Università degli Studi di Padova

Facoltà di Ingegneria

Corso di Laurea Triennale in Ingegneria dell'Informazione

Analisi di Tangle, un grafo aciclico diretto utilizzato come registro distribuito

Relatore: Nicola Laurenti

Laureando: Luca Secchieri

Anno accademico 2021-2022

19 settembre 2022

Indice

Capitolo 1

Introduzione

1.1 Il registro distribuito

Per capire cosa sia Tangle e quale sia il suo scopo è prima necessario introdurre il concetto di registro distribuito (DLT-*Distributed Ledger Technology*).

Un DLT è un insieme di registri ripartiti in diversi nodi della rete,che vengono utilizzati per archiviare dati di vario genere, tra cui transazioni: ogni nodo contiene una copia identica del registro ed è proprio questa ridondanza delle informazioni a garantire la sicurezza del sistema. Una caratteristica importante dei DLT è la decentralizzazione: non c'è una entità centrale a cui fare riferimento ma tutti i partecipanti possono visualizzare e modificare la propria copia del registro per poi condividerla con gli altri nodi della rete. Una volta che una transazione viene approvata non è più possibile modificarla, proprio per garantire il corretto funzionamento del sistema. Infatti se un nodo provasse a modificare una vecchia transazione, aggiornando così la propria copia del registro, questo creerebbe un conflitto con tutti gli altri partecipanti, che avendo a disposizione la propria copia si accorgerebbero del problema. Per questo motivo sono necessarie una serie di regole e protocolli che i nodi devono seguire per raggiungere un consenso sulle transazioni da approvare. Questo criterio prende il nome di *regola del consenso* e stabilire il suo funzionamento è di vitale importanza per analizzare l'efficienza e la sicurezza di un DLT.

Grazie alle caratteristiche di sicurezza e trasparenza, il registro è visualizzabile da chiunque, i DLT trovano impiego nel campo della finanza: in questo ambito i nodi sono, ad esempio, i conti bancari mentre nel registro vengono scritte le transazioni tra i vari utenti. Inoltre l'immutabilità delle informazioni contenute nel registro rende il DLT lo strumento adatto a

tutti gli ambiti in cui è richiesta l'identificazione univoca di un prodotto, come ad esempio nella supply chain o per combattere la contraffazione di beni di vario genere, oppure di una persona, come nel caso di pagamento delle tasse o anche servizi sanitari.

1.2 Tangle e Blockchain, due diverse implementazioni per il registro distribuito

Un registro distribuito può essere implementato in moltissimi modi diversi, tra cui il più noto è senza dubbio la Blockchain.

La Blockchain, presentata nel 2008 dall'anonimo Satoshi Nakamoto, rappresenta la prima applicazione concreta di registro distribuito, utilizzato in questo caso per memorizzare le transazioni di una nuova valuta digitale chiamata Bitcoin. Questo protocollo è nato con lo scopo di permettere pagamenti online peer-to-peer, ovvero senza l'intermediazione di una terza entità esterna, e per riuscirci presenta una serie di soluzioni innovative che fanno ampio uso di crittografia. Sostanzialmente il registro è costituito da una catena ordinata e crescente di blocchi contenenti le varie transazioni. In particolare la rete è formata da due tipologie di partecipanti differenti: quelli che emettono le transazioni e coloro che hanno il compito di approvarle. Un nodo appartenente a quest'ultima categoria ha il compito di registrare tutte le nuove transazioni all'interno di un blocco candidato, per poi essere considerato valido a tale blocco è richiesta una cosiddetta *proof of work*, ovvero la soluzione di un problema crittografico che consiste nel calcolare un numero tale che l'hash del blocco rispetti determinate caratteristiche. Questo numero viene poi scritto alla fine del blocco, insieme all'hash del blocco precedente che viene scritto invece all'inizio. Questa operazione ha due funzioni: serve a stabilire un ordine preciso tra i vari blocchi e allo stesso tempo serve a garantire sicurezza al sistema. Infatti se un nodo provasse a modificare una vecchia transazione, modificherebbe di conseguenza l'hash di tutto il blocco e anche di tutti quelli successivi, il che richiederebbe un ricalcolo impossibile da completare nella pratica, a meno che non si disponga di una capacità computazionale maggiore di quella di tutto il resto della rete. Una volta aggiunto il nuovo blocco, la nuova versione del registro viene condivisa con il resto dei partecipanti alla rete. Nel caso di un eventuale conflitto fra transazioni, e conseguente biforcazione della catena, la regola del consenso adottata dal protocollo stabilisce che sarà ritenuta valida la catena di blocchi più lunga, ovvero quella che ha svolto un proof of work maggiore.

Nonostante gli innumerevoli pregi la blockchain presenta inevitabilmente alcuni limiti, in particolare:

1. Dal momento che solo il primo nodo a risolvere il problema crittografico ha diritto ad aggiungere il blocco alla catena, ne consegue uno spreco di risorse per tutti gli altri nodi, che dovranno ricominciare da capo per l'elaborazione del blocco successivo.

2. L'aggiunta di un blocco alla volta determina una limitazione alla quantità di transazioni che posso essere processate per unità di tempo.

3. Le differenti categorie di partecipanti alla rete porta a inevitabili discriminazioni che portano a sprechi di risorse per essere risolte.

Nel tentativo di superare i problemi intrinsecamente legati alla blockchain è stato ideato Tangle. Tangle rappresenta un'implementazione di registro distribuito radicalmente diversa: è stato abbandonato il concetto di catena, ora più blocchi possono essere processati contemporaneamente, inoltre tutti i partecipanti svolgono sia la funzione di emissione che approvazione delle transazioni. Nel seguito verrà fornita un'analisi approfondita di questo protocollo per quanto riguarda il suo funzionamento, stabilità e sicurezza.

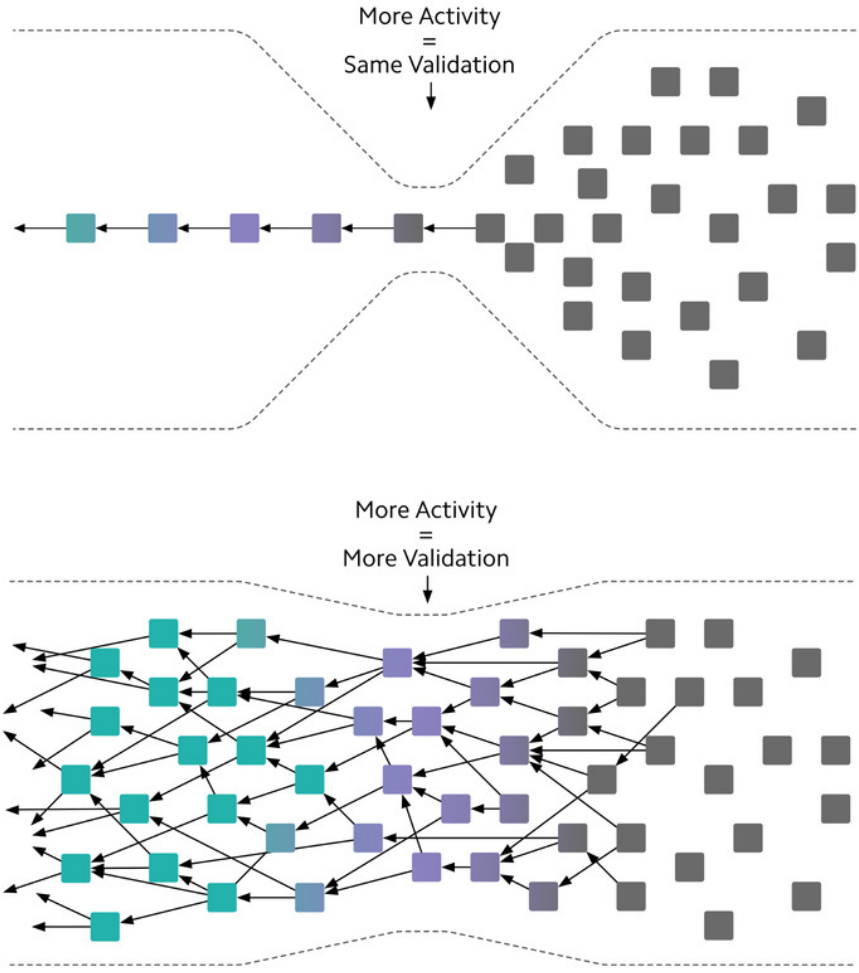

Figura 1.1: Confronto tra la struttura di Blockchain (a sinistra) e Tangle (a destra). Immagine tratta da iota.org.

Capitolo 2

La struttura di Tangle

In questo capitolo viene presentata la struttura di Tangle e il suo funzionamento generale. Successivamente vengono introdotti alcuni concetti utili per effettuare analisi più approfondite nei capitoli successivi.

Attualmente Tangle trova un'applicazione concreta in una cryptovaluta chiamata IOTA, e nonostante un registro distribuito possa trovare impiego in vari ambiti nel seguito si assumerà che Tangle venga usato come registro contabile.

2.1 Descrizione della struttura di Tangle

Un modo innovativo di implementare un registro distribuito è quello di utilizzare un grafo aciclico diretto, o DAG, chiamato Tangle per memorizzare le transazioni emesse dai nodi partecipanti alla rete. Queste transazioni vengono memorizzate una per ciascun vertice, o sito, del grafo. I vertici del grafo sono organizzati nella seguente maniera: ognuno di essi deve approvare due transazioni precedenti, il che equivale a dire che per ogni vertice devono esserci due archi diretti che lo collegano ad una coppia di transazioni precedenti. Se un arco collega x con y, si dice che x approva direttamente y. Se invece esiste un cammino di lunghezza maggiore di 1 che unisce questi due vertici, si dice che x approva indirettamente y. Quando Tangle viene inizializzato è presente un unico sito chiamato *genesi*: questo vertice speciale è l'unico a non approvare nessuna transazione, essendo la prima di tutte, e inoltre viene approvato direttamente o indirettamente da qualsiasi altra transazione. All'estremo opposto invece si trova l'insieme di siti che non hanno ancora ricevuto alcuna approvazione, questi vertici prendono il nome di *tips*.

Con Tangle non c'è più la distinzione tra le entità che approvano transazioni e quelle che le emettono, infatti ogni nodo per poter partecipare alla rete deve necessariamente svolgere entrambi i ruoli. Un nodo per aggiungere una transazione al grafo esegue le seguenti operazioni:

1. Vengono scelti due siti appartenenti al grafo. Per il momento si può assumere che vengano scelti casualmente, nei capitoli successivi verrà approfondita maggiormente la questione.

2. Il nodo controlla che le transazioni selezionate non siano in conflitto tra loro, nel caso lo fossero ne sceglie altre fino a quando non ne trova due adeguate.

3. Infine per completare correttamente la procedura di approvazione il nodo deve svolgere una proof of work, similmente a quanto accadeva per blockchain.

Sebbene due siti in conflitto tra loro non possano venire approvati dalla stessa tip, nulla impedisce ad entrambi di appartenere a Tangle contemporaneamente. Infatti i nodi non hanno la necessità di raggiungere un consenso su quali transazioni possono fare parte del grafo. Per gestire i conflitti però si utilizza un meccanismo in cui la transazione ritenuta non legittima rimane orfana, cioè smette di ricevere approvazioni.

2.2 Il peso di una transazione

In Tangle ad ogni transazione è associato un numero intero positivo n, appartenente ad un intervallo di valori prestabilito, che viene chiamato peso. Il peso assegnato a una transazione è direttamente proporzionale alla quantità di lavoro computazionale che un nodo ha dovuto svolgere per emetterla, e dato che generalmente maggiore è il peso maggiore è il tempo di elaborazione, si ritengono più "importanti" i vertici con un peso elevato. L'idea infatti è di impedire ad una singola entità di generare tante transazioni con un peso elevato in poco tempo, per evitare di congestionare la rete.

Un'altra metrica importante riguardo le transazioni è il peso cumulativo: esso equivale al peso proprio di un vertice sommato a tutti i pesi delle transazioni che lo approvano direttamente o indirettamente.

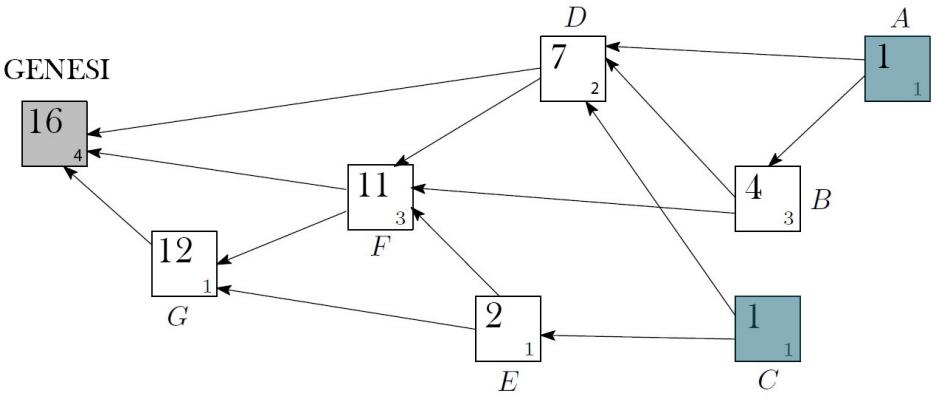

Figura 2.1: Esempio di DAG, il peso di un vertice è indicato in basso a destra, mentre il rispettivo peso cumulativo più in grande a sinistra. Le tips sono indicate in azzurro (adattato da [1]).

Capitolo 3

Stabilità del sistema

Riprendendo l'analisi effettuta in [1], in questo capitolo si analizzano alcuni aspetti della stabilità di Tangle, per poi affrontare la questione dei tempi di approvazione e di crescita del peso cumulativo, che come si vedrà in seguito, ha un ruolo chiave nella sicurezza del sistema.

3.1 Equilibrio delle tips

Si indica con $N(t)$ il numero di tips nel sistema in un dato istante t. $N(t)$ è una catena di Markov con un numero infinito di stati $[1, \infty)$. Inoltre, come affermato in [1], si dice che è ricorrente positiva dato che partendo da un qualsiasi stato s_i c'è una probabilità maggiore di zero di raggiungere ogni altro stato s_j. Una conseguenza di questa proprietà è che $\lim_{t\to\infty} \mathbb{P}[N(t) = k]$ esiste ed è positivo per ogni $k \geq 1$. Dal momento che se valesse $\lim_{t\to\infty} N(t) \to \infty$ molte transazioni non verrebbero mai approvate, e di conseguenza il sistema non sarebbe più funzionale, è ragionevole aspettarsi che $N(t)$ abbia una distribuzione stazionaria con media finita, ovvero $\mathbb{E}[N(t)] = N_0 > 0$. Per procedere con un'analisi più approfondita è necessario fare alcune assunzioni:

1. Le transazioni vengono emesse da un gran numero di nodi indipendenti, di conseguenza è possibile modellare il processo di arrivo tramite un processo puntuale di Poisson omogeneo di paramentro λ.

2. Tutti i dispositivi dispongono della stessa capacità computazionale e i tempi di elaborazione dei nodi sono dati da h_i, variabili i.i.d. con $\mathbb{E}[h_i] = m_h$.

3. Per semplicità, tutti i nodi per aggiungere la propria transazione scelgono in modo casuale due tips distinte da approvare. Questo

metodo di selezione casuale prende il nome di *Uniform Random Tip Selection Algorithm* e verrà trattato in modo più approfondito nel prossimo capitolo.

4. A causa della latenza del sistema, ogni nodo nel momento t in cui aggiunge la propria transazione non osserva lo stato attuale di Tangle, ma bensì quello a $t - m_h$ unità di tempo precedenti. Questo perché se l'entità i aggiunge una transazione al tempo t, questa sarà visibile al resto dei partecipanti solo al momento $t + h_i$.

5. Come anticipato si assume che il numero di tips rimanga stazionario nel tempo e oscilli intorno al valore $N_0 > 0$.

Per ricavare un'espressione di N_0 in funzione di λ e h_i innanzitutto bisogna fare una distinzione tra le tips *nascoste* e quelle *visibili*. Infatti dato che il processo degli arrivi è poissoniano all'istante t ci saranno $\mathbb{E}[\lambda h_i] = \lambda m_h$ tips nascoste, ovvero tutte quelle attaccate al DAG durante l'intervallo $[t - m_h, t]$ e pertanto non ancora visibili al resto dei nodi. Invece le tips visibili, indicate con r sono tutte quelle che erano presenti nel sistema prima dell'istante $t - m_h$ e che sono rimaste tali fino a t. Quindi il numero totale di tips sarà semplicemente la somma tra le due componenti

$$N_0 = r + m_h\lambda. \tag{3.1}$$

Grazie all'ipotesi di stazionarietà si può inoltre affermare che sempre all'istante t ci saranno mediamente $m_h\lambda$ siti che non sono più tips ma lo erano all'istante $t - h_i$.

Ora la probabilità che una transazione approvi una tip è pari a $\frac{r}{(r+m_h\lambda)}$, cioè il numero di quelle visibili diviso il totale. Dato che ogni transazione per venire attaccata al DAG deve approvare esattamente due siti, il numero medio di tips selezionate sarà $\frac{2r}{(r+m_h\lambda)}$. Sfruttando sempre la stazionarietà si nota che il questo numero medio deve necessariamente essere uguale a 1: se fosse maggiore allora ogni transazione in arrivo approverebbe due tips ma lei ne diventerebbe solo una. Chiaramente asintoticamente questo comportamento porterebbe $N_0 \to 0$ (per il caso <1 allora $N_0 \to \infty$).

A questo punto si ricava immediatamente $r = m_h\lambda$, a conferma dell'intuizione che il numero di tips rimaste tali nell'intervallo $[t - m_h, t]$ è uguale a quello dei siti che erano tips ma sono stati approvati. In conclusione si ottiene

$$N_0 = 2m_h\lambda. \tag{3.2}$$

3.1.1 Stima del tempo di conferma di una transazione

Con tempo di conferma si indica il tempo che trascorre dall'emissione di una transazione fino alla sua prima approvazione. Per affrontare la questione è utile distinguere due regimi di carico in base al numero di tips in arrivo nel sistema.

- **Basso carico:** in questa situazione il numero di tips è piccolo cioè $\mathbb{P}[N(t) > 1] < \epsilon$ con $\epsilon \ll 1$, di conseguenza è poco probabile che diverse transazioni approvino la stessa tip.

  Il tempo di conferma, sempre supponendo arrivi poissoniani, sarà quindi nell'ordine di λ^{-1} perchè una data tip verrà approvata da una delle prime transazioni in arrivo.

- **Alto carico:** il numero di tips è elevato, ovvero $\mathbb{E}[N(t)] \gg 1$. Questa situazione è tipica di un sistema con alta latenza unita ad un grande flusso di transazioni, in questo caso è probabile che una stessa tip venga approvata direttamente da transazioni diverse.

  Dato $m_h\lambda$ grande, il processo degli arrivi si può considerare indipendente per ogni tip e come dimostrato in [1] avrà tasso pari a $2\lambda/N_0$. Il tempo di conferma quindi sarà pari a $N_0/(2\lambda) = m_h$.

3.2 Come cresce il peso cumulativo

Per l'analisi della crescita del peso cumulativo conviene riprendere la distinzione tra i due regimi.

Nel caso di basso carico, dal momento che la tip spesso è una sola, essa verrà approvata indirettamente da tutte le transazioni successive, di conseguenza il suo peso cumulativo crescerà con velocità pari al tasso di arrivo λ.

Per quanto riguarda il regime ad alto carico invece, l'analisi si fa più complessa. Innanzitutto si introducono due nuove grandezze: $H(t)$ che indica il peso cumulativo al tempo t, e $K(t)$ il numero atteso di tips che approvano la transazione in questione al tempo t. Come dimostrato nei paragrafi precedenti, si può assumere N_0 costante. Dato che una transazione in arrivo all'istante t, vede il sistema come era al tempo $t - m_h$, bisogna tenere in considerazione che alcuni siti che erano tip potrebbero non esserlo più e in questo caso $K(t)$ aumenterebbe di 1. Ricordando che generalmente il numero di tips è $2m_h\lambda$, in un intervallo m_h una quantità pari $m_h\lambda$ di essere verrà sostituito da delle nuove tips. Ne consegue che la probabiltà che una tip al tempo $t - m_h$ rimanga tale all'istante t è proprio

1/2. Prende il nome di A questo insieme di $K(t-m_h)/2$ tips, mentre le rimanenti che sono state invece approvate durante l'intervallo $[t-m_h, t]$ appartengono all'insieme B. Ora si può calcolare la probabilità che il numero di tips $K(t)$ vari con l'arrivo di una nuova transazione. Ciò può accadere soltanto in due casi: se viene approvata almeno una transazione da B e nessuna da A allora il numero aumenta di 1, al contrario se le transazioni approvate appartengono entrambe all'insieme A, allora diminuisce di 1. Le probabilità di questi due eventi sono rispettivamente p_1 e p_2, e come ricavato in[1] valgono:

$$p_1 = \left(\frac{K(t-m_h)}{2N_0}\right)^2 + 2\frac{K(t-m_h)}{2N_0}\left(1 - \frac{K(t-m_h)}{N_0}\right) \tag{3.3}$$

$$p_2 = \left(\frac{K(t-m_h)}{2N_0}\right)^2 \tag{3.4}$$

A questo punto risolvendo l'equazione differenziale descritta in [1], si può ricavare il comportamento del peso cumulativo in funzione del tempo in regime di alto carico. La crescita si divide in due fasi: un cosiddetto "periodo di adattamento", a crescita esponenziale, che rappresenta la fase iniziale in cui non ancora tutte le tips in arrivo approvano indirettamente la transazione in questione, a cui segue una fase di crescita lineare di paramentro λ del tutto analoga alla situazione di basso carico.

Si ottiene che durante la finestra di adattamento $[0, t_0]$ la crescita è esponenziale e il peso cumulativo è dato da

$$H(t) \approx 2\exp\left(W\left(\tfrac{1}{2}\right)\frac{t}{m_h}\right) \tag{3.5}$$

con $W\left(\frac{1}{2}\right) \approx 0.352$, dove $W(\cdot)$ è la funzione di Lambert. Invece da t_0 in poi la crescita avviene con velocità pari a λ. L'istante t_0 è pari a

$$t_0 \lesssim \frac{h\ln(N_0)}{W\left(\frac{1}{2}\right)}. \tag{3.6}$$

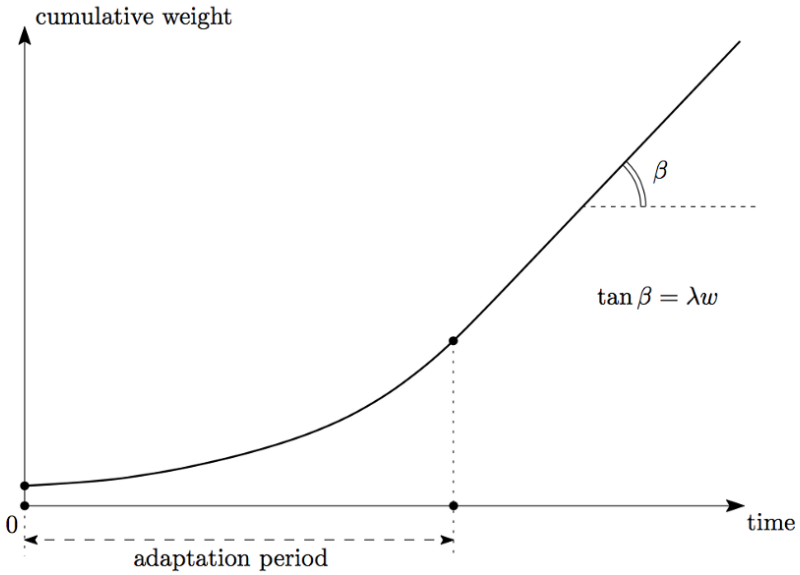

Figura 3.1: Come evolve il peso cumulativo nel caso di alto carico. Si distingue la fase di adattamento iniziale, con crescita esponenziale, da quella successiva a crescita lineare [1].

Capitolo 4

Vulnerabilità, attacchi e contromisure

Quando si utilizza un registro distribuito è importante essere consapevoli delle sue vulnerabilità, in particolare in questo capitolo verrà affrontato il problema degli attacchi a doppia spesa ed eventuali strategie da adottare per fronteggiarli. In seguito verranno analizzati alcuni algoritmi di selezione del tips per capire l'impatto che hanno sulla struttura e sicurezza di Tangle.

4.1 L'attacco a doppia spesa

Un *double-spending attack* è un tipo di attacco che, sfruttando la struttura di un DLT, permette di utilizzare gli stessi fondi per effettuare due pagamenti differenti. In generale i passi per eseguire un attacco simile sono i seguenti:

1. L'attaccante effettua un acquisto da un venditore e invia i fondi richiesti.

2. Quando il venditore ritiene il pagamento valido (es. la transazione ha un peso cumulativo sufficientemente grande), invia all'acquirente il bene in questione.

3. A questo punto l'attaccante aggiunge un'altra transazione al registro in modo da creare appositamente un conflitto con quella precedente.

4. La prima transazione viene scartata a scapito della seconda, a questo punto il venditore si trova senza pagamento e l'attaccante ha riottenuto i suoi fondi, concludendo così l'attacco.

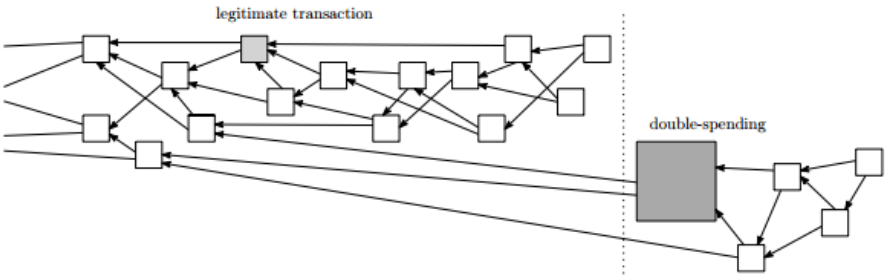

Figura 4.1: Esempio di attacco a doppia spesa[1]

4.1.1 Attacco al peso cumulativo

Tangle essendo un DLT può essere soggetto a questa tipologia di attacchi. In particolare, nel caso di conflitto tra due transazioni si potrebbe pensare di considerare come principale il subtangle con peso cumulativo maggiore. In questo caso l'attaccante ha due alternative per portare a termine la doppia spesa:

- emettere una grande quantità di piccole transazioni, che approvino la doppia spesa ma non quella al venditore, in modo da incrementare il peso cumulativo del subtangle disonesto.

- emettere una singola transazione molto pesante utilizzando tutto la capacità computazionale a disposizione, sempre con l'obiettivo di avere il peso maggiore possibile e facendo attenzione ad approvare transazioni precedenti al pagamento al venditore.

Entrambe le strategie puntano a massimizzare il peso cumulativo del proprio subtangle disonesto in modo che le nuove tips continuino ad approvarlo, mentre il ramo contenente la transazione originale, di peso minore, rimane orfano.

Per quanto riguarda la seconda tipologia di attacco, è stato dimostrato in [1] che questa avrà sempre successo. Infatti, definita $W^{(n)}$ come una variabile aleatoria esponenziale di parametro $\mu 3^{-n}$, con μ che rappresenta la capacità computazionale del nodo attaccante, si ottiene che il tempo necessario a creare una transazione di peso almeno pari 3^n è dato proprio da $W^{(n)}$. Come spiegato in precedenza, quando all'istante t_0 il peso cumulativo della transazione raggiunge una quota arbitraria w_0, allora il venditore ritiene il pagamento valido. Assumendo che ciò avvenga dopo il periodo di adattamento, il peso cumulativo crescerà con velocità λw, dove w è il

peso medio di una transazione. Di conseguenza il peso totale del subtangle onesto sarà pari a $w_1 = \lambda w t_0$. Perché l'attacco abbia successo è necessario che l'attaccante riesca a generare una singola transazione di peso 3^{n_0}, con $n_0 = \lceil \frac{lnw_1}{ln3} \rceil$, in modo da avere $3^{n_0} \geq w_1$, ovvero il peso della transazione confliggente deve superare quello dell'intero ramo di Tangle onesto. La probabilità che accada è data da

$$\mathbb{P}[W^{(n_0)} < t_0] = 1 - \exp\left(\frac{-t_0\mu}{3^{n_0}}\right) \approx 1 - \exp\left(\frac{-t_0\mu}{w_1}\right) \approx \frac{t_0\mu}{w_1} \tag{4.1}$$

Se l'attacco non dovesse avere successo al primo tentativo, per l'attaccante è sufficiente riprovare fino a quando non riesce ad emettere una transazione di peso 3^n, con $n > n_0$, che sia maggiore del peso cumulativo del subtangle onesto. La probabilità di questo evento è

$$\mathbb{P}[\lambda w W^{(n_0)} < 3^n] = 1 - \exp\left(-\mu 3^{-n_0}\left(\frac{3^{n_0}}{\lambda w}\right)\right) = 1 - \exp\left(\frac{-\mu}{\lambda w}\right) \approx \frac{\mu}{\lambda w} \tag{4.2}$$

Dato che questa probabilità è costante all'aumentare di n, questo attacco avrà successo asintoticamente, con tempo medio pari a $3^{\frac{\lambda w}{\mu}}$. Come appena evidenziato utilizzare il peso cumulativo come metrica di consenso comporta delle vulnerabilità. Una prima soluzione potrebbe essere quella di limitare superiormente il peso che una singola transazione può avere, in modo da neutralizzare in partenza una strategia di attacco di questo tipo.

Per analizzare uno scenario con peso limitato conviene assumere che questo sia uguale a 1 per ogni transazione. Come spiegato in precedenza, anche qui l'obiettivo dell'attaccante è quello di creare un subtangle disonesto che riesca a incentivare le nuove tips ad approvarlo a scapito del ramo contenente la transazione originale. L'unico mezzo a disposizione per fare ciò è quello di preparare la transazione doppia spesa in anticipo, per poi emettere una grande quantità di transazioni nel minore tempo possibile per incrementarne il peso cumulativo. Se questo ramo disonesto riesce a crescere più in fretta dell'altro, ovviamente deve ciò avvenire dopo che il venditore ha considerato il pagamento come valido, allora l'attacco andrà a buon fine.

Per l'analisi seguente si fa riferimento a quanto dimostrato in [1]. Definita la funzione $\phi(\alpha) = -\ln(\alpha) + \alpha - 1$, e sempre considerando μ come la capacità computazionale disponibile all'attaccante, si può calcolare la probabilità che la doppia spesa abbia peso cumulativo maggiore del ramo onesto all'istante t_0. Innanzitutto perché ciò accada l'attaccante deve riuscire ad emettere almeno w_0 transazioni. Infatti supponendo che la transazione originale abbia peso cumulativo w_0 all'istante t_0, e ricordando

che ognuna ha peso proprio limitato a 1, l'attaccante per avere successo dovrà crearne almeno w_0 prima di quell'istante. Indicato l'evento successo della doppia spesa con s al tempo t_s la probabilità che avvenga entro t_0 è data da

$$\mathbb{P}[t_s \leq t_0] \approx \exp\left(-w_0\phi\left(\frac{\mu t_0}{w_0}\right)\right) \tag{4.3}$$

È opportuno precisare che la probabilità è approssimata perché si sta trascurando il tempo di propagazione h_i delle tips. Per quanto riguarda il caso in cui l'attacco vada a buon fine in un momento successivo $t \geq t_0$ la probabilità di successo diventa

$$\mathbb{P}[t_s \geq t_0] \approx \exp\left(-\left(w_0 + \lambda(t-t_0)\right)\phi\left(\frac{\mu t}{w_0 + \lambda(t-t_0)}\right)\right) \tag{4.4}$$

dove il peso della transazione legittima al tempo $t \geq t_0$ sarà $w_0 + \lambda(t - t_0)$, perché si assume che il periodo di adattamento sia finito e la crescita sia lineare. È importante notare come nella fase di adattamento nonostante la crescita sia esponenziale, questa avvenga più lentamente che nella fase successiva di parametro λ. Di conseguenza si avrà che $\frac{\mu t_0}{w_0} \geq \frac{\mu}{\lambda}$, e quindi la probabilità che la doppia spesa abbia successo è

$$\mathbb{P}[s] \approx \exp\left(-w_0\phi\left(\max\left(\frac{\mu t_0}{w_0}, \frac{\mu}{\lambda}\right)\right)\right). \tag{4.5}$$

Grazie a (Equazione 4.5) è facile visualizzare come la sicurezza del sistema sia dipendente dai parametri μ e λ. Infatti è assolutamente necessario che la disuguaglianza $\lambda > \mu$ sia rispettata, altrimenti se l'attaccante avesse a disposizione una maggiore capacità computazionale rispetto ai nodi onesti tutta la struttura sarebbe compromessa.

Questa dipendenza appare ancora più evidente guardando il grafico in Figura 4.2 ottenuto da (Equazione 4.5) con i seguenti valori: $w_0 = 32$, $t_0 = 12$, $\lambda = 3$ mentre $\mu \in [0, 2.5]$ è variabile indipendente. Infatti per valori piccoli di $\mu(< 1)$ la probabilità di successo è nell'ordine di 10^{-6}, ma già se un'avversario avesse un capacità pari al 60% di quello dei nodi onesti l'attacco avrebbe esito positivo una volta su dieci.

L'ipotesi di maggioranza onesta e assidua

Per il corretto funzionamento di Tangle occorre fare una cosiddeta ipotesi di "maggioranza onesta e assidua"[2]: tutti i nodi onesti devono processare transazioni continuamente usando così tutto la capacità computazionale a

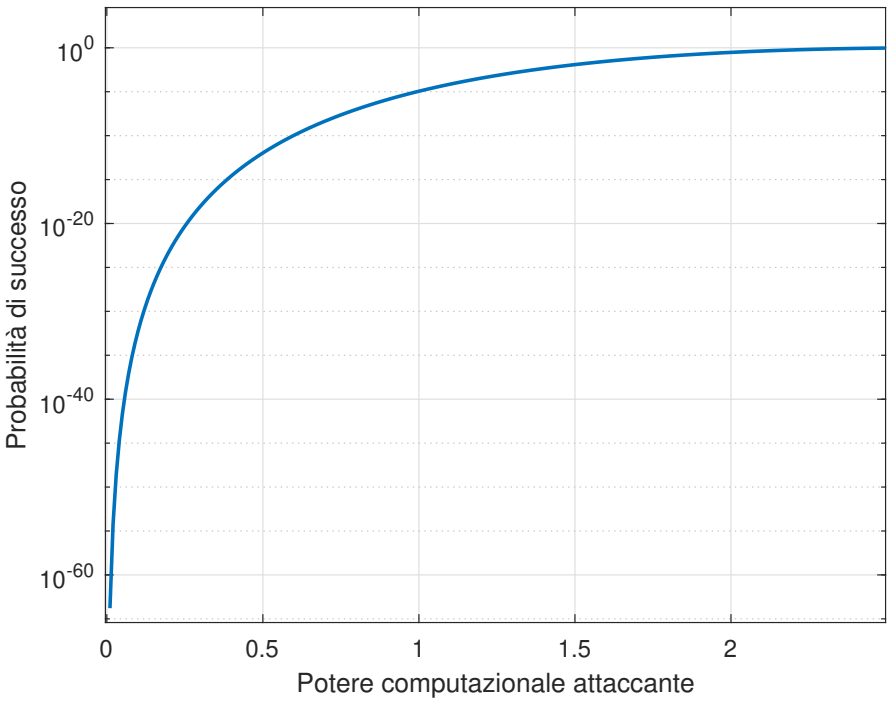

Figura 4.2: Come varia la probabilità di successo di un attacco a doppia spesa in funzione di μ(TFLOPS).

loro disponibile, che deve tuttavia essere strettamente maggiore di quello di un eventuale avversario. Infatti come è emerso anche nel paragrafo precedente, dove una condizione necessaria alla sicurezza del sistema era $\lambda > \mu$, se i nodi onesti non usassero costantemente la loro capacità computazionale, il tasso di crescita λ *effettivo* sarebbe inferiore alle aspettative e di conseguenza lo sarebbe anche il peso cumulativo, andando così a compromettere la sicurezza del sistema. Senza questa ipotesi un avversario sarebbe in grado di generare più transazioni rispetto ai nodi onesti pur avendo a disposizione un capacità computazionale inferiore. In particolare, come ampiamente dimostrato in [2], è importante notare come questa ipotesi sia necessaria per qualsiasi algoritmo di selezione delle tips.

4.1.2 Attacco tramite catena parassita

Per cercare di aumentare la probabilità di successo di un attacco a doppia spesa, un attaccante può pensare di effettuare un attacco costruendo una catena parassita. L'attacco consiste nella creazione in anticipo di una

catena di transazioni parallela al DAG principale, contenente la transazione disonesta. Successivamente l'attaccante attende che il venditore accetti il pagamento per poi emettere una grande quantità di transazioni che si attacchino alla tip disonesta. Avendo ora moltissime tips collegate al ramo parassita, aumentano le probabilità che questo cresca a scapito del ramo principale. Questa strategia permette di sfruttare a proprio vantaggio tutti gli algoritmi che selezionano le tips in base al loro numero e senza tenere conto del peso cumulativo.

4.2 Gli algoritmi di selezione delle tips

Quando un nodo emette una transazione, questa deve andare ad attaccarsi al Tangle approvando così due delle tips presenti. Per decidere quali tips selezionare tra tutte quelle disponibili, il nodo che ha emesso la transazione esegue un algoritmo di selezione. In questa sezione si analizzano alcuni tra i possibili algoritmi, con particolare attenzione alle vulnerabilità che essi possono generare e come eventualmente poterle superare.

Come dimostrato nella prima parte del capitolo, utilizzare il peso cumulativo come metrica di giudizio nel caso di conflitto di transazioni non è una strategia efficace, in quanto soggetta ad attacchi di diverso tipo. Un'idea alternativa è quella di affidarsi ad un algoritmo di selezione, definendo un *livello di confidenza*[3] di una transazione x, che equivale alla probabilità che l'algoritmo scelga una tip che vada ad approvare indirettamente x. Nel caso di conflitto tra due siti allora verrebbe considerato come legittimo quello con il livello di confidenza maggiore. Per concludere è importante sottolineare come gli algoritmi di selezione delle tips abbiano un ruolo chiave sia per quanto riguarda la crescita di Tangle ma allo stesso tempo regolino il consenso tra i vari nodi del registro distribuito.

4.2.1 Algoritmo di selezione casuale

Il primo algoritmo in analisi si chiama *Uniform Random Tip Selection*, abbreviato URTS, ed è il più semplice tra tutti: quando viene emessa una transazione si seleziona in modo casuale, seguendo una distribuzione di probabilità uniforme, una coppia di tips tra tutte quelle disponibili. Pur essendo casuali, le due tips selezionate non devono approvare direttamente o indirettamente transazioni in conflitto tra loro. Nel Tangle possono essere presenti transazioni in conflitto, tuttavia con il passare del tempo una delle due non verrà più approvata indirettamente dalle nuove tips e pertanto rimarrà orfana. Per il corretto funzionamento di questo meccani-

smo è necessario che una transazione in arrivo non approvi entrambi i siti confliggenti.

4.2.2 Catena di Markov Monte Carlo

Il secondo algoritmo di selezione è basato sulla catena di Markov Monte Carlo (MCMC). L'idea è quella di selezionare in modo casuale N siti appartenenti a Tangle, risalire il grafo effettuando una "camminata" verso le tips, e una volta raggiunte raggiunte le prime due esse verranno approvate dalla transazione in arrivo. Più nel dettaglio[1] l'algoritmo viene descritto così:

1. Si considerano tutti i siti appartenenti ad un intervallo arbitrario $[W, 2W]$. Non è necessario partire dalla genesi di Tangle altrimenti il tempo per risalire alla tip crescerebbe troppo, ma non conviene nemmeno iniziare troppo vicino alle tips.

2. Tra questi siti se ne selezionano N distinti che saranno il punto di partenza.

3. Si effettuano N camminate indipendenti a tempo discreto verso le tips. In particolare la transizione da un nodo x a y può avvenire se e solo se y approva direttamente x.

4. Le prime due tips raggiunte verranno selezionate per l'approvazione.

La probabilità di transizione da x a y è calcolata nella seguente maniera:

$$P_{xy} = \frac{\exp(-\alpha(H_x - H_y))}{\sum_{z:z\to x} \exp(-\alpha(H_x - H_z))} \tag{4.6}$$

dove H_x indica il peso cumulativo del nodo x, mentre $\alpha > 0$ è un importante parametro da stabilire che regola la probabilità di saltare alla transazione con peso cumulativo maggiore. La sommatoria invece è calcolata su tutte le transazioni z che approvano x.

L'utilizzo di un algoritmo MCMC inoltre scoraggia la creazione delle cosiddette "tips pigre": sono quelle tips che per risparmiarsi il lavoro di verifica vanno ad approvare due siti vecchi anziché due tips. Infatti avendo peso cumulativo molto inferiore alle tips normali, quelle pigre difficilmente verranno scelte dall'algoritmo, di conseguenza rimarranno orfane e nessun nodo sarà incentivato ad emettere transazioni di questo tipo.

L'utilizzo di N punti di partenza invece di soli due è legato a ragioni di sicurezza. Infatti se risalendo il grafo si finisse su una eventuale catena parassita, non ci sarebbe modo di tornare indietro nel ramo principale.

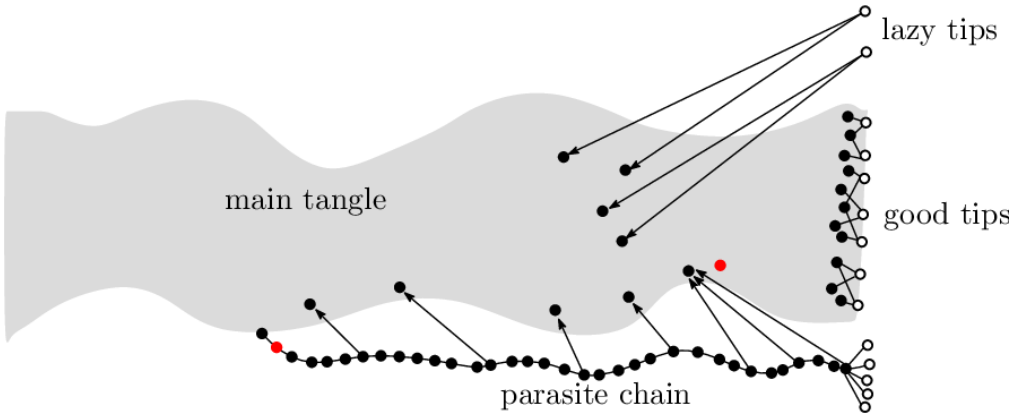

Figura 4.3: Esempio di attacco tramite catena parassita

Tuttavia essendo la catena per sua natura molto lunga, trascorrerà del tempo prima che venga raggiunta una tip disonesta, e durante questo intervallo è probabile che altre due tips siano già state selezionate. Al contrario, per evitare che vengano selezionate delle tips pigre, è opportuno scartare le tips raggiunte troppo velocemente dall'algoritmo di selezione. Come anticipato, il parametro α che compare in (Equazione 4.6) serve a regolare il bias dell'algoritmo MCMC. Infatti nel caso di $\alpha = 0$ la probabilità di saltare su un sito piuttosto che un altro diventa la stessa per tutti, e in questo caso l'algoritmo prende il nome di *Unbiased Random Walk* o *URW*. Invece per $\alpha \neq 0$ l'algoritmo viene chiamato *Biased Random Walk* o *BRW*. In questo caso man mano che il valore di $\alpha > 0$ aumenta la tendenza sarà quella di favorire sempre di più i vertici con peso cumulativo maggiore, fino a raggiungere un comportamento deterministico per $\alpha \to \infty$.

4.3 Confronto tra URTS e URW

Tra gli algoritmi presentati è utile concentrarsi su due in particolare, URTS e URW, in quanto godono di alcune proprietà importanti. Innanzitutto entrambi gli algoritmi non lasciano siti orfani, significa che tutte le transazioni verranno approvate almeno una volta. Inoltre l'URTS è meno computazionalmente oneroso rispetto agli algoritmi basati su catene di Markov. La semplicità di questi due algoritmi permette di ottenere scenari semplificati che sono utili per lo studio e la creazione di modelli matematici di Tangle.

Come in precedenza si indica con $N(t)$ il numero di tips in funzione del tempo t. Per quanto riguarda gli arrivi si assume sempre un processo di Poisson omogeneo di parametro λ. Ricordando che le proprietà di Tangle

sono in funzione di λm_h, e che m_h è espresso in unità di tempo, mentre λ in transazioni per unità di tempo, si può fissare $m_h = 1$ per semplicità, senza però perdere di generalità. Tutti i risultati ottenuti nei paragrafi successivi sono stati ottenuti in [3]. I parametri utilizzati nelle simulazioni sono i seguenti: $\lambda \in \{1, 10, 10^2, 10^3, 10^4\}$ per simulare un condizione di alto carico, mentre Tangle con un numero di siti nell'ordine di $10^5 - 10^7$. I dati sono stati ottenuti dopo una fase iniziale pari a $50m_h$ dalla genesi.

4.3.1 Peso cumulativo

Per quanto riguarda il peso cumulativo non ci particolari differenze tra i due algoritmi. Il comportamento è lo stesso presentato nella sezione 3.2, dove si distingue una fase iniziale a crescita esponenziale a cui segue una fase lineare.

4.3.2 Numero di tips

Per quanto riguardo il numero di tips $N(t)$, il comportamento è quello atteso: distribuzione stazionaria con media limitata. Dalle simulazioni sono state ricavate le due distribuzioni di probabilità, visibili nella figura 4.4, dove si può notare una differenza tra i diversi algoritmi. Infatti per quanto riguarda l'URTS il valore atteso di tips è proprio 2λ, uguale al valore ottenuto in precedenza (Equazione 3.2). Per la URW, si ricava un $N(t) = 2.1\lambda$ che è maggiore di un 5% rispetto all'altro. Questa differenza dimostra come Tangle cresca in modo differente a seconda dell'algoritmo di selezione utilizzato.

λ	URTS		URW	
	$\mathbb{E}(N)$	σ	$\mathbb{E}(N)$	σ
1	2.858	1.115	3.035	1.167
10	20.69	3.140	21.80	3.278
100	200.9	10.21	208.1	10.27
1000	1999	28.57	2079	34.36
10000	19983	86.28	20722	90.41

Tabella 4.1: Numero medio di tips e relativa deviazione standard σ in funzione di λ

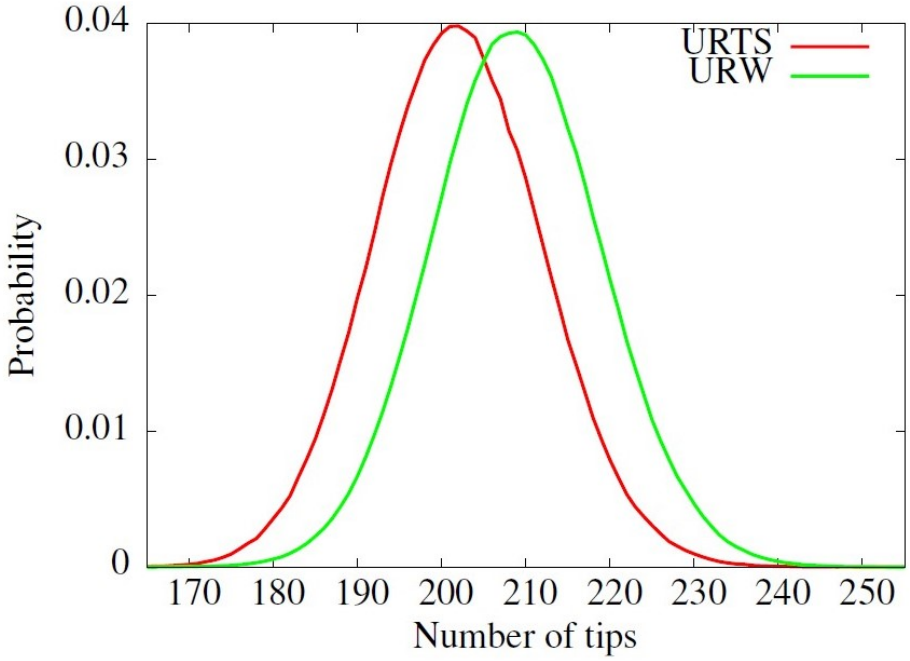

Figura 4.4: Funzione di distribuzione di probabilità del numero di tips $N(t)$ con $\lambda = 100$ per URTS (in rosso) e URW (in verde).

4.3.3 Tempo di conferma di una transazione

Il tempo che trascorre dall'emissione di una transazione fino alla sua prima approvazione, ovvero il tempo di conferma, dipende dal numero di tips presenti nel sistema.

Nel caso dell'URTS, dato λ tasso di arrivo e ricordando che ogni transazione approva 2 tips, dal momento che la selezione è casuale una determinata tips verrà approvata con probabilità pari a $1/N(t)$. Di conseguenza il tasso di approvazione R sarà uguale a

$$R = \frac{2\lambda}{N(t)} \tag{4.7}$$

Ora il tempo di approvazione t_A è l'inverso del tasso R, quindi si ha $t_A = 1/R$ e sostituendo in (Equazione 4.7) si ottiene

$$N(t_A) = t_A 2\lambda \tag{4.8}$$

In conclusione, sapendo che $N(t) = 2\lambda$ e che le transazioni prima di diventare visibili al resto del network necessitano di un tempo pari a $1h$, si ha che $t_A = 2$.

Per quanto riguarda l'URW invece, non è possibile applicare un ragionamento simile perchè la probabilità che una tip venga selezionata non è uniforme. Tuttavia i risultati ottenuti dalle simulazioni (Tabella 4.2) mostrano che i valori per il tempo di conferma sono simili per i due algoritmi, più precisamente: l'URTS garantisce un tempo minore e allo stesso tempo la sua deviazione standard σ è inferiore.

λ	URTS		URW	
	t_A	σ	t_A	σ
1	2.884	1.951	3.042	2.464
10	2.075	1.082	2.177	1.482
100	2.009	1.011	2.087	1.349
1000	1.997	0.997	2.078	1.345

Tabella 4.2: Tempo medio di attesa fino alla prima approvazione e relativa deviazione standard σ in funzione di λ

4.3.4 Sicurezza da attacchi tramite catena parassita

In questo paragrafo verrà discusso come entrambi gli algoritmi, URTS e URW, non possano essere utilizzati in un'applicazione reale di Tangle

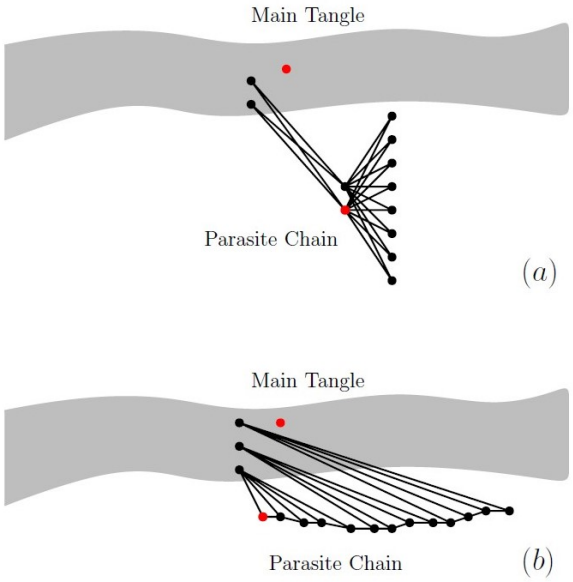

Figura 4.5: Due differenti topologie di catene parassite: la prima efficace contro l'URTS mentre la seconda contro l'URW [3].

in quanto vulnerabili ad attacchi tramite catena parassita. Il loro scopo quindi rimane quello di strumenti per studiare il comportamento del sistema.

Nel caso dell'URTS, dato che l'algoritmo seleziona casualmente le tips, l'obiettivo dell'attaccante è quello di massimizzare il proprio numero di tips. Infatti dopo aver creato segretamente la transazione a doppia spesa (il pallino rosso in Figura 4.5(a)), l'attaccante emetterà esclusivamente nuove transazioni che la approvino direttamente. Dato che il numero di tips nel ramo onesto è stabile intorno al valore 2λ, mentre nella catena parassita questo limite non c'è dato che l'attaccante può aggiungere tutte le transazioni che vuole, gli basterà attendere che il numero di tips "disoneste" superi di una quantità sufficientemente grande quelle del ramo principale. A quel punto l'attaccante sarà libero di rendere pubblica alle altre entità la catena parassita, che avendo più tip a disposizione verrà selezionata con probabilità maggiore, portando così all'approvazione la doppia spesa.

L'URW invece, offre protezione contro questo tipo di catena parassita, perchè è indifferente al numero di tips presenti. Tuttavia può essere attaccata utilizzando una catena come quella presente nella Figura 4.5(b). In questo caso l'attaccante seleziona un insieme Y di siti tali che qualsiasi cammino dalla genesi alle tips passi necessariamente per un elemento di Y.

Poi in segreto emette la transazione doppia spesa x_0 e di seguito crea una sequenza di transazioni tali che ogni x_i approvi direttamente sia x_{i-1} che un elemento di Y. Ora supponendo che le transazioni vengano accettate solo dopo che il loro livello di confidenza ha superato il valore $k \in (0, 1)$. L'attaccante continua a costruire la catena fino a che per ogni $y \in Y$, la proporzione di siti appartenenti alla catena che approvano y è maggiore di k. A questo punto effettua il pagamento al venditore, e sarà proprio questa transazione a creare il conflitto con x_0, che tuttavia non è ancora visibile al resto della rete. Solo una volta accettata la transazione dal venditore, l'attaccante pubblicherà la catena. A questo punto la probabilità che la camminata dell'URW finisca sulla catena è maggiore di k, inoltre una volta che ci finisce dentro non è possibile uscirne. Per questo motivo il livello di confidenza di x_0 crescerà sempre di più concludendo con successo l'attacco.

Come anticipato, nessuno degli algoritmi è in grado di difendersi da un attacco di questo tipo. Una soluzione è quella di utilizzare una Biased Random Walk, in quanto questa tipologia di camminata è meno influenzata da piccole modifiche alla topologia del grafo. Inoltre portare a termine con successo un attacco sarebbe molto più costoso, dato che sarebbe necessario creare siti il più pesanti possibile per cercare di sbilanciare il grafo a favore della catena parassita.

4.4 Soluzione al problema dei siti orfani

Uno dei principali problemi con gli algoritmi basati su MCMC è l'individuazione del parametro α ottimale che garantisca allo stesso tempo sia la sicurezza del sistema che un throughput adeguato. Innanzitutto bisogna ricordare che il ruolo di α, il parametro che compare in (Equazione 4.6), è quello di regolare il bias dell'algoritmo MCMC. Intuitivamente un valore di α elevato implica che il sistema è "freddo", e quindi le camminate avvengono lungo una minoranza di cammini pesanti, al contrario un valore piccolo implica un sistema "caldo" che si comporta in modo più caotico.

Più in dettaglio una scelta di α grande aumenta la sicurezza del sistema: in questo modo solo le tips più affidabili vengono selezionate, perché appartenenti a cammini con peso cumulativo maggiore. Tuttavia scegliendo un valore elevato aumenta anche il numero di tips oneste che non riescono a venire approvate, infatti queste tips non avendo un gran peso faticano ad attirare l'algoritmo di selezione, che al contrario preferirà percorrere i soliti cammini più pesanti e finirà con l'approvare esclusivamente le tips al termine di questi percorsi. Si crea così un feedback positivo dove questi percorsi accumulano sempre più peso cumulativo e attirano in questo

modo tutte le nuove tips. Di conseguenza però il peso di tutte le altre transazioni non appartenenti a questi cammini non riesce più a crescere e vengono di conseguenza considerate orfane. Al contrario, scegliere un valore di α troppo piccolo, come ad esempio pari a 0 nel caso della URW, per quanto possa aumentare il numero di tips oneste che vengono approvate, renderebbe il sistema più vulnerabile ad attacchi a doppia spesa.

Al momento l'unico modo per trovare un valore di α appropriato è quello di procedere tramite simulazioni. In particolare essendo Tangle un processo stocastico sono state effettuate 50 realizzazioni per ottenere dei risultati significativi. Dallo studio svolto in [4], con parametri $\lambda = 30$, $m_h = 5$, sono emersi due comportamenti differenti: per $\alpha = 0.1$ il numero di tips diverge nel tempo, invece per $\alpha = 0.001$ il valore $N(t)$ ha una distribuzione stazionaria con media finita.

Una possibile soluzione a questo compromesso, presentata sempre in [4], consiste nella realizzazione di un cosiddetto algoritmo di selezione ibrido. Per superare i problemi di sicurezza e allo stesso tempo non lasciare siti orfani l'algoritmo è diviso in due passi:

- *Security Step*: Il primo insieme di tips viene selezionato tramite BRW con un valore di α elevato per garantire che vengano scelte le tips oneste.

- *Swipe Step*: Nel secondo step viene eseguito un algoritmo tra URTS o URW in modo da andare a selezionare quelle tips che sono state escluse nel primo step, per evitare di lasciarne qualcuna indietro creando così dei siti orfani.

È stato dimostrato[4] che questo algoritmo ibrido assicura che tutte le transazioni vengano approvate in un tempo finito, che il numero di tips $N(t)$ è limitato e che relativamente alla sicurezza mantiene le caratteristiche degli algoritmi di selezione basati su MCMC.

Capitolo 5

Conclusioni

Tangle si presenta come un'alternativa a blockchain per l'implementazione di un DLT. Sfruttando un grafo aciclico diretto riesce a elaborare più transazioni in parallelo, inoltre l'innovativo meccanismo di attaccamento al grafo permette di superare la differenza nei ruoli dei partecipanti alla rete. Nel secondo capitolo è stata presentata la struttura di Tangle e descritto il suo funzionamento, sono state introdotte alcune nozioni come il peso e peso cumulativo. In seguito, per quanto riguarda la stabilità del sistema, l'analisi si è concentrata su due aspetti: l'equilibrio delle tips e la crescita del peso cumulativo. Per il numero di tips $N(t)$ è risultata una distribuzione stazionaria con media finita, il cui valore è dato da (Equazione 3.2), mentre dallo studio sull'evoluzione del peso cumulativo sono state distinte due fasi: un periodo di adattamento a crescita esponenziale a cui segue una fase di crescita lineare secondo il parametro λ.

Nel capitolo successivo invece l'analisi si è concentrata sulle vulnerabilità di Tangle rispetto agli attacchi a doppia spesa ed eventuali contromisure da adottare che includono l'utilizzo di determinati algoritmi di selezione delle tips. Per gli attacchi sono emerse criticità riguardo all'utilizzo del peso cumulativo come metrica per stabilire l'affidabilità di una transazione nel caso di conflitti. Nemmeno limitare superiormente il peso di una singola transazione si è rivelata una strategia sufficiente a prevenire questa tipologia di attacco. L'unica soluzione efficace e anche imprescindibile per il corretto funzionamento del sistema è "l'ipotesi di maggioranza onesta e assidua", secondo la quale è richiesto che i nodi onesti abbiano a disposizione un potere computazionale maggiore di un nodo attaccante e soprattutto che esso venga impiegato incessantemente per l'aggiunta di nuovi siti. In seguito sono stati presentati due particolare algoritmi di selezione delle tips: lo Uniform Random Tip Selection, o selezione casuale, e la Unbiased Random Walk. Sono stati confrontati sotto vari aspetti ed è risultato dalle

simulazioni che il comportamento di entrambi rispecchia l'analisi teorica riguardante la stabilità e il peso cumulativo; tuttavia utilizzando il secondo di essi si è notato un $N(t)$ e un tempo di approvazione maggiori del 5% rispetto alla selezione casuale. Per concludere il confronto si è studiato il comportamento di questi due algoritmi rispetto ad attacchi tramite catena parassita, e nessuno dei due è stato in grado di offrire sicurezza da minacce di questo tipo, rendendoli di fatto inutilizzabili in implementazioni reali. Nell'ultimo paragrafo invece è stato introdotto un terzo di tipo di algoritmo di selezione ibrido in grado di fornire al contempo sicurezza e in grado anche di superare il problema dei siti orfani.

Bibliografia

[1] S. Popov, *The Tangle*. April 2018. Version 1.4.3.

[2] Q. Bramas, "The stability and the security of the tangle," 2018. hal-01716111v2.

[3] B. Kuśmierz, W. Sanders, A. Penzkofer, A. Capossele, and A. Gal, "Properties of the tangle for uniform random and random walk tip selection," 2020.

[4] P. Ferraro, C. King, and R. Shorten, "On the stability of unverified transaction in a dag-based distributed ledger," 2020.

[5] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system." Online available http://bitcoin.org/bitcoin.pdf, 2008.

[6] N. Benvenuto and M. Zorzi, *Principles of Communication Networks and Systems*. John Wiley and Sons Ltd, 2011.

[7] L. Finesso, *Lezioni di probabilità*. Edizioni Libreria Progetto Padova, 2018.

Elenco delle tabelle

Elenco delle figure