e tre le proprietà verificate contemporaneamente. La blockchain è sempre disponibile ad essere consultata, ma non è detto che ogni componente della rete abbia la stessa immagine di essa, poiché entrano in gioco problematiche relative alla distribuzione del sistema, ovvero ritardi della rete fisica o addirittura guasti che rendono la sincronizzazione non istantanea. Per soddisfare la consistenza bisognerebbe mostrare un risultato di una richiesta relativa ad una transazione solo quando questa sia confermata dal sistema, ma questo renderebbe non disponibile l'informazione fino a quel momento.

1.2.1 Tipi di Nodo

Nell'architettura Blockchain si può effettuare una distinzione tra i tipi di nodo che danno vita al sistema:

- I nodi core o *Full Node* durante la configurazione effettuano il download di tutto il database della Blockchain, il loro compito sarà quello di mantenere sincronizzata la loro copia di dati e di rimanere in collegamento con gli altri nodi della rete.

- I nodi client o *Lightweight Node* emettono semplicemente transazioni e leggono i dati relativi al proprio wallet, essi non hanno una copia dell'intera base dati, ma sfruttano un meccanismo chiamato SPV (Simplified payment verification) il quale permette la verifica delle transazioni con la quantità minore di dati possibile.

- I nodi *Miner* sono quelli dedicati alla creazione dei blocchi, oltre a mantenere una copia dei dati al loro interno, hanno anche il compito di eseguire le onerose operazioni di calcolo che permettono la formazione di un blocco.

Come mostrato in figura 1.2 la rete si costruisce a partire da questi tre tipi di nodo.

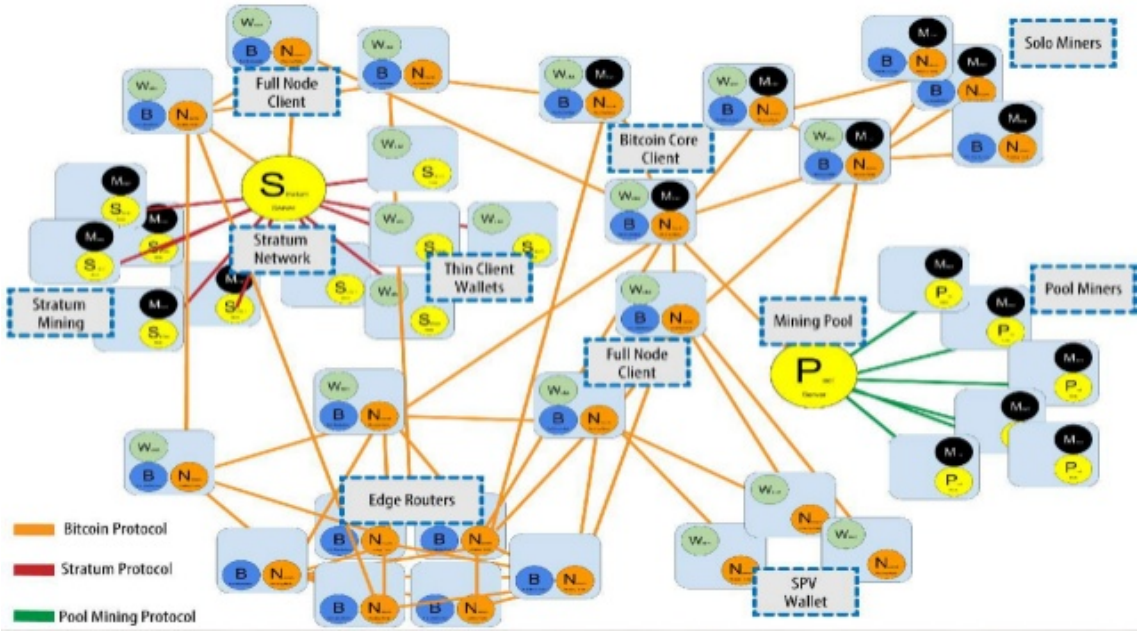

Figura 1.2: Rappresentazione della rete Bitcoin

1.2.2 Mining e Fee

Il *mining* è l'operazione necessaria per l'aggiunta di una transazione al registro distribuito, questo avviene attraverso la creazione di un blocco da parte dei miner.

Il calcolo eseguito per la generazione di un blocco è un'operazione di forza bruta molto costosa in termini di energia elettrica e tempo che prende il nome di *Proof of Work* (PoW). Un gran numero di miner concorrono per trovare la stessa soluzione e solo uno di essi avrà successo. La difficoltà della PoW viene modificata dinamicamente.

La difficoltà dell'operazione di mining è intenzionalmente costosa in termini di risorse utilizzate ed è variabile nel tempo per riuscire mantenere una frequenza di generazione blocchi mediamente costante, con una soluzione valida ogni circa dieci minuti (dati relativi alla blockchain di Bitcoin). Il secondo motivo della difficoltà di questa operazione è quello di rendere la blockchain immodificabile, in quanto lo sforzo che un attaccante dovrebbe fare sarebbe insostenibile.

L'operazione è complicata poiché consiste nel trovare un hash SHA256 dell'header del blocco che risulti inferiore ad un certo valore *target*. Per fare ciò è necessario un *nonce*, inserito nell'header ed incrementato ad ogni tentativo. La probabilità che un hash risulti inferiore ad un certo valore è bassa, perciò servono mediamente moltissimi tentativi prima di ottenere una soluzione valida.

La *difficoltà* è il parametro che indica il quanto sia complicata l'operazione di mining. Il suo valore viene ricalcolato ogni 2016 blocchi, in modo da mantenere costante la frequenza di generazione (un blocco ogni circa dieci minuti). L'arrivo di nuovi miner e quindi di potenza di calcolo comporta un incremento del rate di generazione e conseguentemente anche della difficoltà.

Il *block reward* è il compenso che un miner riceve in seguito all'avvenuta creazione di un blocco valido. Questo consiste in un certo numero di token di criptovaluta, attualmente pari a 12.5 Bitcoin ma il suo numero viene ricalcolato ogni 210.000 blocchi. La *Fee* invece è una quantità di criptomoneta che viene associata ad una transazione per remunerare colui che la inserirà con successo in un blocco, più alta sarà la somma e più sarà appetibile per i miner che dovranno selezionare le transazioni.

Colui che per primo riuscirà a portare a termine con successo l'operazione avrà *minato* un blocco e verrà ricompensato con una determinata quantità di criptovaluta, il cui valore dipende da una parte fissa (*block reward*) e una variabile (*transaction fees*) data dalla somma di tutte le fee associate ad ogni transazione all'interno del blocco.

Il mining veniva inizialmente praticato sfruttando la potenza di calcolo delle CPU. Successivamente ha cominciato a prendere piede il mining tramite GPU, le quali offrono una potenza di hash superiore e molto più vantaggiosa in termini economici. Negli ultimi anni invece sono stati sviluppati degli ASIC dedicati a svolgere l'attività di mining, raggiungendo livelli di efficienza altissimi e soppiantando i vecchi metodi. Per accrescere ancor di più la potenza di hash con lo scopo di minare più blocchi rispetto agli altri miner, alcuni di essi hanno unito le loro forze nelle cosiddette *mining pool*. All'interno di questi gruppi vengono divisi i compensi provenienti dai blocchi minati. Attualmente non è più possibile per un singolo miner competere contro le grandi mining pool, sono quindi obbligati ad entrare in uno di questi gruppi per operare insieme agli altri.

1.2.3 Sicurezza

L'integrità dei dati all'interno del sistema è garantita dalla quantità enorme di calcolo che viene sfruttata per costruire la blockchain. Prendendo in esame lo schema descritto precedentemente é possibile comprendere quanto sia difficile modificare a proprio piacimento una porzione di dati salvata in questo tipo di DLT.

Essendo ogni blocco descritto dal risultato di una funzione di hash, ogniqualvolta si modificherà uno dei dati in esso contenuti il suo digest cambierà. Siccome ogni blocco contiene l'hash del precedente, la modifica di un blocco già confermato (con almeno un blocco davanti) comporterebbe l'inconsistenza con la sotto-catena successiva. Supponendo di modificare un blocco con altri N blocchi davanti ci si troverà in uno scenario in cui sarà necessario ricalcolare anche i successivi N, in quanto dovranno essere forgiati a partire dal nuovo hash del blocco modificato. Questa operazione si rivela essere impraticabile considerando la potenza di calcolo che dovrebbe avere a disposizione un attore malevolo intenzionato a modificare i dati. Come già detto la generazione di un blocco è un'operazione molto costosa in termini energetici e temporali, il successo di tale operazione a intervalli di dieci minuti è garantito da calcoli probabilistici, basati sul fatto che tutti i miner della rete concorreranno per risolvere la PoW. Un singolo attore difficilmente riuscirebbe a forgiare blocchi più velocemente del resto della rete.

Questa difficoltà intrinseca assicura quindi che blocchi abbastanza profondi non possano essere modificati, ma i blocchi vicini alla testa della catena non sono del tutto immuni ad attacchi, in quanto è possibile che degli attaccanti riescano con successo a modificare la porzione più giovane della blockchain. Rimane comunque un'operazione costosa, non

conveniente per gli attori malevoli nel caso l'azione non comporti più entrate rispetto alle spese necessarie a compiere l'attacco stesso. In ogni caso è consigliato attendere che un blocco sia distante in altezza (height) di almeno cinque unità prima di considerare "sicura" una transazione, e spendere i token derivanti da essa con una tranquillità data dalla probabilità pressoché nulla che un attacco venga messo a segno.

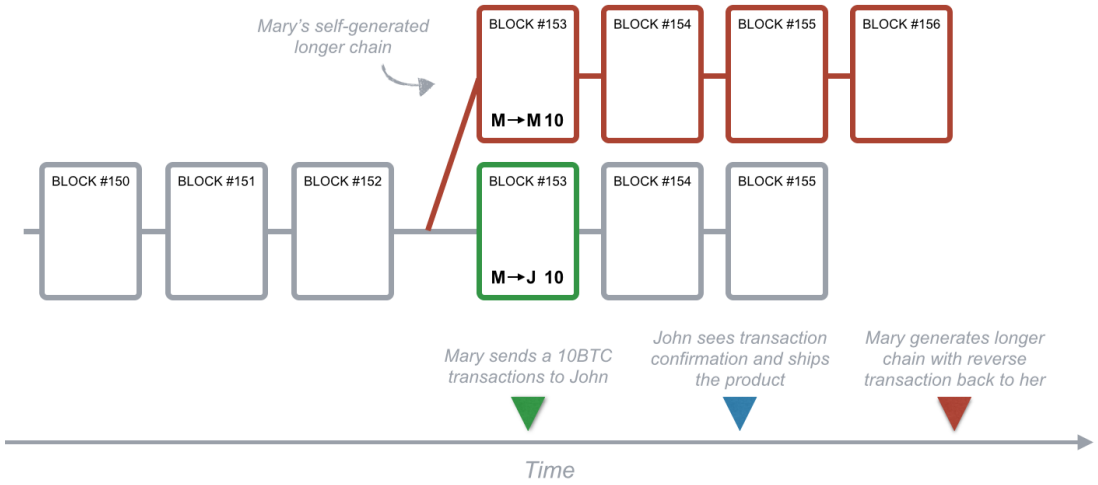

Figura 1.3: Esempio di attacco Double Spending

Un possibile attacco basato su questo concetto è quello del *Double Spending* [3]. Tenendo a mente il concetto di fork, questa tecnica prevede la sostituzione di una parte della catena in modo tale da imporre blocchi modificati dall'attaccante (come mostrato nella figura 1.3). In questo modo sarebbe possibile spendere una volta i propri token nella catena principale e, una volta ottenuti i beni, spendere nuovamente i soldi in una catena secondaria forgiata ad-hoc e più lunga di quella principale. Ci sono diverse tipologie di questo attacco, il più famoso è sicuramente l'*attacco del 51%*. Questo è possibile se si detiene il 51% della potenza di hash di tutta la rete, non ci sono contromisure contro questo tipo di attacco poiché la rete sarebbe in mano ad attori malevoli e completamente corrotta. La blockchain è basata sulla distribuzione e decentralizzazione della potenza di calcolo quindi è inverosimile che una singola mining pool possa detenere più di metà di essa. Ed è questo un concetto fondamentale su cui si basa tutto il sistema, in caso questo presupposto dovesse venire a mancare il sistema smetterebbe di funzionare.

1.2.4 Problemi Principali

Nonostante la blockchain associata a Bitcoin abbia introdotto una vera e propria rivoluzioni, essa ha evidenziato alcuni problemi nel suo sistema. Di seguito verranno presentati i problemi più interessanti per il seguito di questo documento.

Centralizzazione del mining Come precedentemente detto, è diventato controproducente per un singolo miner cercare di creare nuovi blocchi da solo a causa della grande

difficoltà della Proof-of-Work. Questo li costringe ad entrare a far parte delle mining pool per riuscire a trarne un profitto. L'aggregazione di potenza di calcolo introduce un controsenso nell'architettura del sistema stesso: un protocollo pensato per essere completamente distribuito e decentralizzato si ritrova con tutto il potere concentrato nelle mani di poche organizzazioni. Questo rende la prospettiva di un attacco del 51% non così improbabile come si supponeva inizialmente, quando c'era la convinzione che tutto sarebbe rimasto completamente distribuito. Nel caso qualche malintenzionato riuscisse a prendere il controllo di una o più mining pool per un totale pari o maggiore al 51% dell'hash power complessivo, metterebbe il sistema in uno stato di insicurezza, poiché avrebbe il controllo sull'emissione di nuovi blocchi.

Scalabilità Con un crescente numero di token, utenti, investitori e tutte le persone interessate ad utilizzare la blockchain di Bitcoin, la scalabilità si è rivelata essere un grande problema di questa architettura. Il numero delle transazioni ha cominciato ad aumentare a dismisura, mettendo alla luce le limitazioni in termini di *transactions per second* (TPS) gestibili dalla rete.

Il TPS è uno dei parametri fondamentali per determinare le prestazioni di un sistema distribuito con funzionalità di tipo finanziario, poiché spiega quale sarà il carico massimo di transazioni gestibili.

Anche se la rete Bitcoin ha la capacità di crescere in modo illimitato sia in termini di nodi che in termini di potenza di calcolo, non si può dire lo stesso della quantità di transazioni gestibili. La limitazione è dovuta alle dimensioni di un blocco, originariamente fissata a 1MB ovvero circa 2000 transazioni (dimensione minima 250 Byte). Dato che una caratteristica del sistema è quella di generare un nuovo blocco ogni circa dieci minuti ci troviamo con un TPS che non supera l'ordine della decina, qualunque sia la disponibilità di risorse del sistema.

Un tradizionale circuito di pagamento come Visa ha un TPS nell'ordine delle decine di migliaia di transazioni al secondo, non paragonabile con quello di Bitcoin.

Fee Il problema della scalabilità si porta dietro un'altra controindicazione, ovvero quella dell'aumento smisurato delle fee. Inizialmente per immettere una transazione era sufficiente "pagare" ai miner una frazione di token di valore equivalente nell'ordine dei centesimi di dollaro. Con il crescente utilizzo del sistema e l'aumento del numero di transazioni, unite alla scarsa scalabilità, a Dicembre 2017 si è giunti ad un punto in cui affinché una transazione venisse scelta dai miner durante la creazione di un blocco, era necessario pagare una fee di un valore superiore a 20 dollari.

Quantum Resistance Il sistema Bitcoin sfrutta ECDSA-256 per eseguire la firma. Questo algoritmo non è resistente ad attacchi eseguiti con computer quantistici, in quanto, con queste macchine, sarebbe possibile rompere gli schemi di crittografia a chiave pubblica Attualmente i computer quantistici sono in fase di studio e non raggiungono una potenza necessaria per effettuare questo tipo di attacco. Si stima che potrebbero esserne in grado entro il 2040.

Attualmente ci sono dei meccanismi utilizzati per mitigare questa limitazione inseriti tramite *hard fork*, con cui è possibile cambiare parametri dell'architettura a patto che una gran parte della comunità sia a favore delle modifiche.

1.3 Tangle

Il *Tangle* [13] è una delle tante DLT che sono state sviluppate dopo l'avvento della prima Blockchain. L'architettura di questa tecnologia tenta di rimodellare la classica blockchain utilizzando una struttura a *Grafo Diretto Aciclico* (DAG) e non più una catena lineare di blocchi. Inoltre i ruoli dei nodi sono completamente ridisegnati, con la conseguente modifica del funzionamento generale di consenso e PoW. Lo stesso termine "tangle" significa "garbuglio", questo fa riferimento alla struttura dei dati, in cui le singole transazioni si legano ad altre due transazioni precedenti, dando origine ad una struttura molto più complessa della classica blockchain.

Nel corso del documento verrà analizzata più nel dettaglio questa architettura per comprenderne al meglio le proprietà e i meccanismi.

Capitolo 2

IOTA

2.1 Introduzione a IOTA

IOTA è stata fondata nel 2015 da David Sønstebø, Sergey Ivancheglo, Dominik Schiener, e Dr. Serguei Popov.

Con il termine *IOTA* ci si riferisce all'ecosistema che ha come due componenti principali il Tangle, come architettura del registro distributo, e la criptovaluta IOTA che prende il nome dal progetto stesso.

Figura 2.1: Il logo di IOTA

Il progetto nasce specificamente per creare una DLT che potesse svolgere il ruolo di backbone per i dati del mondo *Internet of Things* [8] (IoT), da cui le prime tre lettere del nome. Gli esperti stimano che entro il 2020 saranno attivi 30 miliardi di dispositivi IoT, distribuiti in ogni ambito immaginabile. I problemi derivanti dalla necessità di gestire la grande quantità di dati generata dai dispositivi IoT potrebbero essere risolti dalle tecnologie DLT e IOTA è uno dei progetti che si propone di fare ciò. Uno dei paradigmi di cui IOTA vorrebbe essere un fattore abilitante, è il *Machine to Machine* (M2M) Payments, che consiste in un'economia gestita in modo del tutto autonomo dalle macchine. Per raggiungere l'obiettivo prefissato vi è stata la necessità si superare alcuni limiti offerti dalle classiche architetture blockchain, e per questo sono state introdotte diverse innovazioni e sono state apportate diverse modifiche agli algoritmi normalmente utilizzati nelle DLT precedenti.

IOTA è un sistema per lo scambio di token (criptovalute) che in questo caso prendono il nome stesso del sistema ovvero IOTA. Al contrario di altre criptovalute come Bitcoin, che vedono la generazione di nuovi token ad ogni nuovo blocco minato (sotto forma di ricompensa per i miner), IOTA non è inflazionata. Tutti i token sono stati generati con la *genesis transaction*, la prima transazione generata quando il sistema è stato creato. Il numero di IOTA disponibili è di $(3^{33}-1)/2$ che è pari ad un totale di 2.779.530.283.277.761 IOTA. Questo elevato numero dovrebbe permettere una granularità molto fine, ottima per abilitare i sistemi di microtransazioni.

Il sistema IOTA utilizza un'aritmetica ternaria, anziché la comune binaria che si trova su praticamente tutti i calcolatori al mondo. Non vi è nessuna motivazione ufficiale riguardo a questa scelta, si parla di una presunta efficienza maggiore in termini di consumi e prestazioni dell'hardware ternario rispetto a quello binario, ma ciò è ancora da confermare. Unica notizia certa è quella del coinvolgimento di uno dei cofondatori di IOTA in una start-up di nome "Jinn Labs", la quale sta attualmente lavorando ad un coprocessore su base ternaria pensato per essere inserito nei dispositivi IOT e per scaricare il lavoro di PoW su di esso.

L'unità più piccola che contiene le informazioni nell'aritmetica ternaria prende il nome di *trit*, l'analogo del bit, e può assumere un valore compreso tra -1 e 1. Il *tryte*, analogo del byte, invece è composto da tre trits ed è rappresentabile con le lettere maiuscole dell'alfabeto latino [A-Z] con l'aggiunta del numero 9, per un totale di 3^3 caratteri.

Una stringa di caratteri molto usata in IOTA è il *seed* [4], una parola di 81 trytes che funziona come *master private key*. Il seed è il componente principale per la gestione del proprio portafoglio IOTA, da esso vengono generate tutte le chiavi private per inviare transazioni e tutte le chiavi pubbliche per riceverle. In IOTA la chiave pubblica è essa stessa l'indirizzo di una transazione.

Il possesso di un seed determina anche la proprietà del *wallet* ad esso associato, dando il diritto a chi lo controlla di spendere e ricevere criptovaluta da qualsiasi indirizzo generato da esso. Il seed deve essere mantenuto al sicuro, poiché la sua perdita comporterà la conseguente inaccessibilità ai propri token.

2.2 Caratteristiche Principali

In questa sezione verranno presentate le caratteristiche distintive dell'architettura IOTA.

2.2.1 Decentralizzazione Reale

Nelle classiche blockchain derivate da quella di Bitcoin solo il miner ha il compito di creare il blocco da inserire nel registro, questo vuol dire che è l'unico che validerà le transazioni. Negli ultimi anni si è assistito alla nascita di enormi agglomerati di miner (chiamati *Mining Pools*), con l'intenzione di unire le forze pe risolvere la PoW più velocemente degli altri e ottenere così il compenso. Ne è risultato che l'effettivo potere sulla blockchain si spostasse nelle mani di questi colossi, vanificando in parte l'intenzione originale di avere un sistema completamente decentralizzato e democratico.

IOTA, per risolvere questo problema, introduce una modifica sostanziale nell'architettura generale, eliminando il concetto di miner e di blocco. Si noti che in questo sistema la singola transazione viene inserita nel grafo rappresentato dal Tangle, è quindi necessario eseguire dei calcoli per ogni singolo invio. Il compito di validazione è quindi nelle mani di chiunque voglia eseguire transazione: ad ogni utente viene richiesto, al momento dell'invio, di validare altre due transazioni ed eseguire un PoW (più leggera di quella di Bitcoin).

In definitiva si può affermare che non vi è distinzione tra chi emette una transazione e chi la valida, come invece avviene nelle blockchain tradizionali.

2.2.2 Scalabilità

Nella blockchain di Bitcoin (ma anche nelle sue derivate) è previsto il bilanciamento della difficoltà della PoW in modo tale da limitare la creazione di un blocco in un tempo medio di dieci minuti. Questo da origine ai problemi di scalabilità di cui si è discusso nel capitolo precedente.

In IOTA l'assenza di blocchi implica che il tempo di attesa per la validazione delle transazioni sia molto basso. Essendo il consenso parallelizzato, e non concorrente come quello della blockchain, si ha la capacità di scalare infinitamente con il crescere del numero dei nodi.

Il TPS in IOTA cresce proporzionalmente con la capacità della rete. Più nodi ne fanno parte e più velocemente viene messo in atto il meccanismo di consenso e la conferma delle transazioni.

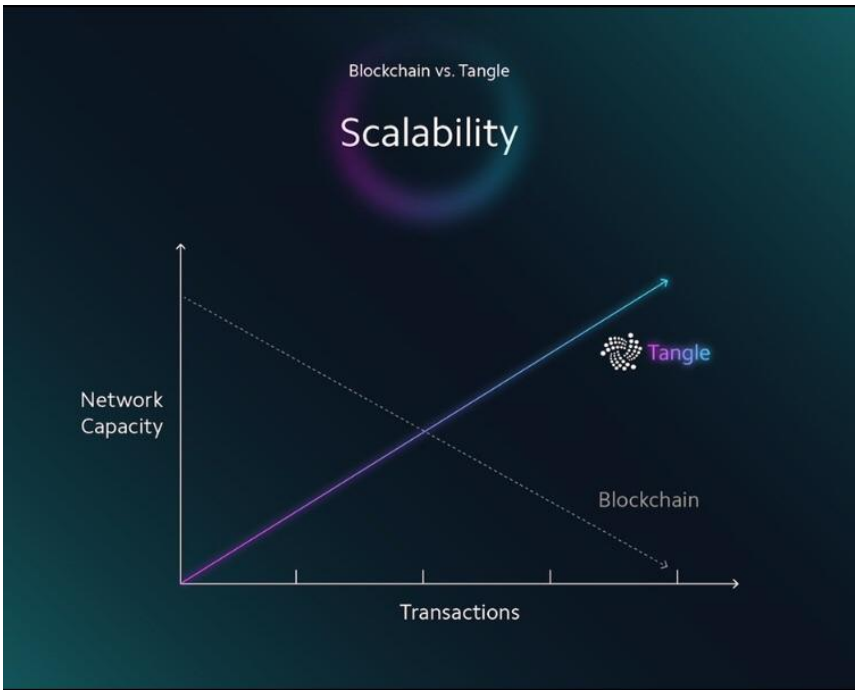

Figura 2.2: Scalabilità di IOTA contro Bitcoin

Questo è in perfetta linea con il funzionamento del mondo IOT, dove i sensori hanno la necessità di inviare dati con alta frequenza.

2.2.3 Zero Fee

Nella architettura di IOTA non è prevista la figura del miner. Coloro che emettono una transazione devono eseguire anche una PoW. Questa operazione ha la funzione di prevenire il cosiddetto *Sybil Attack* [10].

La difficoltà della PoW è variabile in base alla volontà dell'emettitore della transazione. Una Proof-of-Work più forte garantirà un minor tempo di approvazione, poiché darà un maggior peso in termini di sicurezza alla transazioni emessa.

Questa operazione a carico dell'emettitore sostituisce il meccanismo di fee utilizzato nelle blockchain, non avendo più miner da ricompensare per inserire una data transazione in un blocco, si ha come unico incentivo quello di "offrire" la propria potenza di calcolo per rendere il sistema sicuro.

IOTA si propone come prima DLT che non prevede alcun tipo di fee nel protocollo di scambio di token. Questo è un'importantissima caratteristica abilitante per sistemi che necessitano micropagamenti.

2.2.4 Quantum-Immunity

In IOTA si ha una Proof-of-Work molto meno costosa di quella di Bitcoin, il numero di nonce da provare per trovare un hash valido è sensibilmente inferiore. L'efficienza che si guadagna utilizzando un computer quantistico non impatta in modo pesante sul sistema.

Inoltre IOTA utilizza per la firma l'algoritmo *Winternitz* [5], il quale è resistente ai computer quantistici. Come controindicazione ha però il fatto di rivelare una porzione di chiave significativa ogniqualvolta si esegue l'operazione di firma. Per questo in IOTA è necessario utilizzare una chiave privata diversa ogni volta che si emette una nuova transazione. In termini pratici significa che un indirizzo da cui si è effettuata una transazione in uscita non può essere riutilizzato.

2.3 Il Tangle

Il *Tangle* è uno degli elementi fondamentali nell'architettura IOTA. Presenta una struttura sensibilmente differente dalla classica Blockchain poiché, come già accennato precedentemente, è un DAG, ovvero un Grafo Diretto Aciclico e non una catena lineare di blocchi. Tecnicamente non sarebbe possibile riferirsi al Tangle usando la parola blockchain, poiché "is a blockchain without blocks and without chain", come è stata definita sul web. La sua rappresentazione grafica si può osservare nella figura 2.3.

In questa struttura vengono inserite le transazione singolarmente, non inglobate all'interno di un macro-contenitore. Una nuova transazione prenderà il nome di *tip*. L'origine di questa forma è dovuto all'algoritmo di consenso adottato dal sistema e del quale si parlerà nel seguito del documento. Sostanzialmente ogni transazione che viene inserita nel Tangle si collegherà a due transazione precedenti, scelte in modo pseudo-casuale tra

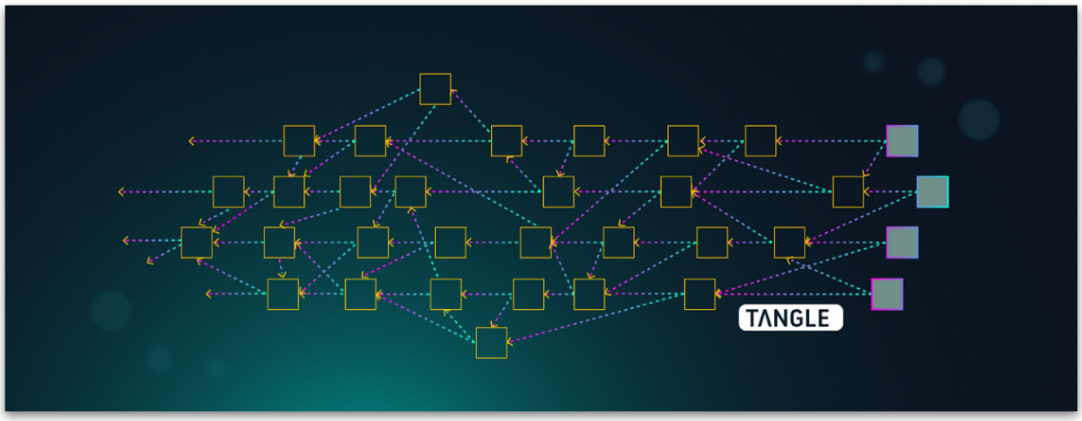

Figura 2.3: Rappresentazione grafica del Tangle

quelle non ancora referenziate (*tips*). Come si può notare nella figura 2.3 le transazioni colorate in grigio vengono inserite nella parte destra dell'immagine e si collegano ad altre due precedenti "confermandole".

2.4 I Nodi

I nodi IOTA sono la base su cui si fonda l'architettura del sistema. Si tratta di programmi software installati sulle macchine che partecipano alla rete, la comunicazione tra di essi, secondo un protocollo Peer-to-Peer (P2P) dà origine alle connessioni logiche che formano la rete IOTA.
Non tutti i nodi sono uguali, alcuni hanno funzionalità limitate in modo da permettere a macchine non troppo prestazionali di avere comunque la possibilità di utilizzare la rete. Come detto in precedenza IOTA è pensato soprattutto per il mondo IOT, perciò è facile immaginarsi piccoli dispositivi dalle limitate capacità computazionali che comunicano con degli hub o direttamente con la rete internet. Questi dovranno incapsulare solo alcuni strati del protocollo, delegando parte del lavoro ad altri nodi con piene funzionalità e dalla potenza di calcolo più elevata.
I due tipi di nodo che attualmente si trovano nell'architettura IOTA prendono il nome di *Full Node* e di *Light Node*.

2.4.1 Full Node

Come indicato dal nome il "Full Node" è il più completo e offre una serie di funzionalità per l'interazione con il Tangle e con il resto della rete IOTA. Per essere incluso all'interno di quest'ultima, il Full Node necessita uno o più collegamenti con altri nodi IOTA che prenderanno il nome di *neighbors* (vicini) una volta stabilita la connessione. Le macchine con a bordo un Full Node avranno l'onere di mantenere al proprio interno tutto il database

e di mantenerlo aggiornato rimanendo in costante comunicazione con i propri vicini.
Il modulo software che implementa le funzionalità di un Full Node prende il nome di *IOTA Reference Implementation* (IRI) ed espone una serie di API per la gestione dei vicini e delle transazioni sul Tangle.

IOTA Reference Implementation L'IRI è un processo software Java che espone un determinato numero di API JSON-REST HTTP. Queste permettono di eseguire operazioni inerenti alla gestione dei vicini e a tutti i tipi di interazioni possibili con il Tangle. La lista dei comandi messi a disposizione dell'IRI è riportata di seguito.

getNodeInfo Restituisce le informazioni relative al nodo tra cui: versione del software, occupazione della memoria relativa al processo IRI, stato di aggiornamento del nodo.

getNeighbors Restituisce un la lista dei vicini a cui il nodo è collegato.

addNeighbors Permette di aggiungere un nodo alla lista dei vicini.

removeNeighbors Permette di rimuovere un nodo dalla lista dei vicini.

getTips Restituisce la lista di tutte le tips.

findTransactions Restituisce la lista delle transazioni che corrispondono a indirizzi, tag o bundle hash passati come input.

getTrytes Restituisce i dati relativi ad una data transazione convertiti in trytes.

getInclusionStates Restituisce lo stato di validazione di una data transazione.

getBalances Restituisce il bilancio confermato relativo ad un dato indirizzo.

getTransactionsToApprove Restituisce i riferimenti delle due tips selezionate per l'algoritmo di consenso.

attachToTangle Permette di attaccare una transazione nel Tangle eseguendo la Poof-of-Work.

interruptAttachingToTangle Permette di interrompere l'operazione di attachToTangle.

broadcastTransactions Permette di notificare i vicini riguardo ad una nuova transazione.

storeTransactions Permette di salvare nel proprio Tangle locale una data transazione.

wereAddressesSpentFrom Permette di verificare se un indirizzo sia o meno stato utilizzato per inviare transazioni.

Queste funzioni rappresentano i blocchi fondamentali su cui si basa tutto il sistema di interazione con la rete e con il Tangle. Tutti i metodi offerti dalle librerie IOTA fanno da wrapper a queste funzioni base.

2.4.2 Light Node

Questo tipo di nodo è una versione "leggera" di quella precedente, non ha la necessità di mantenere una copia locale del Tangle e fondamentalmente neanche quella di avere un pacchetto software installato. Si intenda un Light Node come qualsiasi dispositivo che abbia bisogno di fare riferimento ad un Full Node per avere accesso alle funzionalità di IOTA.

Tendenzialmente si opta per questo tipo di soluzione quando si ha a che fare con macchine con scarse capacita di memoria e di calcolo.

Le librerie ufficiali offrono la possibilità di scrivere il proprio software volto ad operare come Light Node, con la gestione dei propri indirizzi e con la possibilità di operare con le transazioni che siano esse da inviare o da consultare nel Tangle, il tutto appoggiandosi ad un Full Node di riferimento.

L'applicazione base che rappresenta questo tipo di nodo è il *Wallet*: un software che permette di gestire gli indirizzi associati al proprio seed, di inviare e ricevere transazioni e di tenere sotto controllo il proprio bilancio.

Wallet Con la parola *wallet* (portafoglio), si intende uno strumento che permette ad un utente di possedere e tenere al sicuro una certa quantità di valuta, in questo caso criptovaluta IOTA. Tecnicamente per possederne uno è sufficiente avere il controllo di un *seed*, dal quale sarà possibile generare gli indirizzi con cui ricevere o inviare transazioni.

La parola wallet da anche il nome al software che permette la gestione di questa componente finanziaria, ovvero un programma che permetta di tenere sotto controllo le proprie chiavi o indirizzi, inviare o ricevere transazioni e visualizzare il proprio bilancio.

Il *IOTA Wallet* ufficiale è reperibile dalla repository ufficiali Github e, in quanto Light Node, va impostato un Full Node di riferimento al quale chiedere tutte le informazioni.

2.5 Transazioni

La transazione [1] è la struttura dati base di una qualsiasi DLT, essa descrive il trasferimento di token da una indirizzo ad un altro. Le transazioni vengono registrate nel database della DLT e sono visibili in chiaro all'interno del registro.

Ma come è fatta una transazione? Nelle tabelle 2.2 e 2.1 sono descritti i campi di una transazione.

Nella tabella 2.1 si trovano elencati i campi che compongono il cosiddetto *bundle essence*. Il nome è dovuto al fatto che sono i campi che vengono utilizzati per il calcolo del *bundle hash*, che verrà descritto in seguito.

Come si può notare dalla tabella 2.2 una transazione contiene un solo campo *address* (indirizzo), perciò è possibile indicare solo da chi vengono oppure a chi vanno i token. Per questo motivo in IOTA si ha il concetto di *bundle* [2], un contenitore di più transazioni appartenenti alla stessa operazione.

Per comprendere meglio come vengono create transazioni e bundle verrà descritto il processo che ha luogo ogniqualvolta si vogliano trasferire dei token da un indirizzo ad un altro.

Tabella 2.1: Anatomia di una transazione: bundle essence

Campo	Descrizione	Trytes
***address**	In caso di output, sarà l'indirizzo del ricevente. In caso di input, sarà l'indirizzo usato per mandare token.	27
***currentIndex**	Indice della transazione nel bundle	9
***lastIndex**	Indice dell'ultima transazione nel bundle	9
***obsoleteTag**	Tag definito dall'utente (deprecato)	27
***timestamp**	Timestamp arbitrario	int
***value**	Valore della transazione	9

Tabella 2.2: Anatomia di una Transazione

Campo	Descrizione	Trytes
attachmentTimestamp	Timestamp dopo la terminazione della PoW	9
attachmentTimestampLowerBound	Lower Bound	9
attachmentTimestampUpperBound	Upper Bound	9
branchTransaction	Hash della prima transazione approvata	81
bundle	Hash usato per raggruppare più transazioni in un bundle.	81
hash	Hash della transazione	81
nonce	Stringa generata durante la risoluzione della PoW	27
signatureMessageFragment	Nel caso di input speso contiene la firma con chiave privata, altrimenti può essere usato per inserire un messaggio.	2187
tag	Tag definito dall'utente	27
trunkTransaction	Hash della seconda transazione approvata	81

2.5.1 Generazione degli Indirizzi

Per prima cosa è necessario conoscere come viene generato un indirizzo, ovvero l'informazione che permette di spostare i token da un wallet ad un altro.

Gli indirizzi vengono generati da una chiave privata creata a partire dal seed e dall' *index*. Il *security level* in IOTA determina la difficoltà utilizzata nella generazione di chiavi e indirizzi e può assumere un valore compreso tra uno e tre. In base ad esso cambierà il numero di cicli di hash e la lunghezza delle chiavi, risultando in un processo più o meno lungo. Partendo dallo stesso seed è possibile avere index uguali in processi di generazione con livelli di sicurezza diversi, poiché il risultato sarà diverso. Si può dire che il seed controlla un "account" per ogni livello di sicurezza. Gli indirizzi generati con diversi livelli di sicurezza avranno comunque la stessa lunghezza pari a 27 trytes anche se le chiavi private avranno un lunghezza pari a $2187 \times sec$ trytes, con *sec* rappresentante il livello di sicurezza.

Il livello di sicurezza più basso è pensato per essere utilizzato da macchine con scarsa capacità di calcolo e wallet contenenti un numero di token di poco valore. Il livello medio è pensato per un utilizzo standard, ovvero per sfruttare il wallet allo stesso modo di un tradizionale portafoglio fisico. Il livello alto è pensato per associazioni che spostano grandi cifre e che non possono permettersi di subire attacchi al proprio wallet.

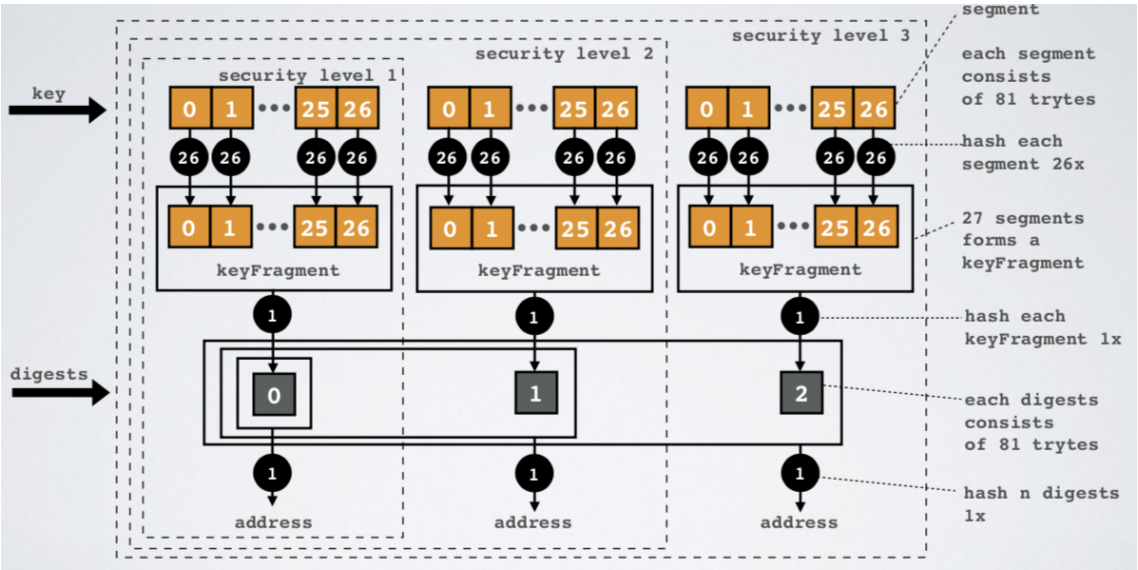

Figura 2.4: Schema di generazione di un indirizzo.

Il processo di generazione degli indirizzi parte calcolando un *subseed* sommando i valori del seed e dell'index convertiti in trits. Il subseed viene utilizzato come input della classe *Kerl* che utilizza internamente l'algoritmo di hashing *Keccak-384*. All'interno di questa classe viene generata una chiave privata secondo il livello di sicurezza desiderato. Questa chiave viene suddivisa in segmenti di 81 trytes e ad ognuno di essi viene applicata la funzione di hash per 26 volte. Da questa operazione risulterà un numero di *keyFragment* compreso tra uno e tre, in base al livello di sicurezza scelto. Ai keyFragment verrà applicato un ulteriore ciclo di hash per ottenere un *digest* per ognuno di essi. I digest verranno utilizzati insieme per calcolare l'address finale tramite un nuovo ciclo di hash.

2.5.2 Consolidamento del Bundle

Un utente che chiameremo Bob è intenzionato a spostare dei soldi dal proprio wallet verso quello di Alice. Bob possiede tre indirizzi B1, B2 e B3. I primi due contengono 100 IOTA l'uno e il terzo è vuoto. Alice possiede solamente un indirizzo A1 con 10 IOTA associati.

Bob vuole trasferire 150 IOTA sull'indirizzo A1 di Alice e questo è ciò che avviene:

1. **Creazione del Bundle**

   Per eseguire questo trasferimento è necessaria la creazione di un bundle di sei transazioni:

   - La prima transazione rappresenta un output e ha un valore positivo pari a 150 IOTA e descrive l'accredito di 150 IOTA all'indirizzo A1 di Alice
   - La seconda transazione rappresenta un input e ha un valore negativo pari a -100 IOTA e descrive la rimozione di 100 IOTA dall'indirizzo B1 di Bob
   - La terza transazione ha un valore pari a 0 IOTA e serve a contenere metà della firma della prima transazione di input
   - La quarta transazione rappresenta un input e ha un valore negativo pari a -100 IOTA e descrive la rimozione di 100 IOTA dall'indirizzo B2 di Bob
   - La quinta transazione ha un valore pari a 0 IOTA e serve a contenere metà della firma della seconda transazione di input
   - La sesta transazione rappresenta un output e ha un valore pari a 50 IOTA e descrive l'accredito di 150 IOTA all'indirizzo B3 di Bob

2. **Finalizzazione del Bundle**

   Il *bundle hash* viene calcolato usando l'algoritmo "Kerl", prendendo in considerazione soli i campi essenziali (bundle essence) delle transazioni al suo interno.

   Ogni campo di ogni transazione viene convertito nel valore corrispondente in trits per poi eseguirne la somma e generare quello che chiameremo *bundleEssence*, di dimensione totale pari a 486 trits. Tutti i bundleEssence generati dalle transazioni che fanno parte del bundle vengono assorbiti uno alla volta da una funzione spugna (come mostrato nella figura 2.5), per poi generare un hash di dimensione pari a 243 trits.

   Questo hash è il *bundle hash* finale e viene inserito nell'apposito campo bundle presente in ogni transazione e sarà uguale per tutte le transazioni appartenenti allo stesso bundle.

3. **Aggiunta delle Firme**

   Per ogni transazione di input viene generato il *signatureMessageFragment*, a partire dalla chiave privata associata a quell'input e dal *normalized bundle hash*.

   Quest'ultimo parametro si ottiene ricalcolando il bundle hash in modo tale che al suo interno non compaia nessun tryte di valore "M".

   L'operazione di normalizzazione è un processo di ridistribuzione del peso.

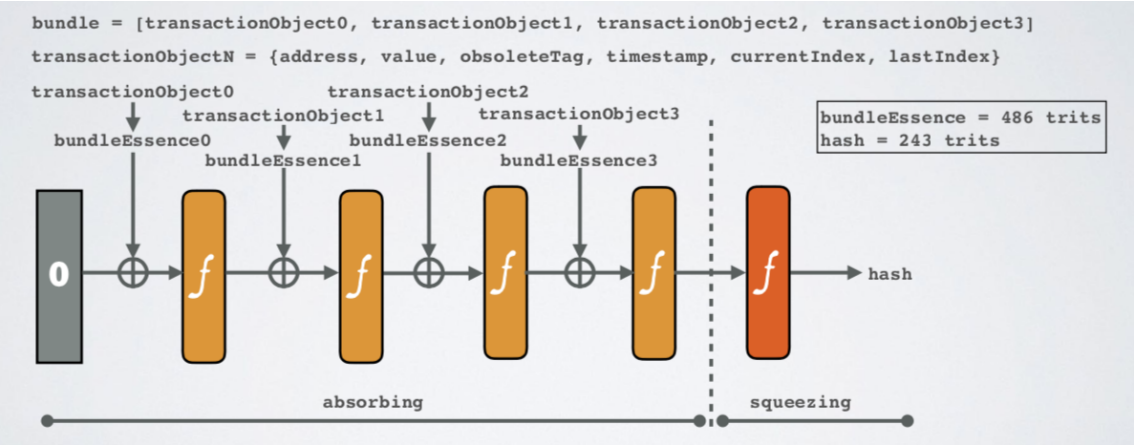

Figura 2.5: Processo di generazione del bundle hash. I campi essenziali di una transazione vengono convertiti in trits e sommati originando i bundleEssence che vengono poi usati per il calcolo del bundle hash.

Il bundle hash viene diviso in tre parti composte da 27 trytes ciascuna. Per ogni parte si converte ogni tryte nel suo valore decimale e ne viene calcolata la somma. Se questa somma è maggiore o uguale a zero allora significa che quella parte ha più "peso" a destra, perciò si parte dal primo tryte, si decrementa di uno e si ricalcola la somma. Se il valore della somma non risulta pari o inferiore a zero si ripete l'operazione di decremento. Se il tryte modificato raggiunge il valore -13 allora si passerà al tryte successivo se sarà necessario decrementare ulteriormente. Una volta che la somma ha raggiunto un valore inferiore a zero il processo si ferma e la parte modificata del bundle hash in questione viene utilizzata per la creazione del bundle hash normalizzato.

Se la somma è maggiore di zero allora la parte di bundle hash ha più "peso" a sinistra. Si ripeteranno quindi le operazioni precedenti con la sola differenza data dall'operazione di incremento anziché quella di decremento e il passaggio al tryte successivo ogniqualvolta venga raggiunto il valore 13. Una volta che la somma ha raggiunto un valore pari o maggiore di zero il processo si ferma e la parte di bundle hash viene consolidata.

La normalizzazione è necessaria poiché nello schema del calcolo della firma il valore "M" rivela una parte della chiave privata, come verrà messo in evidenza in seguito.

La chiave privata viene utilizzata per firmare il bundle. Essa viene suddivisa in 27 segmenti di 81 trytes ed ad ognuno vengono applicati K cicli di hash generando così il *signatureFragment*. Il valore K viene calcolato a partire dal bundle hash normalizzato secondo la formula $K = 13 - dec$, dove dec rappresenta il valore decimale di un tryte dell'hash.

La figura 2.7 rappresenta lo schema secondo il quale viene generato il signatureFragment e successivamente verificato sfruttando i valori K ricavati dal normalized

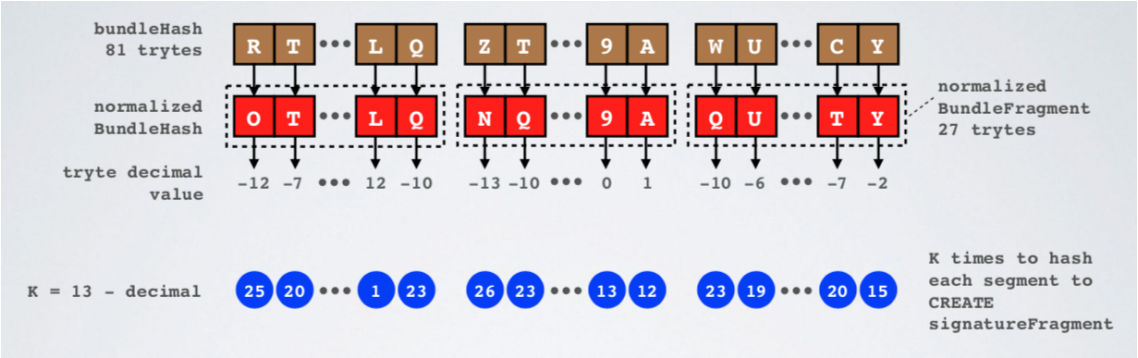

Figura 2.6: Calcolo del valore K

bundle hash. Si noti come un tryte di valore "M" all'interno del bundle hash rivelerebbe una parte di chiave poiché il valore K risulterebbe pari a zero e al segmento corrispondente non verrebbe applicata una funzione di hash. Questo risulterebbe in una parte di chiave in chiaro all'interno del campo signatureMessageFragment. Per questo motivo viene calcolato il bundle hash normalizzato.

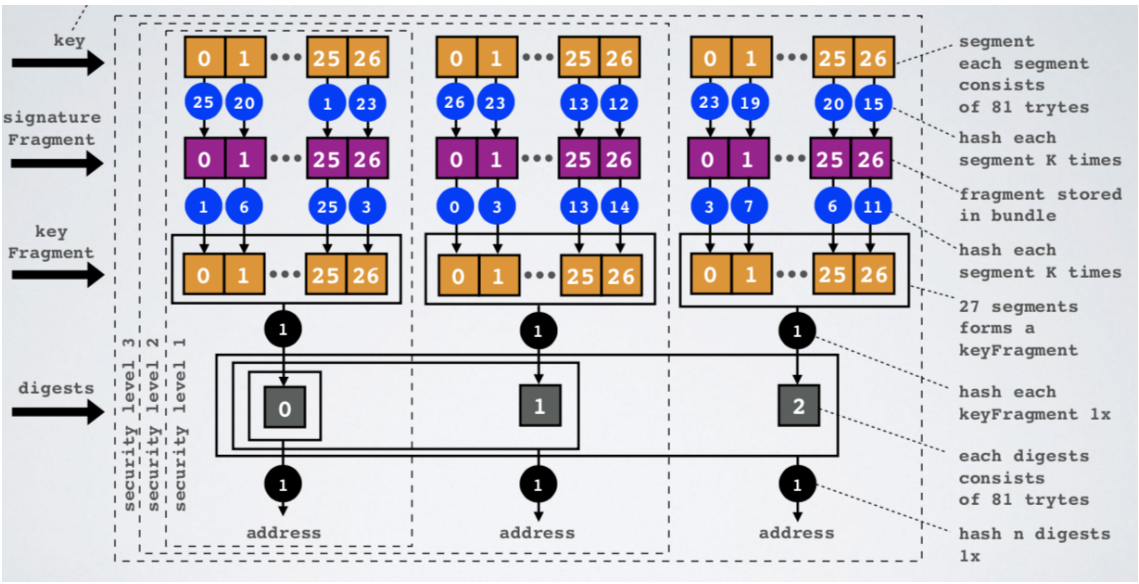

Figura 2.7: Schema di generazione del signatureFragment e della su verifica

Il signatureFragment calcolato viene inserito nei 2187 trytes del campo signatureMessageFragment delle transazioni di output del bundle.

Durante il processo di verifica viene calcolato il valore K secondo la formula $K = 13 + dec$ e applicata la stessa operazione precedente sui 27 segmenti del signatureFragment estratto dalle transazioni del bundle. In questo modo si ottiene il *keyFragment* utilizzato per generare l'address da confrontare con quello presente nella transazione. Se i due indirizzi corrispondono allora la firma verificata.

4. **Selezione delle Tip**

   Per connettere una transazione al Tangle è necessario selezionare due tip a cui fare riferimento. Questa scelta viene normalmente effettuata da un Full Node in seguito ad una richiesta, questo poiché è l'unico nodo con una visione sul Tangle. L'algoritmo di selezione è conosciuto come MCMC (*Markov Chain Monte Carlo*).

   La selezione delle tip passa attraverso un processo di validazione, in cui esse vengono sottoposte ad un controllo secondo una certa serie di criteri. Le tip invalide verranno scartate e mai referenziate. Ci si aspetta che tutti i nodi benevoli agiscano nello stesso modo, non confermando transazioni che non soddisfano i criteri di validità.

   Nonostante un bundle contenga molte transazioni, esso referenzia solamente due transazioni in totale e non $2N$ (con N pari al numero delle transazioni) come ci si aspetterebbe. Questo poiché le transazioni all'interno di un bundle sono referenziate tra di loro secondo lo schema mostrato nella figura 2.8. I puntatori alle due tip selezionate in questo processo vengono salvati nei campi della transazione *trunkTransaction* e *branchTransaction*, indicati nella tabella 2.2.

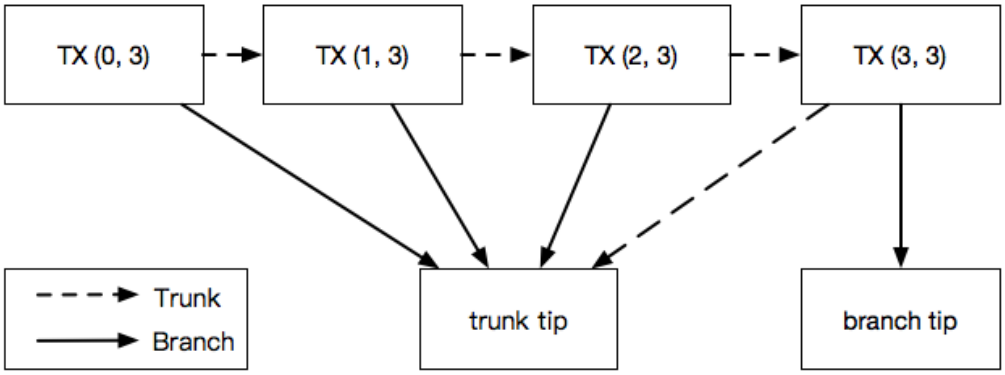

Figura 2.8: Tutte le TX appartengono allo stesso bundle. Viene mostrato come solo due tip esterne vengano referenziate

   Nella figura 2.8 si nota anche come la trunkTransaction venga referenziata da (N-1) transazioni, mentre la branchTransaction solamente dall'ultima transazione del bundle.

5. **Proof of Work**

   Questo è l'ultimo passaggio per la corretta emissione di una transazione. La Proof of Work utilizzata in IOTA è simile al sistema di Hashcash, diversa e molto più semplice di quella usata nella Blockchain di Bitcoin. Lo scopo fondamentale di questo calcolo è quello di prevenire lo spam e il sybil attack.

La PoW viene eseguita per ogni transazione presente nel bundle. Questo viene fatto attraverso una funzione chiamata *Curl* usata in una delle sue due possibili implementazioni: *CCurl* e *Webgl 2 Curl*. La prima basata su calcolo tramite CPU, l'altra su calcolo tramite GPU.

La funzione di PoW prende in input il *transactionObjectTrytes* (l'oggetto transazione codificato in trytes) e il *Minimum Weight Magnitude* o MWM (la difficoltà della PoW), e restituisce un *nonce*. Quest'ultimo, unito al transactionObjectTrytes, viene convertito in trits e gli viene applicata la funzione Curl per la validazione. Se il numero di zeri finali è uguale o superiore al valore MWM, allora il nonce è valido.

6. **Broadcast del Bundle**

   Una volta creato un bundle viene inviato ad un Full Node il quale avrà il compito di inoltrarlo ai suoi vicini, che a loro volta faranno lo stesso.

   Ogni nodo che vedrà le nuove transazioni potrà decidere se inoltrarle a sua volta o se scartarle. É possibile che una transazione venga scartata per una PoW insufficiente.

   I criteri usati per decidere se inoltrare o meno una transazione sono diversi da quelli usati durante l'emissione di una nuova transazione. Un nodo può inoltrare transazioni invalide ma queste non verranno mai referenziate e rimarranno *unconfirmed* per sempre. Questo perché l'algoritmo di validazione, usato durante la selezione delle tip, non confermerà transazioni invalide (a meno che non si tratti di un nodo alterato per poter eseguire azioni malevole).

2.6 Consenso

Con il termine "consenso" si intende il processo che porta tutti i nodi a concordare sullo stesso stato della DLT.

In IOTA l'intera rete di full node è responsabile del consenso generale, al contrario di quanto avviene in Bitcoin, dove solo i miner hanno il diritto di decidere.

Il consenso in IOTA è intrinseco nel processo di emissione delle transazioni ed una delle caratteristiche principali di questa DLT.

Si dice che una transazione ne *approva* un'altra referenziandola direttamente o indirettamente. Questo significa che si considera come valida una transazione, ovvero conforme ai termini dettati dal protocollo. L'approvazione diretta si ha quando una transazione ne referenzia altre due tramite i suoi campi trunkTransaction e branchTransaction, è quindi direttamente collegata. L'approvazione indiretta si ha invece quando esiste un cammino tra la nuova transazione e una qualsiasi altra nel Tangle. Come già detto la *tip* è una transazione non ancora approvata da nessun altra. Una transazione si dice confermata quando viene validata direttamente o indirettamente da tutte le tip. Il parametro *weight* misura il peso della transazione ed è proporzionale alla quantità di Proof-of-Work eseguita durante l'operazione di attach (attualmente questo parametro viene impostato a 1 per tutte le transazioni, ma è in programma lo sviluppo dell'algoritmo basato sulla difficoltà della PoW). Il *cumulative weight* è definito come il peso di una transazione più la somma di tutti i pesi di tutte quelle che la approvano direttamente o indirettamente.

Algoritmo MCMC Il già citato algoritmo *Markov Chain Monte Carlo* è usato per la selezione delle due tip da referenziare. Esso gioca un ruolo fondamentale nella sicurezza del sistema, poiché la protezione dagli attacchi di double spending deriva da una giusta scelta delle tip. L'idea è quella di mettere alcuni *random walkers* (camminatori casuali) nella zona del Tangle che presenza transazioni ormai confermate, per poi lasciarli muovere verso le tip in modo pseudo-casuale. Le tip così raggiunte sono quelle che vengono prese per essere approvate. Durante il cammino la scelta della prossima transazione su cui muoversi dipende dal cumulative weight di ognuna di esse. Partendo da un punto qualsiasi, il set delle prossime transazioni è definito da tutte le transazioni che referenziano direttamente quella corrente. Si supponga che il walker si trovi sulla transazione x. Il set delle prossime viene indicato con z. Una transazione y del set z ha una probabilità P_{xy} di essere selezionata per proseguire il cammino data da

$$P_{xy} = \exp\big(-\alpha(\mathcal{H}_x - \mathcal{H}_y)\big)\Big(\sum_{z:z\rightsquigarrow x} \exp\big(-\alpha(\mathcal{H}_x - \mathcal{H}_z)\big)\Big)^{-1} \tag{2.1}$$

L'equazione indica che il walker tende a scegliere una transazione con il gradiente minimo di cumulative weight tra quella corrente e la prossima transazione potenziale.

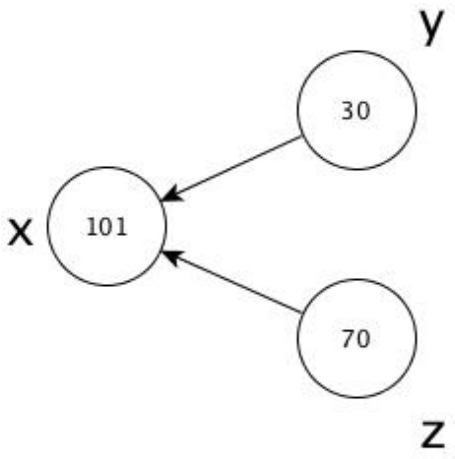

Figura 2.9: Rappresentazione di tre transazioni con il loro cumulative weight.

Nella figura 2.9 il walker si trova su x e deve scegliere la prossima transazione su cui muoversi tra z e y. Dal calcolo delle probabilità P_{xy} e P_{xz} (considerando $\alpha = 1$ per semplicità) si ottiene

$$P_{xy} = \exp\big(-(101-30)\big)\Big(\exp\big(-(101-70)\big) + \exp\big(-(101-30)\big)\Big)^{-1} =$$
$$= \exp(-71)\Big(\exp(-31) + \exp(-71)\Big)^{-1} \approx 4.25 * 10^{-18}$$

$$P_{xz} = \exp\big(-(101-70)\big)\Big(\exp\big(-(101-70)\big) + \exp\big(-(101-30)\big)\Big)^{-1} =$$
$$= \exp(-31)\Big(\exp(-31) + \exp(-71)\Big)^{-1} \approx 1$$

Come si può notare la probabilità di andare verso y è trascurabile. Questo perché z è già approvata da molte transazioni e il suo cumulative weight è alto, mentre il ramo di y ha un peso molto inferiore.

Stato del Tangle IOTA non è costruita con blocchi temporalmente allineati come la blockchain, ogni singola transazione è attaccata nel Tangle in modo autonomo e parallelo alle altre, dando origine ad un DAG.

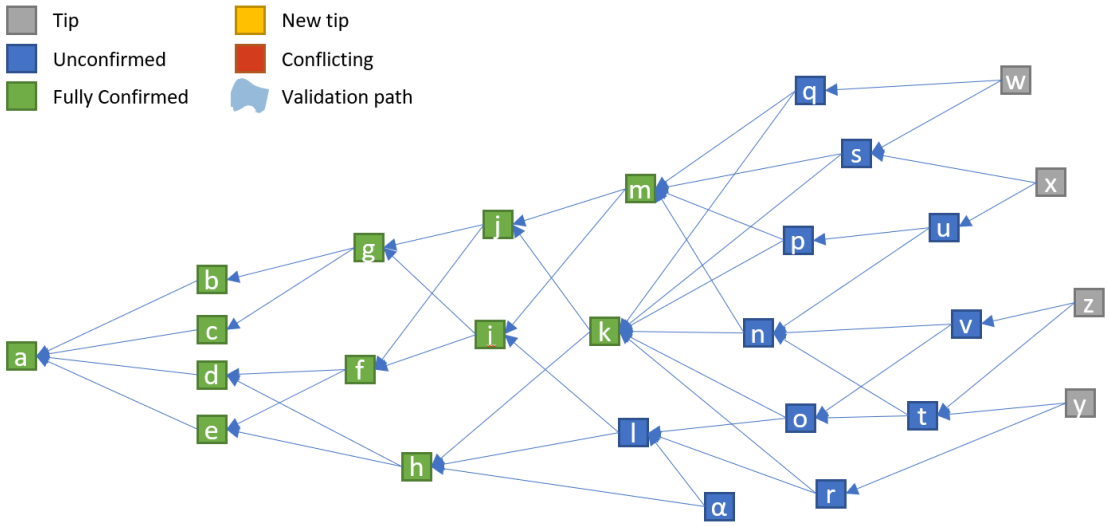

Figura 2.10: Raffigurazione del Tangle

Nella figura 2.10 le transazioni verdi sono quelle già completamente confermate dalla rete, mentre quelle blu non sono ancora del tutto confermate. Quelle grigie invece non hanno avuto ancora nessuna validazione.

Bisogna ricordare che, come nella blockchain, non c'è mai la certezza assoluta che un dato non possa essere cambiato, ma si assume come tale poiché la probabilità che esso venga modificato è molto bassa.

La transazione α in figura 2.10 è inusuale e pare non essere tollerata dalla rete. Essa referenzia due transazioni h e l ma la prima è già referenziata dalla seconda. Questo significa che al momento della selezione delle tip h non la era.

Inserimento di una Transazione Per aggiungere una nuova transazione al Tangle, un utente deve prendere due tip e validarle. Questo significa verificare la firma, la PoW ed essere sicuri che le tip non siano in conflitto con nessuna delle transazioni precedenti.

Se il processo di validazione ha successo allora le due tip vengono referenziate dalla nuova transazione e questa viene inserita nel Tangle.

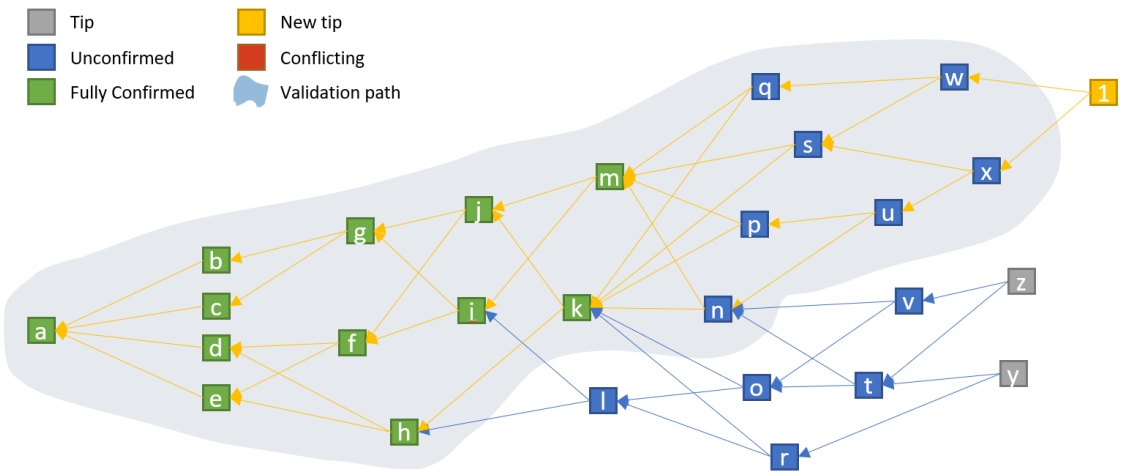

Figura 2.11: Inserimento di una nuova transazione nel Tangle

Il nodo che emette la transazione si preoccupa di validare solo il subtangle composto dalle transazioni referenziate direttamente o indirettamente dalle tip scelte. La porzione di Tangle in questione è evidenziata nella figura 2.11 tramite dall'area grigia. Al suo interno non devono esserci transazioni conflittuali. La porzione di Tangle rimanente potrebbe però contenere elementi non compatibili con quelli all'interno del subtangle precedentemente considerato. Questi conflitti verranno eventualmente rilevati in successive operazioni di inserimento, quando altri nodi selezioneranno altre tip e verificheranno un'altra porzione di Tangle.

Inserimento di una seconda Transazione Se un altro utente aggiunge un'altra transazione verificando altre due tip, si troverà a validare un subtangle diverso dal precedente ma con molti elementi in comune, sicuramente tutti quelli appartenenti alla zona verde, quella delle transazioni "confermate".

Nuovo Stato del Tangle Dopo le due operazioni di inserimento è possibile osservare come sia cambiato lo stato del Tangle. Sono presenti due nuove tip, mentre le precedenti sono entrate nella zona blu. Inoltre sovrapponendo i due subtangle considerati precedentemente si può notare come una nuova transazione (etichettata con n) sia entrata nella zona verde. Questa è stata completamente confermata poiché si è creato un cammino che da essa porta a tutte le tip del Tangle.

Per verificare la conferma di una transazione e sufficiente guardare se essa è referenziata direttamente o indirettamente da tutte le tip, senza dover eseguire nessun processo di validazione.

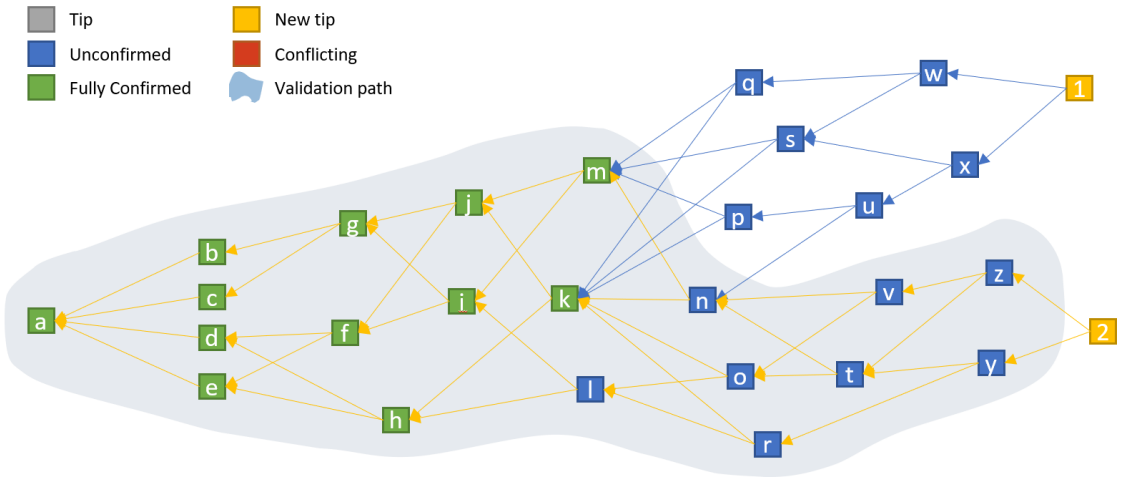

Figura 2.12: Inserimento di una seconda nuova transazione nel Tangle

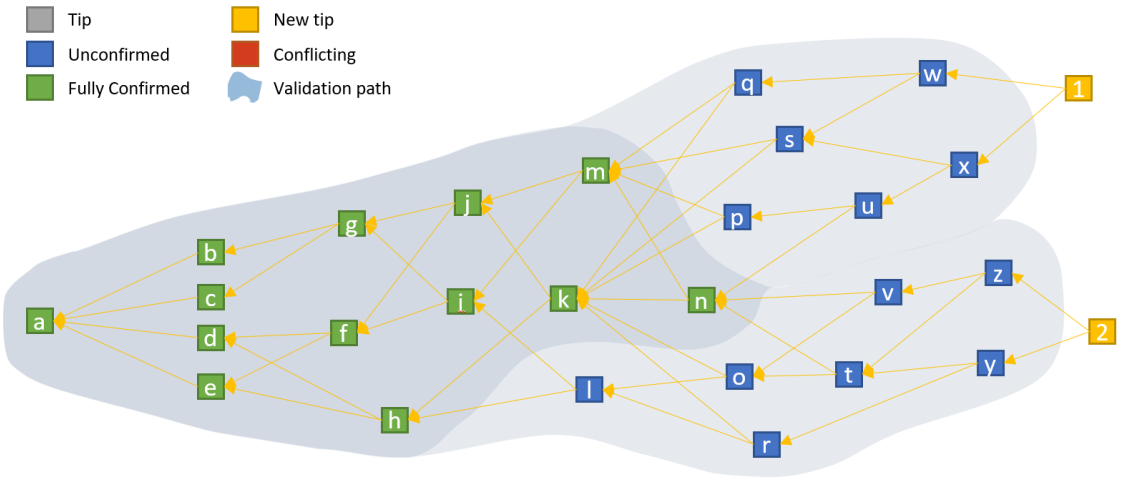

Figura 2.13: Stato del Tangle dopo l'aggiunta di due nuove transazioni.

Livello di Conferma Come precedentemente detto la certezza assoluta che una transazione non possa mai essere modificata non esiste. Il livello di conferma accettabile è stabilito da colui che deve vendere un bene e ricevere il pagamento, in base al rischio che è disposto a correre. Quando vende qualcosa di poco valore può accontentarsi di un livello di conferma relativamente basso. In caso di transazioni più ingenti sarebbe invece preferibile raggiungere una percentuale di conferma più alta.

Ritardo di Propagazione I nodi sono collegati attraverso la rete Internet, perciò è inevitabile che vi siano dei ritardi di propagazione e che non tutti vedano lo stesso Tangle nello stesso momento.

Nella figura 2.14 le transazioni q ed s non sono più tip e non sono più selezionabili

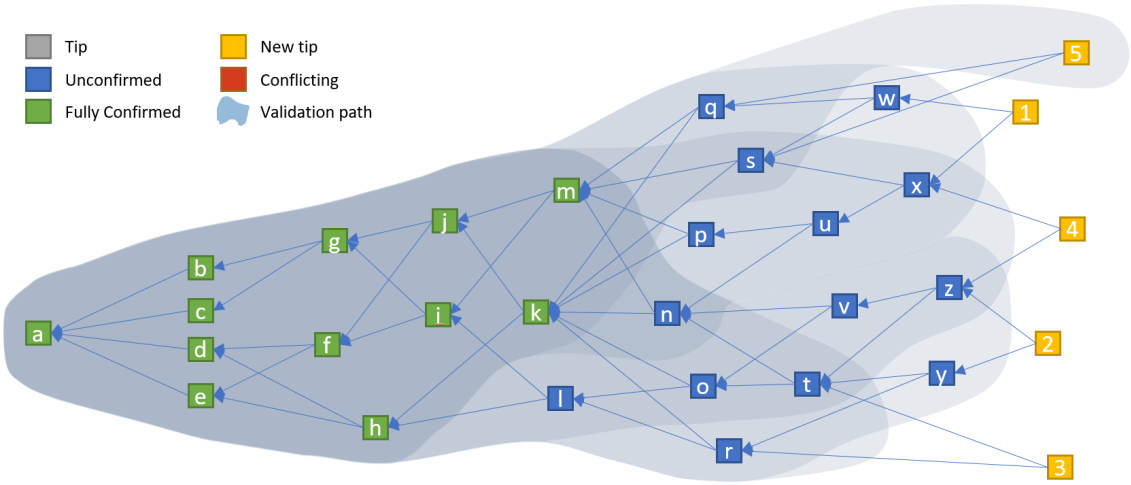

Figura 2.14: Stato del Tangle con nuove transazioni.

utilizzando l'algoritmo MCMC. É possibile però che un altro nodo non sia completamente aggiornato e che veda q ed s come tip. In questo contesto il nodo potrebbe selezionare quelle due transazioni come trunk e branch. Dopo che la transazione 5 viene inserita dal nodo non aggiornato e propagata al resto della rete, si entra in uno stato in cui n non è più nella zona verde. Il suo livello di conferma è comunque molto alto dato che è referenziato indirettamente da quattro tip su cinque. Si noti che la transazione n non torna nello stato "non confermato", cambia solamente la sua percentuale di conferma, non più pari ad un ideale 100% ma comunque molto alta. Transazioni successive possono riportare n nella zona verde referenziando la transazione 5 e, ad esempio, la 1. Nel mondo reale il 100% di conferma è difficile da raggiungere a causa del ritardo di propagazione ma anche a causa di tip che rimangono tali poiché non vengono selezionate dall'algoritmo MCMC.

Double Spending Il principale problema che tutte le criptovalute devono affrontare è legato al double spending. La natura digitale di questa valuta permette ad un utente di provare a spendere gli stessi token per acquistare due beni diversi. Le due transazioni non possono essere considerate valide entrambe, esse rappresentano un conflitto. Solo una delle due verrà considerata valida. Un venditore che consegna un prodotto ad un acquirente dopo aver ricevuto un pagamento viziato da double spending potrebbe essere incapace di incassare effettivamente i soldi, poiché la transazione potrebbe essere considerata invalida dalla rete. Per questo motivo è importante gestire il problema del double spending e dare modo all'utente di valutare una transazione come confermata o meno.

In IOTA le transazioni indicano il passaggio di token da un indirizzo ad un altro, questo significa che il double spending consiste nell'usare come input di una transazione un indirizzo con token che sono già stati spesi.

La figura 2.15 raffigura lo scenario in cui un utente aggiunge le due transazioni conflittuali w e y. Utenti successivi potrebbero includere le due transazioni in percorsi di validazioni differenti, mantenendo i due elementi conflittuali in due subtangle separati.

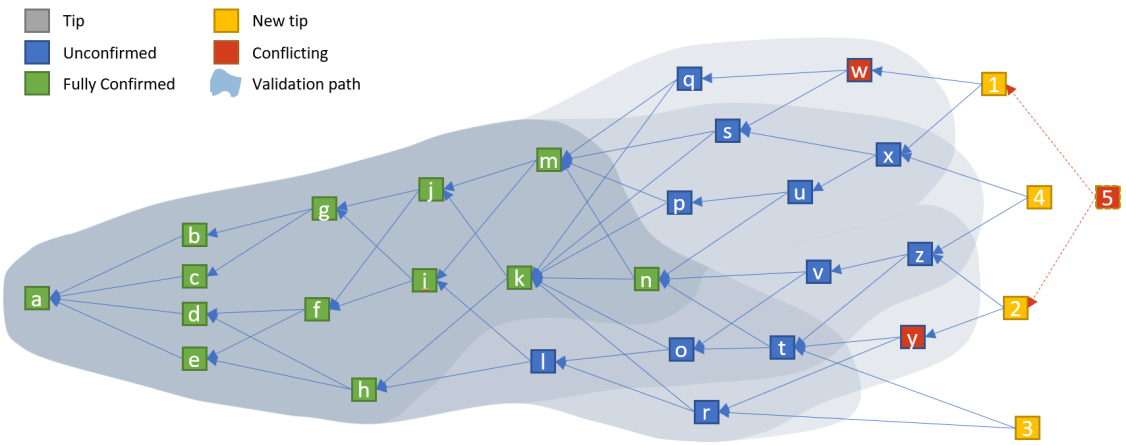

Figura 2.15: Tangle con transazioni conflittuali.

Per esempio le transazioni 1 e 2 non rilevano il conflitto durante la validazione perciò w e y ricevono una prima conferma. Prima o poi queste due transazioni entreranno in un singolo percorso di validazione e le due tip selezionate per inserire la nuova transazione verranno scartate e ne verranno scelte altre.

Con gli inserimenti di transazioni successive uno dei due subtangle contenenti w o y crescerà più dell'altro e una delle due verrà confermata mentre l'altra sarà abbandonata, con il conseguente abbandono di tutte le transazioni che la referenziano direttamente o indirettamente.

Tutte le transazioni valide abbandonate a causa di un riferimento ad un elemento conflittuale possono essere reinserite (*reattach*) nel Tangle per rientrare nel processo di conferma. In questo caso solo la PoW deve essere ricalcolata.

Risoluzione del Double Spending Nella figura 2.15 la selezione delle tip 1 e 2 per l'inserimento di 5 viene scartata a causa del conflitto e si decide di prendere le tip 1 e 4 per inserire la nuova transazione.

Un altro utente sceglie le tip 2 e 3 per la nuova transazione 7. Questo porta al delineamento di due rami separati, dei quali solo uno potrà sopravvivere. Il ramo dal quale verranno selezionate più tip crescerà più rapidamente, incrementando il cumulative weight delle transazioni al suo interno. L'altro ramo verrà abbandonato e le sue transazioni non raggiungeranno mai lo stato di conferma.

Come già detto le transazioni valide del ramo abbandonato possono essere reinserite nel Tangle per rientrare nel meccanismo di conferma. La transazione conflittuale del ramo non sarà mai considerata come valida anche se reinserita con la stessa modalità delle altre.

Tangle Offline IOTA permette agli utenti di costruire parti di Tangle mentre si è offline. Per fare ciò le transazioni devono essere create e collegate tra di loro come descritto dal protocollo.

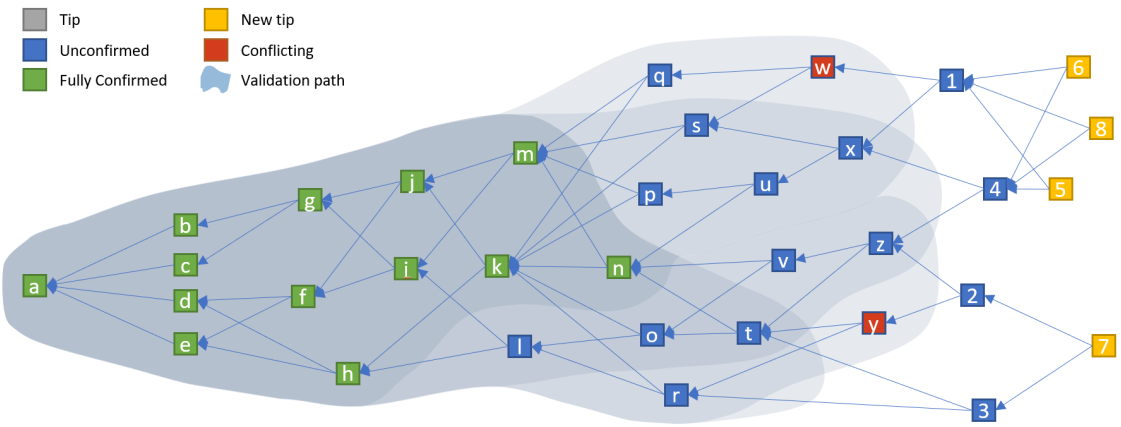

Figura 2.16: Tangle con risoluzione del double spending.

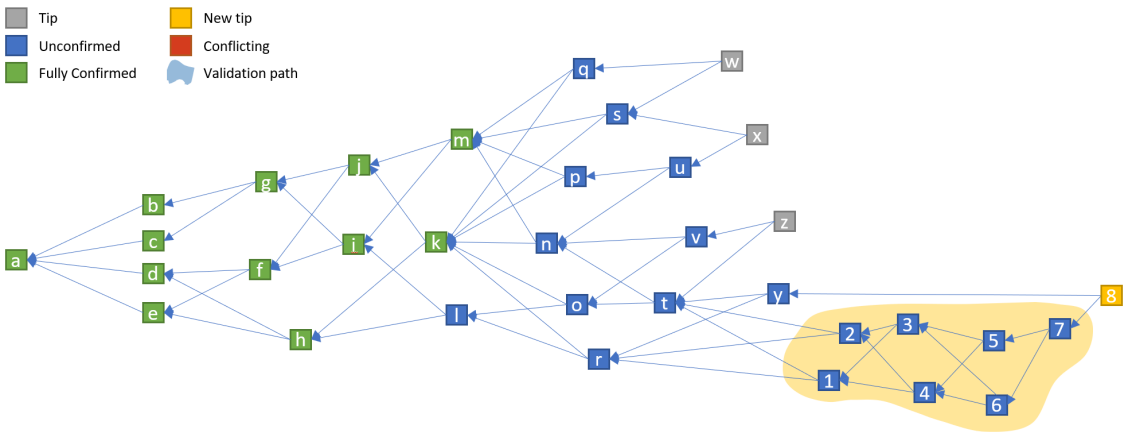

Figura 2.17: Inserimento di una parte di Tangle generata offline.

La figura 2.17 mostra come dalle le transazioni 1 e 2 inizi la costruzione di un subtangle offline. Esse vengono collegate a due delle ultime tip conosciute e successivamente vengono referenziate direttamente o indirettamente dalle nuove transazioni emesse offline.

Per consolidare il subtangle generato in questo modo è necessario inserire una nuova transazione (la 8 nella figura) che unisca la parte offline con le tip più recenti. Questa operazione non è obbligatoria ma è raccomandata per promuovere il ramo che si vuole immettere. In caso contrario l'algoritmo di selezione potrebbe non considerare le tip del nuovo ramo poiché troppo vecchie (se il subtangle è rimasto offline per troppo tempo).

Gli utenti che selezioneranno la transazione 8 come tip da validare includeranno tutte le transazioni eseguite offline nel meccanismo di conferma.

Una volta inserite nel Tangle, le transazioni offline possono entrare nello stato di conferma esattamente come le altre. Se queste transazioni generano conflitti verranno trattate secondo gli stessi criteri descritti nei paragrafi precedenti.

2.7 Sicurezza

IOTA, a differenza della Blockchain, è basata sul Tangle, ovvero un grafo diretto aciclico. Non ci sono miner, blocchi e hash rate da mantenere costante. Il consenso passa attraverso i nodi che devono approvare due transazioni per poterne inserire una.

É possibile eseguire con successo un'operazione di double spending sfruttando l'attacco del 51%?

Nella blockchain di Bitcoin l'unico requisito è quello di possedere una certa percentuale della potenza di hash per poter eseguire questo tipo di attacco. In IOTA bisogna considerare la topologia della rete dove ogni nodo vede solamente una piccola parte del Tangle attraverso i propri vicini.

Per avere successo nell'attacco sono necessari tre elementi:

1. **Percentuale della potenza di hash della rete** Una porzione sufficientemente grande della potenza di hash totale. Esattamente come in Bitcoin l'attaccante deve possedere un potenza di calcolo molto elevata.

2. **Onnipresenza** Per utilizzare le proprie risorse in modo efficiente, l'attaccante dovrebbe avere una visione estesa su tutta la rete per ottimizzare la propagazione dell'attacco avendo più punti di accesso al Tangle. In IOTA le connessioni tra full node sono private e non è possibile mappare la rete. Attualmente l'aggiunta di un vicino comporta l'inserimento manuale degli indirizzi da parte di entrambi i nodi. Questi devono quindi eseguire l'operazione consapevolmente. Non è possibile ottenere una lista di tutti i nodi disponibili.

3. **Percentuale di onnipotenza** Avere vicini appartenenti ad una grande porzione di rete. Questo è necessario a causa delle limitazioni di banda in una rete composta principalmente da dispositivi IoT. L'attaccante vorrebbe poter eseguire un attacco consistente sfruttando tutta la sua potenza di hash, ma verrebbe limitato dal numero dei vicini, riuscendo a raggiungere solo una piccola parte della rete. L'attaccante non riuscirebbe a diffondere abbastanza velocemente il suo attacco, nonostante sia in grado di generare transazioni molto velocemente.

Potenza di hash e onnipotenza sono i due elementi fondamentali per portare a termine un attacco. Avere risorse infinite ma scarso controllo sulla rete risulterà in un attacco fallito. Allo stesso modo avere un grandissimo numero di collegamenti senza una potenza adeguata porterà allo stesso risultato. Se una certa percentuale di nodi viene attaccata questi, prima o poi, si riallineeranno agli altri nodi.

Nonostante una Proof-of-Work più semplice, la sicurezza in IOTA non viene compromessa grazie alla topologia della rete e all'architettura del sistema stesso.

2.8 Coordinator

Nella fase iniziale di IOTA, in cui ci si trova ora, la rete è ancora composta da un numero relativamente basso di nodi. Per questo motivo potrebbe essere più facile attaccare

insediando un certo numero di nodi malevoli nel sistema atti a generare un gran numero transazioni maligne. L'algoritmo MCMC seleziona le tip in modo casuale e potrebbe esserci la possibilità che proprio le transazioni di questi nodi vengano approvate.

Per proteggere la rete dall'attacco del 51% precedentemente descritto è stato implementato un meccanismo temporaneo di sicurezza chiamato *Coordinator*. Si tratta di un nodo speciale controllato dalla IOTA Foundation e viene usato per confermare direttamente o indirettamente le transazioni del Tangle.

Nonostante questa unica entità posta a "vegliare" sul Tangle la rete mantiene comunque la sua decentralizzazione, poiché ogni nodo della rete controlla che il Coordinator non infranga le regole di consenso del protocollo IOTA.

Il Coordinator emette transazioni chiamate *milestone*, esse sono numerate secondo un numero progressivo e presentano un valore nullo. Ogni milestone è composta da un bundle di due transazioni contenenti la firma. Le milestone hanno come trunk la milestone precedente e come branch una normale transazione del Tangle. Le transazione verificate direttamente o indirettamente da una milestone sono verificate, inoltre queste transazioni speciali non sono falsificabili poiché firmate direttamente dal Coordinator.

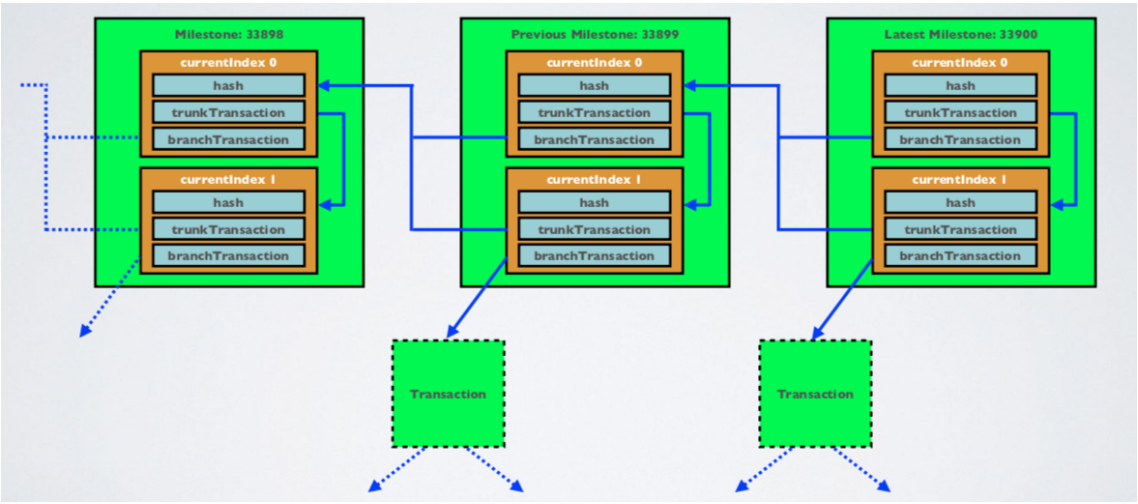

Figura 2.18: Le milestone sono bundle composti da due transazioni e referenziano la milestone precedente e una normale transazione del Tangle.

Se si esegue la chiamata all'API getNodeInfo (descritta nel paragrafo relativo ai Full Node) verso un qualsiasi nodo della rete si otterrà in risposta un oggetto json contenente i campi *latestMilestone*, *latestMilestoneIndex*, *latestSolidSubtangleMilestone* e *latestSolidSubtangleMilestoneIndex* (come mostrato nel codice riportato).

```
{
"appName": "IRI",
"appVersion": "1.4.2.4",
"jreAvailableProcessors": 8,
"jreFreeMemory": 133573312,
"jreVersion": "1.8.0_171",
```

```
7  "jreMaxMemory": 4772593664,
8  "jreTotalMemory": 3099066368,
9  "latestMilestone": "9EMUKSWHITOFFHYVXPVRVZURQISRIEQ9
       NNHLIZGGGFWKRMNYAXLMPYARLSDRHNGWJIMDOUAQHBEVZ9999",
10 "latestMilestoneIndex": 528457,
11 "latestSolidSubtangleMilestone": "9
       EMUKSWHITOFFHYVXPVRVZURQISRIEQ9
       NNHLIZGGGFWKRMNYAXLMPYARLSDRHNGWJIMDOUAQHBEVZ9999",
12 "latestSolidSubtangleMilestoneIndex": 528457,
13 "neighbors": 2,
14 "packetsQueueSize": 0,
15 "time": 1528898437495,
16 "tips": 5398,
17 "transactionsToRequest": 16,
18 "duration": 0
19 }
```

Listing 2.1: Risposta a getNodeInfo

latestMilestone hash dell'ultima transazione milestone ricevuta.

latestMilestoneIndex indice dell'ultima milestone ricevuta.

latestSolidSubtangleMilestone quando un Coordinator emette una transazione, questa referenzia e conferma un certo subtangle. Una milestone diventa solida per un nodo quando questo possiede tutte le transazioni referenziate direttamente o indirettamente da una milestone. Se alcune di queste mancano la milestone non sarà solida e il nodo non sarà perfettamente sincronizzato.

latestSolidSubtangleMilestoneIndex indice dell'ultima milestone solida per un nodo.

Si noti che dal campo latestSolidSubtangleMilestoneIndex è possibile determinare lo stato di sincronizzazione di un nodo. Il suo valore deve essere uguale a quello indicato da latestMilestoneIndex che a sua volta dovrebbe combaciare con il numero dell'ultima milestone emessa dal Coordinator. Un nodo non aggiornato non riesce a funzionare correttamente, avendo problemi ad emettere nuove transazioni nel Tangle.

La disattivazione di questo nodo speciale avverrà non appena la rete avrà raggiunto un alto valore di *confirmed transaction per second* (CTPS) e potrà proseguire in modo autonomo e sicuro. Il valore soglia per il CTPS non è determinato a priori ma vengono attuati dei test periodici per controllare lo stato della rete e la resistenza a certi tipi di attacchi in assenza del Coordinator.

2.9 Snapshot e Permanode

Gli *Snapshot* sono una delle peculiarità di IOTA. Essi permettono di ridurre la dimensione del Tangle eliminando gran parte delle transazioni, lasciandone solamente una per ogni

indirizzo con bilancio positivo. In questo modo il Tangle riparte da una lista di indirizzi e dal loro bilancio attuale.

Questa funzionalità è ancora in fase di sviluppo e attualmente gli snapshot sono decisi dai coordinator e annunciati attraverso i canali di comunicazione della community IOTA. In futuro è previsto che ogni nodo sia in grado di eseguire in modo automatico uno snapshot sul proprio Tangle.

Ma se tutte le transazioni con valore zero vengono eliminate, come è possibile utilizzare il Tangle come database su cui salvare dati provenienti dall'IoT in modo che rimangano per sempre? A questa domanda non c'è attualmente una risposta precisa. Gli snapshot sono tagli netti del Tangle in cui tutta la storia passata viene eliminata, quindi si può supporre che anche i vecchi messaggi vengano eliminati e questo sarebbe fonte di grande limitazione per alcuni tipi di applicazioni. Insieme agli snapshot sono però in fase si sviluppo alcuni full node particolari chiamati *Permanode*, i quali permetterebbero di mantenere al proprio interno tutta la storia del Tangle o una parte di essa.

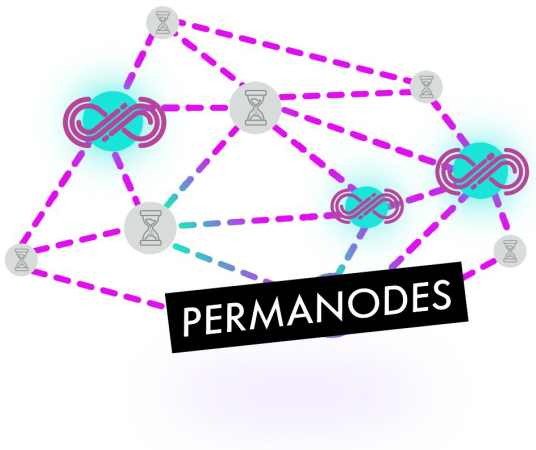

Figura 2.19: Permanode

Attualmente il sistema e in fase di alpha testing da parte della IOTA Foundation e verranno rilasciati aggiornamenti man mano che lo sviluppo procederà.

2.10 Moduli di Estensione

Il protocollo base di IOTA è pensato per essere leggero. Ha come scopo principale quello di abilitare l'invio di transazioni senza pagare nessuna fee in un ambiente trustless. Il sistema è stato progettato per essere completamente modulare, in modo che l'aggiunta di funzionalità potesse essere eseguito senza modificare il core di IOTA. I moduli di estensione sono chiamati *Iota eXtension Interface* (IXI). Con questa architettura è possibile sfruttare funzionalità aggiuntive solo se necessario, in modo tale da non appesantire il protocollo

base. Gli IXI rendono l'architettura scalabile e molto versatile. Tra i moduli principali e in stato di sviluppo avanzato si hanno:

- **Flash Channels**

  Permette lo scambio di transazioni istantanee off-Tangle.

- **Masked Authenticated Messaging**

  Permette la creazione di flussi di messaggi all'interno del Tangle, i quali possono essere letti ottenendo un punto di accesso.

- **Snapshot**

  Permette di leggere lo stato dell'ultimo snapshot dall'IRI.

Capitolo 3

Masked Authenticated Messaging

3.1 Introduzione al MAM

Come già detto IOTA si propone come tecnologia fondamentale per la gestione di dati e transazioni relative al contesto dell'Internet of Things, che si suppone sia caratterizzato da un numero estremamente elevato di piccole operazioni, flussi di dati e micropagamenti.

L' IXI denominato *Masked Authenticated Messaging* [12] è un modulo sperimentale introdotto nell'architettura IOTA nella seconda metà del 2017. Si tratta di un protocollo di comunicazione che aggiunge le funzionalità di trasmettere o consultare uno stream di dati cifrati sul Tangle. Il MAM si candida ad essere uno dei moduli più utili ed interessanti per il futuro dell'ecosistema IOTA.

La maggior parte delle DLT permette di appendere messaggi arbitrari alle transazioni ma in IOTA questo può essere fatto senza pagare delle fee. In ogni caso persiste la limitazione di poter pubblicare solo un messaggio alla volta e non messaggi consecutivi collegati. Se ad esempio si volesse pubblicare sul Tangle la temperatura letta da un sensore ogni 15 minuti in modo ordinato, sarebbe necessario inviare transazioni contenenti l'informazione tutte allo stesso indirizzo. Per un attaccante sarebbe però semplice identificare questo indirizzo e cominciare a inviare messaggi di spam per corrompere la serie di dati. Nel caso si volesse cambiare indirizzo su cui salvare i dati bisognerebbe tenere traccia di tutte le locazioni delle transazioni per poterle recuperare, operazione abbastanza costosa e inefficiente in termini di risorse.

Il MAM permette di creare una catena di messaggi, protetta dallo spam e libera dall'onere di mantenere traccia degli indirizzi. Ogni transazione viene inviata ad un indirizzo diverso, ma al suo interno contiene informazioni che permettono di conoscere come prosegue il flusso. Messaggi vecchi conducono solamente a messaggi nuovi, la catena procede in una sola direzione.

Si definisce quindi il concetto di *canale* in cui, come su di una qualsiasi piattaforma di trasmissione, il proprietario può pubblicare contenuti e gli utenti possono iscriversi al canale e vedere quello che viene pubblicato al suo interno. Il possesso del canale è sancito dal controllo di un seed IOTA, chiunque ne entri in possesso potrà pubblicare dati come se fosse il proprietario del canale.

Un utente a conoscenza di un punto di accesso al canale, ovvero un *channel ID* (rappresentato da un indirizzo), riuscirà a leggere tutti i messaggi da quel punto in poi, compresi quelli pubblicati nel momento in cui si è in ascolto sul canale con una modalità real time. Si potrebbe fare una similitudine con la radio: tutti coloro che si sintonizzano sulla frequenza desiderata sono in grado di ascoltare la trasmissione da quel momento in poi; nell'ambiente del MAM la frequenza radio è rappresentata dal channel ID, basterà sintonizzarsi a quell'indirizzo per intercettare tutti i messaggi di quella trasmissione.

Nell'ambiente del MAM le transazioni vengono inviate ad indirizzi derivati dalla cosiddetta *root*, ovvero la radice di un Merkle Tree. Ma a cosa serve questa struttura dati? Si supponga che un utente sia in possesso di una root e possa accedere ad un canale e leggerne il contenuto. Se questo utente prendesse l'ultimo elemento dello stream e leggesse l'indirizzo del prossimo messaggio ancora non esistente, potrebbe continuare il flusso pubblicando lui stesso un nuovo messaggio a quell'indirizzo. In IOTA le transazioni con valore zero non contengono una firma quindi sarebbe lecito. Ed è proprio per risolvere questo problema che c'è la necessita da parte dell'emittente di firmare i propri messaggi ed in questo meccanismo entra in gioco il Merkle Tree, riguardo al quale verranno dati più dettagli in seguito [11].

I messaggi possono avere qualsiasi dimensione ma, secondo una valutazione euristica, si dimostra che messaggi piccoli hanno un maggior potenziale per mantenere l'integrità dei dati. Un grosso video caricato sfruttando il MAM provocherà una saturazione della rete e quindi un'esperienza pessima per l'utente che dovrà scaricarlo per la riproduzione. Per questo si ritiene più efficiente utilizzare il protocollo per lo scambio di segnali o piccoli messaggi per il controllo di servizi off-tangle, ad esempio comandi per il controllo remoto.

Tutti questi messaggi sono inclusi nel Tangle e contribuiscono quindi alla sicurezza della rete, incrementando, come di consueto, il cumulative weight di altre transazioni a beneficio della proprietà di integrità della rete.

La diffusione dei messaggi nella rete sfrutta il cosiddetto *Gossip Protocol*. Un messaggio viene propagato sfruttando i meccanismi del protocollo IOTA e le interconnessioni tra nodi.

3.2 Considerazioni sul MAM

Una delle questioni più interessanti del MAM è il fatto che questo modulo punti fortemente verso il disaccoppiamento della componente finanziaria da quella legata al dato.

Fino ad oggi la componente finanziaria è stata sempre presente nelle DLT più conosciute. Come è stato già detto in precedenza, le transazioni nelle blockchain possono tranquillamente trasportare dei dati, ma questo può essere fatto al prezzo imposto dalla fee. IOTA è una DLT che permette di inviare informazioni senza alcun tipo di pagamento in token e questo rende il sistema utilizzabile a prescindere dalla criptovaluta ad esso associata.

Sicuramente IOTA non sarà immune dai discorsi legati alla speculazione finanziaria, infatti il valore dei token IOTA è molto seguito dagli investitori, ma è percepibile come il mondo legato alla tecnologia e quello legato alla finanza siano facilmente disaccoppiabili. IOTA permette di utilizzare tutti i benefici delle DLT senza dover necessariamente eseguire dei pagamenti.

Il Masked Authenticated Messaging si focalizza proprio sul lato puramente legato al dato, dando la possibilità di creare sistemi che sfruttano la rete solo per il trasferimento di informazioni e non per quello, voluto o meno, di token.

3.3 Modalità di Funzionamento

Il MAM permette di controllare la visibilità e l'accesso ai dati. Questo viene fatto attraverso tre modalità di funzionamento che verranno descritte in seguito.

3.3.1 Public

La modalità pubblica usa la root del Merkle Tree come indirizzo a cui inviare transazione che contiene il messaggio. Un utente casuale può trovare il messaggio e decodificarlo usando semplicemente il suo indirizzo. Da lì in poi potrà seguire il flusso e recuperare tutti gli elementi successivi.

Questa modalità è simile alle radio amatoriali e può essere usata ad esempio per annunci pubblici o comunque per trasmettere messaggi di cui non si ha interesse nel mantenere la segretezza.

3.3.2 Private

Nella modalità privata, l'hash della radice è usata come indirizzo. Questo impedisce ad un utente qualunque di decifrare il messaggio se è solo in possesso dell'indirizzo, poiché la root non è ricavabile dal suo hash. Ciò abilita alla lettura solo chi è in possesso della root stessa.

La modalità privata essere usata per cifrare flussi di dati pensati per non essere di pubblico dominio. Tutti possono vedere il flusso ma solo chi possiede la root può decifrarlo. Questo rende questa modalità utilizzabile da dispositivi che vogliono comunicare privatamente tra di loro.

3.3.3 Restricted

La modalità ristretta aggiunge una chiave di autorizzazione alla modalità privata. L'indirizzo usato per l'emissione del messaggio è l'hash della root. In più per poter decifrare un messaggio è necessaria una chiave aggiuntiva, con la quale vengono cifrate le informazioni. Un emittente può modificare la chiave in ogni punto dello stream, revocando così l'accesso agli utenti in ascolto.

Coloro che vogliono avere accesso al canale hanno quindi bisogno di un elemento aggiuntivo oltre alla root, ovvero la chiave di autorizzazione (indicata con il nome di *sidekey*).

3.4 Proprietà

3.4.1 Forward Secrecy

Siccome ogni messaggio contiene la root del messaggio successivo, è semplice seguire lo stream data la root di uno dei messaggi (a patto di essere a conoscenza della modalità e delle eventuali informazioni aggiuntive).

Poiché ogni messaggio punta solamente al successivo, non c'è alcun modo di leggere i messaggi precedenti al punto di accesso di cui si è in possesso. Non è dunque possibile seguire lo stream in senso contrario. Il flusso di dati è visualizzabile come un'autostrada: iniziare a leggere lo stream è come entrare in autostrada, non è possibile procedere contromano. Questa immagine sarà utile per comprendere meglio la combinazione di questa proprietà con quella descritta di seguito.

3.4.2 Channel Splitting

Un emittente può decidere di dividere in due parti il canale in qualunque momento, creando una biforcazione. Ne consegue la possibilità di canali alternativi derivati da quello principale, utilizzabili per un sub-set di dati ai quali non si vuole dare la stessa visibilità del canale padre. Questa proprietà permette di dare un accesso ai dati con granularità fine.

Può essere preso come esempio un record di identità personale. Esso può avere due sezioni principali, una di dati pubblici e una di dati privati. La parte pubblica userà la modalità Private del MAM, dando la possibilità a tutti gli utenti in possesso della root di leggere i messaggi contenenti informazioni come il nome, una lista degli interessi e un'immagine. Supponendo che l'utente voglia andare dal medico, potrebbe voler condividere con lui dati relativi alla sua salute. Questi dati sono di natura privata, perciò sarebbe opportuno utilizzare una modalità di tipo "Restricted" con una chiave di accesso a un certo tipo di informazioni relative alla salute dell'utente.

Il discorso può essere ripetuto per un qualsiasi tipo e numero di informazioni di cui si voglia gestire la visibilità. Ogniqualvolta si decida di modificare la visibilità sui dati è sufficiente dividere il canale modificando i parametri di accesso.

Grazie alla proprietà della Forward Secrecy chiunque ottenga accesso ad un ramo del canale non potrà navigarlo a ritroso per aver accesso a informazioni a lui non destinate.

Un'altra applicazione di questa proprietà si può ritrovare in sotto-canali di breve durata, creati per avere una granularità più fine dei dati. Si immagini un dispositivo che emette dei report su dati ambientali con una certa cadenza. In seguito alla rilevazione di un'anomalia potrebbe dividere il canale per iniziare a pubblicare dati con una frequenza maggiore, in modo da poter aver più informazioni per l'analisi dell'anomalia. Ciò permette di avere, in modo molto flessibile, lo stream principale invariato e degli stream secondari con dati pubblicati ad una frequenza maggiore, utili a coloro che dovranno svolgere l'analisi specifica sull'anomalia riscontrata.

3.5 Merkle Tree

In crittografia il Merkle Tree è un albero binario in cui le foglie sono etichettate con l'hash di un blocco dati ed ogni nodo è etichettato con l'hash delle etichette dei suoi figli. Gli alberi di hash permettono una sicura ed efficiente verifica dei contenuti di un'ampia struttura dati.

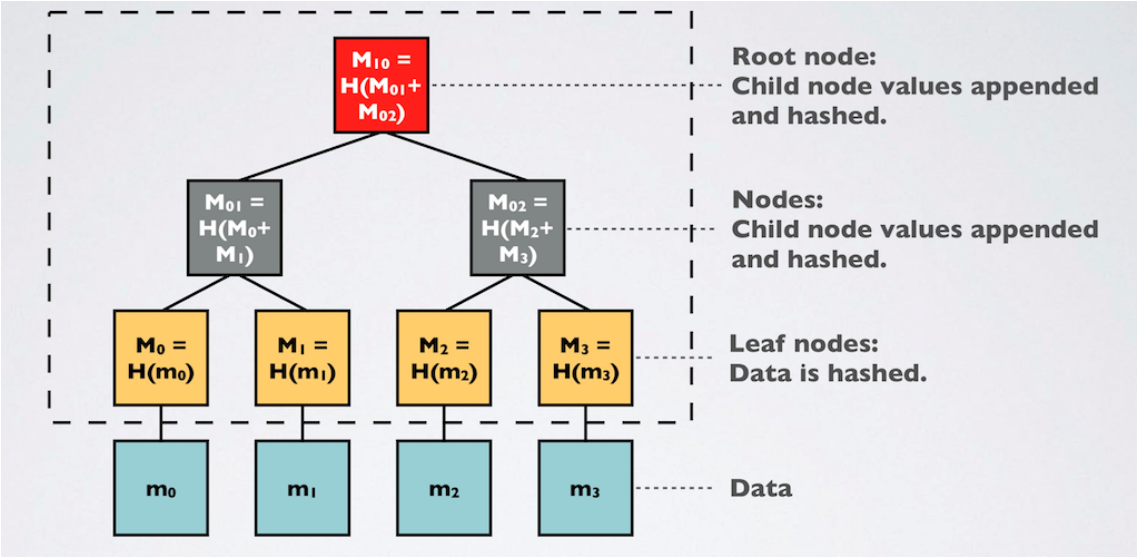

Figura 3.1: Merkle Tree

Nella figura 3.1 viene mostrata la rappresentazione del Merkle Tree. Si noti come il dato in sè non venga considerato come parte dell'albero, il quale ha come foglie solamente l'hash del dato.

Dimostrare che una foglia è parte di un dato albero richiede il calcolo di un numero di hash proporzionale al logaritmo del numero delle foglie. Ed è questo il motivo dell'efficienza di questa struttura.

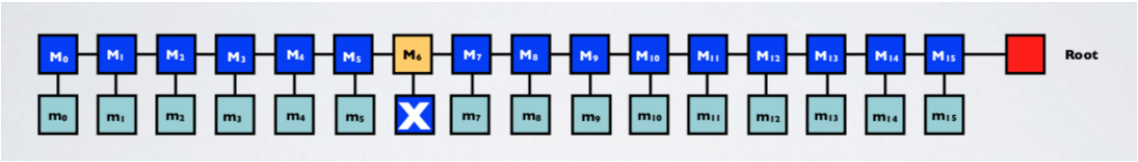

Figura 3.2: Merkle Tree

Si faccia riferimento allo schema mostrato in figura 3.2, nel quale viene semplicemente calcolata la root come hash degli hash di ogni singolo blocco di dati. In una struttura del genere, un qualsiasi utente che sia in possesso della root autentica e di un qualsiasi blocco dati e che volesse verificare che esso non sia stato alterato, dovrebbe ricevere anche gli hash di tutti i blocchi di cui non è in possesso, in modo da poter calcolare l'hash degli hash e verificare che corrisponda alla root.

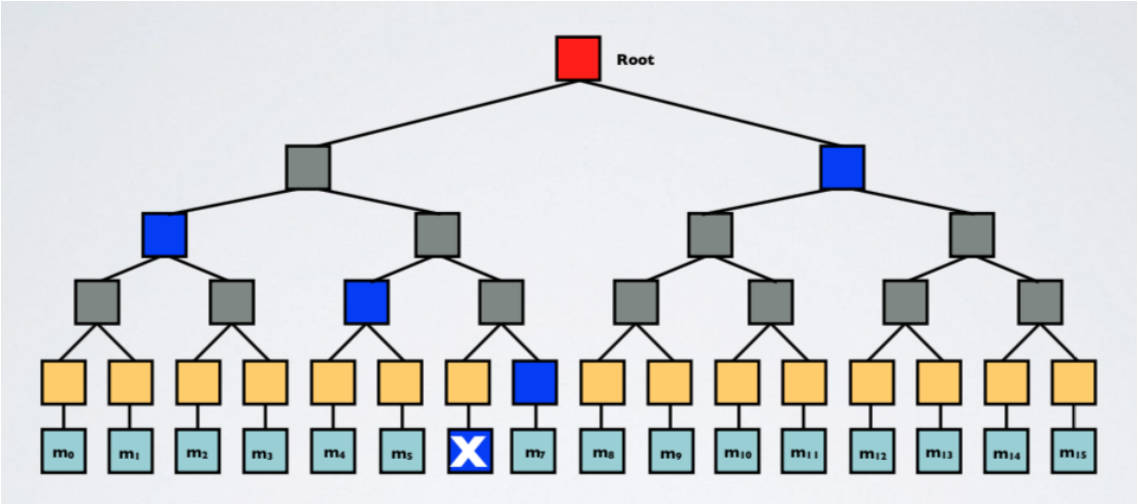

Figura 3.3: Merkle Tree

Si osservi ora lo schema mostrato in figura 3.3. Per eseguire l'operazione di verifica descritta in precedenza non è più necessario che all'utente vengano forniti tutti gli hash dei messaggi, ma gli sarà sufficiente conoscere gli hash indicati con il colore blu (in numero pari a $\log_2 N$ con N pari al numero delle foglie). Tutti i nodi in blu prendono il nome di *root sibilings*, ovvero gli elementi che permettono di ricavare il percorso che va dalla foglia alla radice, il quale viene chiamato *root branch*. La quantità di dati necessari è quindi molto inferiore e l'operazione risulta essere molto più efficiente. Alberi composti da un grandissimo numero di foglie permetteranno di eseguire la verifica ricevendo un numero di hash piccolissimo se paragonato al totale delle foglie, come evidenziato dalla figura 3.4.

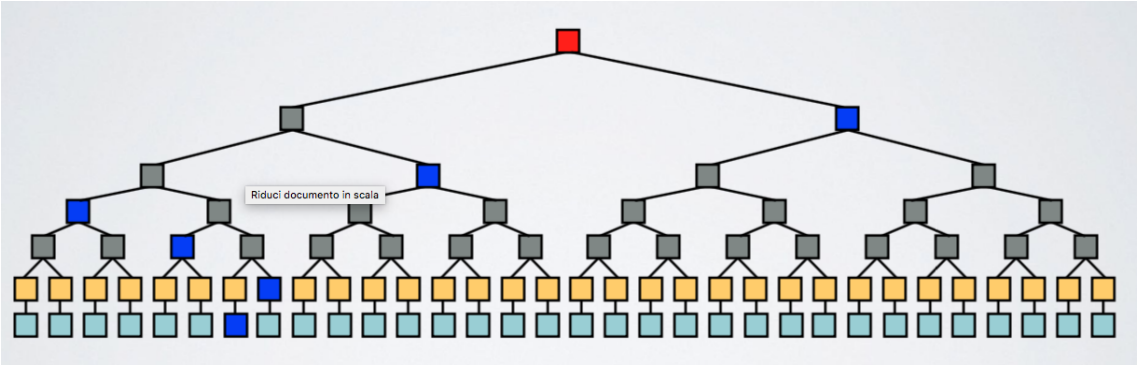

Figura 3.4: Merkle Tree

3.6 MAM Object

Il *MAM Object* è la struttura dati che contiene tutte le informazioni necessarie alla gestione di un canale. Essa viene utilizzata ogniqualvolta si desideri pubblicare un nuovo messaggio, al suo interno mantiene tutte le informazioni per costruire il flusso di dati con le caratteristiche descritte in precedenza.

```
{"subscribed":[],
"channel":
        {"side_key":null,
        "mode":"public",
        "next_root":null,
        "security":2,
        "start":0,
        "count":1,
        "next_count":1,
        "index":0},
"seed":"BLTVJFAMLBS9
   TMDSJFUSGBFOQGQKEFMTTAWPDGFQBWGKELNWYUKVVUENDMM9
   LRDNXGTBCCZVLABXDJWQD"}
```

Listing 3.1: MAM Object JSON

L'elemento *channel* contiene tutti i parametri per la descrizione del canale.

side_key è la stinga utilizzata nella modalità "restricted" che funge da chiave per la cifratura e decodifica del messaggio, con uno schema di tipo simmetrico.

mode è il parametro che indica la modalità di funzionamento del canale, potrà avere uno dei tre valore "public", "private" o "restricted".

next_root contiene la root del messaggio successivo, necessaria per poter continuare lo stream. In modalità pubblica l'indirizzo del messaggio successivo è uguale alla root, mentre nelle modalità ristretta e privata l'indirizzo sarà l'hash della root.

security indica il livello di sicurezza. Questo parametro influenza la difficoltà del calcolo delle chiavi, rendendo più o meno sicura l'operazione di cifratura.

start si riferisce alla foglia del Merkle Tree dalla quale è necessario partire per la creazione del messaggio. Siccome ogni foglia dell'albero va utilizzata una sola volta, sarà necessario memorizzare l'indice della prima chiave valida da utilizzare.

count indica il numero delle foglie del Merkle Tree corrente.

next_count indica il numero delle foglie del prossimo Merkle Tree.

index è l'indice utilizzato per selezionare ed utilizzare una delle chiavi del Merkle Tree.

Il seed viene inserito nel MAM Object quando deve essere utilizzato per generare chiavi ed indirizzi.

3.7 MAM Payload

Oltre al messaggio, vengono utilizzati una serie di campi per salvare informazioni aggiuntive necessarie durante la lettura e la verifica del messaggio stesso. Tutti questi elementi compongono il *masked payload* [9], il quale viene codificato nei 2187 trytes del campo signatureMessageFragment di una transazione IOTA. Siccome il payload ha una dimensione superiore a quella offerta da una singola transazione IOTA, questo verrà suddiviso in più parti racchiuse all'interno di un bundle.

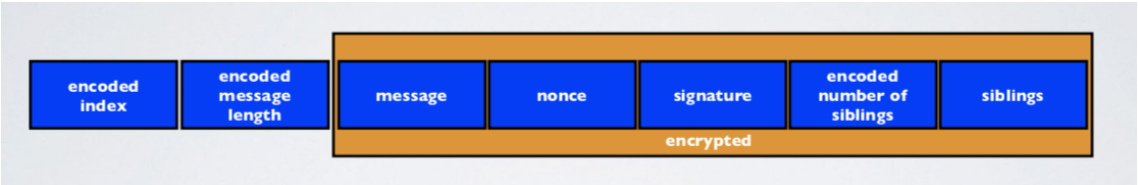

Figura 3.5: Masked Payload

Nella figura 3.5 si possono osservare tutti i segmenti che compongono il payload di un messaggio. Come è possibile notare, gran parte di esso viene inserito cifrato.

Di seguito verranno descritti tutti i passaggi che portano alla creazione del masked payload, il procedimento è stato dedotto in gran parte dal codice delle librerie disponibili nella repository GitHub ufficiale di IOTA.

Eccetto le operazioni descritte nella sezione riguardante la generazione del Merkle Tree, tutte le altre hanno origine dalla funzione *create*, contenuta nella libreria rust del MAM. Questa riceve in input tutti i parametri necessari per la creazione del payload.

```
pub fn create(
        seed: &[Trit], message: &[Trit], side_key: &[Trit],
        root: &[Trit], siblings: &[Trit], next: &[Trit],
        start: isize, index: usize, security: u8,
..)
```

Listing 3.2: Funzione create per la generazione del masked payload.

Il processo sfrutta una funzione spugna [7], ovvero un algoritmo che preso un input di qualsiasi lunghezza, produce un output di lunghezza desiderata. Essa è caratterizzata da una memoria a stati, una funzione f e una funzione di padding p (per rendere input troppo corti di una lunghezza compatibile con f). Ad ogni passaggio, vengono aggiunti alcuni degli input allo stato interno corrente e viene applicata la funzione f. Gli stati interni rappresentano o degli interi segmenti del payload o componenti utili al loro calcolo.

3.7.1 Generazione del Merkle Tree

Il Merkle Tree viene creato con un processo bottom-up a partire da 2^n indirizzi generati a priori secondo il seguente schema.

La chiave è composta da un numero di segmenti variabile in base al livello di sicurezza: il primo livello vede una chiave composta da 27 segmenti di 81 trytes ciascuno; ad ogni

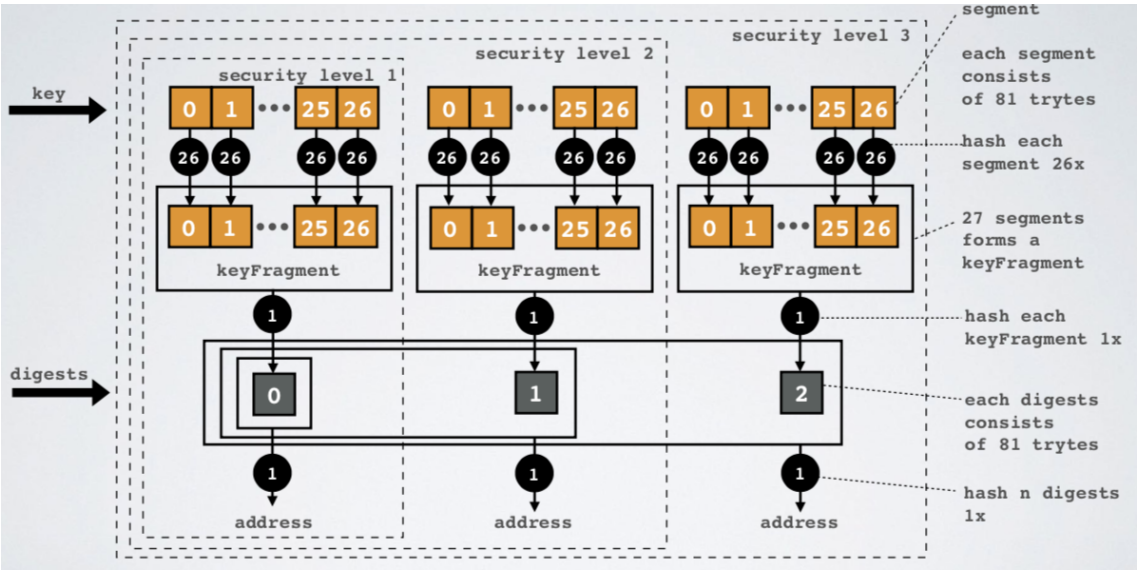

Figura 3.6: Processo di creazione di un indirizzo

livello verranno aggiunti 27 segmenti della stessa dimensione; il massimo livello di sicurezza è tre e quello di default è il secondo.

Per la creazione di un indirizzo (che si ricorda essere di 81 trytes) ad ogni segmento viene applicata la funzione di hash per 26 volte. Questa operazione dà origine ad un *keyFragment* per ogni livello di sicurezza. Ad ognuno di questi frammenti viene successivamente applicata la funzione di hash una sola volta. Ne risulta un digest da 81 trytes per ogni keyFragment. Viene calcolato l'hash dei digest come ultima operazione per generare l'indirizzo finale di 81 trytes (processo mostrato in figura 3.6).

Figura 3.7: Merkle Tree basato sugli indirizzi

Una volta ottenuti gli indirizzi si passa alla generazione del Merkle Tree. Viene calcolato l'hash di ogni indirizzo per generare le foglie dell'albero, dopodiché ogni nodo viene

generato come l'hash della concatenazione di due foglie, le quali diventano i figli del nuovo nodo. Ogni nodo di livello superiore viene calcolato come l'hash dei due nodi figlio fino ad arrivare alla radice.

Durante le esperienze che verranno descritte nei capitoli precedenti sono stati utilizzati alberi di dimensione pari a uno, in cui l'indirizzo coincide con la radice del Merkle Tree. Questo è dovuto ad alcune limitazione attuali della libreria javascript del MAM. Non si tratta comunque di un fattore limitante ai fini del risultato, in quanto, come precedentemente detto, questa struttura viene utilizzata per rendere più efficiente la verifica della firma.

3.7.2 Inserimento del Messaggio

Il messaggio contente le informazioni che si vogliono trasmettere utilizzando questo protocollo viene inserito come primo segmento della sezione cifrata del payload. Il campo *message* contiene due sezioni ed è di lunghezza variabile, poiché la parte dedicata al messaggio vero e proprio non ha una dimensione fissa ma dipende dal suo contenuto. La lunghezza del campo message è indicata in uno dei due segmenti non cifrati del masked payload, ovvero il campo *message length*. La seconda sezione del segmento message è invece occupata dal campo *next*, nel quale è indicata la root del prossimo messaggio.

3.7.3 Calcolo del Nonce

Il *nonce* è un numero basato su key, root, index, message length, next root e message. L'insieme di questi parametri, sotto forma di stato corrente della funzione spugna mostrata nella figura 3.8, viene usato come input per la funzione *search* insieme al valore relativo al livello di sicurezza. Come output del processo si ottiene un nonce di 81 trits. Si noti che il nonce qui calcolato è completamente diverso da quello che si genera dalla PoW eseguita per inserire una transazione nel Tangle.

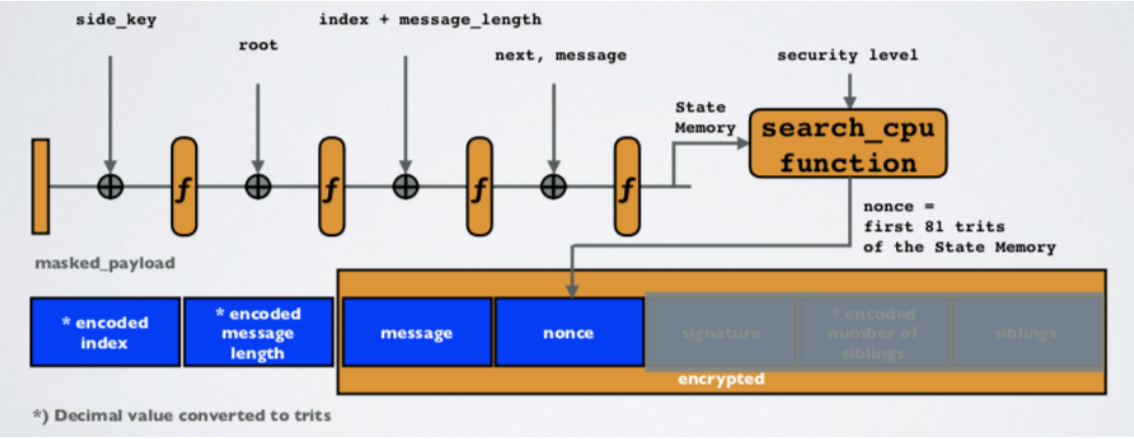

Figura 3.8: Calcolo del nonce.

3.7.4 Calcolo della Firma

Dopo il calcolo della nonce, questo viene reimmesso nella funzione spugna come input e vengono estratti dallo stato corrente i primi 243 trits che determinano il *bundle* o *bundleHash*. A partire dal bundle viene calcolata la firma.

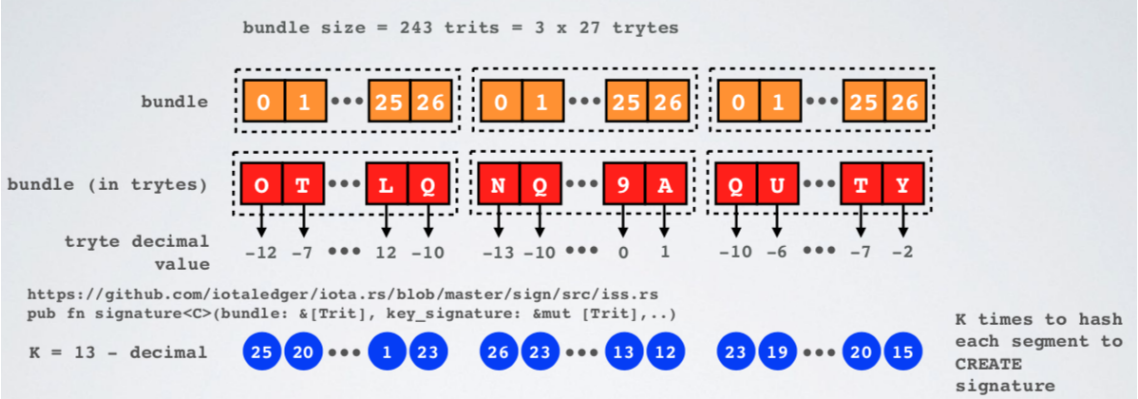

Figura 3.9: Calcolo dei valori K per la firma

Il bundle viene suddiviso in tre blocchi da 27 trytes l'uno. Il numero di blocchi utilizzati nell'operazione di firma è stabilito dal livello di sicurezza utilizzato. Ogni trytes viene convertito in decimale (*dec*) dopodiché, per ogni trytes viene calcolato il valore K tramite la formula $K = 13 - dec$ come descritto dalla figura 3.9. Ne risulterà un vettore di valori.

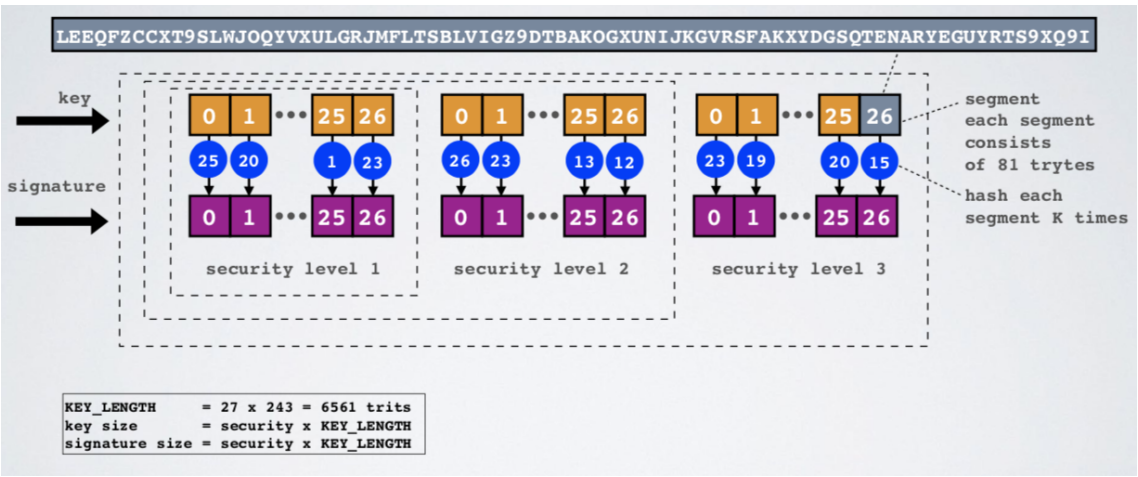

Figura 3.10: Calcolo della firma

Il valore di K indica il numero di cicli di hash che vengono applicati ad un segmento della chiave usata per la firma (ogni segmento ha una dimensione 81 trytes). Ogni elemento del vettore di valori K viene associato al segmento di chiave di pari posizione, come mostrato nella figura 3.10 La concatenazione dei segmenti risultanti dall'hash è la firma.

3.7.5 Calcolo dei Figli

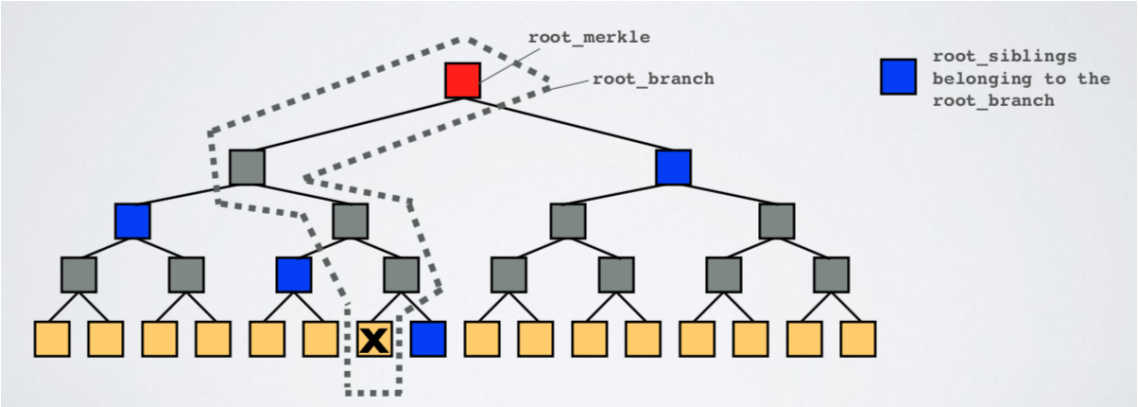

Figura 3.11: Conteggio e selezione dei sibilings

Per completare il payload si prende in considerazione la foglia del Merkle Tree associata al messaggio. Si tiene come riferimento il root branch (ovvero il cammino dalla foglia alla radice) e si contano i root sibilings (ovvero tutti i fratelli dei nodi appartenenti al root branch). Questo numero viene codificato in trytes e inserito nell'apposito segmento del payload (*number of sibilings*). I figli veri e propri, ovvero gli hash che identificano i nodi, vengono inseriti in un altro campo del payload chiamato *sibilings*. Come si può intuire questo è un campo di lunghezza variabile, che cambia in funzione della dimensione dell'albero e quindi del numero di sibilings. Questi campi saranno necessari per la verifica della firma.

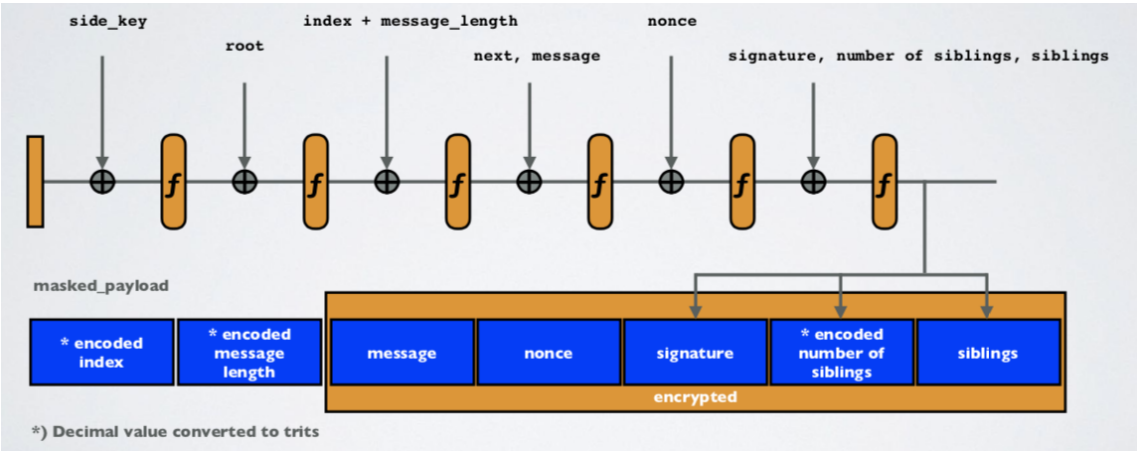

Figura 3.12: Inserimento degli ultimi tre campi del payload

3.8 Lettura del Flusso

La lettura di uno stream presuppone la conoscenza di un punto di accesso, identificato da un indirizzo. Come precedentemente detto, l'indirizzo sarà la radice del Merkle Tree in caso di modalità pubblica e l'hash della radice in caso di modalità privata o ristretta. Quest'ultima modalità presuppone la conoscenza di un'informazione ulteriore: la sidekey.

Una volta trovato lo stream ne si prende il primo messaggio puntato e lo si passa attraverso un processo in cui esso viene validato, decifrato e letto. Dopodiché si passa al messaggio successivo e si ripete il processo. Si noti che è possibile partire da qualsiasi punto del flusso, non solo dal primo messaggio.

Quando si consuma uno stream MAM la prima cosa da fare è validare la firma del messaggio che si sta leggendo, verificando se essa appartiene ad una delle foglie del Merkle Tree. Se la verifica fallisce l'intero messaggio viene considerato invalido.

Per prima cosa è necessario estrarre dal payload il valore del bundle. Questo viene fatto immettendo nella funzione spugna, oltre a sidekey e root, i primi quattro segmenti del payload. Estratti i 243 trits (81 trytes) del bundle dallo stato della spugna si passa all'operazione vera e propria di validazione.

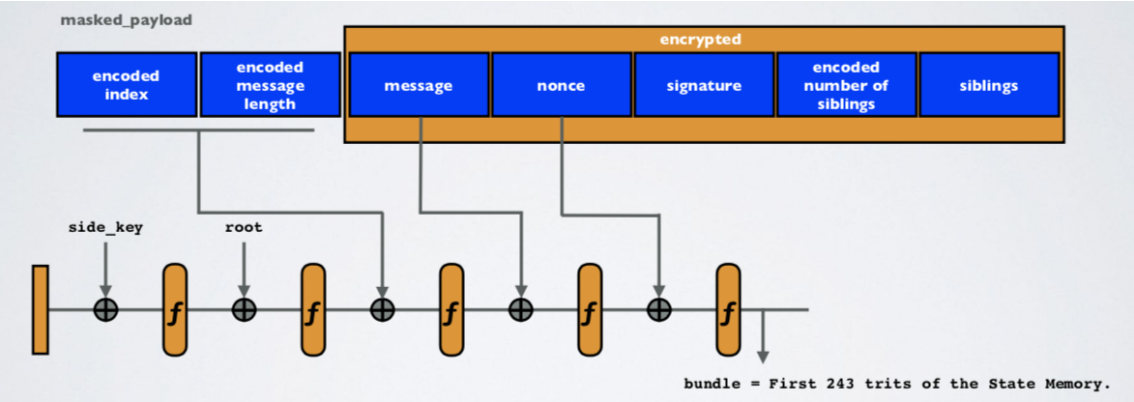

Figura 3.13: Estrazione del bundle

Viene dedotto dal bundle il livello di sicurezza, dopodiché, come nell'operazione di firma, viene suddiviso il bundle in tre blocchi da 27 trytes. Per ogni trytes viene calcolato il valore decimale (*dec*) e calcolato un valore K, questa volta secondo la formula $K = 13+dec$ come mostrato nella figura 3.14.

A questo punto viene estratta la firma dal relativo segmento del payload e ad ogni sua parte vengono applicati K cicli di hash secondo lo stesso schema visto per l'operazione di firma e descritto nell'immagine 3.15. La concatenazione di 27 segmenti determina un *key fragment*, il loro numero sarà pari al livello di sicurezza utilizzato. Ad ogni singolo key fragment viene applicato un primo ciclo di hash, per poi applicarne un secondo sulla concatenazione dei risultati del primo.

L'indirizzo risultante delle operazioni precedenti viene utilizzato per la verifica attraverso il Merkle Tree, insieme ai campi del masked payload relativi all'index e ai sibilings.

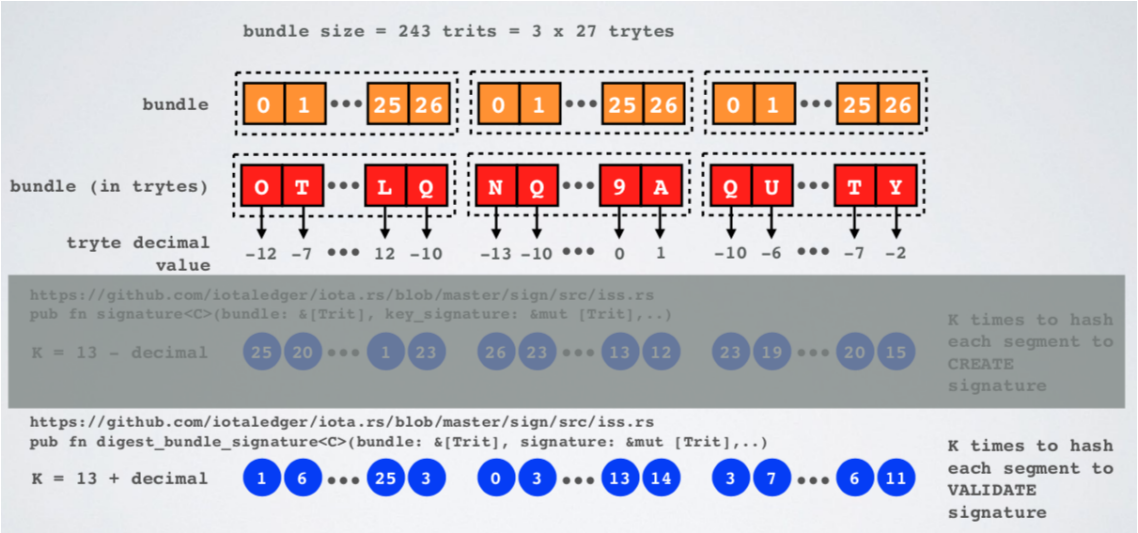

Figura 3.14: Calcolo dei valori K per la validazione

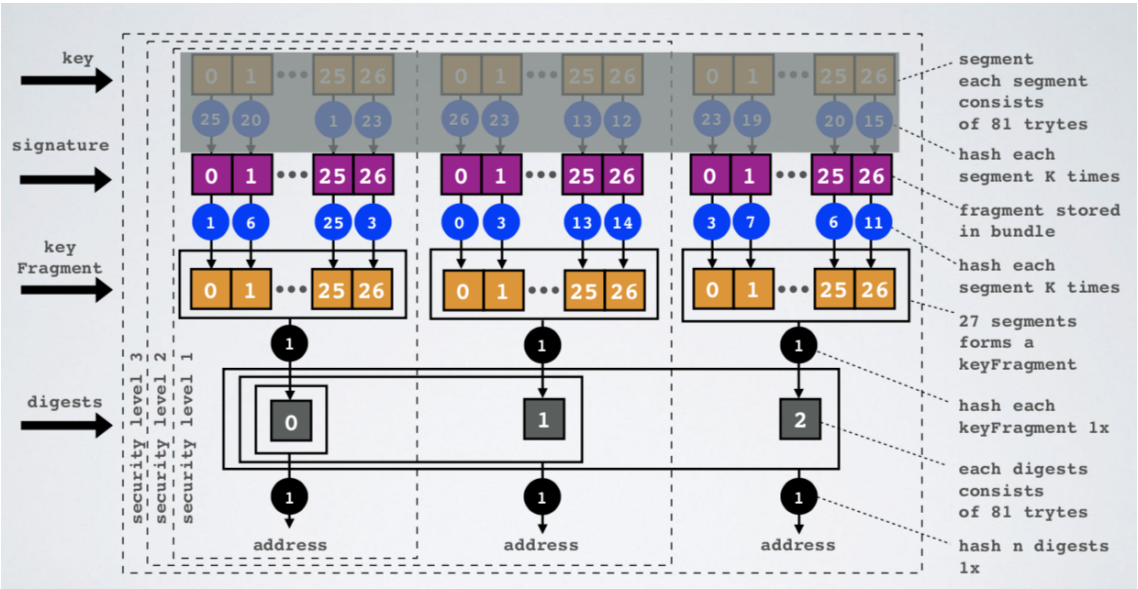

Figura 3.15: Validazione della firma

Conoscendo questi ultimi, si verifica se la root calcolata ricreando il root branch corrisponde con la root (conosciuta a priori) dello stream. In caso affermativo il messaggio viene validato poiché la firma appartiene ad una delle foglie dell'albero. Si passa quindi alla decodifica del segmento message del masked payload, estraendone le informazioni contenute nel messaggio e il valore "next root" grazie al quale sarà possibile continuare lo stream.

Capitolo 4

Mobility Data Marketplace

4.1 Introduzione

IOTA è una tecnologia che nasce per soddisfare alcuni bisogni del mondo Internet-of-Things. Secondo gli ideatori di IOTA nei prossimi decenni si assisterà ad una grandissima crescita di modelli di business basati sulla decentralizzazione in un contesto peer-to-peer in cui saranno le macchine gli attori protagonisti, si parlerà quindi del già citato paradigma Machine-to-Machine (M2M). L'IoT ha trovato terreno fertile nel campo della mobilità, dove i veicoli sono diventati dei centri di raccolta informazioni in movimento. Tutti i dati devono essere trasmessi e consumati nel modo più efficiente possibile. IOTA ha come obiettivo quello di inserirsi come struttura portante in questa rivoluzione, sia dal punto di vista del sistema di pagamento, sia dal punto di vista del trasporto e dell'immagazzinamento dei dati.

Oltre a questi aspetti, si trova intrinseco nel concetto di DLT anche la totale assenza di centralizzazione. Nessun attore è in possesso del potere necessario per decidere in modo autonomo le sorti del sistema. Questo meccanismo non è attualmente in uso nei più diffusi sistemi di raccolta e distribuzione dei dati. Nello scenario attuale si hanno tipicamente degli utenti che generano dati in modo passivo inviandoli, in modo più o meno consapevole, ad uno o più operatori, i quali da quel momento ne detengono il controllo assoluto e possono utilizzarli in totale libertà. Si prenda come esempio il mondo della mobilità in cui un veicolo, di proprietà di un individuo, genera dati destinati alla casa produttrice della vettura senza che il proprietario possa aver alcun controllo su di essi. Un altro esempio sono i dati di posizione che giornalmente escono dagli smartphone delle persone e finiscono nei data center di un gigante come Google o Apple, i quali hanno il completo controllo su di essi e sulla loro economia senza che l'utente ne tragga alcun beneficio.

Come sarebbe se si ribaltasse l'attuale paradigma e si portasse il meccanismo attuale in un contesto di totale decentralizzazione, in cui il dato è sotto il controllo di chi lo ha generato? Questo è la domanda che ha dato le basi al sistema che è stato pensato nel corso del lavoro di tesi e che sarà analizzato nelle prossime sezioni del documento.

4.2 Concept

I servizi di mobilità traggono enorme beneficio nel conoscere le abitudini dei propri clienti, questo permette di organizzare al meglio le proprie risorse, limitando gli sprechi e coprendo in modo opportuno le esigenze dei propri utenti. Questo progetto vuole partire da questa semplice idea per creare un sistema che possa portare beneficio sia agli utenti che ai fornitori dei servizi.

In questo contesto si hanno due attori principali: l'utente comune e un'azienda interessata all'acquisto del dato per analizzarlo e decidere se e come migliorare e ottimizzare il proprio servizio. Allo stato attuale l'azienda è obbligata a rivolgersi ad altre compagnie per acquistare questi dati oppure ad investire altre risorse per costruire un proprio sistema di raccolta dati.

Nel nuovo modello l'azienda sarà in grado di comprare i dati dall'utente, generando un compenso per il diretto generatore del dato senza che un attore terzo ne tragga un beneficio economico a suo discapito.

L'economia del dato passa sotto il diretto controllo dell'utente, i dati sono di chi li genera. Ogni informazione che garantisca un controllo sul dato rimane in possesso dell'utente fino al momento in cui non viene acquistato da qualcuno. In questo modo non è importante che le informazioni arrivino direttamente dall'utente, ma possono essere memorizzate su di un dispositivo remoto e recuperate solo grazie alle informazione mantenute segrete.

Questo sistema è stato battezzato *Mobility Data Marketplace* poiché pensato come infrastruttura per la compravendita di dati legati alla mobilità, un vero e proprio store in cui è possibile acquistare informazioni generate dalle persone direttamente da loro.

La parte del sistema sviluppata nel corso del lavoro è l'architettura che permette il salvataggio di un dato in arrivo da un dispositivo, in modo che le informazioni vegano memorizzate in remoto, senza che attori diversi dall'utente che ha generato il dato possano averne il controllo. Tutti gli altri elementi del sistema sono stati simulati tramite processi elementari.

4.3 Vantaggi del Mobility Data Marketplace

Riassumendo i vantaggi di questo nuovo concetto di economia del dato, è possibile evidenziarne i vantaggi principali:

Il dato è di chi lo genera. Colui che produce il bene di scambio principale ne è anche il possessore, senza che un attore terzo sia in grado di controllarlo e trarne un guadagno a discapito dell'utente. Questo comporta una maggiore consapevolezza degli utenti riguardo al valore dei dati che loro stessi producono senza rendersene conto e che regalano a compagnie senza pensare a quanto possano effettivamente valere. Inoltre genera un compenso economico diretto che prima era totalmente assente.

Totale assenza di centralizzazione. I dati vengono salvati su di un ledger distribuito, non sono quindi nelle mani di alcuna entità che possa modificarli o trarne un beneficio all'insaputa degli utenti. La visibilità sulle informazioni è quindi pubblica, ma grazie

alle proprietà offerte dal sistema solo il generatore del dato può autorizzare qualcuno a leggerle. Nessuna informazione sensibile passa attraverso un server centrale.

Dati sempre disponibili e immodificabili. Le DLT forniscono la garanzia che i dati siano disponibili ad essere recuperati in qualsiasi momento, inoltre ogni dato inserito non può più essere modificato. Non è più necessaria la fiducia in un attore terzo poiché questa è intrinseca nel sistema.

Anonimato del creatore del dato. Ogni utente che genera un dato è reso anonimo utilizzando solo le stringhe che identificano i suoi indirizzi, nessuno avrà visibilità sulle informazioni personali. Questo garantisce il completo rispetto della privacy e una garanzia in più per un utente che possa sentirsi spaventato nell'esporre in questo modo informazioni private.

Registrazione non necessaria. Chiunque potrà inserire dati senza essere registrato in alcun modo al servizio, e nello stesso modo potrà essere remunerato. Nessuna operazione preliminare sarà richiesta se non quella di entrare in possesso di un wallet relativo alla DLT, cosa che sarà facilitata dal software.

4.4 Problemi da Affrontare

Attualmente IOTA è ancora in fase di sviluppo, alcune librerie presentano limitazioni e non permettono di sfruttare a pieno le funzionalità del sistema. Altri moduli aggiuntivi di IOTA sono stati solamente annunciati e sono ancora in fase di testing o addirittura in fase di progettazione.

Smart Contract In particolare gli *Smart Contract* sono una funzionalità molto discussa nell'ambito delle DLT. Essi sono costituiti da porzioni di codice che modellano un contratto e sono salvati all'interno delle DLT sotto forma di una normale transazione. Il contratto è costituito da clausole e condizioni che, una volta verificate, portano allo sblocco ed esecuzione del codice. Questo avviene in modo del tutto distribuito poiché saranno i nodi stessi della rete a controllare che tutte le condizioni siano effettivamente verificate e che non ci siano operazioni conflittuali in corso, esattamente come succede nel meccanismo di consenso base delle DLT.

In IOTA questa funzionalità non è supportata in modo nativo. Il motivo principale di questa mancanza è dovuto alla struttura del Tangle che non ha un vero e proprio ordinamento temporale, anche se tutte le transazioni contengono un timestamp non si può essere sicuri riguardo alla loro accuratezza [14]. La IOTA Foundation è al lavoro sul progetto Qubic per sviluppare uno strato aggiuntivo per rendere disponibile questa funzionalità entro la fine del 2018 [15]. Dalle informazioni che si possono reperire si comprende che il meccanismo sarà basato su un *quorum*, ovvero sarà sufficiente che un certo numero di nodi verifichi la corretta attivazione di uno smart contract e non tutta la rete come avviene in DLT come Ethereum (la più famosa per questo tipo di funzionalità). Ne risulterebbe quindi un processo più leggero e meno dispendioso in termini di risorse,

che porterebbe ad avere un numero maggiore di transazioni processate per unità di tempo rispetto ad altre DLT.

Gli smart contract abilitano alcune funzioni molto importanti in un ambiente decentralizzato poiché permettono di eseguire operazioni in modo automatico senza dover passare da un punto di centralizzazione che prenda le decisioni.

Nel caso del Mobility Data Marketplace gli smart contract permetterebbero di inserire all'interno del Tangle porzioni di codice per regolare la distribuzione delle credenziali di accesso ad un pacchetto dati cifrato ogniqualvolta ne venga effettuato l'acquisto. Questo consisterebbe semplicemente nell'inviare una transazione verso l'indirizzo in cui è stato registrato il contratto per poter ricevere le chiavi necessarie alla lettura del dato.

Limiti delle Librerie Le librerie IOTA sono in continua evoluzione e aggiornamento. In particolare la libreria javascript che implementa le funzionalità del MAM è stata rilasciata solo da qualche mese e non permette di avere un controllo completo su alcune operazioni come la gestione del Merkle Tree.

Questo impedisce di sfruttare al meglio la proprietà di channel splitting offerta da questa tecnologia.

4.5 Architettura del Sistema

L'architettura ha come base principale la Distributed Ledger Technology IOTA e il suo modulo di estensione Masked Authenticated Messaging. Queste tecnologie sono incluse nei moduli software che compongono il sistema e vengono utilizzate per eseguire le operazioni più rilevanti.

Durante lo sviluppo sono state utilizzate due librerie per tutte le funzioni relative all'ambiente IOTA:

mam.client.js è versione ufficiale in javascript della libreria utilizzata per l'utilizzo dell'estensione MAM. Essa ha due varianti: una per essere utilizzata in un ambiente Node.js, mentre l'altra per essere utilizzata all'interno di un normale client web.

iota.js è libreria javascript principale del progetto IOTA, permette di eseguire qualsiasi tipo di operazione base, dalla gestione del seed con la generazione di chiavi e indirizzi, alla pubblicazione e alla lettura delle transazioni sul Tangle. Permette inoltre di interfacciarsi con i nodi per ottenere informazioni riguardo al loro stato.

Gli attori in gioco principali sono gli utenti che generano i dati e le compagnie interessate al loro acquisto, in più è presente un server intermediario per salvare informazioni utili a catalogare e visualizzare le tipologie di dato disponibili. Queste tre componenti principali verranno analizzate nel dettaglio nei prossimi paragrafi.

4.5.1 Utente

L'utente è rappresentato da un software che avrà due tipologie di compiti:

- Generare le informazioni di mobilità riguardo alla posizione geografica.

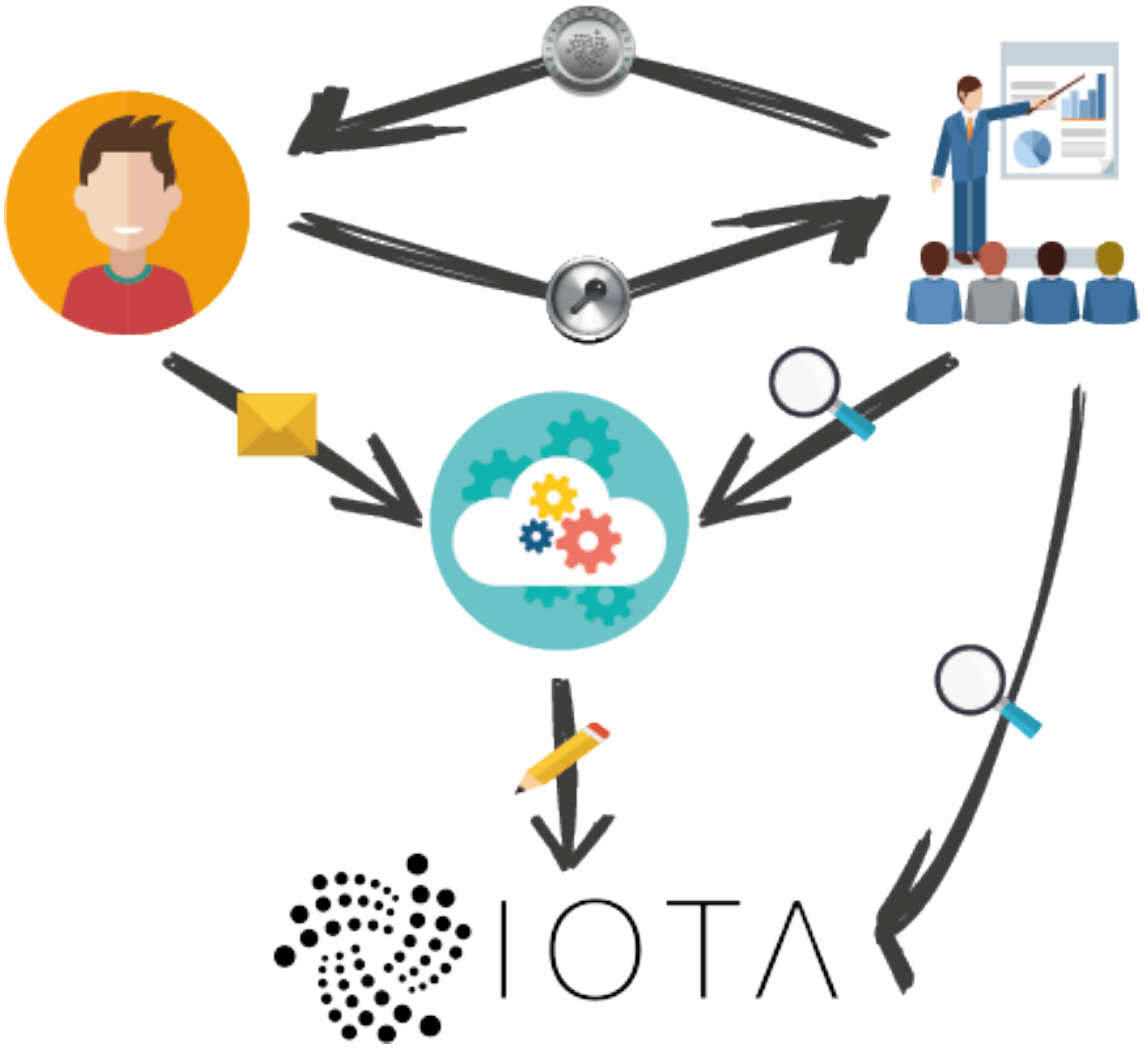

Figura 4.1: Architettura del sistema Mobility Data Marketplace

- Gestire l'interfacciamento con IOTA, inviando i dati di mobilità tramite il protocollo MAM e gestendo uno o più wallet necessari per generare le chiavi utili a mantenere il controllo sui dati, oltre che a ricevere pagamenti in seguito alla vendita del dato.

Il primo punto è coperto da un modulo preesistente e non è stato sviluppato, durante i test del sistema è semplicemente stato simulato attraverso dati generati in modo casuale.

Il secondo punto è invece stato oggetto di sviluppo approfondito in quanto sfrutta le tecnologie che sono argomenti fondamentali di questa tesi.

L'utente è rappresentato da un seed IOTA, il quale abilita tutte le operazioni di interazione con il Tangle. Il seed è l'informazione più importante e va tenuta segreta e al sicuro, la sua diffusione comporta la perdita di controllo sul proprio wallet e sui propri token. Inoltre chiunque verrà a conoscenza del seed potrà eseguire transazioni come se

fosse il proprietario del wallet, ottenendo quindi il diritto di pubblicare dati partendo dagli indirizzi generati da quel seed. Un seed corrotto non ha senso di essere utilizzato da nessuno, se non da chi lo vuole ottenere per rubare la criptovaluta gestita dalle sue chiavi.

Il seed viene utilizzato per generare l'oggetto cuore del sofware utente ovvero il MAM Object. Essa è la struttura dati fondamentale per il funzionamento del sistema viene creata sfruttando le funzionalità della libreria *mam.client.js*.

La funzione di libreria utilizzata per l'inizializzazione del MAM Object è la seguente:

```
1  Mam.init(iota, seed, security)
```

I tre parametri di input hanno il seguente significato:

iota: `Object` oggetto generato sfruttando le funzioni della libreria iota.js. Contiene le informazioni relative al punto di accesso all'ambiente IOTA, come l'indirizzo del full node scelto per interfacciarsi con il Tangle.

seed: `String` stringa di 81 trytes che determina il controllo su di un wallet IOTA.

security: `Integer` livello di sicurezza scelto per le chiavi. Il valore di default è pari a 2.

Il MAM Object viene restituito come output della funzione `init`, inizializzato con il `seed` passato come input e con un nuovo canale in modalità pubblica (`side_key` nulla) con i campi `start` e `index` a 0, il livello di sicurezza di default (2), i parametri `count` e `next_count` riguardanti un Merkle Tree con una singola foglia, `next_root` nulla poichè non ancora calcolata.

La seconda funzione di libreria interessante é `Mam.changeMode(state, mode, sidekey)` la quale permette di cambiare la modalità del canale e in caso la scelta cada su "ristretta", permette di impostare una chiave con cui cifrare il messaggio. Il parametro di input `state` stabilisce su quale MAM Object attuare le modifiche.

Il MAM Object viene mantenuto all'interno della memoria del software utente configurato in modo da poter pubblicare messaggi in modalità ristretta, cosicché l'utente possa cifrare messaggi con una sidekey mantenuta segreta fino al momento dell'acquisto.

La creazione del messaggio avviene attraverso la funzione

```
1  Mam.create(state, message)
```

i cui parametri di input sono rispettivamente lo stato attuale del MAM Object e il messaggio in chiaro codificato in tryte (tramite apposita funzione). Più interessante è invece l'output, rappresentato da un oggetto composto nel seguente modo:

state: `Object` MAM Object aggiornato incrementando o resettando in modo opportuno i quattro campi relativi al Merkle Tree.

payload: `String` masked payload definitivo (composto secondo gli schemi descritti nel capitolo 3 sezione 3.7) pronto ad essere pubblicato.

root: `String` root del Merkle Tree associato al messaggio.

address: `String` indirizzo utilizzato per la pubblicazione del messaggio sul Tangle.

Una volta generato il masked payload sarà sufficiente pubblicarlo sul Tangle mediante la funzione apposita della libreria MAM. Questa operazione viene delegata al server intermedio per questioni relative alle operazioni autorizzate da parte del full node che è stato appositamente configurato. Il payload verrà quindi inviato al server insieme a tutte le altre informazioni necessarie alla presentazione dei dati.

La chiave di cifratura dei dati viene mantenuta all'interno di un classico database SQL a bordo del dispositivo su cui viene eseguito il software dell'utente. Questa chiave viene cambiata in base alla granularità con cui si vogliono creare i pacchetti dati nello stream. I dati sono intrinsecamente organizzati in modo temporale, ogni pacchetto dati acquistabile può essere creato su questa scala semplicemente scegliendo una chiave diversa per ognuno di essi. La granularità è modificabile a piacimento con grande flessibilità. Pacchetti di dati che includono un lasso di tempo breve avranno pochi dati al loro interno, inoltre se un acquirente sarà interessato a un lasso di tempo ampio sarà necessario l'invio di molte chiavi per avere accesso a tutti i dati richiesti. Per ottimizzare il processo di scambio delle chiavi è necessario trovare il giusto equilibrio tra la grandezza dei pacchetti e le tipologie di richieste da parte degli acquirenti.

Ogniqualvolta si genera un nuovo dato viene recuperato dal database il MAM Object aggiornato all'ultima operazione eseguita e viene utilizzato per creare il nuovo masked payload da inviare al server. La chiave da utilizzare viene estratta anch'essa dal database e utilizzata per cifrare i dati. Nel caso sia necessario iniziare un nuovo pacchetto dati si genera una nuova chiave e si aggiunge al database insieme al punto di accesso al blocco di dati associato. Viene infine inviato al server un oggetto json contenete la root che identifica la porzione di stream e l'indirizzo su cui l'utente desidera ricevere il pagamento. La porzione di dati in chiaro inviati viene accompagnata dalla root, questa permette al server di identificare l'indirizzo al quale deve essere inviato il pagamento una volta che i dati vengono scelti da un acquirente.

4.5.2 Server

L'attuale assenza di smart contract nell'ecosistema IOTA ha obbligato la scelta di inserire un server intermedio tra le due parti fondamentali del sistema, ovvero gli utenti e gli acquirenti. Questo porta la creazione di un punto di centralizzazione e single point of failure.

Nonostante questo il server è progettato in modo da non poter trarre vantaggio dai dati in chiaro che riceve neanche se volesse agire in modo malevolo.

Uno dei suoi compiti è quello di offrire un'interfaccia per la visualizzazione dei dati in modo che l'acquirente possa selezionarne uno slot da acquistare. Successivamente il server deve risalire alla provenienza i dati selezionati, in modo da ricavare gli indirizzi per effettuare il pagamento verso colui che ha generato i dati e notificare gli utenti coinvolti nell'operazione della necessità di comunicare all'acquirente le chiavi per leggere lo stream.

Un altro compito delegato al server è quello di interfacciarsi con il full node IOTA e pubblicare il payload per conto dell'utente. Per fare questo deve implementare funzioni della libreria del MAM. La funzione più significativa è

```
1 Mam.attach(payload, address)
```

I suoi parametri di input sono:

payload: `String` masked payload ricevuto dall'utente per essere pubblicato sul Tangle.

address: `String` indirizzo utilizzato per la pubblicazione del messaggio sul Tangle, restituito dalla funzione utilizzata per la creazione del payload e comunicato dall'utente.

La funzione descritta ingloba al suo interno *attachToTangle* ovvero l'operazione che permette di inserire nel Tangle la nuova transazione creata. Quest'ultima viene restituita come output.

La comunicazione tra l'utente e il server avviene tramite API REST esposte da quest'ultimo. Esso è stato programmato utilizzando il framework Express in ambiente Node.js poiché garantisce estrema leggerezza e flessibilità. Grazie alle funzioni POST l'applicazione utente è in grado di inviare al server i dati da pubblicare sul Tangle, oppure i sample per la visualizzazione dei dati.

Full Node Per questo progetto è stato configurato appositamente un full node IOTA per garantire totale libertà di accesso al Tangle senza incorrere nelle limitazioni creare da nodi di terze parti. Questa necessità nasce dal fatto che la maggior parte dei full node pubblici non si prendono carico Proof-of-Work delegata da altri, risulta perciò impossibile effettuare l'operazione di attachToTangle. Avere il pieno controllo di un nodo garantisce grande libertà e permette di gestire a piacimento le operazioni consentite dalla macchina e i diritti di accesso ad essa.

Il full node è stato installato su una macchina di medie prestazioni ma sufficienti a tenere il nodo aggiornato e sempre disponibile quando richiesto. La PoW viene eseguita dalla GPU della macchina, la quale, essendo solamente quella integrata nel processore, non garantisce una grande potenza di hash ma comunque in grado di smaltire un'operazione di questo tipo nell'ordine di qualche decina di secondi.

I vicini del nodo sono stati aggiunti contattando direttamente la IOTA Foundation, poiché in questa fase iniziale di IOTA non è facile trovare nella community persone che eseguano un full node 24/7 perfettamente aggiornato. Una volta trovati nodi affidabili come vicini è stato semplice mantenere il Tangle perfettamente aggiornato.

Dati i continui aggiornamenti dell'IRI (il software che implementa il full node) è stato necessario eseguire continua manutenzione per non rimanere indietro con le numerose patch rilasciate.

4.5.3 Acquirente

L'acquirente deve poter essere in grado di visualizzare i dati su di una mappa e selezionare lo slot spazio-temporale a cui è interessato per poi valutarne l'acquisto.

Una volta presa una decisione il server comunica all'interessato gli indirizzi IOTA verso i quali effettuare i pagamenti e le root degli stream inclusi nella selezione.

Successivamente al pagamento l'acquirente riceve le chiavi per decifrare i dati acquistati direttamente dall'utente che li ha generati. L'operazione è gestita da un software che tramite gli indirizzi legge gli stream direttamente dal Tangle e con le chiavi ne decifra il contenuto, salvando il tutto su file.

Anche a bordo del modulo software eseguito dall'acquirente sono presenti funzioni della libreria MAM. La più significativa è

```
1 Mam.fetch(root, mode, sidekey)
```

I suoi parametri di input sono:

root: `String` root del Merkle Tree utilizzato come punto di accesso allo stream (diverso da address).

mode: `String` modalità dello stream che si vuole leggere, nel caso di questa applicazione sarà impostato su "restricted".

sidekey: `String` chiave ricevuta dall'utente che ha generato i dati e utilizzata per la decodifica del messaggio.

Come output saranno ritornati i due elementi seguenti:

nextroot: `String` root del prossimo Merkle Tree, necessario per continuare la lettura dello stream di dati.

messages: `Array` vettore contenente tutti i messaggi dello stream decodificati.

4.6 Risultati

Il modulo Masked Authenticated Messaging è stato sviluppato per dare la possibilità di creare flussi di dati ordinati all'interno del Tangle. Questa caratteristica è stata il centro del progetto del Mobility Data Marketplace. Essendo tutto basato sul sistema IOTA, ne vengono ereditati tutti i benefici e i difetti.

L'integrità dei dati è la prima proprietà ad essere garantita grazie all'architettura della DLT, come spiegato precedentemente il messaggio viene firmato ed inserito all'interno del Tangle, una sua modifica comporterebbe il ricalcolo del bundle e della firma delle transazioni. Un messaggio confermato dalla rete è reso immodificabile, la probabilità che un attacco nei suoi confronti abbia successo non è nulla ma trascurabile. Anche se un attaccante riuscisse nell'intento di modificare una parte dei dati contenuti nel Tangle questo vorrebbe dire aver speso soldi e risorse per farlo. I dati derivanti dal modo dell'IoT sono moltissimi e singolarmente hanno un bassissimo valore economico, nessuno avrebbe convenienza nel compiere un'azione di attacco di questo tipo, in cui il danno provocato avrebbe sicuramente meno valore di quello delle risorse spese. Si ricordi inoltre che il MAM è basato su transazioni che non contengono al loro interno nessun token, attacchi come il double spending non hanno alcun senso su transazioni di questo tipo.

Come altra caratteristica del sistema si ha la completa disponibilità dei dati in ogni momento poiché la rete IOTA è completamente distribuita e non ha problemi di down time.

4.6.1 Prestazioni

Per quanto riguarda le prestazioni ci sono varie operazioni che devono essere valutate. Tutti i dati sono stati raccolti con simulazioni sfruttando un laptop con CPU "Intel i7-3520M" e GPU integrata "Intel HD Graphics 4000" per svolgere le operazioni lato client, mentre per la parte server e soprattutto per il full node IOTA è stato utilizzato un pc desktop dotato di CPU "Intel i7-4770" e GPU integrata "Intel HD 4600".

Inizializzazione del MAM Object Preparare un nuovo MAM Object comporta la generazione di due Merkle Tree. Si ricordi che le limitazioni dovute alla libreria javascript permettono di generare alberi di dimensione uno. Ognuno di essi comporta il calcolo di un nuovo indirizzo a partire dal seed e due digest. L'operazione richiede un tempo di circa 1ms che si rivelerà essere un tempo trascurabile rispetto a quello impiegato dalle altre azioni necessarie nel resto della catena.

Creazione di un messaggio Un nuovo dato da parte dell'utente comporta la creazione di un messaggio. L'operazione include il calcolo del masked payload che comprende la cifratura delle informazioni trasmesse. L'operazione `Mam.create` impiega un tempo medio di 550ms con un minimo misurato pari a 540ms e un tempo massimo di 591ms. Queste tempistiche garantiscono la libertà di impacchettare un dato al secondo, soglia più che sufficiente per la trasmissione di dati completi.

Invio del messaggio al server L'invio al server del messaggio impacchettato è un'operazione limitata solamente dalla latenza di rete. La quantità di dati da inviare è veramente bassa. Una transazione misura 2673 trytes che convertiti in byte equivalgono a circa 1.55kB.

Inserimento della transazione nel Tangle Questa operazione è sicuramente la più costosa in termini di tempo. Il pc desktop utilizzato come full node riesce a calcolare una PoW con valore di difficoltà MWM (Minimum Weight Magnitude) impostata a 14 in un tempo che va dai 10 ai 60 secondi, un tempo eterno per un calcolatore e molto maggiore se paragonato ai tempi necessari per le altre operazioni. Visti questi numeri è chiaro che attualmente un solo full node non riuscirebbe a smaltire la grande mole di dati proveniente da tutti i dispositivi mobili.

Lettura dello stream dal Tangle Leggere uno stream di dati dal Tangle comporta l'esecuzione ripetuta delle operazioni di recupero dei messaggi online e decodifica. Il MAM impone che per ottenere il puntatore al prossimo elemento debba essere prima decifrato quello precedente. Questo meccanismo comporta la necessità di contattare un full node IOTA per ogni singolo messaggio, con il conseguente impatto del ritardo di rete ad ogni singola lettura. In ogni caso, supponendo una latenza tra i 10 e i 50 ms, questa impatterà al massimo sul 10% del tempo totale necessario per eseguire la lettura, poiché questo valore ha una media di circa 0.5s. Queste tempistiche sono dovute all'operazione di decodifica

della parte cifrata del messaggio e sono simili a quelle riscontrate durante l'operazione inversa.

Come è possibile dedurre da quanto detto, l'operazione di lettura richiede moltissimo tempo. Mezzo secondo per un singolo dato è una quantità di tempo enorme considerando che una compagnia interessata ai dati necessiterà di dataset molto grandi. Leggendo i dati in modo seriale si avrebbe un tempo di "download" altissimo. Se si considerano dataset di decine o centinaia di migliaia di dati il tempo diventa insostenibile, 100.000 dati sarebbero disponibili in circa 14 ore secondo le tempistiche calcolate.

4.6.2 Discussione delle Problematiche

Nella sezione precedente sono state evidenziate diverse problematiche riguardanti l'architettura del sistema e la tecnologia stessa. Di seguito verranno discusse alcune soluzioni.

Costo della PoW Questa operazione è sicuramente la più costosa in termini temporali. É possibile considerare varie alternative rispetto a quella proposta.

La Proof-of-Work è un'operazione che potrebbe essere delegata al client, questo permetterebbe di scaricare il full node da questo compito e diventerebbe solamente un punto di comunicazione con il Tangle. Vista la difficoltà della PoW questa soluzione non è attualmente praticabile visto che la capacità di calcolo di uno smartphone è nettamente inferiore a quella di un laptop. In un contesto in cui si ha un ecosistema di sensori collegati ad un gateway questo scenario acquisterebbe concretezza, dato che il centro stella potrebbe avere sufficiente capacità di calcolo da poter sopportare operazioni di questo peso.

La IOTA Foundation, in collaborazione con Jinn Labs, è però al lavoro su un dispositivo hardware che farebbe da acceleratore per il calcolo ternario, il tipo di matematica su cui è basato tutto l'ecosistema IOTA. Non si hanno attualmente ulteriori informazioni a riguardo, eccetto che sia un chip pensato per permettere anche a dispositivi dalle scarse capacità, come quelli IoT, di effettuare la PoW. Viste le difficoltà nell'eseguire la PoW da parte di un desktop comune probabilmente renderebbe più veloci le operazioni se adottato anche sui full node.

Una soluzione più interessante sarebbe quella di creare una rete composta da più calcolatori adibiti a full node, il che permetterebbe di scaricare il lavoro su più macchine, selezionandone una piuttosto che un'altra quando necessario. Va anche considerato il fatto che la rete IOTA potrebbe svilupparsi con la maggior parte dei full node pubblici e disponibili ad effettuare la PoW per tutti quelli che ne abbiano bisogno.

Attualmente questa è la problematica principale poiché un ritardo di questo tipo per ogni singolo dato da pubblicare non è sostenibile e nonostante i lati positivi di questa tecnologia non riuscirebbe minimamente a competere in termini pratici contro altri tipi di soluzione.

Un'ulteriore possibilità sarebbe quella di creare messaggi con più di un dato all'interno, il risultato sarebbe una diminuzione del numero di operazioni in modo direttamente proporzionale a quello dei dati inseriti in un masked payload. Uno stream con messaggi contenenti cinque dati ciascuno avrà bisogno di un quinto delle operazioni di attachToTangle. Un risparmio in termini di tempo non indifferente.

Tempi di download Come sottolineato precedentemente il download dei dati dal Tangle è molto lento e inefficiente a causa del gran numero di letture e decodifiche da effettuare.

La soluzione di impacchettamento multiplo di dati in un messaggio, proposta nel precedente paragrafo, permetterebbe di ridurre anche il numero delle operazioni di lettura oltre che quelle di scrittura. Il tempo di download decrescerebbe in modo direttamente proporzionale con il numero di dati in una transazione.

Il software di download potrebbe inoltre essere ottimizzato per la lettura in parallelo di stream provenienti da diversi utenti. La lettura sequenziale è obbligata all'interno di un singolo stream poiché il puntatore al messaggio successivo è criptato e deve essere estratto prima di poter proseguire. Siccome ogni utente genera un dato e i dati scaricati saranno di diversi utenti, è possibile eseguire in parallelo la lettura di stream diversi. Adottata insieme alla soluzione precedente diminuirebbe ulteriormente i tempi di download rendendoli più praticabili.

Si consideri il dataset di 100.000 elementi preso come esempio nella sezione precedente, per il quale erano state stimate 14 ore. Ricalcolando il tempo di download con messaggi contenenti 10 dati ciascuno e stimando 20 stream per dataset, si raggiungerebbe un tempo ideale di circa 4 minuti. Un tempo notevolmente inferiore a quello calcolato precedentemente.

4.7 Sviluppi Futuri

Il concept sviluppato sfruttando la tecnologia IOTA e in particolare il modulo MAM, offre molte possibilità di ampliamento poiché rappresenta solo la base di un sistema che potrebbe esserci costruito sopra. Considerando ambiti esterni a quello informatico, potrebbe essere molto interessante da studiare l'economia che ruoterebbe intorno ad un sistema di questo tipo.

Lo sviluppo del MDM potrebbe sicuramente continuare con la messa a punto di tutti i suoi moduli, a partire da una applicazione mobile che generi i dati di mobilità e li invii al server, oppure che comunichi direttamente con un full node per inserirli nel Tangle. Inoltre potrebbe mantenere al suo interno un portafoglio per tenere sotto controllo i propri token IOTA, utilizzabili per eseguire anche altri tipi di pagamento.

Dovrà essere implementato il portale per la visualizzazione dei dati così da rendere semplice e veloce la selezione dei dati.

Con l'avvento di una funzionalità simile agli Smart Contract si potrebbe ridisegnare il sistema eliminando persino il server intermedio, rendendo il tutto completamente distribuito e decentralizzato.

Il modulo MAM è in fase di sviluppo e alcune funzionalità non sono ancora disponibili nella libreria javascript. É in programma l'aggiunta della funzionalità *subscribe*, la quale permetterebbe a gli interessati di ricevere i dati in tempo reale, non appena questi vengano pubblicati sul Tangle. Si concretizzerebbe uno scenario in cui le informazioni arrivano man mano che vengono generate direttamente alle compagnie che le vogliono analizzare. Nel contesto del MDM si potrebbe selezionare un'area di interesse e cominciare a ricevere dati non appena questi venissero generati.

IOTA è una tecnologia giovane e in rapido sviluppo, le possibilità offerte da questo tipo di architettura sono infinite e attualmente esplorate solo in minima parte. Il MDM vuole essere un esempio di caso d'uso di IOTA per poterne studiare i vantaggi e le problematiche in applicazioni di questo tipo. In futuro possono essere esplorati altri modi di utilizzare il MAM per scopi differenti, effettuando valutazioni e confronti con tecnologie già esistenti.

L'interesse verso le DLT è in forte crescita e la completa decentralizzazione di questi sistemi porta alla creazione di nuovi concept, i quali prima non erano immaginabili. Il paradigma Machine-to-Machine è di forte interesse per la IOTA Foundation e verrà sviluppato fortemente nei prossimi anni. Si vedrà se IOTA riuscirà a raggiungere l'obiettivo di essere un elemento portante di questa rivoluzione.

Capitolo 5

Blockathon EU

5.1 Introduzione

Dal 22 al 25 Giugno 2018 si è svolta a Bruxelles la prima Blockathon EU, una competizione organizzata da EUIPO, l'Ufficio dell'Unione Europea per la proprietà intellettuale.

La sfida consisteva nel progettare e realizzare un sistema che andasse a combattere la contraffazione di beni basato su architetture distribuite quali le DLT.

La soluzione doveva coprire almeno una delle problematiche riscontrabili sulla catena che va dall'azienda manifatturiera fino al consumatore, passando per compagnie logistiche e dalle dogane.

Esperti di ognuno di questi settori hanno tenuto presentazioni in diverse sessioni organizzate prima e durante i lavori per esporre i problemi maggiori che ogni giorno devono essere affrontati lungo la catena. Le stesse persone sono rimaste disponibili per tutta la durata della gara per discutere con le squadre partecipanti riguardo alla bontà dei processi dedicati a risolvere specifiche problematiche.

Il team Links composto da quattro elementi dell'Istituto Superiore Mario Boella e due tesisti del Politecnico di Torino è stato selezionato tra le undici squadre partecipanti alla competizioni.

5.2 Le Sfide

I team si sono impegnati per risolvere almeno una delle tre sfide scelte tra consumatori, autorità doganali e operatori logistici.

Consumatori Un consumatore non vuole comprare prodotti falsi, consapevole del fatto che dietro un oggetto contraffatto c'è un lavoro non controllato che produce danni alle persone e alle economie. Sono necessari strumenti per capire se un prodotto sia falso o meno.

In campi come quello farmaceutico o automobilistico la contraffazione potrebbe avere conseguenze tragiche. É facile immaginare quali controindicazioni potrebbero esserci nell'assumere un farmaco non conforme alle regole o guidare un veicolo con componenti non a norma.

Ogni volta che si effettua un acquisto online sarebbe necessario essere sicuri che tutta la catena sia sicura, a partire dal produttore sino alla consegna del prodotto. Serve un meccanismo che porti la fiducia tre le parti coinvolte e che non ci sia nessuno che possa attaccare la catena di autenticità.

Dogane La dogana lavora per intercettare prodotti contraffatti, ha bisogno di una soluzione per svolgere in modo più efficiente il suo lavoro, fermando la criminalità e mantenendo i falsi fuori dal mercato. Attualmente è possibile controllare solo una piccola parte dei beni che transitano, si ha la necessita di aumentare il numero di prodotti controllati e la percentuale di controlli con esito positivo.

All'arrivo di una spedizione le autorità devono stimarne il rischio per decidere se controllarla o meno. Questa stima viene fatta sulla base dei dati fornite e dell'esperienza degli operatori.

Ci sono moli enormi di documenti da gestire e avere buoni dati e un sistema efficiente aiuterebbe a velocizzare il lavoro di scansione e verifica delle spedizioni.

Strumenti unificati per il calcolo dei rischi e la notifica di possibili irregolarità da parte di altri operatori della catena e verso le forze dell'ordine sarebbero utilissimi per le dogane.

Operatori Logistici Gli operatori logistici vogliono garantire che i prodotti che loro spediscono siano legittimi e non vogliono essere immischiati in attività criminali. Vorrebbero inoltre essere protetti da beni pericolosi che mettono in pericolo il business della compagnia e le persone a cui vengono spediti.

L'operatore logistico è una connessione chiave nella supply chain e i clienti possono essere dei tipi più svariati, dal privato alle aziende più grandi.

La reputazione di essere veloci, sicuri ed economici è molto importante per la compagnia logistica.

Ogni trasporto che si ferma per controlli causa ritardi e danni alla reputazione dell'azienda, serve un modo per assicurare i consumatori e le autorità che tutto quello che transita sia fidato.

Dato il numero di compagnie coinvolte nella rete logistica è necessario avere un metodo di comunicazione esteso e uniforme per aumentare la certezza sulla qualità dei beni.

5.3 Il Progetto

Il progetto proposto dal team Links è un'applicazione basata su tre elementi principali: una tecnologia in fase di brevetto (ViSeQR) di una startup toscana di nome VidiTrust, l'EUIPO's Enforcement Database e IOTA.

ViSeQR Una tecnologia basata sul riconoscimento dell'immagine di un codice QR e della sua cornice per identificare gli errori di stampa e associarli ad una determinata

macchina tramite una cosiddetta impronta digitale. Questo viene fatto addestrando preventivamente di un classificatore capace di riconoscere a quale stampate è associato un determinato tipo di errore.

EUIPO's Enforcement Database Un sistema creato da EUIPO per digitalizzare le informazioni dei prodotti e monitorare il loro stato lungo tutta la catena. Permette di segnalare merci sospette e aiuta gli operatori nel calcolo dei rischi.

IOTA Utilizzato per il salvataggio dei dati sui prodotti certificati dalla tecnologia ViSeQR sul Tangle, con il fine di garantire sicurezza dei dati e tracciabilità dei beni. La loro presenza nel registro distribuito ne garantisce la completa immutabilità e facilità di lettura.

Partendo da queste due tecnologie è stato costruito un sistema nominato *Dagoo* per offrire un servizio di controllo e certificazione lungo tutta la supply chain.

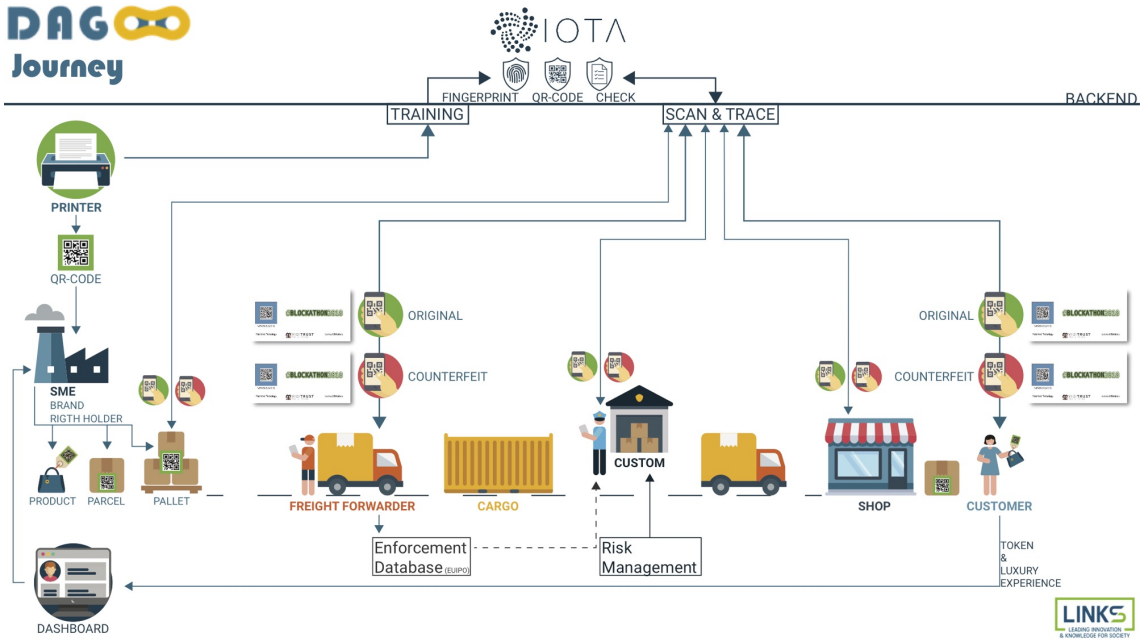

Figura 5.1: Supply chain con controlli tramite il sistema Dagoo

Nella figura 5.1 è mostrato il percorso che deve seguire un bene lungo tutta la catena e sono rappresentate le operazioni che i vari operatori dovranno eseguire per verificare lo stato della spedizione che sta transitando, con l'intento di intercettare o segnalare tutti i pacchi sospetti con la maggior efficienza possibile.

Il processo parte dall'azienda manifatturiera che registra una stampante utilizzando il servizio ViSeQR e salva nel Tangle la sua impronta digitale in modo che possa essere verificata in sicurezza. Successivamente vengono stampate delle etichette contenenti il codice QR con una cornice che darà la possibilità all'algoritmo di VidiTrust di riconoscere da quale stampante proviene l'etichetta. Quest'ultima verrà attaccata sulle singole scatole di un prodotto e su imballi più grandi.

La spedizione contrassegnata dai tag verrà verificata lungo tutta la catena sfruttando una semplice applicazione che scansiona l'immagine e comunica con il server che effettuerà una valutazione riguardo all'autenticità dell'etichetta. Ogni lettura verrà salvata nel Tangle in modo da rendere tracciabile ed immutabile il processo della spedizione. Qualunque tentativo di contraffazione del tag scatenerebbe una serie di contromisure che consistono nel registrare l'avvenuta scansione di merce sospetta nell'Enforcement Database di EUIPO.

Ad ogni scansione del tag l'operatore potrà verificare sull'applicazione lo storico della spedizione scaricato direttamente dal Tangle.

Alla dogana si potranno sfruttare le informazioni per calcolare al meglio i rischi di una determinata spedizione. Un lotto contenente pacchi verificati sarà meno a rischio di uno con merce dubbia.

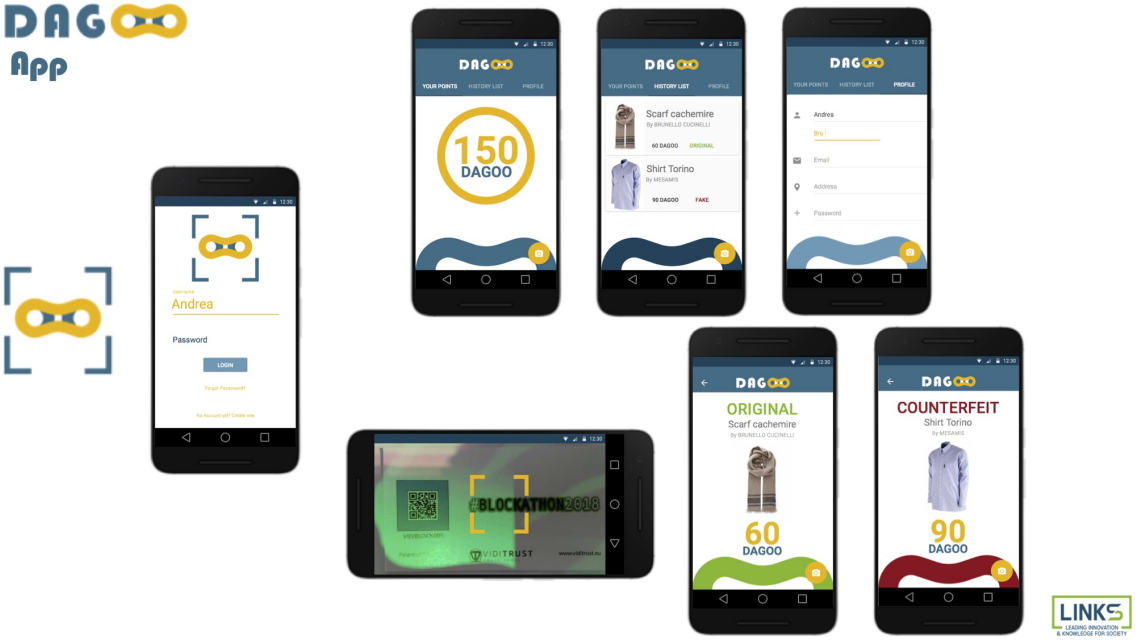

Figura 5.2: Applicazione per la verifica del prodotto

Il consumatore finale avrà un'applicazione a lui dedicata che gli permetterà di verificare l'autenticità del prodotto acquistato, generando feedback per l'azienda produttrice che ricompenserà l'utente con dei servizi extra.

5.4 Utilizzo di IOTA

All'interno di questo sistema IOTA è stato utilizzato per salvare in modo sicuro ed immutabile le informazioni relative ad una spedizione.

In particolare è stato riadattato il modulo sviluppato per l'applicazione Mobility Data Marketplace in modo che potesse essere eseguito sul server di gestione del sistema. Il MAM è stato utilizzato per creare degli stream di dati relativi ad un prodotto o un insieme di essi.

In seguito ad una scansione l'applicazione invia i dati per la verifica del tag al server. Quest'ultimo comunica con un database MongoDB per recuperare il MAM Object relativo allo stream del bene in questione, nel caso non siano presenti entrate ne inizializza uno nuovo. Ogni tag contiene un codice univoco che funge da identificativo del bene su cui viene applicato. Il risultato della scansione viene salvato come nuovo elemento dello stream relativo all'oggetto e la root dello stream viene memorizzata sul server in modo da poter consultare il flusso di dati sul Tangle ogniqualvolta venga richiesto da un operatore.

L'architettura di questo sistema, per quanto riguarda IOTA, è leggermente diversa da quella descritta nel concept del MDM. Tutto il pacchetto di moduli che implementano le librerie IOTA sono stati spostati lato server. Questo ha il completo controllo sui seed e sulle chiavi che permettono di pubblicare dati e accedere agli stream. Tutte le funzioni che operano sul Tangle avvengono lato server e sono esposte alle applicazioni client tramite API REST.

Il software a bordo dei dispositivi degli operatori sulla supply chain non mantiene al suo interno nessuna informazione sensibile ed esegue solamente chiamate al server per ricevere o inviare tutte le informazioni necessarie.

5.5 Considerazioni

L'applicazione descritta presenta dinamiche differenti da quelle che si sono osservate nel MDM.

Ogni lettura del codice QR genera necessariamente un solo dato, non è consentito raggruppare più dati in una singola transazione nel Tangle poiché le letture avvengono in momenti temporali lontani. Per questo motivo non ci saranno problemi di ritardo dovuti alla PoW, poiché essa non bloccherà le operazioni successive, le quali avverranno ben distanziate.

Un bene può "generare" solo una limitata quantità di dati, pari al numero di scansioni che vengono eseguite durante la catena. La verifica dello storico di una spedizione comporta la visualizzazione dei dati salvati nello stream. Il download richiede pochi secondi poiché l'operazione coinvolge un piccola quantità di messaggi.

Volendo si potrebbe permettere al software client di consultare direttamente gli stream di dati dal Tangle, sarebbe sufficiente generare il primo elemento quando viene stampata l'etichetta e inserire la root del flusso di dati direttamente nel codice QR. In questo modo il client potrebbe parlare direttamente con un full node per recuperare tutte le informazioni necessarie.

La differenza principale di questa architettura con quella del MDM è la presenza di un server che, agendo in modo malevolo, potrebbe corrompere tutto il sistema. Questo è un difetto che va contro la filosofia di totale distribuzione dettata dalle DLT. Bisogna però considerare che l'utilizzo di funzionalità come gli Smart Contract permetterebbe di ridisegnare l'architettura per eliminare il punto di centralizzazione.

5.6 Risultati

Il sistema è stato discusso durante tutta la durata della progettazione con gli esperti dei vari settori i quali si sono sempre rivelati interessati al prodotto, esprimendo giudizi positivi riguardo alle soluzioni proposte.

Nonostante la competizione sia stata vinta da un altra squadra, il team Links ha raggiunto un ottimo risultato attirando l'attenzione dei rappresentanti delle dogane e degli operatori logistici, segno della bontà del lavoro svolto.

Per quanto riguarda IOTA il team Links è stato l'unico ad aver utilizzato questa tecnologia e questo ha generato discussioni e confronti molto costruttivi con gli altri team e con gli esperti dell'area tecnologica. Questo ha dato la possibilità di conoscere le opinioni di persone esperte nel campo delle DLT e di valutare insieme a loro le caratteristiche principale.

Capitolo 6

Conclusioni

IOTA è una tecnologia nuova e ha bisogno di tempo per dimostrare tutto il suo potenziale e vedere se riuscirà a raggiungere i suoi obiettivi soddisfacendo le aspettative. Sicuramente le innovazioni adottate per risolvere alcuni problemi delle altre DLT sono molto interessanti. Tutte le sue caratteristiche peculiari puntano a far diventare IOTA un punto cardine per il mondo dell'IoT e abilitare nuovi paradigmi come quello del M2M.

Queste intenzioni sono ben chiare in un protocollo come il MAM, basato sul trasferimento di messaggi tramite il Tangle. L'esperienza di studio di questo modulo ha evidenziato come sia possibile utilizzare il registro per il salvataggio di qualsiasi tipo di informazione. Nonostante gli attuali limiti delle librerie appare chiaro come la comunicazione attraverso il Tangle possa avvenire in modo semplice e rapido.

L'IoT può sicuramente trarre vantaggio da queste funzionalità innovative, IOTA potrebbe diventare il sistema di trasferimento dati unificato, senza dover avere migliaia di piattaforme con strati intermedi per permettere la comunicazione tra le varie parti.

IOTA è un ecosistema che deve sicuramente crescere ancora molto poiché per ora ci sono dei limiti che possono far dubitare della sua reale efficacia. Il tempo di esecuzione della PoW è per ora un limite di questa tecnologia poiché, in un mondo in cui miliardi di dispositivi IoT dovranno comunicare inviando messaggi e pagamenti, le transazioni saranno moltissime e nonostante il grande numero di nodi non si sa come reagirà la rete all'enorme carico di PoW da svolgere.

La sicurezza è ancora argomento di studio e discussione. Vari stress test e simulazioni di attacco vengono effettuate periodicamente per osservare come si comporta la rete.

I due concept in cui è stata implementata questa tecnologia hanno dimostrato come il MAM possa essere adattato con estrema flessibilità ai vari casi d'uso. Il completo controllo sui dati dello stream è sicuramente uno strumento potente che permette di decidere con granularità molto fine chi abbia accesso ai dati e quali di essi possa visualizzare.

Inoltre con i prossimi aggiornamenti del modulo MAM si otterrà un ulteriore controllo sui flussi, introducendo la possibilità di creare dei veri e propri fork per inserire dati più specifici non accessibili agli utenti che hanno solo le credenziali per il ramo principale.

IOTA e tecnologie della stessa famiglia permettono di utilizzare un registro distribuito come database introducendo certe proprietà e alcuni limiti rispetto alle architetture tradizionali. Una delle domande che più spesso si leggono è: perché non farlo con un database

standard? La risposta è che si potrebbe tranquillamente fare. Ad oggi siamo abituati ad avere sistemi del tutto centralizzati, in cui si ripone la fiducia in qualcuno che detiene il completo controllo e che, agendo in modo malevolo potrebbe compromettere tutto. Quello che introduce una DLT a prescindere dalle caratteristiche tecniche è la possibilità di utilizzare un sistema senza doversi fidare di nessuno. Creare sistemi in totale assenza di fiducia è la vera sfida proposta dalle DLT e solo il tempo potrà mostrare se questa novità concettuale verrà applicata con successo.

Bibliografia

[1] The anatomy of a transaction. https://iota.readme.io/docs/the-anatomy-of-a-transaction.

[2] Bundles. https://iota.readme.io/docs/bundles.

[3] Irreversible transactions.

[4] Seeds, private keys and accounts. https://iota.readme.io/docs/seeds-private-keys-and-accounts.

[5] Very basic estimates on the risk of reusing the winternitz signatures.

[6] Cap theorem. https://en.wikipedia.org/wiki/CAP_theorem, Sep 2018.

[7] Funzione spugna. https://it.wikipedia.org/wiki/Funzione_spugna, Sep 2018.

[8] Internet of things. https://en.wikipedia.org/wiki/Internet_of_things, Sep 2018.

[9] Masked payload. https://www.mobilefish.com/download/iota/masked_payload_part20.pdf, 2018.

[10] Sybil attack. https://en.wikipedia.org/wiki/Sybil_attack, Sep 2018.

[11] ABmushi. Iota: Mam eloquently explained – abmushi – medium. https://medium.com/@abmushi/iota-mam-eloquently-explained-d7505863b413, Feb 2018.

[12] Paul Handy. Introducing masked authenticated messaging – iota. https://blog.iota.org/introducing-masked-authenticated-messaging-e55c1822d50e, Nov 2017.

[13] Serguei Popov. The tangle. *cit. on*, page 131, 2016.

[14] Serguei Popov. On the timestamps in the tangle. 2017.

[15] Ralf Rottmann. About smart contracts in iota – ralf rottmann – medium. https://medium.com/@ralf/about-smart-contracts-in-iota-626d2bd3619e, Mar 2018.