Alma Mater Studiorum
Università di Bologna
Campus di Cesena

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA
Corso di Laurea in Ingegneria e Scienze Informatiche

BLOCKCHAIN ED ETHEREUM

Elaborato in:
CRITTOGRAFIA

Relatore:
Prof.
Margara Luciano

Presentata da:
Paganelli Alberto

Sessione II
Anno Accademico 2021-2022

Ringrazio
i miei amici,
la mia famiglia,
ma soprattutto i miei compagni di corso

Indice

Elenco delle figure

Capitolo 1

Introduzione

La crittografia ha da sempre avuto un ruolo fondamentale nella vita di tutti noi. La richiesta di canali di comunicazioni sicuri è sempre in crescita, soprattutto in questo periodo storico dove i dati hanno un vero e proprio valore economico. Con il passare del tempo si sono sviluppate metodologie sempre più avanzate e tra queste troviamo l'insieme di tecnologie che chiamiamo comunemente *blockchain*.
Lo scopo di questa tesi è introdurre la tecnologia blockchain parlando del suo avvento e di come è possibile usufruirne oggi. Saranno presenti considerazioni sia sull'aspetto di decentralizzazione della blockchain sia sulle diverse metodologie di lavoro e verranno definiti i componenti fondamentali.
Si vedranno le principali differenze tra vari tipi di blockchain e alcune possibili nuove applicazioni introdotte da Ethereum, diverso da Bitcoin al quale attribuiamo la nascita e l'esclusivo utilizzo in ambito monetario.

Sarà quindi presente un approfondimento sulla blockchain Ethereum considerando non solo la sua potenza come valuta ma anche tutte le possibilità offerte agli utenti in termini di servizi. Non verranno tralasciati gli aspetti di crittografia presenti in blockchain come i vari algoritmi utilizzati al suo interno, le curve ellittiche sulla quale si basa e la metodologia di lavoro.

Verrà analizzato come una transazione avviene e descritte le nuove applicazioni come Smart Contract, Token fungibili, DAO e ENS: il DNS decentralizzato basato su blockchain Ethereum.

Inoltre sarà presente un capitolo nel quale si introdurrà l'interoperabilità tra le diverse blockchain ed infine qualche cenno su possibili attacchi rivolti sia all'infrastruttura che all'utente finale.

Capitolo 2

Introduzione Blockchain

La blockchain trova la sua miglior presentazione in un registro condiviso ed immutabile composto da un insieme di tecnologie avanzate che permettono e facilitano la tracciabilità e la trasparenza delle transazioni registrate su di esso. Troviamo diversi componenti fondamentali in una blockchain ognuno dei quali fondamentale per il corretto funzionamento e mantenimento della chain.

La nascita della tecnologia blockchain viene attribuita a Satoshi Nakamoto creatore della criptovaluta Bitcoin ma sulla verità vi sono ancora molte controversie e aspetti poco chiari. L'idea di valuta digitale non è relativamente nuova ma solo con BTC è stata implementata con successo. Nel suo primo stadio l'idea era il trasferimento di denaro attraverso una connessione peer-to-peer ed un uso efficace della crittografia. [22]
Come possiamo dedurre dal nome, la blockchain è un insieme di blocchi concatenati tra essi che godono della proprietà fondamentale di *immutabilità*. Con questo termine stiamo ad indicare che una catena di blocchi, una volta creata, non è più modificabile. Tutto ciò è garantito dal fatto che ogni blocco presente in chain è cifrato facendo uso della cifratura del blocco precedente. Verrà analizzato successivamente, attraverso una spiegazione basata sulla crittografia, come questa metodologia garantisce immutabilità.

Non solo immutabilità ma anche decentralizzazione, trasparenza, sicurezza, consenso, responsabilità e programmabilità sono garantite dalla tecnologia blockchain e nel corso di questo documento verranno approfondite le metodologie attraverso le quali si riescono a garantire tali caratteristiche.

Per poter inizialmente semplificare la tecnologia si possono suddividere le aree fondamentali della blockchain in base alle tecnologie utilizzate sulla

stessa. Questa classificazione consente di fare un'idea propria di blockchain e valutare, secondo i parametri più importanti, quale blockchain è preferibile rispetto ad altre.

Come anticipato la blockchain funziona mediante un meccanismo di consenso condiviso, nodi o partecipanti alla chain e delle fortissime fondamenta di crittografia per garantire trasparenza, integrità dei dati e tracciabilità.

Gli elementi che verranno approfonditi successivamente sono libro mastro o Ledger nel quale sono mantenute tutte le informazioni, il meccanismo di consenso condiviso al quale tutti i partecipanti in rete devono sottostare e i vari tipi di nodi con differenti ruoli.

2.1 Cosa si intende con "decentralizzazione"

Il termite decentralizzazione viene utilizzato molto spesso quando si parla di blockchain. E' una delle prime tecnologie che riesce a garantire fiducia, responsabilità e trasparenza attraverso un meccanismo condiviso e verificato da tutti i partecipanti della rete. La caratteristica fondamentale introdotta dalla blockchain è il fatto che quest'ultima garantisce il corretto funzionamento autoregolandosi da sè senza la necessità di enti terzi.

Con decentralizzazione si intende quel fenomeno per il quale in una blockchain, una qualsiasi transazione o informazione che deve essere salvata, rispetta il meccanismo di consenso condiviso. E' facilmente intuibile il vantaggio della decentralizzazione quando per poter inserire informazioni in blockchain si deve sottostare a un meccanismo verificato da tutti i partecipanti in rete. E' sicuramente una caratteristica che porterebbe grandi vantaggi se applicata a dovere e sarà un punto fondamentale trattato nel capitolo inerente alle diverse tipologie di blockchain: **permissioned** o **permissionless**. La differenza tra le due sta proprio nel come viene garantito, applicato e verificato questo meccanismo di consenso finalizzato a mantenere alto il livello di decentralizzazione.

Non avere un ente centrale spesso viene considerato positivamente ma a differenza di ciò che si potrebbe pensare in un primo momento la decentralizzazione potrebbe scaturire diverse problematiche. Come anche "Remy Prud'homme" sottolinea ne The dangers of decentralization [24] la decentralizzazione sotto determinati aspetti favorisce la corruzione ed è una metodolo-

gia che non sempre porta benefici. Sin dall'introduzione, Remy Prud'homme, fornisce una descrizione attraverso una metafora "Decentralization measures are like some potent drugs, however: when prescribed for the relevant illness, at the appropriate moment and in the correct dose, they can have the desired salutary effect; but in the wrong circumstances, they can harm rather than heal.".

Capitolo 3

Componenti principali in blockchain

3.1 Ledger

Per poter introdurre al meglio il concetto di blockchain bisogna comprendere il rapporto tra la stessa e il *Distributed Ledger*. Con Distributed Ledger o DL è inteso quel registro condiviso nel quale sono presenti tutte le operazioni verificatasi in blockchain. La blockchain possiamo identificarla come una *DLT*, ovvero una Distributed Ledger Technology che fa riferimento ad un registro distribuito (DL) gestito in maniera tale da rendere la possibilità di modifica e l'accesso solo ad alcuni partecipanti (nodi).

Il Ledger è come un vero e proprio libro mastro nel quale sono annotate tutte le operazioni effettuate, verificate e inserite in blockchain. Ogni transazione e i dati che la rappresentano, intesi non solo come transazioni o dati monetari, vengono crittografati e inseriti nel DL attraverso un meccanismo di firma a doppia chiave simmetrica senza la necessità di enti certificatori centralizzati (necessità di un meccanismo di consenso condiviso dai nodi).

Come si vedrà successivamente le distributed tecnology utilizzano algoritmi crittografici e abilitano l'utente all'utilizzo della tecnologia fornendogli una coppia di chiavi generate attraverso le quali potrà sottoscrivere Smart Contract **Ethereum** o verificare transazioni altrui.

Quindi, per come è strutturata, si può intendere la blockchain come un sistema decentralizzato che, attraverso un meccanismo condiviso, mantiene il libro mastro distribuito. La differenza con altre DLT è il fatto che nella blockchain le informazioni sono contenute in blocchi che rispettano le regole stabilite e si "concatenano" all'ultimo blocco inserito.

Chi ha il compito di verificare transazioni, inserirle e mantenere la chain sono i nodi.

3.2 Meccanismo di consenso condiviso

Una transazione presente in blockchain è da considerarsi verificata e quindi approvata dai nodi solo se rispetta il meccanismo di consenso condiviso. L'approvazione della transazione, ovvero quel meccanismo attraverso il quale si riesce a raggiungere il consenso, è un compito distribuito ai nodi.

Il meccanismo di consenso condiviso comprende un *algoritmo di consenso* il quale è definibile come lo strumento necessario per trovare un accordo sulla validità di una data operazione all'interno di un sistema distribuito. Gli algoritmi di consenso hanno il ruolo fondamentale di assicurare affidabilità in sistemi che coinvolgono nodi multipli oltre a rendere attuabile la caratteristica di decentralizzazione tipica della blockchain.

Quando un nuovo insieme di transazioni, impacchettato in un blocco, viene trasmesso all'intera rete da uno dei nodi, ogni altro nodo deve verificarne la validità. Il consenso della rete si ottiene quando quel blocco viene aggiunto a tutte le copie della blockchain ed è proprio in quel momento che tutte le transazioni presenti iniziano a godere della proprietà di immutabilità rendendole così impossibili da modificare o contraffare.

3.2.1 Il problema dei Generali Bizantini

Il problema dei generali bizantini è uno dei maggiori scogli da affrontare quando si introducono i sistemi distribuiti affinché questi ultimi possano operare in modo sicuro ed efficiente. Per ogni sistema distribuito è essenziale garantire una tolleranza ai guasti bizantini. Con tolleranza ai guasti bizantini (BFT: *Byzantine Fault Tolerance*) intendiamo la capacità del sistema a riadattarsi nel momento in cui si presenta un errore provocato dal problema dei generali bizantini illustrato qui sotto.

Le problematiche sviluppate si notano maggiormente nelle reti distribuite nelle quali spesso il guasto o l'alterazione voluta di un componente del sistema genera dei conflitti sulle informazioni presenti creando una situazione di inconsistenza dei dati che è proprio ciò che si vuole evitare in un sistema distribuito.

L'unico modo per poter far fronte all'inconsistenza dei dati è raggiungere un consenso condiviso. Questo problema è stato introdotto da Lamport, Shostak, e Pease nell'opera *"The Byzantine's Generals' Problem"* nel 1982 nella quale gli autori spiegarono la problematica attraverso una metafora molto semplice ma non di semplice risoluzione. [17]
Si supponga che un gruppo di generali bizantini debba decidere se attaccare o meno una città nemica. I generali possono comunicare tra loro solo tramite dei messaggeri e qualunque decisione venga presa vi è la necessità che tutti ne siano a conoscenza. Questo è il problema di base aggravato dal fatto che alcuni generali possono essere traditori, quindi che cercano di impedire l'eventuale attacco insieme ad alcuni messaggeri che potrebbero essere inaffidabili.
Sorge la necessità di soddisfare i seguenti requisiti:

1. tutti i generali fedeli devono concordare uno stesso piano di azione.

2. un numero ristretto di traditori non può impedire ai generali fedeli di mettere in azione il proprio piano.

Nonostante il problema dei generali bizantini possa sembrare di facile soluzione, un semplice calcolo algebrico dimostra che occorrono almeno $\mathbf{3 \cdot n + 1}$ generali fedeli per contrastare l'azione prodotta da $\mathbf{n}$ traditori.

In altre parole, il problema può essere risolto soltanto se almeno due terzi dei generali sono fedeli, un singolo traditore può da solo sabotare l'operato di due generali fedeli. Il problema dei generali bizantini rientra nella classe più generale di problemi in cui occorre trovare un accordo o soluzione comune in un sistema distribuito. Si tratta in altre parole di un problema di consenso, il quale richiede un algoritmo di consenso per la sua risoluzione. L'obiettivo degli algoritmi di consenso è quello di assicurare l'affidabilità del sistema trovando un accordo sulla validità dei dati tra un certo numero di soggetti coinvolti o processi. Alcuni di questi soggetti potrebbero agire secondo una modalità dovuta a guasti, malfunzionamenti o persino a malintenzionati. L'algoritmo di consenso deve quindi essere tollerante sopportando tali evenienze: la consistenza dei dati deve essere anteposta a qualunque altro tipo di considerazione. Si ritornerà sul problema del consenso nel capitolo relativo alla sicurezza dell'infrastruttura blockchain, specificatamente quando si introdurrà il "DDoS Attack", Distributed Denial of Service per un singolo nodo oppure quando verrà introdotto in "Majority Attack".

3.3 Nodi

I nodi sono tutti quei dispositivi che fanno parte, supportano e mantengono la blockchain. In quanto la blockchain è una *DLT*, le problematiche legate al mantenimento, alla sicurezza e alla trasparenza del registro pubblico delle transazioni sono compiti dei nodi.

Vi sono vari tipi di nodi: *Light nodes* o regolari, *full nodes* e *masternodes*, ciascuno dei quali con ruoli e livelli di responsabilità diversi. I tipi più potenti di nodi, i masternodes, sono stati introdotti per la prima volta nella rete Dash nel marzo 2014 e da allora sono stati adottati da un'ampia varietà di altre reti blockchain. [9]

Per poter comprendere a pieno le differenze tra i vari tipi di nodi bisogna tenere presente che le singole reti blockchain potrebbero differire tra di loro e perciò tutti i tipi di nodi potrebbero presentare ruoli e responsabilità leggermente diversi. La caratteristica comune tra tutti i nodi di tutte le blockchain rimane l'esecuzione del software al loro interno che viene indicato come protocollo e che determina regole, funzionalità e dettagli di una particolare blockchain.

3.3.1 Light nodes

Un light node detto anche regolare è la tipologia di nodo più elementare. Comprende tutti i dispositivi informatici che supportano la rete blockchain. In senso molto generale, una blockchain è l'insieme di tutti i nodi che la supportano. Nei sistemi *Proof of Work* (PoW) i minatori sono i nodi, nei sistemi *Proof of Stake* (PoS) i portafogli di staking sono i nodi. PoW e PoS sono meccanismi di consenso che consentono ai nodi di mettersi d'accordo sulla validità delle transazioni e sullo stato dell'intera blockchain come si vedrà nel capitolo a loro dedicato.

La vera differenza con i *full nodes* sta proprio nella quantità di dati e informazioni della blockchain che il nodo scarica dalla rete per poter elaborare e verificare nuove transazioni. Il loro carico è minimo e questo serve a non rallentare l'esecuzione analizzando quantità di dati esagerate ed inutili.

3.3.2 Full nodes

Un *full node* elabora e verifica transazioni proprio come un *light node* con la differenza che i dati presenti in un full node sono una vera e propria copia del Ledger distribuito, in tempo reale. Un full node continua a scaricare le transazioni avvenute su una blockchain e le verifica completamente con altri nodi.

Una caratteristica fondamentale della decentralizzazione e del mantenimento dei full nodes è che, nell'estremo caso di una catastrofe nella quale la maggioranza dei nodi viene distrutta, **attraverso un singolo full node si potrebbe ripristinare l'intera rete blockchain**. Inoltre, con una copia del registro condiviso tra tutti i nodi sparsi nel mondo, viene garantito che la rete rimanga sempre attiva a meno che tutti i nodi non siano distrutti.

3.3.3 Masternodes

I *masternodes* non validano e inseriscono transazioni in blockchain ma ne verificano la correttezza. La differenza sostanziale sta nel fatto che i masternodes hanno anche un ruolo di gestione, government e regolamentazione a seconda del protocollo blockchain al quale aderiscono. Ad esempio, i masternodes possono governare gli eventi di voto ed eseguire operazioni di protocollo.

I masternodes operano su un sistema basato su garanzie in modo molto simile ad un Proof of Stake e solitamente si tratta di server con elevate prestazioni i quali devono essere fruibili in ogni momento.

Chi opera con i masternodes deve possedere una quantità significativa della criptovaluta alla base della blockchain stessa e a loro volta, agli operatori, vengono garantiti guadagni annuali per i loro servizi. Attraverso questa metodologia la rete garantisce che i nodi più importanti siano incentivati attraverso il denaro, rafforzando l'affidabilità della rete stessa. La logica è simile a PoS e i masternodes infatti sono anche noti come *sistemi di validazione vincolati.*

I vantaggi dell'utilizzo dei Masternodes

Indipendentemente dal protocollo e struttura della rete, quindi che sia PoW, PoS o che utilizzi una metodologia di lavoro diversa, tutte le reti comprendono un modo di premiare in denaro i nodi che la sostengono.

Chiaramente i masternodes ricevono maggior parte della ricompensa essendo nodi con maggiore responsabilità. Inoltre per poter elaborare e generare nuovi blocchi nei sistemi PoS il livello di elaborazione richiesto è molto inferiore rispetto ai sistemi PoW dove viene utilizzata la tecnica del mining.

Questo facilmente porta alla conclusione che in teoria, a parità di potenza computazionale, governare un masternode in ambiente PoS generi più guadagni con **costi inferiori** rispetto che in un ambiente PoW.

Difficoltà di utilizzo dei Masternodes

Il mantenimento di un masternode è un ottimo mezzo per generare rendite passive ma presenta diversi ostacoli da superare per poterne ottenere uno.

I masternodes richiedono molte più risorse di un light node o full node ed il funzionamento di un masternode richiede conoscenze, capitale e tempo. E' necessario anche fornire delle garanzie alla rete attraverso la moneta nativa della rete blockchain, rispettare le funzioni assegnate qualunque esse siano e mantenere tutto il sistema acceso facendosi carico delle spese come costi energetici e dell'infrastruttura stessa.

Capitolo 4

Tipologie di Blockchain

Tutti i principali protocolli blockchain sottostanno ad un meccanismo di consenso condiviso. Essendo sempre presente il problema dei generali bizantini analizzato nel capitolo precedente è di facile comprensione il disservizio portato dalla blockchain nel caso in cui la maggior parte dei nodi che compongono una blockchain siano malevoli.

La differenza sostanziale tra blockchain *permissionless* e *permissioned* è proprio la scelta e la partecipazione dei nodi che la compongono: mentre nella tipologia permissionless ogni nodo può fungere da validatore nella permissioned i nodi vengono accuratamente scelti. [21]

4.1 Permissionless

La maggior parte delle permissionless, non avendo la possibilità di scegliere direttamente i nodi, tentano di attirare come nodi validatori quelli che, guadagnando attraverso la validazione delle transazioni, mirano a incrementare l'utilizzo e l'espansione della blockchain. I cosiddetti **self-select** nodes sono coloro che si sono elevati da soli a nodi validatori della blockchain permissionless. Per poter definirsi un self-select node si necessita di molta potenza computazionale nel caso di Proof of Work oppure di una grande quantità di criptovaluta nel caso delle Proof of Stake ed in ognuno dei due casi il nodo deve effettuare una spesa iniziale considerevole per hardware come CPUs (PoW) o di criptovalute (PoS). Questo metodo viene utilizzato anche per incentivare i nodi stessi a continuare ad alimentare e supportare la chain.

4.2 Permissioned

Nel caso invece delle blockchain permissioned la scelta dei nodi partecipanti è molto differente. Mentre nella permissionless tutti i partecipanti possono esserlo nella permissioned si fa affidamento ad input di un qualche processo di selezione. Solitamente chi ha diritto di scelta sono le istituzioni, i governi o qualche gruppo industriale.

Questa tipologia di blockchain limita considerevolmente le potenzialità del DL (distributed ledger). Mentre nelle permissionless tutti possono consultare il Ledger distribuito, nelle permissioned questa caratteristica può essere modificata rendendo il Ledger accessibile solo da eventuali partecipanti.
Di facile comprensione come questa tipologia sia la più stretta e legata alle istituzioni e di come questa generi altre problematiche legate alla sicurezza. Potrebbero generarsi falle dovute alla manomissione di qualche specifico nodo piuttosto che un nodo casuale di rete, aumentare le probabilità di corruzione ed anche generare possibilità di agire in maniera poco trasparente da parte di alcuni nodi.

Capitolo 5

Metodologie di lavoro

Fino ad ora si è parlato di **Proof of Work** e **Proof of Stake** ma senza porre abbastanza attenzione sul significato e sulle conseguenze che porta l'implementazione di una metodologia piuttosto dell'altra. [23] L'algoritmo di consenso è un meccanismo che consente di proteggere la rete dagli attacchi. Il lavoro dell'algoritmo consiste nel fornire regole che agiscano sui membri della rete.
Proof of Work è una delle metodologie di lavoro utilizzabili dalle blockchain, il principio cardine è basare la prova su un calcolo oneroso a livello di CPU sfruttando la difficile risoluzione di un problema algoritmico complesso. Questo algoritmo richiede che l'hardware sul quale viene eseguito sia prestante e che abbia una potenza di calcolo sufficiente per trovare una soluzione in un periodo di tempo accettabile oltre che per mantenere le sue caratteristiche di sicurezza.
Un'altra possibilità la troviamo nel *Proof of Stake*: quest'ultimo non richiede così tante risorse per mantenere le prestazioni della rete, ma presenta una serie di carenze e limitazioni che verranno approfondite durante il capitolo.

5.1 Proof of Work

L'algoritmo Proof of Work fornisce la sicurezza della rete sotto forma di *block mining*. Il punto principale di PoW è che ogni nodo che vuole partecipare al mining deve risolvere il problema computazionalmente difficile per garantire la validità del blocco appena estratto.
Per ogni nuovo blocco estratto il minatore riceve una ricompensa. Il proto-

collo viene applicato correttamente quando un minatore ha una probabilità p uguale alla frazione tra la sua e la totale potenza di calcolo di estrarre e risolvere un blocco.

Per un attaccante quindi è necessario risolvere gli stessi problemi degli altri partecipanti della rete protetta da PoW.

La problematica principale dei sistemi PoW è che la sicurezza viene garantita da risorse effettivamente scarse o costose come hardware specializzati nel mining ed elettricità.
Ciò rende i sistemi protetti da PoW inefficienti dal punto di vista delle risorse dato che per aumentare il profitto i minatori sono costretti a distribuire continuamente più risorse per il mining.
Questo rende praticamente impossibile un attacco ad un sistema protetto da PoW ma l'incompatibilità ecologica del protocollo ha portato a sforzi per costruire un protocollo di consenso simile con una richiesta molto inferiore di risorse.

5.1.1 Esempio in Bitcoin

Nel caso della blockchain **BTC**, che fa uso della metodologia PoW, per valutare come avviene il processo di validazione bisogna prima considerare come il blocco è composto.
Ogni blocco in Bitcoin è suddiviso in due parti:

- Intestazione del blocco con parametri chiave, incluso il tempo di creazione, il riferimento al blocco precedente e la radice dell'albero Merkle del blocco delle transazioni.

- Block list delle transazioni.

Per fare riferimento a un blocco specifico si usa il relativo header che viene sottoposto a due cicli di hash attraverso la funzione *SHA-256*.
Il processo dispendioso di validazione consiste nel risolvere un'equazione nella quale è presente un parametro incrementale nel tempo ai fini di garantire una sufficiente *difficoltà algortimica* e poter far fronte a improvvisi ed eventuali incrementi di potenze computazionali da parte delle macchine oggigiorno diffuse.
Attraverso questa metodologia basterà incrementare esponenzialmente il parametro e l'algoritmo di consenso rimane valido. Il parametro utilizzato in

blockchain BTC è il numero di cifre a 0 con le quali inizia un indirizzo BTC. L'unico modo per trovare stringhe soddisfacenti è effettuare il processo di hash in maniera enumerativa fino a che non risulterà un output soddisfacente. Chiaramente più alto è il valore della difficoltà, più iterazioni sono necessarie per trovare un blocco valido; il numero previsto di operazioni è esattamente la difficoltà.
Il periodo di tempo T(r) per un minatore con hardware in grado di eseguire k operazioni al secondo per trovare un blocco valido è distribuito esponenzialmente con la velocità k/D:

$$\mathrm{P}\ \ \mathrm{T}(k) \leq t = 1\text{–}exp(-kt/D)$$

Si considerino **n** minatori di Bitcoin con hash rate k_1, k_2, ...,k_n. Il periodo di tempo per trovare un blocco T è uguale al valore minimo delle variabili casuali T(k_i) assumendo che il minatore pubblichi un blocco trovato e raggiunga immediatamente altri minatori. Secondo le proprietà della distribuzione esponenziale, anche T è distribuito esponenzialmente:

$$\mathrm{P}\ \{\ \mathrm{T}\ def\ = \min(\mathrm{T}_1, \ldots, \mathrm{T}_\mathrm{n}) \leq t\} = 1\text{–}exp(-tD \textstyle\sum_n^{i=1} k_\mathrm{i})$$

$$\mathrm{P}\ \{\ \mathrm{T} = \mathrm{T}_\mathrm{i}\ \} = \mathrm{k}_\mathrm{i}\ \textstyle\sum_n^{j=1} k_\mathrm{j}.$$

L'ultima equazione mostra che il mining è giusto: un minatore con una quota di risorse per mining p ha la stessa probabilità p di risolvere un blocco prima degli altri minatori.

5.2 Proof of Stake

Si basa su un'implementazione di un particolare algoritmo in maniera tale da garantire sicurezza senza necessità di svolgere operazioni dispendiose in termini di hardware. L'idea è semplice: invece del potere di mining, la probabilità di creare un blocco e ricevere il premio è proporzionale alla quota di proprietà di un utente nel sistema.

Considerando un singolo *stakeholder*, la probabilità p che ha di risolvere un blocco sarà uguale alla frazione tra le sue e il numero totale di monete in circolazione sulla rete. Quindi gli utenti che detengono la quota più alta nel sistema hanno il maggiore interesse a mantenere la rete sicura, poiché soffriranno di più se la reputazione e il prezzo della criptovaluta sottostante diminuisse a causa di attacchi. La sicurezza in questo caso sta nel fatto che per sferrare un attacco un malintenzionato dovrebbe acquisire la maggior parte della valuta, il che sarebbe proibitivo per un sistema popolare oltre che controproducente da parte dell'attaccante che si ritroverebbe una moneta con un valore drasticamente basso.

5.2.1 Esempio

Si consideri un utente con indirizzo A e saldo *saldo(A)*. Un algoritmo Proof of Stake comunemente usato utilizza una condizione:

$$\text{Hash}(\ \text{Hash}(\ \text{Block}_{\text{prev}}),\ \text{A},\ \textit{time}) \leq \textit{saldo(A)}\ \text{M}\ /\ \text{D}$$

dove:

- $\text{Block}_{\text{prev}}$ indica il blocco su cui l'utente sta costruendo.
- *time* è il timestamp UTC corrente.

L'unica variabile che l'utente può modificare è il timestamp t nella parte sinistra dell'equazione mentre il saldo degli indirizzi è bloccato dal protocollo: il protocollo calcola il saldo sulla base di fondi non movimentati per un giorno e coinvolge calcoli costosi nella prova di partecipazione.

Insieme ad un indirizzo A ed un timestamp t soddisfacente un utente deve fornire una prova di proprietà dell'indirizzo perciò deve disporre di una chiave privata relativa all'indirizzo A. Il tempo per risolvere un blocco per l'indirizzo A è distribuito esponenzialmente con velocità *saldo*(A)/D. Di conseguenza l'implementazione della Proof of Stake è equa: la probabilità di generare un blocco valido è uguale al rapporto tra il saldo dei fondi dell'utente e l'importo

totale della valuta in circolazione. Il tempo per trovare un blocco per l'intera rete è distribuito esponenzialmente con velocità $\sum a \cdot saldo(a)/D$.

5.3 Delegated Proof of Stake

La *prova di partecipazione delegata* (DPoS) è un termine generico che descrive un'evoluzione dei protocolli di consenso con base PoS. I blocchi sono coniati da un set predeterminato di utenti del sistema (*delegati*), che vengono premiati per il loro dovere e puniti per comportamenti dannosi (come la partecipazione ad attacchi a doppia spesa). Negli algoritmi DPoS, i delegati partecipano a due processi separati:

- costruire un blocco di transazioni.
- verificare la validità del blocco generato mediante il processo di verifica della firma digitale.

Sebbene un blocco venga creato da un singolo utente, per essere considerato valido, deve essere firmato da più delegati. L'elenco degli utenti idonei per la firma dei blocchi viene modificato periodicamente in base a determinate regole. L'insieme di delegati per ogni blocco è in genere piccolo. In qualche versione di DPoS, un delegato deve mostrare impegno depositando i suoi fondi in un conto di sicurezza con un blocco temporale (garanzia). Questa versione di DPoS è spesso indicata come *Proof of Stake basata sul deposito*. La quantità viene calcolata in DPoS con uno dei seguenti metodi:

- i delegati possono essere eletti in base alla loro partecipazione al sistema.
- i delegati possono ricevere voti da tutti gli utenti del sistema con potere di voto dipendente dalla quantità di moneta in stake dell' elettore. i voti dei delegati sui blocchi validi possono avere poteri proporzionali all'ammontare del loro deposito cauzionale.

5.4 Confronto tra PoW e PoS

In questa sezione verranno analizzate alcune problematiche di sicurezza per poter confrontare al meglio le due tipologie di lavoro e poter sottolineare

peculiarità o difetti dei due protocolli.
Si considerano gli attacchi a doppia spesa ed altre tipologie di attacco frequenti che mirano proprio ai componenti adibiti a validare i blocchi: i nodi. Nel caso di vulnerabilità della tipologia "*double spending*" la possibilità di un attacco a doppia spesa riuscito diminuisce quando una transazione ottiene conferme basate su PoW e dipendenti dalla quantità di potere di mining posseduto dall'attaccante. Per diminuire ancor di più il rischio di doppia spesa dei fondi, si consiglia di attendere un certo numero di conferme. Inoltre, ci sono meccanismi di sicurezza per ridurre il rischio nei pagamenti veloci, nella quale la doppia spesa è di più facile effettuazione essendo queste microtransazioni inserite in blockchain con un tempo maggiore.

Per entrambi i consensi, Proof of Work e Proof of Stake, i tipi di attacchi sono comuni. Uno degli attacchi più comuni è l'attacco DoS il quale scopo è interrompere il normale lavoro della rete inondando i nodi di richieste inutili o non valide. Un altro tipo di attacco è il Sybil: interrompe la rete creando una serie di comportamenti scorretti sui nodi. La vulnerabilità della rete agli attacchi DoS e Sybil dipende anche dai dettagli del protocollo di rete. Non ci sono ragioni che renderebbero PoS meno vulnerabile a questi tipi di attacchi rispetto a PoW. In un mining corrotto, un utente (che dispone di molta potenza computazionale) rivela selettivamente i blocchi estratti allo scopo di far sprecare le risorse di calcolo ai minatori onesti.
L'attacco è inefficace per le valute PoS dato che nella generazione dei blocchi non vengono coinvolte risorse costose. D'altra parte, non sono noti casi di attacco attraverso la corruzione del mining eseguiti con successo in blockchain BTC.
Per il consenso PoW, un grado di vulnerabilità agli attacchi può essere previsto semplicemente in base all'hashrate totale del sistema mentre nel caso dei sistemi PoS non esiste una misura equivalente alla "salute" della rete.
La vulnerabilità dei meccanismi di Proof of Work e Proof of Stake per i tipi di attacco vengono riportate nella figura sottostante tratta da [23], documento nel quale è possibile trovare un ulteriore approfondimento sulle vulnerabilità.

Attack type	Vulnerability		
	PoW	PoS	Delegated PoS
Short range attack (e.g., bribe)	−	+	−
Long range attack	−	+	+
Coin age accumulation attack	−	+/–	−
Pre computing attack	−	+	−
Denial of service	+	+	+
Sybil attack	+	+	+
Selfish mining	+/–	−	−

Figura 5.1: The vulnerability of proof of work and proof of stake consensus mechanisms to attack types

5.4.1 Conclusioni

Come riportato anche nel documento [23], un aspetto negativo è che il costo dell'elettricità esercita una pressione aggiuntiva al tasso di cambio delle criptovalute dato che ancora i pagamenti delle bollette avvengono mediante valuta fiat. Quindi, è ovvio che il protocollo PoW deve essere sostituito.

D'altro canto un protocollo di consenso PoS non protegge adeguatamente i sistemi distribuiti e comunque presenta dei costi di mantenimento. Problematica dovuta al fatto che quando si utilizza il PoS, il validatore deve avere fondi in valuta, mentre il minatore PoW non ha necessariamente bisogno della valuta. Ecco perché la situazione ottimale per tutte le criptovalute sarebbe basarsi su Proof of Stake e utilizzare meccanismi aggiuntivi per risolvere specifici problemi di sicurezza.

Ad esempio il problema della distribuzione iniziale, quel problema che si presenta nelle criptovalute di tipo PoS quando vengono lanciate (non essendoci ancora valute in staking da parte dei partecipanti alla rete non si sa chi ha più o meno probabilità di convalida dei blocchi), può essere risolto utilizzando una versione a tempo limitato di Proof of Work. Si può prevenire invece un attacco a doppio spesa includendo informazioni su ultimi blocchi della transazione. Tuttavia queste features sono ancora incomplete.

Capitolo 6

Blockchain Ethereum

6.1 Introduzione

La nascita della tecnologia blockchain, come anticipato nell'introduzione, viene attribuita a Satoshi Nakamoto nel 2008 anche se vi sono ancora molte controversie sulla verità. A differenza delle origini della blockchain, la rivoluzione tecnologica che ha iniziato, risulta molto più chiara.

Diversamente da Bitcoin che nasce come sola valuta, **Vitalik Buterin**, creatore di Ethereum, ha ideato una blockchain nata per superare i limiti del protocollo Bitcoin estendendo il protocollo a Smart Contract con i quali ha reso possibile la creazione di applicazioni decentralizzate, organizzazioni autonome, token e molto altro. [12]

Ether, non essendo principalmente creata per essere una moneta digitale, insieme a Bitcoin detiene la maggior parte della capitalizzazione di mercato legata alle criptovalute.

Nel protocollo è compreso anche un linguaggio di programmazione integrato che fornisce un layer di astrazione consentendo a chiunque la creazione di proprie regole di proprietà, formati delle transazioni e funzioni di cambio stato delle transazioni. Questa caratteristica è stata resa possibile dalla creazione di Smart Contracts, ovvero insiemi di regole crittografiche eseguite solo se determinate condizioni prestabilite a monte vengono soddisfatte.

Il consenso in Ethereum è basato su un protocollo detto **GHOST** (*Greedy Heaviest Observed Subtree*) che verrà approfondito nel capitolo successivo.

6.2 Differenze con Bitcoin

Le differenze tra Ethereum e Bitcoin sono particolarmente evidenti [13] ed in questa sezione si andranno ad analizzare le differenze tra le due tipologie di blockchain. Oltre al dettaglio dei creatori che in BTC rimane velato ed in ETH è di pubblico dominio, la differenza sostanziale la notiamo nel vero e proprio utilizzo del protocollo.
Mentre BTC è nato come sola criptovaluta ETH, come detto, supporta anche altre caratteristiche come Smart Contracts che, letteralmente contratti intelligenti, sono contratti che non richiedono alle due parti sottoscriventi fiducia reciproca. (approfodimento nel capitolo Smart Contracts).

Un'altra differenza importante la troviamo nella struttura del blocco presente in blockchain. Mentre quello di BTC contiene *block number*, *difficulty*, *nonce*, ecc., quello di ETH riesce a mantenere anche la *transaction list* ed il relativo stato.

L'header del blocco presente nella blockchain ETH consiste in una serie di informazioni inerenti alla transazione.
La prima è una *stringa generata a partire dall'header del blocco padre* processato attraverso l'algoritmo di hash Keccak 256-bit, dopodiché è presente l' *indirizzo del destinatario* della mining fee, l'*hash* dello stato radice, **Transaction Trie** e **Transaction Receips Trie**: due parametri fondamentali che contengono una serie di informazioni a loro volta.

La *Transaction Trie* è composta da *nonce* (indicante il numero di transazioni effettuate dall'indirizzo mittente), prezzo delle gas fee, limite delle gas fee, destinatario, valore da trasferire, valori di firma della transazione e *account initialization*, nel caso in cui la transazione sia di tipo **contratto**, oppure *transaction data* nel caso la transazione sia di tipo **message call**.

La *Transaction Receipts Trie* invece comprende lo stato della transazione quando avvenuta, le gas fee utilizzate, l'insieme dei log generati dall'esecuzio-

ne della transazione e il *Bloom Filter*, composto anch'esso dalle informazioni contenute nei logs.

Sono presenti anche timestamp e diversi hash extra utilizzati a fine di verifica.

Un'altra differenza la troviamo tra i diversi utilizzi: può essere usato come derivato finanziario, sistema di identità e reputazione, archiviazione di file, assicurazioni, cloud computing, previsioni di mercato, ecc.

6.3 Differenze dovute ai due sistemi: PoW e PoS

Un problema di Bitcoin è rappresentato dall'ASIC (*application specific integrated circuit*) mining, ovvero la capacità di un dispositivo, progettato appositamente, di minare solo Bitcoin. Essendo un processo dispendioso sia in termini energetici che in termini di tempo (caratteristiche strettamente correlate), attraverso questo tipo di mining è possibile ottimizzare il processo favorendo i minatori che riescono ad ottenere un *hashrate maggiore*. Con hashrate o hash per secondo intendiamo l'unità di misura che rappresenta il numero di computazioni doppi di SHA-256 eseguiti in un secondo nella rete Bitcoin.
Un altra differenza dovuta alla progettazione del protocollo Bitcoin è che il nodo validatore che riceve una ricompensa in criptovaluta è solo il nodo che riesce con successo a validare il blocco di transazioni per primo.

Nel protocollo Ethereum, invece, viene usato il l'algortimo **Ethash** come Proof of Work che, essendo pesante in memoria, non è così adatto all' ASIC mining.
Sia per l'algoritmo Ethash che per i dispendi in termini di energia e tempo oltre ad un fattore di scalabilità, la blockchain Ethereum sta andando nella direzione del Proof of Stake prendendo il nome di **Ethereum 2.0**.

Sebbene ci siano preoccupazioni sulla scalabilità di Ethereum, è stato registrato che la rete Ethereum gestisce con successo oltre un milione di transazioni uniche in 24 ore con una media di circa 11 transazioni al secondo.

Come anticipato, il consenso nella rete Ethereum è basato sul protocollo GHOST modificato. E' stato rivisto per poter far fronte ai problemi dei blocchi obsoleti nella rete.
I blocchi obsoleti possono generarsi se un gruppo di miners ha più potenza di calcolo degli altri, il che significa che i blocchi del primo pool lo faranno contribuire maggiormente alla rete creando così un problema di centralizzazione. Attraverso il protocollo GHOST si riesce a far fronte alla problematica.

6.4 Il protocollo GHOST modificato

Il protocollo *Greedy Heaviest Observed Subtree* (GHOST) è una novità introdotta da Yonatan Sompolinsky e Aviv Zohar nel 2013 [29]. La necessità che ha portato allo sviluppo del protocollo è dovuta ai tempi di conferma. Si richiedono tempistiche brevi per garantire sicurezza di rete dato che i blocchi necessitano di un certo periodo di tempo per propagarsi a tutti i nodi.

6.4.1 Scenari risoluzione dei problemi

Problema spreco potenza computazionale

Se il miner A esegue il mining di un blocco e poi il miner B esegue il mining di un altro blocco prima che il blocco del miner A si propaghi a B, il blocco del miner B andrà sprecato e non contribuirà alla sicurezza della rete.

Problema di centralizzazione

Se il miner A è costituito da un pool di mining con il 30% di hashpower e B ha il 10% di hashpower, A corre il rischio di produrre un blocco obsoleto per il 70% del tempo (dato che per il rimanente 30% del tempo produce l'ultimo blocco e quindi ottiene subito i dati di mining) mentre B corre il rischio di produrre un blocco obsoleto per il 90% del tempo.

Quindi, se l'intervallo di validazione del blocco è abbastanza breve affinché il tasso di obsolescenza sia elevato (B non riesce a produrre blocchi in tempo), A sarà notevolmente più efficiente in virtù della sua potenza computazionale. Con la combinazione di questi due effetti, nelle blockchain che producono blocchi rapidamente, è probabile che un pool di mining con una

percentuale abbastanza elevata di *hashpower* in rete abbia di fatto il controllo sul processo di mining.

Come descritto da Sompolinsky e Zohar nel documento [29] il protocollo GHOST modificato, risolve il primo problema di perdita di sicurezza della rete includendo blocchi obsoleti nel calcolo della catena "più lunga"; cioè, al calcolo del blocco con la Proof of Work più elevata vengono aggiunti non solo il padre e gli elementi superiori di un blocco, ma anche i discendenti obsoleti dell'antenato del blocco. Per risolvere il secondo problema della propensione alla centralizzazione, si va oltre al protocollo descritto da Sompolinsky e Zohar fornendo ricompense anche per i blocchi obsoleti: un blocco obsoleto riceve l'87,5% della ricompensa di base dovuta e il nuovo blocco che include il blocco obsoleto il restante 12,5%.

Conseguenze

Attraverso l'implementazione del protocollo GHOST la velocità della rete è stata incrementata drasticamente. Il tempo di conferma di un blocco Ethereum è di circa 15 secondi con dei picchi a 30 dipendenti dal congestionamento della rete, a differenza di Bitcoin dove si parla di minuti dovuti proprio alla PoW.

Capitolo 7

Cryptography in Ethereum

7.1 Algortimi di consenso in Ethereum: Ethash

Questa funzione hash progettata per Ethereum è la funzione adibita all'operazione di mining. Un algoritmo di alta qualità che usa tecniche avanzate per assicurare la maggior sicurezza possibile. [6]
E' un algoritmo *memory-hard* ovvero necessita di una grande quantità di memoria per essere eseguito dato che è basato completamente su un DAG esteso (*grafo diretto aciclico*) di grandi dimensioni. Si basa su una funzione chiamata Keccak o SHA-3, oltre a utilizzare versioni di hash Dagger-Hashimoto, motivo per cui l'algoritmo inizialmente era noto proprio con questo nome e poi solo successivamente è diventato Ethash.

7.1.1 Cos'è Dagger-Hashimoto?

Per poter comprendere al meglio Ethash bisogna introdurre l'algoritmo **Dagger-Hashimoto**.
Questo algoritmo sta alla base del recente Ethash ed è stato sviluppato *by design* integrando due principi cardine:

1. L'algoritmo deve essere resistente all'ASIC mining.

2. L'algoritmo deve essere molto efficiente e facilmente verificabile da clients che non dispongono di elevate risorse.

Come si può comprendere dal nome, questo algortimo ha preso il nome dall'unione di altri due algoritmi sviluppati rispettivamente da **Vitalik Buterin**

e **Thaddeus Dryja**: Dagger e Hashimoto.

Dagger, l'algoritmo sviluppato dal creatore di Ethereum, fa grande uso di grafi aciclici estesi per costruire strutture dati pesanti. Su queste strutture sono eseguiti poi una serie di calcoli ospitati in memoria per aggiungere ulteriore carico computazione alla macchina.

Hashimoto invece, sviluppato da Thaddeus Dryja, include una resistenza ai sistemi ASIC. Tecnica effettuata mediante un sovraccarico della RAM che comporta una limitazione appunto per i sistemi ASIC.

Quello che fa in maniera semplificata è prendere i DAG (grafi diretti aciclici) generati dal Dagger, aggiungere informazioni riguardo la difficoltà e le transazioni per poi calcolare un hash che identifichi il blocco che, per poter essere estratto, genera una mole di lavoro non sottovalutatile dalla macchina. Si può facilmente comprendere che se i due algoritmi vengono combinati ne otteniamo uno che è complesso abbastanza per i miner di tipo ASIC affinché riescano ad implementarlo efficientemente.

Anche se apparentemente questa combinazione può sembrare vincente con lo sviluppo e l'evoluzione si è arrivati ad implementare Ethash.
La prima volta è stato presentato dal team di sviluppatori Ethereum che ha totalmente rivisto il funzionamento dell'algoritmo di base *Dagger-Hashimoto* mantenendo però i principi uguali.
Il nuovo funzionamento di Ethash si può riassumere nei seguenti passaggi:

1. Viene calcolato un seme utilizzando le intestazioni dei blocchi precedenti fino al punto in cui inizia l'estrazione.

2. Questo seme viene quindi utilizzato per calcolare e generare una cache pseudocasuale da 16 MB.

3. Questa cache viene quindi utilizzata per generare un set di dati di oltre 4 GB (il DAG). Questo set di dati è semipermanente e viene aggiornato ogni 30mila blocchi. In questo modo, il DAG varia per ogni *mining season*.

4. Una volta generato il DAG, inizia il mining. Questo processo prende valori casuali dal DAG e li combina utilizzando i dati forniti dalla rete e le transazioni da verificare.

5. Infine la verifica viene eseguita con un processo che rigenera parti specifiche del set di dati utilizzando la memoria cache, velocizzando il processo.

Questa versione è quella attualmente in esecuzione e corrisponde alla numero 23 dell'algoritmo. Durante tutto il processo vengono utilizzate le funzioni **Keccak-256** e **Keccak-512**, un algoritmo da cui è stato derivato lo standard **SHA-3**.

Il motivo di tutte queste modifiche o aggiornamenti sono proprio risoluzioni di problemi, vulnerabilità, ottimizzazioni e modifiche per rendere il lavoro più complesso agli ASIC miners.

7.1.2 Features di Ethash

- E' fortemente dipendente dalle operazioni in RAM e consuma anche grandi quantità di banda proprio per far fronte alla problematica degli ASIC miners e quindi mantenere il funzionamento di Ethash stabile. Le operazioni richieste per creare il DAG e la cache di lavoro generano questa feature.

- L'algoritmo è compatibile con le GPU avendo una capacità di memoria molto maggiore rispetto ad alcune CPU dato che ad oggi, la grafica in movimento come quella dei giochi, richiede enormi quantità di memoria per un'elaborazione parallela. Le GPU hanno influito molto nel mining perché riescono a mantenere in memoria tutto il DAG insieme alla cache così che tutti i calcoli eseguiti rimangano su un'unica area di lavoro estraendo nuovi blocchi molto più velocemente.

- Offre eccellenti capacità di verifica per i client con poche risorse. Con circa 16 MB di RAM è possibile creare un thin client in grado di verificare le transazioni in modo molto semplice e veloce. Inoltre, un thin client può essere attivo e funzionante ed eseguire il processo di verifica in appena 30 secondi.

7.2 Pro e contro algoritmo Ethash

7.2.1 Pro

1. È un algoritmo semplice da implementare, sicuro e pratico quando si tratta di fare un ottimo lavoro di resistenza agli ASIC.

2. È veloce: l'uso di un grafo in memoria, l'uso della cache e l'uso della funzione Keccak rendono questo algoritmo un processo efficiente nella produzione di blocchi.

Attraverso questo meccanismo Ethereum può rendere il tempo di produzione dei blocchi adattabile alle esigenze della rete cercando il compromesso tra sicurezza e scalabilità.

7.2.2 Contro

1. La resistenza ASIC è stata infranta nel 2018 quando Bitmain ha introdotto il suo primo miner in Ethash [8]. Da allora, sono stati cercati meccanismi per impedire un'ulteriore centralizzazione del mining su Ethereum. Tuttavia, lo sviluppo di Ethereum 2.0 prevede l'abbandono del Proof of Work (PoW) e quindi con esso Ethash come algoritmo di mining a favore di un Proof of Stake (PoS).

2. L'elevato consumo di memoria del tuo DAG rende molto difficile l'estrazione in macchine più modeste il che genera dei costi elevati.

3. Lo *scaling della sicurezza* dell'algoritmo è molto rapido perciò se si arrivasse a un punto troppo elevato di difficoltà si renderebbe impossibile l'estrazione per qualsiasi minatore che ci provasse. Questo è un serio rischio per la sicurezza della blockchain Ethereum (anche questo superficiale nel caso si abbandoni il PoW).

7.3 ECDSA e sicurezza

7.3.1 Introduzione

Per autorizzare tutte le transazioni presenti in rete Bitcoin ed Ethereum fanno uso di algortimi crittografici basati su curve ellittiche (**ECDSA**: *Elliptic Curve Digital Signature Algorithm*). [20]
I componenti fondamentali dell'algoritmo sono una funzione hash resistente e una curva ellittica.

Sia Bitcoin che Ethereum fanno uso della curva ellittica **secp256k1**: curva raccomandata dallo *Standards for Efficient Cryptography Group* perché rispetta determinate caratteristiche di sicurezza approfondite successivamente.

Prima di proseguire con i dettagli più tecnici bisogna fare la precisazione che spesso, quando questi tipi di algoritmi vengono applicati a situazioni reali, non è garantita la stessa complessità matematica teorica proprio perché vi è l'implementazione di mezzo che può modificare, seppur leggermente, la casualità dell'algoritmo.

Tutta la crittografia su curve ellittiche si basa sul problema del logaritmo discreto che viene considerato un problema difficile da risolvere. Chiaramente, questa è una considerazione vera ad oggi, poiché il problema ora non è risolvibile in tempo polinomiale ma nessuno ci può assicurare che non venga scoperto un metodo di risoluzione nei prossimi anni. Inoltre, solo un gruppo molto stretto di curve ellittiche, quelle definite sui campi binari, vengono ritenute di complessità sub-esponenziale e non esponenziale.

7.3.2 Il Problema del Logaritmo Discreto sulle curve ellittiche e ECDSA

Matematicamente, una curva ellittica E su un campo finito $\mathrm{F_q}$, $\mathrm{q = p^n}$ con n numero primo (ad esempio se $n = 1$ allora $\mathrm{F_p} = Z/pZ$), è l'insieme di soluzioni dell'equazione

$$E : \mathrm{y}^2 = x^3 + Ax + B \qquad (A,\ B \in F_\mathrm{q}),$$

ovvero :

$$E(F_\mathrm{q}) := \{(x, y) \in F_\mathrm{q} x F_\mathrm{q} \mid y^2 = x^3 + Ax + B\}.$$

Su una curva ellittica E abbiamo diversi gruppi di operazioni applicabili come nel caso del gruppo '+' sugli interi Z.

Lo scheletro del problema del logaritmo discreto (DLP) invece è formulato nel gruppo moltiplicativo $(\mathrm{F_p}^{\mathrm{x}}, \cdot)$, *ovvero* :

dato

$$g, h \in F_\mathrm{p}{}^\mathrm{x},$$

trovare

$$k \in Z/(p-1)Z \mid g^\mathrm{k} = h \ mod\ (p).$$

Formulare lo stesso problema per le curve ellittiche utilizzando il gruppo di operazioni '+' piuttosto del moltiplicativo ci porta al problema del logaritmo discreto della curva ellittica (ECDLP).
Dati due punti $P,\ Q \in E(F_q)$ *su* E, *trovare un intero* a, *se esiste*, *tale che l'equazione* $Q = aP$ *risulti valida. In alternativa*, *usando la notazione del logaritmo in modo intuitivo*, *calcolare* $a = log_\mathrm{p}(Q)$ *su* E.

L'algoritmo di firma utilizzato da ECDSA (Elliptic Curve Digital Signature Algorithm) utilizza l'intero a come chiave privata.
Lo scenario base dal quale facilmente si può comprendere il funzionamento si può riassumere in un utente U che vuole firmare un messaggio m per autorizzare un pagamento ad un commerciante M. Data (pubblica) una curva

ellittica E/F_q su un campo finito F_q, un punto base $P \in E(F_q)$ di ordine r, e una funzione hash H (i suoi valori saranno considerati modulo r per rientrare nello spazio della curva) che è pre-immagine e resistente alle collisioni allora:

- Private key: U sceglie come intero segreto $a \, mod \, r, es. \, a \in Z/rZ$.
- Public key: U calcola $Q = aP$ e usa Q. (In Bitcoin l'indirizzo dell'utente U è derivato da questo dato).

Dopodiché per firmare e verificare le firme apposte in maniera pubblica si possono usare le seguenti procedure:

Firma:

- U sceglie un numero casuale (che differisce tutte le volte) $1 \leq k < r$ *e calcola* $R = (x_R, y_R) = kP \in E(F_q)$.
  Quindi U calcola $s = k^{-1}(H(m) + ax_R)$ in Z/rZ (ben definito). La firma di U è quindi: (x_R, s).

Verifica della firma:

- M calcola $u_1 = s^{-1}H(m)$ e $u_2 = s^{-1}x_R$ in Z/rZ, e quindi $V = (x_V, y_V) = u_1P + u_2Q$ su E.
- M verifica la firma dell'utente U controllando se $x_R = x_V$ in Z/pZ è valido.

Se qualcuno potesse calcolare logaritmi discreti molto velocemente sarebbe in grado di calcolare la chiave privata a da P e Q e quindi firmare pagamenti a nome dell'utente U.

7.3.3 Sicurezza delle curve ellittiche

Esistono molti standard per proteggere la crittografia a curve ellittiche da attacchi.

Scenario: Sia E/F_q una curva ellittica su F_q con $q = p^n$.

Assumiamo che l'ordine del gruppo $E(F_q)$ è

$$\#\mathrm{E}(\mathrm{F_q}) = f \cdot r$$

con r primo e funzione arbitraria $f \in N$.

La scelta del primo p (che ovviamente deve essere grande), riguarda l'efficienza. Per misurare il livello di sicurezza di ECDLP per una specifica curva E è necessario conoscere il tempo di esecuzione (medio) dei possibili attacchi. Il meno complesso sarebbe calcolare tutti i multipli di P nel gruppo $\mathrm{E}(\mathrm{F_q})$ finché non troviamo Q. Questo ci porterebbe nel peggiore dei casi r operazioni di gruppo e in media $r/2$. Quindi per proteggere la nostra chiave privata da questo attacco si dovrebbe scegliere r in modo che $r/2$ sia abbastanza grande. Lo standard *Safecurve* invece raccomanda che il tempo di esecuzione sia maggiore di 2^{100} operazioni di gruppo su E.

La curva ellittica **secp256k1** è quella raccomandata dal gruppo *Standards for Efficient Cryptography*. È progettata per un numero primo a 256 bit. La curva ellittica secp256k1 è definita sul campo finito F_p con

$p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$, L'equazione di questa curva ellittica è

$$\mathrm{E} : \mathrm{y}^2 = x^3 + 7.$$

Capitolo 8

Ciclo di vita di una transazione in Ethereum

Per poter esplicare il ciclo di vita delle transazioni in Ethereum bisogna prima introdurre quali tipologie di account possono usufruirne. Per ogni account ETH vi sono delle regole di gestione delle transazioni definite dal protocollo Ethereum differenti ed in questo capitolo, oltre ad introdurre i vari tipi di account esistenti, verranno analizzate le interazioni tra questi.

8.1 EOA: Account Ethereum

Con account Ethereum intendiamo tutte quelle entità che, possedendo un saldo in Ether, possono inviare transazioni sulla rete. I conti sono controllabili da utenti oppure sotto forma di Smart Contract.

Ci sono due tipi di conto: quelli di *proprietà esterna* alla rete e i *contratti* [3]. Nei conti di proprietà esterna chi può agire sul conto, quindi chi può eseguire transazioni, è il controllore delle chiavi private. Per quanto riguarda invece l'altro tipo di conto, i contratti, vengono usati per poter distribuire in rete del codice eseguibile che però non è controllato da nessuno.

Tutti e due i tipi di account hanno possibilità di ricevere, conservare e inviare ETH e token oppure interagire con altri contratti intelligenti distribuiti.

Le differenze principali sono innanzi tutto di costo: mentre creare un conto posseduto esternamente, quindi con coppia di chiave pubblica e privata, non costa nulla, creare un contratto ha un costo dovuto al fatto che il codice

eseguibile messo in rete utilizza parte dell'archiviazione.
Inoltre, differenti tipologie di account implicano differenti politiche di gestione delle transazioni. I conti posseduti esternamente alla rete possono avviare transazioni e queste, nel caso siano tra due account esterni, possono riguardare esclusivamente trasferimenti di ETH o token. Nel caso di account di tipo contratto invece è possibile solo inviare transazioni in risposta alla ricezione di un'altra transazione ma non avviarne una.

Per costruzione, gli account di tipo contratto, quando ricevono transazioni da un account esterno, possono innescare del codice binario che può eseguire tutto ciò per il quale è stato progettato: dalla creazione di un nuovo contratto al trasferimento di token o ETH.

8.1.1 Da cosa è composto un account?

I conti (o account) di Ethereum hanno 4 campi fondamentali:

- Il primo è il *nonce* che indica il numero di transazioni inviate dal conto. Questo assicura che le transazioni siano elaborate una volta evitando problematiche di double-spending. In un account basato su contratto, questo numero rappresenta il numero di contratti creati dall'account.

- Il *balance* (o saldo) del conto ovvero il numero di "wei" posseduti da questo indirizzo. Wei è una denominazione di ETH e ci sono 10^{18} "wei" per ETH.

- Il *codeHash*, una stringa che si riferisce al codice eseguibile di un account di tipo contratto sulla macchina virtuale EVM. Come visto gli account di tipo contratto mantengono una porzione di codice eseguibile in rete e questa, non modificabile, è mantenuta in un database di stato inferiore agli hash corrispondenti. In questo modo è possibile identificare la porzione di codice a partire da un codeHash: per gli account esterni invece il codeHash è l'hash di una stringa a vuota.

- Lo *storageRoot* oppure hash di archiviazione è un hash a 256 bit del nodo radice di un tree di Merkle Patricia con il quale viene codificato lo spazio di archiviazione disponibile all'account.

8.1.2 Indirizzi e creazione dei due tipi di account

I conti esterni sono composti da una coppia di chiave pubblica e privata generata sulla base di una curva ellittica (in particolare la stessa di Bitcoin, la secp256k) attraverso l'algoritmo ECDSA.

Tutte le transazioni presenti in rete, nel momento in cui vengono spedite, sono sottoposte anche ad un processo di firma digitale attraverso l'utilizzo della chiave privata del mittente. La verifica della firma apposta viene effettuata mediante l'utilizzo della chiave pubblica del mittente (metodologia illustrata nel capitolo precedente), rendendo quindi tutte le transazioni disponibili ai partecipanti di rete favorendo la trasparenza della rete.

I conti di tipo contratto invece non hanno una coppia di chiavi pubblica e privata.

Indirizzi dei due tipi di account

La chiave pubblica è generata dalla chiave privata usando *Elliptic Curve Digital Signature Algorithm*. Si può ottenere l' indirizzo pubblico di un account esterno attraverso gli ultimi 20 byte dell'hash Keccak-256 della chiave pubblica e aggiungendo *0x* all'inizio.

L'indirizzo dell'account contratto viene invece indicato quando un contratto viene distribuito nella blockchain di Ethereum. L'indirizzo deriva da quello del creatore e dal numero di transazioni inviate da tale indirizzo (il *nonce*). Un account **non** è un *wallet*. Un account è la coppia di chiavi per un account Ethereum di proprietà dell'utente mentre un wallet è un'interfaccia o un'applicazione che ti permette di interagire con il tuo account Ethereum.

8.2 Transazione in Ethereum

Ethereum è una macchina a stati basata su transazioni e cambi di stato. Ogni qualvolta che una transazione avviene con successo lo stato della macchina viene aggiornato. Il ciclo di vita di una transazione tocca molteplici componenti della rete Ethereum.

Supponendo lo scenario di uno scambio di denaro, un sender *s* firma digitalmente una transazione e la spedisce sulla rete attraverso un client Ethereum con una chiamata JSON-RPC. JSON-RPC è un protocollo RPC (Remote Procedure Call) senza stato e leggero. In primo luogo, la specifica definisce diverse strutture di dati e le regole relative alla loro elaborazione ed utilizza chiaramente JSON come formato dati.

Ogni transazione è composta dal **destinatario** del messaggio, una **firma** che identifica il mittente, la **quantità** di Ether da inviare, un campo **dati facoltativo** e i valori **STARTGAS**, e **GASPRICE**. [13]

Quando il client ETH ha validato la transazione può spedirla in broadcast sulla rete peer-to-peer di Ethereum.
In questo momento tutti i client di tipo miner che ricevono questa transazione la aggiungono alla loro transaction pool, ovvero un insieme di transazioni in attesa di essere processate dal nodo.
Durante l'esecuzione degli algoritmi di mining il nodo deve scegliere quali delle transazioni processare.

La metodologia di scelta dalla transaction pool ovviamente mette in testa quelle transazioni che hanno un valore di **gas fee** più alto, in modo da ottenere una ricompensa maggiore nel momento dell'eventuale validazione a buon fine.

I campi STARTGAS e GASPRICE, come si può facilmente comprendere, sono vitali nel ciclo di vita della transazione e per farla "sopravvivere" in rete. Con *Gas* si intende un'unità fondamentale di calcolo. Ogni transazione richiede un determinato numero di elaborazioni e il campo STARTGAS denota il numero massimo di passaggi di elaborazione consentiti alla transazione. Il prezzo è 1 gas per 1 passo computazionale più il prezzo aggiuntivo

fisso di 5 gas per byte nell'area dati, questo valore può essere maggiore ed è definito nel campo GASPRICE. Dal momento che i minatori vengono premiati di più se elaborano la transazione con GASPRICE maggiore, il sender deve scegliere attentamente il valore di GASPRICE affinchè la sua transazione venga elaborata. D'altro canto, anche i minatori devono scegliere un GASPRICE minimo in base al quale accettano o si rifiutano di elaborare una transazione. La funzione di transizione dello stato di Ethereum, che cambia gli stati del mittente e del destinatario eseguendo una transazione, inizia con la verifica della correttezza della transazione (la firma è valida e il *nonce* corrisponde al *nonce* del conto del mittente). Se la transazione risulta corretta, viene calcolata la commissione di transazione come STARTGAS * GASPRICE, sottraendo tale valore dal saldo dell'account sender e aumentando il suo campo *nonce*. Se il processo si conclude con successo allora la commissione viene pagata e la quantità richiesta di Ether viene trasferita al destinatario.

L'account ricevente viene creato se non esiste già oppure, nel caso si tratti di un contratto, viene eseguito il codice al suo interno. Se il mittente non ha una quantità sufficiente di Ether per la transazione oppure l'esecuzione del codice spende tutta la quantità di gas disponibile, la funzione di transizione di stato annulla tutte le modifiche effettuate tranne le commissioni di pagamento per i minatori.

Come anticipato esistono due tipi di transazione, il primo è *money-transfer* e il secondo è *contract-creation*. Per una transazione di tipo *money-transfer*, l'importo specificato viene trasferito dall'EOA del mittente all'EOA o all'account destinatario mentre per un'operazione di *contract-creation* dove l'input è un pezzo di bytecode, viene creato un nuovo account contratto ed associato al bytecode generato dal compilatore (vedi capitolo Smart Contract). Per quanto riguarda invece le *contract-invocation* dove il destinatario è un contratto univocamente identificato, la chiamata si limita a caricare il bytecode presente all'interno del contratto nell'ambiente di esecuzione EVM (Ethereum Virtual Machine).

Capitolo 9

Novità introdotte dalla blockchain Ethereum

9.1 Smart Contract

Durante gli anni 90 Nick Szabo ha coniato il termine **smart contract** definendolo come "*a computerised transaction protocol that executes the terms of a contract*" [10]

In altre parole sono come contratti stipulati in linguaggio informatico o tramite l'uso di un determinato protocollo informatico. La peculiarità di uno smart contract è il potersi eseguire nel momento esatto in cui delle condizioni stipulate precedentemente dalle parti si verificano.
In sostanza non corrisponde esattamente ad un contratto fisico non avendo valore legale ma a una procedura attuata automaticamente.

Può sembrare un meccanismo "banale" ma se lo si pensa applicato alla blockchain, nella quale non sempre le due parti sottoscriventi hanno fiducia reciproca, si può comprenderne le potenzialità.

Non solo garantisce fiducia ma rende tutto il sistema blockchain ancora più trasparente garantendo certezza del rispetto delle clausole e dei risultati di queste operazioni a tutti i partecipanti della rete.
Inoltre garantisce immutabilità, caratteristica non sottovalutabile nel momento in cui si stipula un contratto.

Essendo una tecnologia ancora non pienamente regolamentata in tutto il mondo vi sono ancora dei limiti da superare per i quali gli smart contract potrebbero aver bisogno di tempo.
I problemi fondamentali sono legati alla giurisprudenza in materia di blockchain e smart contract essendo ancora pochi i paesi nel mondo che hanno legiferato su questo nuovo mix di tecnologie.

In Europa si è sentita la necessità di regolamentare gli smart contract e si stanno effettuando analisi di costo e beneficio su queste tecnologie mentre in Italia il legislatore ha definito gli smart contract come programmi per elaboratori operanti su un database condiviso e distribuito che vincola due o più parti sulla base di clausole stipulate dagli stessi.

9.1.1 Smart Contract nel dettaglio

Nel contesto di Ehtereum, uno smart contract, si riferisce ad un programma per elaboratori immutabile che esegue deterministicamente nel contesto di una Ethereum Virtual Machine come parte del protocollo della rete Ethereum.

Per poter comprendere al meglio la definizione occorre specificare cosa si intende con alcuni dei termini sopra riportati. Si parla di programmi per elaboratori poiché uno smart contract non è altro che un software e quindi senza alcun valore **legale** come potrebbe sembrare.

Immutabile invece è la caratteristica fondamentale dello smart contract. Una volta aggiunto in rete Ethereum il codice rimane immodificabile. L'unico modo per poter modificare uno smart contract è aggiungerne un altro alla rete.

Deterministico dovuto al fatto che l'output risultante dall'esecuzione del contratto è lo stesso per chiunque lo esegua dato uno stesso contesto iniziale della transazione e la sua esecuzione.

Lo smart contract opera in un contesto molto limitato detto EVM content (Ethereum Virtual Machine). Il loro software può accedere e far riferimento solo al proprio stato, al contesto della transazione richiamante e ad alcune

informazioni del blocco più recente. Inoltre, ogni EVM, viene eseguita su ogni nodo Ethereum ma dato che ogni istanza, partendo da uno stato iniziale comune produce uno stato finale uguale per tutte le istanze, si può considerare come una macchina che opera come un singolo **world computer**.

9.1.2 Ciclo di vita di uno Smart Contract

Gli smart contract sono generalmente scritti in un linguaggio ad alto livello ed in questo capitolo verrà analizzato come possono essere facilmente scritti in Solidity: un linguaggio di programmazione apposito per smart contracts.

Quando uno smart contract viene scritto, per potersi eseguire sulla rete Ethereum, necessita di una traduzione in linguaggio a basso livello eseguibile sulla EVM. Il componente adibito alla traduzione è, come nei tipici linguaggi di programmazione, il compilatore. Attraverso il compilatore **solc** (Solidity compiler) viene effettuata questa traduzione fino ad ottenere un low-level code eseguibile anche sulla EVM.

Una volta compilati, gli smart contract possono essere eseguiti sulla rete Ethereum usando una transazione particolare denominata *contract creation transaction*. Come anticipato ogni contratto è identificato da un indirizzo Ethereum derivato dalla transazione di creazione del contratto in funzione dell'account creatore e del relativo *nonce*.
L'indirizzo Ethereum del contratto può essere usato in una transazione come destinatario per poter inviare moneta oppure richiamare una particolare finzione del contratto. Una caratteristica importante è che nemmeno il creatore ha dei vantaggi sul contratto a meno che non li abbia direttamente codificati al suo interno ed infatti ad ogni contratto non fa riferimento nessuna chiave pubblica o privata (si può dire che un contratto inserito in rete non è posseduto da nessuno se non dal contratto stesso).
Ogni contratto per poter essere eseguito deve essere richiamato da una transazione e nel momento della sua esecuzione potrebbe far riferimento e chiamate a funzioni di altri contratti (anche multilivello). Pur essendo possibile eseguire contratti da altri contratti nessuno di questi può eseguirsi da solo o in parallelo ad altri anche perché la EVM, per metodologia di lavoro, si può intenderla come un computer potente ma single thread quindi nel quale

tutte le transazioni sono *atomiche*.
Tutte le transazioni legate ai contratti vengono eseguite nella loro interezza ed ogni modifica nello stato della rete globale viene registrato solo se tutte le esecuzioni terminano con successo. Con successo si indica che nessuna delle esecuzioni ha generato errori altrimenti la rete procederebbe con il roolback delle modifiche e la registrazione del tentativo in blockchain (chiaramente le gas fee necessarie non verrebbero restituite all'account originario).
Come velocemente anticipato, un contratto non può essere modificato ma può essere eliminato. L'eliminazione avviene rimuovendo il codice presente al suo interno e svuotando la memoria dall'indirizzo Ethereum associato lasciando in rete un account vuoto. Per eliminare un contratto bisogna eseguire sulla EVM un comando chiamato SELFDESTRUCT e questa operazione ha un costo negativo, ovvero viene incentivata l'eliminazione dei contratti per liberare, seppur in maniera minima, lo spazio di archiviazione della rete. La possibilità di eliminazione del contratto attraverso la chiamata SELFDESTRUCT deve essere inizializzata da parte dell'utente creatore nel momento della scrittura altrimenti lo smart contact non può essere rimosso.
Chiaramente, trovandoci su una blockchain immutabile, tutte le transazione che coinvolgevano il contratto prima dell'eliminazione rimangono visibili a tutti senza possibilità di rimozione.

9.1.3 Introduzione a Solidity e alla scrittura di uno Smart Contract

Solidity è stato creato da Dr. Gavin Wood come un linguaggio per scrivere esclusivamente smart contract con features per supportare l'esecuzione nell'ambiente decentralizzato di Ethereum. Il risultato era abbastanza generalizzabile, così, seppur la nascita fosse correlata ai soli smart contracts in Ethereum, oggi Solidity viene usato per interagire con molteplici altre blockchain.

Solidity ad oggi è sviluppato e mantenuto come progetto indipendente su GitHub (link: https://github.com/ethereum/solidity)

Il prodotto principale, come anticipato, è proprio il compilatore solc che riesce a trasformare codice scritto in Solidity in linguaggio macchina eseguibile dalla EVM.

Installazione Solidity

Ci sono molteplici maniere di installare il compilatore Solidity all'interno di un pc, dalla compilazione del codice sorgente alla semplice esecuzione della binary release. La procedura illustrata fa uso del package manager apt su sistemi Ubuntu/Debian.

```
$ sudo add-apt-repository ppa:ethereum/ethereum
$ sudo apt update
$ sudo apt install solc


$ solc --version
solc, the solidity compiler commandline interface
Version: 0.4.24+commit.e67f0147.Linux.g++
```

Figura 9.1: Solidity installation procedure using apt package manager

Alla fine della procedura solc, il compilatore di Solidity sarà installato sulla macchina e pronto all'uso. Possiamo compilare tutti i file che hanno estensione *.sol* per poterli trasformare in codice binario eseguibile nell'EVM.

Per poter sviluppare smart contracts possiamo usare qualsiasi editor di testo anche se sono consigliati Remix IDE o Eth-Fiddle, due ambienti di sviluppo web-based.

Nella prossima sezione vi sarà un esempio di smart contract minimo denominato Faucet scritto usando Remix IDE.

Listing 9.1: My first Smart Contract in Solidity

```
//My first Smart Contract
contract Faucet {

  //Free Ether for anyone!
  function withdraw(uint withdraw_amount) public {
    require(withdraw_amount <= 1000000000);

    msg.sender.transfer(withdraw_amount);
  }

  function () public payable {}
}
```

Compilazione del contratto

Ora, usando il compilatore solidity solc si potrà compilare il file sorgente Faucet.sol con diverse opzioni. Tutte le opzioni disponibili del compilatore possono essere visualizzate eseguendo il comando solc con l'argomento *-help*. Gli argomenti utilizzati in questo caso sono *—bin* e *—optimize* per produrre un binary file ottimizzato del nostro contratto.

```
$ solc --optimize --bin Faucet.sol
======= Faucet.sol:Faucet =======
Binary:
6060604052341561000f57600080fd5b60cf8061001d6000396000f300606060405260043610603e5
763ffffffff7c0100000000000000000000000000000000000000000000000000000006000350416
632e1a7d4d81146040575b005b3415604a57600080fd5b603e60043567016345785d8a00008111156
06357600080fd5b73ffffffffffffffffffffffffffffffffffffffff331681156108fc0282604051
600060405180830381858888f1935050505050151560a057600080fd5b505600a165627a7a723058203
556d79355f2da19e773a9551e95f1ca7457f2b5fbbf4eacf7748ab59d2532130029
```

Figura 9.2: The binary representation of a Smart Contract that can run in the EVM

Il risultato prodotto dal compilatore sarà in formato binario hex-serialized e può essere direttamente sottomesso alla blockchain di Ethereum.

9.2 NFT (ERC721)

Con il termine **NFT** indichiamo tutti quei token che non sono fungibili (*non-fungibile token*) [25]. Solitamente con il termine NFT si pensa ad arte digitale che negli ultimi anni (2020-2021) ha rappresentato uno dei settori nei quali è stato generato maggiore movimento di denaro.

Il concetto di NFT deriva originariamente da uno standard di Ethereum con l'obiettivo di distinguere ogni token con segni distinguibili. Questa tipologia di token può essere associato ad un concetto di proprietà digitale attraverso le identificazioni univoche. Di conseguenza tutte le proprietà digitali possono essere liberamente scambiate attraverso dei *costi* che variano in base a fattori che spesso dipendono dall'opera stessa. Nel caso degli NFT si può parlare di rarità, di età del token ecc.

Seppur un settore molto espanso ad oggi, si crede sia ancora nella sua fase iniziale e nel quale si trova ancora molta incertezza.

9.2.1 NFT nel dettaglio

Come anticipato, nel settore degli NTF vi è ancora molta incertezza ed in questo capitolo vi sarà una panoramica sulla metodologia di questo settore fornendo informazioni su protocollo e standard.

L'ecosistema NFT chiaramente deve essere basato su un Ledger distribuito, consentire transazioni e possibilità di trading con queste opere. Oltre a questo, il settore si basa su un sistema a due ruoli: colui che compra l'NFT e colui che vende l'NFT.
Vi sono inoltre delle macro aree per quanto riguarda il settore degli NFT dato che le opere necessitano, per poter essere oggetto di trading o di vendita, di processi adeguati alle necessità della community.

NFT Digitalize

Questo processo funziona come una sorta di controllo di qualità dell'NFT stesso. Si controlla che il file, il titolo e la descrizione siano accurati dopodiché si procede a digitalizzare il dato grezzo in un proprio formato.

NFT Store

Con store intendiamo quella caratteristica di poter salvare il dato grezzo che rappresenta l'NFT anche in un database esterno che non risiede in blockchain. Questo meccanismo è utilizzato da chi vuole rendere unico il proprio NFT generando meno fee possibili. Da notare che, pur essendo molto più gas-consuming, si può salvare l'NFT anche direttamente in blockchain.

NFT Sign

Processo tramite il quale si firma una transazione che include l'hash dei dati grezzi dell'NFT e lo si spedisce ad uno smart contract.

NFT Mint and Trade

Processo che inizia quando l'NFT è effettivamente pubblico e riguarda la vendita o lo scambio a prezzi totalmente dipendenti dall'opera stessa.

NFT Confirm

Processo che viene avviato quando una transazione è confermata. Attraverso questo processo gli NFT saranno collegati per sempre ad un unico indirizzo blockchain come prova della loro persistenza. Nell'immmagine sottostante riportata dal documento [25] si può osservare come un sistema NFT svolge i processi sopraelencati garantendo trasparenza e immutabilità.

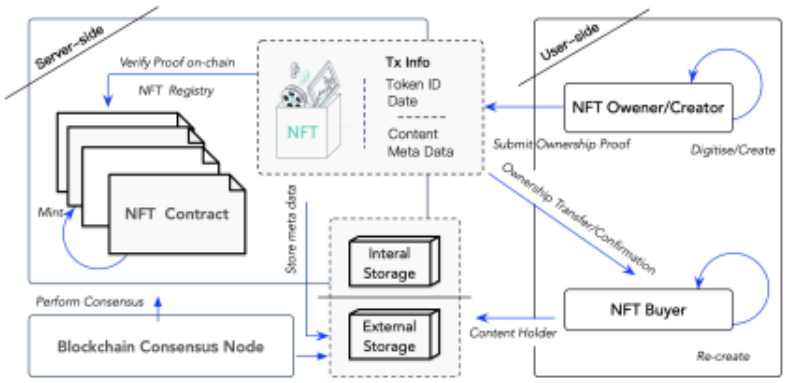

Figura 9.3: Model of NFT system

Quando un NFT viene venduto vi è la necessità di una nuova transazione (come visibile nell'immagine oltre che nel capitolo smart contract) per richiamare lo smart contract collegato. Dopo la conferma di questa transazione l'NFT e i metadati oltre ai nuovi dettagli di proprietà vengono aggiunti ad un nuovo blocco della blockchain così che la storia e la titolarità dell'NFT rimangano preservate.

9.2.2 Utilizzo degli NFT

Le opportunità di utilizzo per gli NFT sono molteplici ed in questo capitolo se ne analizzeranno alcuni ponendo particolare attenzione allo scopo trovato in un altro dei settori più in sviluppo in questi anni: quello del gaming.
Molti giochi infatti già includono la possibilità di acquistare NFT al loro interno e in questo caso possono essere usati per favorire l'acquisto di *edizioni limitate* favorendo il *marketing* del gioco stesso.
Un altro utilizzo sempre all'interno dei videogiochi è il legare benefici agli utenti possessori di NFT creando nel giocatore la sensazione di esclusività.

Inoltre è possibile anche la creazione di NFT-Ticket per evitare rincari dei prezzi stessi oppure evitando frodi nel momento dell'acquisto (si pensi ai problemi di rivendita dei ticket).

Tutto ciò che non è digitale ed ha la necessità di certificare la provenienza può essere protetto mediante l' NFT con identità del proprietario integrata.
Un altro esempio, dovuto alla particolarità degli NFT di rimuovere gli intermediari negli scambi di opere, potrebbe essere per un eventuale artista che sollevato dalla necessità di vendere la proprietà di un'opera ad un agente, decide di vendere direttamente digitalmente le sue opere.
In più gli NFT possono essere programmati in modo che l'autore possa ricevere una predeterminata tassa di royalty ogni volta che la sua opera digitale si scambia nei mercati.
Altro aspetto fondamentale nel quale troveremo sicuramente NFTs è l'ambito del metaverso, settore, anche questo, ad oggi totalmente in sviluppo.

9.3 Token fungibili (ERC20)

I token creati attraverso la tecnologia blockchain esistono da prima dell'invenzione di Ethereum di Vitalink Buterin. In altri termini basti pensare a Bitcoin: la prima blockchain sviluppata è proprio inerente alle criptovalute. Di conseguenza, molte piattaforme offrivano i loro token basati su Ethereum e Bitcoin ma nessuno di questi condivideva uno standard per poter essere scambiato.

Il primo standard introdotto è stato l'**ERC20** che ha portato ad un'esplosione di tokens.

La differenza principale tra i token basati su Ethereum e Ether, la criptovaluta correlata, è proprio il fatto che Ether è una valuta intrinseca alla rete mentre spedire o semplicemente detenere token non lo è. Il bilancio di Ether è mantenuto a livello di protocollo mentre il bilancio dei token è mantenuto nei singoli smart contract. Per creare un nuovo token sulla rete basterà creare uno smart contract. Una volta messo in esecuzione sulla EVM, lo smart contract si occuperà di tutto: se ne può scrivere uno che esegue determinate azioni personalizzate oppure decidere di aderire ad uno standard, caratteristica che viene apprezzata dalla community a favore della trasparenza.

9.3.1 Lo standard ERC20

Il primo standard è stato introdotto nel Novembre del 2015 da Fabian Vogelsteller [2] come un ERC (Ethereum Request for Comments). E' stato assegnato automaticamente da GitHub l'issue numero 20 creando così l'acronimo ERC20. La ERC20 è divenuta poi Ethereum Improvement Proposal 20 ma ha mantenuto il suo ormai nome ERC20.

9.3.2 Lo standard ERC20 nel dettaglio

ERC20 è uno standard per i token fungibili ovvero significa che differenti unità di un token ERC20 sono intercambiabili e non hanno un unico proprietario.

Lo standard definisce delle interfacce comuni ai contratti i quali poi implementano a loro volta i token in modo che, definendo comportamenti comuni,

possano essere usati tutti nello stesso modo. L'interfaccia comprende diverse funzioni che rappresentano una base da implementare presente in ogni token ERC20 al quale poi ogni sviluppatore può aggiungere attributi o funzioni aggiuntive.

9.3.3 Funzione e eventi standard ERC20

Saranno elencati tutti i metodi e gli eventi principali da implementare per la costituzione di un token ERC20 contrassegnati con *M* (method) nel caso di tratti di funzioni oppure *E* (event) nel caso si tratti di eventi.

- M - totalSupply: metodo che ritorna il totale dei token esistenti. I token ERC20 possono avere un totale fisso o variabile.
- balanceOf: metodo che dato un indirizzo ritorna il bilancio.
- M - transfer: Metodo che dato un indirizzo e una quantità, trasferisce la quantità di token dall'indirizzo richiamante all'indirizzo passato come argomento.
- M - transferFrom: Dato un mittente, un destinatario e una quantità trasferire i token da un indirizzo all'altro. Usato in combinazione con approve.
- M - approve: metodo che dato un indirizzo destinatario e un importo autorizza tale indirizzo ad eseguire più bonifici fino a tale importo, dal conto che ha rilasciato l'approvazione.
- M - approvance: metodo che dato un indirizzo proprietario e un indirizzo ricevente ritorna l'ammontare che il ricevente è autorizzato a prelevare dal proprietario.
- E - Transfer: evento che viene attivato quando si verifica un trasferimento di token andato a buon fine. (attraverso metodo transfer o transferFrom).
- E - Approval: evento attivato dopo una chiamata di approvazione riuscita.

Listing 9.2: ERC20 interface

```
interface IERC20 {

    function totalSupply() external view returns (
       uint256);
    function balanceOf(address account) external
       view returns (uint256);
    function allowance(address owner, address
       spender) external view returns (uint256);

    function transfer(address recipient, uint256
       amount) external returns (bool);
    function approve(address spender, uint256 amount
       ) external returns (bool);
    function transferFrom(address sender, address
       recipient, uint256 amount) external returns (
       bool);


    event Transfer(address indexed from, address
       indexed to, uint256 value);
    event Approval(address indexed owner, address
       indexed spender, uint256 value);
}
```

9.3.4 Workflow ERC20

ERC20 consente due tipi di workflow. Il primo è il tipico workflow che comprende una singola transazione ed avviene usando semplicemente l'implementazione della funzione transfer.
L'esecuzione rimane molto semplice: se Alice vuole spedire a Bob 10 token basterà effettuare una chiamata alla funzione *transfer* passando come argomento l'indirizzo di Bob e l'importo 10 in modo che il contratto aggiusti i due bilanci (Alice -10 e Bob +10).

La seconda metodologia di lavoro invece è quella che avviene quando chiamiamo la *transferFrom*. Mentre con la funzione *transfer* il trasferimento di token avveniva direttamente tra due utenti con l'indirizzo e importo, attraverso la funzione *transferFrom*, si può delegare il trasferimento ad un altro indirizzo che si occuperà della distribuzione.

Ad esempio se una compagnia sta vendendo token per ICO possono approvare un contratto di crowdsale per distribuire un certo numero di token. Il contratto crowdsale consente poi di eseguire una *transferFrom* dal proprietario dei token ai vari utenti compratori.

Una ICO è un meccanismo di crowdfunding utilizzato da diverse compagnie per poter guadagnare soldi vendendo token [13].

9.4 ENS: Ethereum Name Service

L' **Ethereum Name Service** è un sistema di classificazione dei nomi distribuito, open-source ed estendibile basato sulla blockchain di Ethereum. Nasce per risolvere la problematica degli indirizzi difficili da ricordare dovuta al fatto che tutti gli indirizzi o stringhe utilizzate in blockchain non sono stati progettati per un uso umano bensì della macchina.

Ethereum Name Service vede i suoi primi sviluppi all'inizio del 2017 [5] ed era originariamente collegato alla Ethereum Foundation. Tuttavia nel 2018 l'ENS si è staccata dalla fondazione come organizzazione separata. L'organizzazione senza scopo di lucro denominata *True Names LTD* [4] ora gestisce lo sviluppo di ENS.

Il ruolo principale dell'ENS è molto simile a quello che il DNS (Domain Name System) svolge per l'attuale web2. Il compito è quello di mappare tutti gli indirizzi identificatori come gli indirizzi Ethereum in nomi comprensibili e di facile consultazione dall'uomo.
Non è solo possibile mappare gli indirizzi Ethereum ma anche indirizzi di criptovaluta, hash di contenuto oppure metadati.

Come anche il DNS è in grado di fare, anche l'ENS supporta la risoluzione inversa degli indirizzi rendendo possibile associare nomi o descrizioni ad indirizzi Ethereum. A differenza del DNS l'**architettura** dell'Ethereum Name Service è **distribuita** ovvero in esecuzione sulla rete Ethereum evitando qualsiasi problematica di Single Point of Failure ma non è l'unica differenza architetturale dovuta ai vincoli della blockchain.

La metodologia di lavoro e risoluzione degli indirizzi è strettamente legata a quella del DNS tant'è che anche l'ENS opera su sistemi di nomi gerarchici separati da punti o domini.
Come nel web2, anche attraverso un dominio ENS è possibile creare sottodomini tutti gestiti dal titolare del dominio principale. Il sistema inoltre è già predisposto ad un importazione dei domini DNS su ENS se già di proprietà di un determinato utente. Un tipico nome di dominio in ENS termina con ".eth".

9.4.1 Architettura dell'ENS

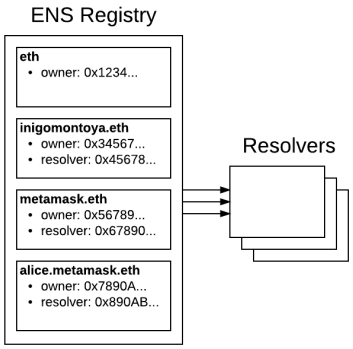

Figura 9.4: Architecture of the Ethereum Name Service

Come possiamo dedurre dall'immagine soprastante i principali ruoli nell'architettura dell'ENS sono due: il **registry** e il **resolver**. I domini di primo livello, come ".test" e ".eth", sono controllati e di proprietà di smart contract e vengono chiamati **registrar**. Questi registrar specificano le regole sulle quali si basa la distribuzione di qualsiasi sottodominio. Chiunque può seguire le regole degli smart contracts di registrazione per ottenere il proprio dominio.

Registry

Il **registry** ENS è costituito da uno smart contract incaricato di mantenere un elenco di tutti i domini e sottodomini. Inoltre, per tener traccia dei domini, il registro deve memorizzare tre informazioni fondamentali:

1. Il proprietario del dominio.

2. Il risolutore per il dominio.

3. Il *time-to-live* di memorizzazione nella cache per tutti i record nel dominio.

Un proprietario di dominio può essere sia uno smart contract che un account esterno. Un registrar è essenzialmente uno smart contract che possiede un dominio. Questi contratti hanno la possibilità di rilasciare sottodomini

agli utenti che rispettano le regole dello smart contract.

Di conseguenza i proprietari di un dominio possono svolgere le seguenti azioni:

1. Specificare il resolver e il TTL del dominio.
2. Trasferimento di proprietà del dominio.
3. Cambia la proprietà dei sottodomini.

Inoltre, il registry ENS è relativamente semplice ed esiste solo per mappare un nome al particolare risolutore responsabile del dominio.

Resolver

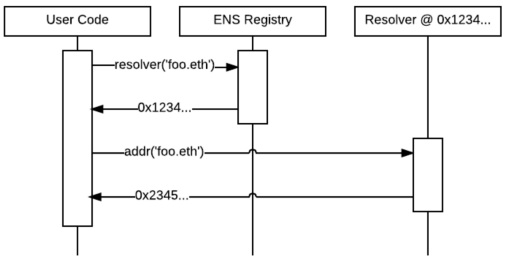

Figura 9.5: The ENS Resolver and how it work

I **resolver** sono i componenti di ENS responsabili del processo di traduzione dei nomi in indirizzi. Qualsiasi smart contract che implementa gli standard correttamente può fungere da risolutore nel sistema. Ogni tipo di identificatore definisce i metodi che un risolutore deve applicare per fornire record per quel tipo specifico. Ad esempio, l'indirizzo di una determinata criptovaluta potrà essere risolto in maniera differente rispetto ad un identificatore di indirizzo Ethereum.
Inoltre, è possibile definire nuovi tipi di identificatori in qualsiasi momento tramite il relativo processo di standardizzazione (ERC e poi EIP).

Il processo di risoluzione dei nomi in ENS consiste in due passaggi. Innanzitutto, dobbiamo interrogare il **registry** per il risolutore responsabile di un nome particolare. In seguito, tutto ciò che si deve fare è chiedere al risolutore corretto la risposta alla richiesta.

9.4.2 La governance di ENS

La *ENS Foundation* ha tre direttori: Nick Johnson, Brantly Millegan e Kevin Gaspar, tutti membri del core team di ENS. Gli amministratori sono responsabili della gestione quotidiana della fondazione ma ciò che effettivamente regola l'ENS è la DAO (o "Consiglio"). Attraverso la DAO di ENS si può votare per:

- Nominare o rimuovere un amministratore, membro o supervisore.
- Proibire l'ammissione di membri in futuro.
- Gestione dei beni della fondazione.
- Sebbene non sia specificato direttamente nello Statuto, la DAO può anche istruire gli amministratori ad agire per conto della Fondazione, come la firma di un contratto, l'assunzione di una società per un servizio richiesto dalla DAO o la delega di alcuni dei poteri degli amministratori a un Gruppo di lavoro DAO.

9.5 DAO: Decentralized Autonomous Organization

Le tecnologie basate su Blockchain come visto nei capitoli precedenti sono sempre più presenti nel mondo di oggi. Le **DAO** (al singolare: Decentralized Autonomous Organization), stanno prendendo sempre più importanza e visione sia nel mondo accademico che nei vari governi.
In questo capitolo verranno affrontate le principali caratteristiche di una DAO ed anche cosa spinge investitori e sviluppatori a portare avanti questo tipo di applicazione della tecnologia blockchain. Inoltre si vedrà come la nascita delle DAO non sia stata supportata totalmente per problemi dovuti ad un attacco hacker e come invece oggi si parli di scalabilità e applicazioni in molti settori, dall'uso nell'IoT fino a trovare una sua applicazione in economia e governi.

9.5.1 Definizione di DAO

La DAO è stata originariamente introdotta da Vitalik Buterin nel paper [12] ed è stata definita come un organizzazione costruita su smart contracts che si possono eseguire autonomamente.
A differenza degli enti tradizionali centralizzati, nella DAO non vi è un ente centrale che controlla o dirige l'organizzazione.

In sostanza una DAO si compone di molteplici smart contracts che traducono in codice regole prestabilite dell'organizzazione alle quali nessuno potrà rifiutarsi nel momento in cui entra a farne parte. Questa caratteristica è ciò che rende la DAO affidabile da parte degli utenti o investitori che non necessitano di fiducia o affidabilità tra di loro. Il meccanismo fondamentale della DAO è, dopo aver stabilito attraverso contratti le regole, accumulare token di governance per poi votare o proporre nuove iniziative alla DAO stessa.

Come vedremo la DAO può essere costruita anche su blockchain diverse da quella di Ethereum dato che vi sono concetti, metodologie di lavoro e applicazioni che se applicate correttamente mantengono lo stesso livello di fiducia e organizzazione.

La prima DAO

Come anticipato nell'introduzione del capitolo, la prima DAO sviluppata ha riscosso inizialmente molto successo ma poi è stata subito soggetta ad un attacco hacker che ha abbassato notevolmente la credibilità e fiducia sia della rete Ethereum che della relativa criptovaluta.

Il progetto **The DAO** è iniziato nell'Aprile del 2016 con più di 11'000 membri raggiungendo i 150 milioni di dollari rappresentando il più grande progetto di crowfounding della storia.
Il progetto rappresentava un successo in quegli anni subito fermato il 18 di Giugno quando un hacker, attraverso chiamate ricorsive a una funzione ben strutturata ma mal implementata, riuscì a recuperare Ether dai token venduti dalla The DAO. L'hacker è riuscito ad ottenere 3.6 milioni di ETH e le perdite, al tempo, furono stimate sui 70 milioni.
I problemi del **The DAO** hanno avuto un impatto negativo sia sulla rete Ethereum che sulla sua criptovaluta. Alla fine, il fondo rimasto è stato restituito agli investitori come se l'organizzazione non fosse mai esistita (approfondimento nel capitolo inerente alla sicurezza nella sezione attacchi all' infrastruttura).

9.5.2 Tecnologie e applicazioni della DAO

E' inevitabile che la DAO abbia avuto notevoli sviluppi nei settori della supply chain, salute, IoT, privacy e gestione di dati [26], [19], [16], [27]. Successivamente anche nel settore crowdfunding e contabilità fino ad arrivare alle auto elettriche e ai sistemi di fatturazione delle stazioni di ricarica delle auto. I vantaggi che la DAO può portare sono la chiarezza nella governance aziendale per poter rendere partecipi tutti alle vicende che riguardano l'organizzazione oppure eliminare intermediazioni attraverso codice e connettività peer-to-peer.
In più, chi detiene i token relativi alla DAO, è interessato allo sviluppo e a portare valore all'organizzazione.

9.5.3 DAO come nuova metodologia di collaborazione?

Ad oggi molte aziende stanno investendo risorse per sviluppare programmi basati su blockchain. Nella governance attuale, tutte le aziende siglano dei contratti che hanno efficacia dipendente dal sistema giuridico nazionale in cui ci si trova.
A differenza dei contratti tradizionali, l'efficacia delle organizzazioni distribuite dipende da come è stato definito il codice e quali sono gli algoritmi presenti in uno smart contract. La blockchain attraverso la DAO potrebbe essere considerata una delle forme di governance che sfrutta le potenze computazionali basate totalmente sul digitale.

Oggi molte aziende stanno investendo risorse per sviluppare e implementare programmi basati su blockchain. Nella tradizionale governance dei contratti, l'efficacia dipende dal sistema giuridico nazionale. Al contrario, la governance DAO non dipende direttamente dai sistemi legali esterni. L'applicazione in una blockchain si ottiene attraverso il codice e gli algoritmi come i contratti intelligenti.

Quindi la blockchain può essere considerata la prima forma di governance che sfrutta veramente le capacità computazionali e basate sui dati della tecnologia digitale, ben oltre le tradizionali forme di governance sociale. DAO, come altri meccanismi di governance, non governa ugualmente bene tutti i tipi di transazioni e come sottolinea anche Lumineau et al. [14] la governance attraverso un sistema DAO può ridurre i tempi di ricerca, monitoraggio e applicazione ma comprende costi di progettazione molto elevati.

9.5.4 Conclusione

La DAO è sicuramente un paradigma molto promettente per le future soluzioni organizzative, si possono trovare altre applicazioni della DAO attraverso la blockchain Ethereum che riguardano tutti i settori elencati nella sezione precedente. La costituzione di diverse DAO nel mondo è sempre più in crescita e nel documento [18] si pensa si possa arrivare anche ad un eGOV-DAO ovvero un governo totalmente basato su smart contract che si auto-regola.

Capitolo 10

Interoperabilità tra blockchain

10.1 Bridges

Ora che attraverso la tecnologia blockchain si riesce a mantenere tutti i dati in maniera sicura, trasparente e accessibili a tutti i partecipanti della rete l'obiettivo è quello di interfacciare più di una blockchain affinché queste collaborino e riescano ad operare come se fossero un'unica blockchain. Come è noto esistono molteplici tipi di blockchain e spesso non ve n'è una migliore di altre ma semplicemente vengono diversificate sulla base dell'implementazione o di scelte progettuali.

Con interoperabilità intendiamo la possibilità di due o più blockchain di scambiarsi informazioni.
Il fattore di interoperabilità sicuramente è uno dei fattori che rallenta l'adozione di massa di questa tecnologia, basti pensare che quando si parla di applicazioni su blockchain inerenti a filiere produttive queste ultime sono composte da numerosi enti o società che possono fare uso di blockchain diverse dato che ognuno avrà la propria gestione personalizzata dei dati. [28]

Proprio per queste motivazioni è nata la necessità di far interagire le varie blockchain ed uno strumento indispensabile in questo caso è un **bridge**. Esistono diverse tipologie di bridge che verranno elencate e descritte mirando ad ottenere il maggior livello di flessibilità possibile. Con bridges si intendono dei sistemi che consentono lo scambio di dati tra due ecosistemi diversi e differenziabili in base al loro scopo. Essendo collegamenti tra due blockchain ed essendo quest'ultima una tecnologia che supporta un'infinità di funzionalità,

un bridge può variare dal semplice scambio con una singola altra blockchain sino ad arrivare ad un bridge globale nel quale tutte le altre blockchain sono supportate.

Si possono categorizzare le funzionalità dei bridge in 4 macrogruppi:

1. **Asset-Specific**: creati unicamente per consentire l'uso di una criptovaluta su blockchain esterne, questa tipologia è la più semplice da realizzare ed implementare ma limitata nelle funzionalità. Consente l'interoperabilità crittografica, ad esempio, il porting di bitcoin sulla rete Ethereum tramite il wrapping di BTC in Wrapped BTC, un token ERC20 compatibile con la rete Ethereum. Sono necessari perché ci si può trovare in situazioni nelle quali i meccanismi di consenso sono diversi.

2. **Chain-Specific**: un bridge tra due blockchain dedicato a operazioni di base come il blocco o lo sblocco dei token sulla blockchain di partenza e la creazione di wrapped token sulla blockchain di destinazione. Sono pratici, ma poco scalabili.

3. **Application-Specific**: bridge che permette di accedere a due o più blockchain col solo scopo di far funzionare l'applicazione stessa. Utilizzata nelle filiere nelle quali sono presenti software che devono interagire tra di loro.

4. **Generalized-Bridge**: protocollo pensato per il trasferimento di informazioni attraverso più blockchain. Una singola integrazione rende il progetto accessibile dall' ecosistema connesso al bridge, causando così un effetto di rete, ossia l'aumento dell'adozione da parte degli utenti del progetto. Il rischio in questo caso è che i progetti sacrifichino la sicurezza e la decentralizzazione per massimizzare l'effetto di scala. Un esempio di bridge di questa tipologia è Chainlink.

10.1.1 Metodologia di lavoro dei bridge

Le tipologie di bridge sono differenti ma ognuno di essi ha dei funzionamenti di base ai quali deve sottostare per poter funzionare correttamente.
Quando si scambiano informazioni tra due blockchain bisogna fare particolare attenzione ai dati e soprattutto alla loro consistenza. Dal punto di vista leggermente più tecnico gli elementi di cui un bridge deve disporre sono il monitoraggio, possibilità di firma crittografica, meccanismo di consenso e possibilità di scambiare messaggi tra le due blockchain.
Attraverso questi elementi che si trovano nelle principali soluzioni bridge analizzate, si riesce a costituire un passaggio di dati mantenendo le caratteristiche richieste dalla blockchain con il vantaggio di averne messe in comunicazione diverse.

Le principali soluzioni di bridge che comprendono gli elementi sopra citati sono 3:

1. **External validators and Federations**
2. **Light Clients e Relays**
3. **Liquidity Network**

Nel momento in cui avviene un passaggio di dati attraverso un bridge che fa uso di *validatori esterni* questi ultimi hanno il compito di verificare che l'operazione avvenga atomicamente e che non vi siano azioni nel frattempo che modificano le quantità di token associati all'indirizzo mittente.
Il gruppo di validatori esterni monitora l'indirizzo sulla blockchain di partenza e, raggiunto il consenso, esegue un'azione sulla blockchain di destinazione.

Solitamente nello scenario più comune come trasferimento di token o criptovalute vengono bloccati i fondi nell'indirizzo di partenza e se ne copia il valore nella blockchain di destinazione. Questo modello è utilizzato anche nell'ecosistema Ethereum.

Per quanto riguarda invece il monitoraggio di eventi viene fatto uso dei *Light Clients and Relays*. Questi bridge funzionano attraverso smart contracts che vengono creati e messi in esecuzione sulle blockchain coinvolte. Con il fine di scatenare eventi sulla blockchain di destinazione generati da eventi accaduti nella blockchain di partenza i light clients monitorano la blockchain

di partenza e, nel momento in cui un evento si verifica, generano prove di eventi. Dopodiché trasmettono queste prove crittografiche ad light clients, ovvero altri smart contracts nella blockchain di destinazione.

E' un metodo molto funzionale e sicuro poiché si basa su smart contract per ogni destinazione i quali, però, richiedono molte gas fee per essere creati ed eseguiti. Questa metodologia è utilizzata ad esempio nei bridge di Polkadot.

Liquidity Networks:

Le *liquidity networks* sono delle vere e proprie pool di liquidità che si trovano sia nella prima che nella seconda blockchain coinvolta e in questo modo, quando si necessitano scambi di rispettivi token, i due pool facilitano queste transazioni abbattendone i costi.

10.2 Uso delle APIs in blockchain

Una **API** (o Application Program Interface) è un'interfaccia costituita tra due applicazioni e nella sua definizione generale non riguarda solo il mondo della blockchain ma ve ne trova un forte uso [7].
Ad esempio, l'utilizzo della sicurezza decentralizzata offerta dalle blockchain per verificare gli scambi di chiavi crittografiche, richiede una qualche forma di API. Inoltre, poiché le blockchain sono molto costose da utilizzare in fase di elaborazione (mining), possono essere utilizzate per scambiare token con enti terzi ai fini di affittare potenza di calcolo.
Etheroll è una di queste applicazioni decentralizzata che si basa sulla blockchain per garantire un bilanciamento di carico pari mantenendo un buon livello di decentralizzazione.
Un'altra applicazione delle API blockchain per la sicurezza è per la gestione della logistica della supply chain (approvvigionamento merci) in VeChain che rappresenta una delle migliori blockchain e utilizza le API per assicurarsi che il prodotto sia autentico e tracciabile in tempo reale.

Questa tecnologia nell'ambito delle criptovalute è forse ancora più importante che nelle supply chain o nelle Proof of Work dato che gran parte del valore delle criptovalute stesse dipendono da come e quando è possibile utilizzarle e che quindi, un utilizzo di API potrebbe aumentare l'adozione della

criptovaluta e di conseguenza il prezzo. Inoltre per le API nelle criptovalute si deve far particolare attenzione a mantenere il livello di privacy e sicurezza alto e costante oltre a garantire una facilità d'uso anche per gli utenti meno esperti.

Un altro settore significativo è il trading. Essendo Ethereum e Bitcoin considerate come titoli di borsa attraverso le API si riescono ad estrapolare moltissime informazioni utili per svolgere al meglio le attività di trading. La maggior parte degli scambi crittografici utilizza proprio specifiche API.

Esistono anche aggregatori di dati che raccolgono informazioni da tali scambi e le trasmettono agli utenti tramite la propria API. Un esempio quasi banale potrebbe essere un'applicazione che, automaticamente, raccoglie le transazioni effettuate da tutti gli account di trading forniti per poi generare un report unico per tutti gli exchange favorendo l'utente finale (report utile poi per eventuali bilanci).

Uso di APIs nell'e-commerce

Uno degli usi più diffusi ad oggi è quello nel settore dell'e-commerce. Chiaramente chi possiede un sito di e-commerce punta a soddisfare nei migliori dei modi i clienti accontentando anche chi preferisce effettuare pagamenti in criptovalute piuttosto che valute fiat.
Anche in questo scenario l'integrazione delle API fornisce soluzioni al problema ma in questo caso si deve tenere conto di qualche fattore in più.

Innanzi tutto esistono miriade di tipologie di API diverse e sicuramente i fattori che bisogna valutare preventivamente come la tecnologia usata, la scalabilità del progetto e il budget. Altro fattore fondamentale è la tipologia di criptovaluta che si vuole accettare: la maggior parte delle API disponibili supporta transazioni in Bitcoin ma se il commerciante volesse accettare anche altre tipologie di criptovalute dovrebbe adottare l'API che supporta le funzionalità per esse.

Capitolo 11

Sicurezza

11.1 Perché proteggersi

Nel settore blockchain gli attacchi hacker non sono una novità e l'informazione non suscita scalpore essendo un settore nel quale grandissimi movimenti di denaro sono all'ordine del giorno.
Pur essendoci grandi strati di crittografia anche le più grandi infrastrutture hanno subito attacchi come quello velocemente introdotto nel capitolo della DAO ma gli attacchi all'infrastruttura non rappresentano l'unico pericolo da cui difendersi.
Nell'ambito delle criptovalute anche gli attacchi ai singoli utenti sono molto sviluppati essendo un ambito nuovo e nel quale le vulnerabilità si riescono a trovare più facilmente sugli utenti piuttosto che sull'infrastruttura. Un' altra caratteristica per la quale gli attacchi agli utenti sono di più facile effettuazione, motivazioni date nella sezione del DDoS (Distribuited Denial of Service), è che è necessaria molta meno potenza computazionale.

Gli attacchi rivolti agli utenti sono principalmente quelli che mirano a carpire chiavi private di wallet salvate all'interno dei pc in modo che i malintenzionati abbiano la possibilità di trasferire denaro dal wallet collegato al proprio. Questo è dovuto al fatto che la sicurezza della maggior parte delle criptovalute è alta ma la sicurezza dei portafogli e degli account attorno alle criptovalute rimane abbastanza bassa.
Mentre gli attacchi all'infrastruttura sono per lo più teorici e difesi attivamente, la falla nella sicurezza di Bitcoin, Ethereum e di qualsiasi altra

criptovaluta è il fatto che gli uomini non sono così esperti nel proteggersi contro tali attacchi.

11.2 Introduzione su possibili attacchi: infrastruttura o utente

Quando si parla di attacchi alla blockchain è bene suddividere in due macrosezioni le tipologie di attacchi effettuati dagli utenti malintenzionati. Come anticipato in una rete blockchain gli **attacchi** possono essere rivolti sia all'**infrastruttura** che agli **utenti finali**. [1]

Gli attacchi rivolti all'infrastruttura sono quelli che mirano a modificare il funzionamento tipico della rete attraverso l'alterazione del comportamento di uno o più componenti di rete. Uno degli esempi che verranno analizzati nella sezione successiva è l'attacco "DDoS" oppure il "Majority attack", tutti e due attacchi all'infrastruttura con due obiettivi diversi: nel primo si vuole ottenere un blocco del servizio e nel secondo invece un ritorno economico attraverso una doppia spesa della moneta coinvolta. In maniera più approfondita si potrebbe parlare di BGP hijacking attack e di eclipse attacks, altre due metodologie di attacchi rivolti all'infrastruttura e specificatamente alla metodologia di validazione e alla crittografia a curve ellittiche.

Gli attacchi rivolti invece agli utenti sono quelli che solitamente vedono come obbiettivo il denaro dell'utente finale. In questo caso gli attacchi sono veramente di ogni tipologia e, nei casi più sofisticati, possono comprendere ciò che viene definita ingegneria sociale. Nella sezione successiva verranno introdotti i principali attacchi rivolti agli utenti solitamente effettuati con il fine di carpire chiavi private ed ottenere la quantità di monete stipate nei wallet a loro collegate.
Gli attacchi di questo genere possono differire veramente tanto tra di loro e sicuramente le metodologie successivamente illustrate non sono le uniche che riconducono allo stesso fine. Si vedrà come vengono effettuati alcuni attacchi come Keylogger, Man in the Middle e PoisonTap.

Inoltre ogni tecnologia introdotta porta con sé novità e nuove applicazioni e servizi ma anche nuove vulnerabilità che devono essere scoperte prima di essere corrette. Un esempio è la blockchain 2.0 che introduce crimini attraverso smart contracts (CSC: *Criminal Smart Contract*) e vulnerabilità.

11.2.1 Analisi principali attacchi alle infrastrutture

Attacco DDoS

Con attacco DDoS viene inteso quel tipo di attacco portato a termine da un gruppo di macchine collegate in rete e diretto all'infrastruttura della blockchain. Essendo di natura decentralizzata gli attacchi alla blockchain possono riguardare più componenti della rete stessa. L' attacco DDoS rivolto a un nodo di rete oppure a un determinato componente della rete mira a paralizzare e rendere inutilizzabile il componente stesso.

Supponendo l'attacco ad un nodo di rete, questo avviene attraverso numerosissime transazioni non valide inviate al nodo che, per poter validarle o meno, deve occupare risorse disponibili impedendo l'elaborazione di richieste legittime. Bisogna notare che il consumo di risorse volto a mantenere il servizio occupato è generato attraverso la creazione di transazioni su semplici client e questo procedimento non necessita di grandi potenze computazionali dalla parte dell'attaccante ma ne richiede molte al nodo in fase di validazione.

Essendo un attacco di semplice realizzazione è anche uno dei più comuni se non fosse per il fatto che non reca troppi danni alla rete ma ne aumenta drasticamente il congestionamento. Inoltre le reti blockchain data la loro vastità sono costantemente sotto attacco di tentativi DDoS ma le reti sono ad oggi progettate sin dall'inizio per risolvere questo tipo di problematica.

I danni recati da attacchi DDoS avvenuti con successo sono solo un arresto dell'attività di rete quindi del servizio.

Attacco "Majority Attack"

Questo attacco sfrutta in maniera significativa la natura decentralizzata della rete. Essendo composta da molti nodi con diverso hashrate la rete in

alcune situazioni potrebbe essere sbilanciata. Supponendo diversi hashrate, quindi potendo validare transazioni con diverse velocità, si può validare una transazione due volte generando la famosa problematica del double spending.

Questa problematica consiste nel creare due transazioni con le stesse monete come se effettivamente potessero essere spese due volte. Viene effettuato mandando in rete la prima transazione e ricevendo il bene per la quale è stata eseguita dopodiché spedire un'altra al nodo con hashrate molto più elevato degli altri in modo che vi siano decisamente più probabilità che la seconda transazione venga validata prima della precedente, creando cosi la situazione desiderata dall'attaccante: ricevere il bene della prima transazione mentre si è in attesa del bene della seconda.

Ottenere una maggioranza di hashpower non consentirebbe ad un utente malintenzionato di creare monete, accedere a indirizzi o compromettere la rete in qualsiasi altro modo, il che limita il danno di questo attacco ma uno dei grandi effetti collaterali potrebbe essere la perdita di fiducia nella rete aggredita e un successivo drastico calo indotto nel prezzo delle attività di qualsiasi token sulla rete.

Questo tipo di attacco richiede all'attaccante una potenza computazione non da poco che richiede un dispendio di energie e denaro molto elevato. Proprio per questa ragione le criptovalute non sono preoccupate particolarmente data anche la ovvia conseguenza: un attaccante che ha speso molto per la realizzazione dell' attacco e che ottiene una determinata criptovaluta non vorrebbe mai che il prezzo di quest'ultima crollasse per un eventuale notizia sull'attacco.

The DAO Attack

Come già anticipato nell'introduzione di questo capitolo il più grande vettore di errori per gli attacchi informatici è l'umano. L'esempio più significativo di errore umano è quello che è stato anticipato anche nel capitolo della Ethereum DAO che, vista la gravità, ha portato alla creazione di una nuova criptovaluta.
L'idea sulla quale era costituita questa specifica organizzazione autonoma decentralizzata era rendere possibile a tutti l'investimento nell'azienda e poi

votare in maniera totalmente trasparente i progetti che volevano in automatico, il tutto gestito in modo sicuro dal codice del contratto intelligente. Per chi decideva di ritirarsi inoltre vi era la possibilità di riconvertire i token DAO utilizzati in questo meccanismo in Ether, in maniera tale da recuperare l'investimento e poter riutilizzare la stessa quantità di moneta investita come si preferiva.

Questo meccanismo è chiamato "Split Return" ed è stato sfruttato dall'attaccante per mettere a punto il suo attacco. Consiste nel restituire la quantità corretta di Ethereum al titolare del token che li ha resi, prima di incassare i token e registrare la transazione sulla blockchain per aggiornare il saldo del titolare del token DAO.

L'hacker riuscì a ripetere il primo passo senza passare al secondo, il che consentì di sottrarre circa $50 milioni di Ethereum dalla rete. Questo ovviamente contribuì ad un abbasso del prezzo di Ethereum dovuto alla fiducia della rete compromessa.

11.2.2 Analisi principali attacchi agli utenti

Man in the middle

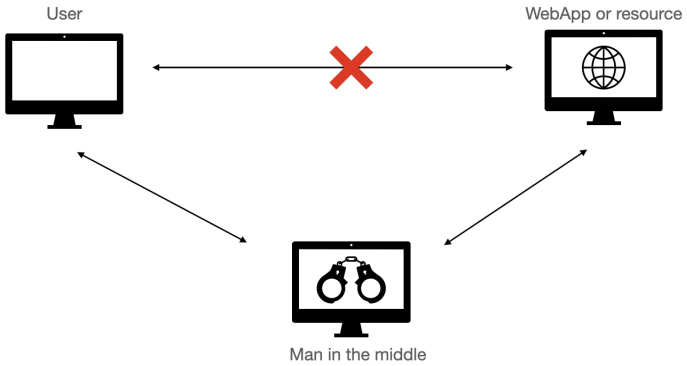

Figura 11.1: How the Man-in-the-middle attack works

L'attacco man in the middle (MITM) è un attacco informatico nel quale un intruso vuole intercettare le comunicazioni fra due utenti e si interpone tra loro. Con questo attacco il nemico può modificare e creare nuovi messaggi fingendosi una delle due parti coinvolte.
Per attuare questo attacco infatti l'intruso deve poter interporsi tra le due parti e, supponendo uno scenario nel quale un utente *mitt* vuole spedire un messaggio m all'utente *dest*, questo avviene nel momento in cui la connessione tra i due utenti viene stabilita. Quando un utente spedisce all'altro utente la sua chiave pubblica l'intruso sostituisce la chiave pubblica dell'utente *mitt* con la propria. In questo modo tutti i messaggi che il mittente spedisce vengono criptati con la K$enemy[pub]$ e l'intruso può decriptarli senza problemi. Dopodiché l'intruso recapiterà il messaggio all'utente *dest* che, senza accorgersi di nulla, crederà di aver ricevuto il messaggio dall'utente *mitt* corretto.

Ad oggi, buona parte dei protocolli fondamentali come TLS, autenticano una o entrambe le parti così da prevenire questo genere di attacchi. L'autenticazione avviene tramite certificati rilasciati da Certificate Authority (CA).

Keylogger: hardware o software

Con Keylogger intendiamo uno strumento hardware o software attraverso il quale è possibile carpire tutto ciò che viene digitato su una tastiera. Vi sono molteplici tipologie di keylogger ma in questa sezione ne analizzeremo solamente 2: **hardware** e **software**.

Il keylogger hardware consiste in uno strumento da collegare al computer oppure all'interno della tastiera stessa. Il dispositivo, attraverso un circuito che rileva i segnali corrispondenti ai tasti utilizzati, memorizza ogni carattere premuto sulla sua memoria interna oppure ne invia i dati in rete. Una peculiarità del keylogger hardware è che non è rilevabile dal SO della macchina non essendoci un software collegato rendendolo invisibile anche agli antivirus.
Inoltre il vantaggio del keylogger hardware risiede nel fatto che, dato che sono completamente indipendenti dall'SO, sono in grado di carpire anche password di bootstrap, la cui digitazione avviene in fase di avvio e prima del caricamento del sistema operativo, quindi non rilevabile via software.
Una misura di sicurezza da tenere in considerazione è utilizzare in maniera molto limitata i computer messi a disposizione in ambienti pubblici come biblioteche o internet point considerandoli non affidabili e non inserendo quindi credenziali o dati sensibili.

Un keylogger invece di tipo software consiste in vero e proprio codice creato appositamente per rilevare la sequenza di tasti premuti dall'utente sulla tastiera. Anche in questo caso i dati memorizzati possono essere estrapolati fisicamente oppure via rete dal computer.
Spesso sono utilizzati anche all'interno di aziende ai fini di monitoraggio delle attività svolte in rete senza che gli utenti o dipendenti ne siano a conoscenza.
Sono stati creati keylogger appositi per smartphone e tablet appositi per SO come Android o iOS, in grado di monitorare anche la cronologia di navigazione web, i testi inseriti, il registro delle chiamate, l'utilizzo delle applicazioni ed il segnale GPS.

PoisonTap e Coockie Siphoning

Tra questi tipi di attacchi elencati forse il PoisonTap è uno di quelli più complessi a livello tecnologico anche se sfrutta principi base molto semplici.

Ideato da Samy Kamkar è l'unico attacco introdotto in questo capitolo che fa uso di una backdoor (letteramente porta sul retro) installata sul client e nel settore blockchain questo tipo di attacco potrebbe portare a conseguenze molto gravi.

L'attacco necessita di un piccolo hardware e di una memoria SD per poter usufruire del sistema operativo scelto e dell'exploit. L'operatività si divide in due fasi, il PoisonTap vero e proprio e successivamente una seconda fase di cookie siphoning.
Quando l'hardware viene connesso al pc via usb viene riconosciuto come una scheda di rete ethernet e, siccome probabilmente è già connesso a internet, imposta la rete dell'hardware collegato come "low priority". Quando una macchina si connette a una nuova rete vengono scambiati dei pacchetti con alcune informazioni: il router ad esempio comunica il suo indirizzo IP e dove il computer può trovare i DNS, ma soprattutto quali sono gli indirizzi IP disponibili nella rete locale (normalmente in una rete classe C privata, da 192.168.1.1 a 192.168.1.255). Tutti gli altri indirizzi IP saranno quindi identificati come "esterni".
Inoltre, qualsiasi sistema operativo gestisce le connessioni in modo che, per la risoluzione di determinati indirizzi IP, ne effettua la ricerca prima nella rete locale nella quale eventualmente il computer è connesso (così facendo sfrutta sia la "high priority" che la "low priority"). Viene sfruttata proprio questa caratteristica della risoluzione: quando l'hardware viene connesso al computer, comunica che la sua rete locale ("finta") ha tutti gli IP da 1.0.0.1 a 255.255.255.255: quello che succede, quindi, è che il computer userà la rete "finta" e non quella "vera" per connettersi a tutti gli indirizzi IP esterni, perché nella rete "finta" tutti gli indirizzi da risolvere figureranno come locali.

Alla seconda fase ci pensa il computer a patto che ci sia un browser connesso a qualche sito con della pubblicità, un'estensione Chrome o qualsiasi componente che richieda al browser di contattare un server: appena questo si verifica, la richiesta verrà intercettata dall'hardware collegato (perché il

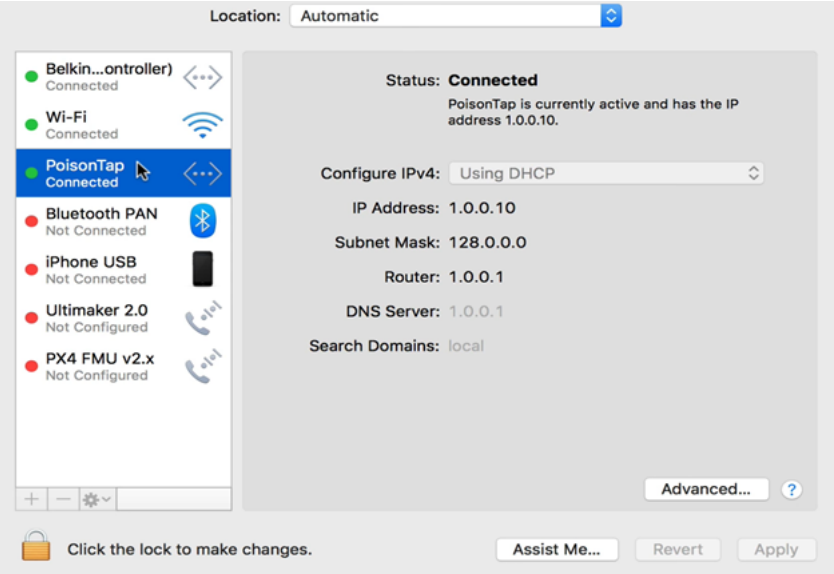

Figura 11.2: The PoisonTap low priority network

computer pensa che sia rivolta a un indirizzo locale della rete finta) che risponde con del codice che è il vero e proprio exploit. Il componente infatti esegue quasi sicuramente il codice che gli è stato mandato creando tanti iframe in ognuno dei quali viene aperto un sito diverso.

La caratteristica semplice ma che consente di recuperare eventuali cookie è dovuta al fatto che per alcuni di questi siti il browser ha una sessione aperta (quelli visitati in passato dall'utente nei quali aveva fatto il login e non logout) e quindi per farsi riconoscere dal sito e saltare la fase di login manderà i cookie di autenticazione salvati in locale, che però non arriveranno mai a destinazione, ma verranno salvati dall'hardware. Tra gli iframes che vengono aperti ne vengono inseriti alcuni collegati ad una backdoor e finché rimane aperta la tab che contiene gli iframes (invisibili), anche se l'hardware sarà disconnesso dal computer, quella backdoor continuerà a funzionare e potrà essere usata da remoto ad esempio per effettuare log di dati o connettersi alla rete locale (la "vera" rete locale continua ad essere disponibile per il computer perché gli indirizzi IP locali sono già mappati al suo interno). Si può quindi portare a termine altri attacchi anche sugli altri host connessi oltre a carpire tutti i cookie di accesso dell'utente: se componiamo richieste con i cookie scovati, si potrà accedere a tutte le informazioni dell'utente perché il sito relativo riconoscerà quella sessione come già autorizzata.

Per quanto riguarda il meccanismo di difesa principale da questo attacco all'utente, oltre a chiudere il browser e spegnere il pc quando non lo si utilizza o non lo si ha a portata, si trova soltanto lato developer quando si deve impostare la flag secure del coockie a "true" nel sito che si sta sviluppando.

11.3 CSC: Criminal Smart Contract

Grazie al loro anonimato e eliminazione di intermediari, le criptovalute hanno rappresentato sin da subito la crescita di nuovi modelli di business. Sfortunatamente alcuni criminali hanno sin da subito compreso le potenzialità delle criptovalute e con l'avvento di Ethereum, quindi possibilità di creare smart contract, anche di questi ultimi nel mondo della criminalità.

In quest'ultima sezione verrà introdotto come il rischio dei **CSC** (Criminal Smart Contract) sia da tenere in considerazione nel mondo della blockchain e in particolare di come, seppur in maniera molto complessa, si potrebbe insidiare nel mondo della blockchain. [11]

L'utente malintenzionato che vuole creare un CSC necessita di sottostare a diverse regole e deve affrontare determinate sfide. Nella prima fase sicuramente si deve saper valutare se un determinato crimine può avvenire o meno attraverso uno SM (Smart Contract) e in secondo luogo dovrà valutare il sistema di remunerazione che in questo caso viene chiamato **commission-fair** [11], intendendo quel modello per cui l'esecuzione del contratto garantisce sia la commissione di un reato che un compenso commisurato all'autore del reato o niente a nessuno dei due.

I CSC possono quindi essere piuttosto delicati e anche se in linea di principio è possibile costruire un CSC, dati i codici esistenti, non è chiaro se il CSC può essere eseguito. Non si è certi poiché in alcuni casi il CSC può essere eseguito ma con sforzi computazionali inevitabilmente dispendiosi significando commissioni di esecuzione troppo elevate addebitate al CSC stesso.

Per un ulteriore approfondimento su come un CSC può facilitare la fuga di informazioni riservate o commettere vari crimini del mondo reale si riporta al documento [11].

11.4 Alcuni esempi di attacchi

11.4.1 Enigma Attack

Enigma, una piattaforma di investimenti decentralizzata, è stata attaccata nell'Agosto del 2017 sfruttando la vulnerabilità della password debole. Attraverso mezzi di ingegneria sociale, l'attaccante ha rubato con successo la password di un fondatore di Enigma, che era stata utilizzata anche all'interno di Enigma. Di conseguenza, l'attaccante ha preso il controllo dei canali della società, dei registri di e-mail e account Google che mantenevano i token da distribuire nelle pre-sale.
L'attaccante ha sostituito l'indirizzo ufficiale del contratto ICO con un suo indirizzo e inviato messaggi per sollecitare gli acquirenti in false prevendite così da incassare quei token.

11.4.2 MyEtherWallet Attack

MyEtherWallet è un wallet web-based ed è stato attaccato nell'Aprile del 2018 sfruttando delle vulnerabilità dei messaggi BGP inaffidabili, delle vulnerabilità del DNS e delle vulnerabilità di reindirizzamento URL.

L'attacco ha sfruttato un dirottamento BGP e DNS congiunto per indurre in errore gli utenti verso una versione falsa del sito per compromettere poi le chiavi private delle vittime. L'hacker ha quindi svuotato i portafogli delle vittime e ha rubato circa 17 milioni di dollari. Quando gli utenti hanno visitato il loro MyEtherWallet, le loro richieste sono state reindirizzate ai falsi server DNS controllati dall'attaccante, i quali hanno restituito l'indirizzo IP di un sito web di phishing. Gli utenti ignari, che hanno trascurato il segnale di avvertimento del certificato TLS/SSL, accedendo al sito hanno spedito le loro password e chiavi private all'attaccante.

Conclusioni

In questo elaborato è stato trattato il mondo della blockchain e alcune nuove applicazioni basate su di essa. E' stato introdotto il funzionamento generale della tecnologia ponendo attenzione ai principali problemi di consenso e tipologie di nodi che si possono trovare in un sistema decentralizzato.

Come in ogni sistema del genere bisogna evitare le problematiche di inconsistenza dei dati e non sottovalutare quelle di centralizzazione.

Per poter comprendere al meglio come questi tipi di problemi possano sorgere sono state confrontate le due principali metodologie di lavoro: PoW e PoS con un accenno alla DPoS, simile a Proof of Stake ma con deposito.

Dopo l'analisi delle possibili metodologie di lavoro, dei principali tipi di nodi e di come il meccanismo di consenso venisse rispettato all'interno di una blockchain, si è posta attenzione su come quella di Ethereum abbia rivoluzionato l'intero settore.

Il vantaggio di questo tipo di blockchain è che oltre alla criptovaluta correlata ha reso possibile la creazione di smart contract.
Attraverso gli smart contract e un linguaggio di programmazione a loro dedicato si possono certificare provenienze di prodotti, creare nuovi token attraverso una procedura standard o creare vere e proprie organizzazioni autonome senza necessità di fiducia o enti certificatori.

Una problematica degli smart contract è rappresentata dal costo di archiviazione in rete e dalle fee necessarie all'esecuzione che, nella blockchain di Ethereum, risultano abbastanza alte. Inoltre altre problematiche di sicurezza vengono attribuite alla nascita dei Criminal Smart Contract, contratti intelligenti dai fini malevoli scritti da malintenzionati.

Con il passare del tempo gli sviluppatori, come era avvenuto anche da

parte di Vitalin Buterik con Bitcoin, hanno creato alternative e miglioramenti agli smart contract di Ethereum come gli Hooks in XRPL che presentano meno fee necessarie all'esecuzione e risultano più prestanti.

Infine è stata analizzata l'interoperabilità tra blockchian ed approfondito l'ambito sicurezza: sia infrastrutturale sia relativa all'utente.

Ad oggi questa tecnologia si trova nei più vari settori: dalla finanza per questioni di trasparenza e fiducia alle supply chain di alcune grandi aziende. Il focus principale di questo elaborato è stato Ethereum, la blockchain rivoluzionaria che ha reso possibile tutto ciò. Sicuramente in futuro questa tecnologia sarà una delle più sviluppate ma anche una delle più inquinanti. Questa attenzione sviluppata e dovuta nei confronti dell'ambiente sicuramente si ripercuoterà sull'intero settore che potrebbe vedere modifiche non banali sulle metodologie di lavoro e sulle architetture; proprio in questo periodo Ethereum si sta convertendo in PoS.
Non è chiaro invece se la blockchain di Ethereum continuerà anche in futuro ad essere il riferimento per le applicazioni decentralizzate.

Bibliografia

[1] *Attacchi alla blockchain cause, conseguenze e contromisure.*

[2] *ERC-20 TOKEN STANDARD.*

[3] *ETHEREUM ACCOUNTS.*

[4] *ETHEREUM NAME SERVICE GOVERNANCE.*

[5] *OFFICIAL ETHEREUM NAME SERVICE DOCUMENTATION.*

[6] *The origin of Ethash.*

[7] *Site blockchain.com.*

[8] *Tweet Bitmain Tech per annunciare Antminer e3.*

[9] *Types of Nodes: Light Nodes, Full Nodes, and Masternodes.* July 2021.

[10] Dr. Gavin Wood Andreas M. Antonopoulos. *Smart Contracts in Mastering Ethereum: building smart contracts and dapps.* 2018.

[11] Elaine Shi Ari Juels, Ahmed Kosba. *The Ring of Gyges: Investigating the Future of Criminal Smart Contracts.* 2016.

[12] V. Buterin. *Ethereum white paper: A next generation smart contract decentralized application platform.* 2013.

[13] Siniša Ranđić Dejan Vujičić, Dijana Jagodić. *Blockchain technology, bitcoin, and Ethereum: A brief overview.* ResearchGate, March 2018.

[14] O.Schilke F.Lumineau, W.Wang. *Blockchain governance - a new way of organizing collaborations?* 2021.

[15] LAURENT NJILLA SHOUHUAI XU HUASHAN CHEN, MARCUS PENDLETON. *A Survey on Ethereum Systems Security: Vulnerabilities, Attacks, and Defenses.* June 2020.

[16] M.A.Vasarhelyi J.Daiand. *Toward blockchain-based accounting and assurance, pp. 5–21.* 2017.

[17] Marshall Pease Leslie Lamport, Robert Shostak. *The Byzantine Generals Problem.* SRI International, July 1982.

[18] HUAWEI HUANG ZIBIN ZHENG LU LIU, SICONG ZHOU. *From Technology to Society: An Overview of Blockchain-Based DAO.* 2021.

[19] S. Ferretti G. D'Angelo M. Zichichi, M. Contu. *Likestarter: A smart-contract based social DAO for crowdfunding, pp. 313–318.* 2019.

[20] Hartwig Mayer. *A Survey on Ethereum Systems Security: Vulnerabilities, Attacks, and Defenses.* CoinFabrik, Revised June 28, 2016.

[21] Andrew Miller. *Permissioned and Permissionless Blockchains. In Blockchain for Distributed Systems Security (pp. 193–204).* March 2019.

[22] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system.* SSRN Electronic Journall, 2008.

[23] R. Shuwar O. Vashchuk. *PROS AND CONS OF CONSENSUS ALGORITHM PROOF OF STAKE. DIFFERENCE IN THE NETWORK SAFETY IN PROOF OF WORK AND PROOF OF STAKE.* 2018.

[24] Rémy Prud'homme. *The dangers of decentralization.* August 1995.

[25] Qi Wang Shiping Chen Qin Wang, Rujia Li. *Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges.* arXiv, May 2021.

[26] J. L. King R. Beck, C. Müller-Bloch. *Governance in the blockchain economy: A framework and research agenda.* 2018.

[27] Y. Lee C. Lee S. Cho S. Jeong, N. Dao. *Blockchain based billing system for electric vehicle and charging station.* 2018.

[28] Philipp Frauenthaler Michael Borkowski Stefan Schulte, Marten Sigwart. *Towards Blockchain Interoperability.* 2019.

[29] A. Zohar Y. Sompolinsky. *Secure High-Rate Transaction Processing in Bitcoin in Financial Cryptography and Data Security.* Springer Berlin Heidelberg, 2013.

Ringraziamenti

Ringrazio il mio relatore, Margara Luciano, per avermi permesso di scrivere questa tesi sulla blockchain, per la disponibilità e per la pazienza.

Ringrazio mia mamma per aver letto la tesi e, pur non capendoci assolutamente nulla, corretto qualche parola e evidenziato punti critici.

Ringrazio i miei amici per avermi sostenuto durante questo percorso universitario e colgo l'occasione per ricordare che per altri due anni (spero) dovranno continuare a farlo. Un ringraziamento particolare a Matteo, un mio caro amico che da anni mi tiene aggiornato su tutte le ultime novità in materia di criptovalute, blockchain e non solo.

Ringrazio i miei colleghi di lavoro per avermi concesso, anche nei periodi più difficili, il giusto tempo da dedicare all'università e in particolare Giacomo che è sempre disponibile e fondamentale nel nostro lavoro.

Ringrazio i miei colleghi dell'università per il sostegno durante il mio percorso di studi con feste e bei momenti, in particolare Manuel, grande amico e persona sempre di buon umore.

Un ringraziamento particolare anche a Mattia, lui è un collega sia all'università che a lavoro oltre che grande amico e spero di poter continuare a studiarci e lavorarci insieme.

Ringrazio Kelvin e la sua ansia nei momenti di consegna ed anche Manuel, Alessandro, Giacomo, tutti amici prima che colleghi.

Ringrazio inoltre tutte le persone che mi sono state vicine e che mi hanno

supportato nel percorso universitario che non ho citato in precedenza ma che comunque in un qualche modo per me sono state speciali.

Alberto Paganelli