POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Informatica

Sviluppo di un sistema NFT su Ethereum

Tesi di Laurea Magistrale

Relatori

Prof. Danilo Bazzanella

Candidato

Lorenzo Marzolla

Anno Accademico 2025-2026

Indice

Introduzione

La presente dissertazione analizza un caso di studio relativo alla creazione di un marketplace di NFT sulla blockchain Ethereum. L'obiettivo di proporre una possibile soluzione di implementazione di smart contract per la gestione di asset digitali unici, superando alcune problematiche che presenta questo settore. Il progetto nasce anche da una collaborazione con l'azienda Young Platform, una delle principali realtà italiane operanti nel settore della compravendita di criptovalute.

Nella prima parte viene introdotto il concetto di blockchain, descrivendone l'evoluzione storica, le principali caratteristiche e i meccanismi di consenso che garantiscono le caratteristiche fondamentali di sicurezza e trasparenza. Successivamente, l'attenzione si concentra su Ethereum, sulla sua filosofia di computer decentralizzato e sul suo funzionamento tecnico. In particolare, vengono approfonditi i meccanismi di consenso e i protocolli per la gestione dei token fungibili e non fungibili, evidenziandone le differenze.

La trattazione prende anche in analisi la possibilità di rappresentare opere fisiche attraverso NFT, integrando l'utilizzo di firme digitali qualificate off-chain per attribuire valore legale e autenticità all'opera rappresentata. In questa sezione viene quindi introdotto anche il concetto di Certification Authority e che ruolo ha nel fornire le chiavi per la firma digitale.

Un ulteriore aspetto affrontato riguarda la conservazione dei dati associati agli NFT, tema cruciale per la loro persistenza e verificabilità nel tempo. In questa prospettiva vengono esaminate soluzioni basate su database distribuiti, come IPFS e Filecoin, approfondendone il funzionamento tecnico e le modalità con cui possono garantire un'archiviazione decentralizzata e resistente alla censura. Viene poi esplorata la possibilità di trasferire NFT tra diverse blockchain, introducendo i concetti di interoperabilità e cross-chain communication anche attraverso una breve analisi delle architetture Polkadot e Moonbeam.

Infine si propone un esempio di smart-contract che implementi le funzionalità descritte sopra, analizzando possibili criticità e relative soluzioni.

1. *Young Platform*

L'idea nasce nel 2017 da un gruppo di sei studenti di informatica dell'Università di Torino (Andrea Ferrero, Alexandru Stefan Gheban, Samuele Raimondo, Andrea Carollo, Marco Ciarmoli e Daniele Rinaldi) con l'obbiettivo di creare una piattaforma per vendere criptovalute in modo più semplice e sicuro. La start-up vera e propria viene fondata nel 2018 e ha presto successo con i primi investitori: in breve tempo riesce a raccogliere oltre €1M anche tramite campagne di crowdfunding e viene incubato presso l'I3P del Politecnico di Torino. Nel 2019 avviene il lancio dell'app marketplace.

L'idea alla base del progetto non era soltanto quella di realizzare una piattaforma sicura e facilmente fruibile per il commercio di criptovalute, ma anche di avvicinare gli utenti alla tecnologia della blockchain e ai diversi componenti che da essa possono essere sviluppati, nonché al tema degli investimenti.
In tal modo si è inteso rendere l'argomento accessibile anche ai meno esperti, consentendo loro di effettuare scelte più consapevoli e ponderate.
Tale obiettivo si collega anche al contesto nazionale, caratterizzato in Italia da una diffusa carenza di educazione finanziaria, che rende particolarmente importante promuovere strumenti e iniziative in grado di colmare questo divario.
Stando a dati aggiornati, "l'Italia è in piena emergenza educazione finanziaria. Solo poco più di 1 italiano su 10 (16,6%) ha competenze finanziarie minime accettabili, posizionando l'Italia al 36° posto su 39 Paesi di tutto il mondo." (Teleborsa, 2025)
Insieme alla piattaforma, quindi, viene lanciato il token Young, un utility token ERC-20 basato su Ethereum. Il token viene utilizzato come meccanismo di ricompensa per specifiche azioni compiute dagli utenti all'interno dell'applicazione.
In una prima fase, esso veniva assegnato come premio al raggiungimento di determinati obiettivi legati al numero di "passi effettuati". Successivamente, sono stati introdotti una serie di quiz dedicati alla tematica della blockchain, con particolare attenzione alle sue applicazioni in ambito finanziario, riprendendo così uno dei principi fondamentali della filosofia di Young Platform, ossia la diffusione della conoscenza e della consapevolezza in campo cripto-finanziario.
I token "Young" risultano inoltre spendibili all'interno dell'applicazione in diversi modi, incentivando ulteriormente la partecipazione e l'interesse degli utenti.
Il token così assume una doppia valenza: aumenta la fidelizzazione degli utenti, che possono spendere gli Young per ottenere vantaggi in app, ma continua a mantenere la sua utilità originaria, ovvero di educazione finanziaria in generale e più nello specifico alla tecnologia blockchain sempre in modo semplice e comprensibile.
L'applicazione ha subito numerosi aggiornamenti nel corso degli anni, che hanno comportato l'introduzione di nuove funzionalità, tra cui un ranking di livelli disponibile

in fase di iscrizione, ciascuno dei quali associato a specifici vantaggi per l'utente. Negli ultimi anni si è posizionata come uno dei principali exchange italiani.

Di seguito sono riportati alcuni punti di svolta della storia aziendale:

- *28 giugno 2021*: Series A da €3.5M guidata da United Ventures
- *Gennaio 2022*: Acquisizione / acqui-hiring di Arithmos Trading Ltd
- *4 Gennaio 2023*: ottiene la registrazione come **PSAN/DASP** presso l'Autorité des Marchés Financiers. Questo ha permesso l'operatività regolamentata in Francia
- *13 Giugno 2022*: Round Azimut – importante crescita
- *4 Gennaio 2023:* Espansione regolatoria: registrazione/licenza in Francia. Young Platform ottiene la registrazione come **PSAN/DASP** presso l'**AMF** (Autorité des Marchés Financiers). Questo ha permesso l'operatività regolamentata in Francia.

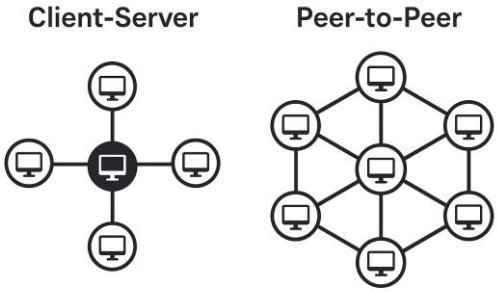
2. *Blockchain*

2.1 Cos'è la Blockchain?

Negli anni '90 iniziarono a emergere studi e progetti dedicati allo sviluppo di soluzioni decentralizzate per la creazione di valute digitali concepite per operare senza l'intervento di alcuna autorità di controllo o organismo di regolamentazione. Le prime opere teoriche relative a un sistema basato su blockchain e sull'utilizzo della crittografia risalgono al 1991 e si evolsero progressivamente fino al 1998, anno in cui Wei Dai propose una soluzione decentralizzata per una valuta digitale interamente fondata sulla crittografia a chiave pubblica.
Ma la sua prima applicazione pratica viene descritta nell' Ottobre del 2008 con il paper di Satoshi Nakamoto "Bitcoin: a peer-to-peer electronic cash system", in cui viene introdotta l'idea di Bitcoin. Nakamoto intendeva creare un sistema di pagamento peer-to-peer slegato dal controllo di terze parti (banche o enti statali), eliminando la necessità di intermediari, abbassando i costi di transazione e garantendo la privacy degli utenti. Per fare questo è necessario un registro distribuito pubblico e immutabile che tenga conto di tutte le transazioni effettuate nel sistema: la blockchain, appunto. Partendo da questo concetto, la tecnologia blockchain è stata successivamente reinterpretata, adattata a diversi contesti e applicata a una vasta gamma di settori.
In generale una blockchain può essere definita come un database distribuito su più nodi, che memorizza le transazioni in blocchi concatenati tra loro in ordine cronologico, formando una catena in cui ogni nuovo blocco è collegato al precedente, proprio come gli anelli di una catena.

2.2 Peer-to-Peer

Per poter parlare del funzionamento della blockchain è fondamentale introdurre l'architettura di rete su cui questa si basa: il peer to peer. Questo modello si contrappone al modello di comunicazione Client-Server dove le informazioni sono centralizzate sui nodi server e i client devono comunicare con questo per recuperarle.
Nel modello peer to peer invece, la rete di comunicazione è formata da diversi nodi (i cosiddetti "peer") che si comportano alternativamente da server e da client. Non si hanno più comunicazioni dirette tra un server e tutti i client ma una rete di connessioni tra i vari peer. Ogni nodo ha gli stessi permessi e lo stesso livello gerarchico rispetto agli altri nodi della rete.



In assenza di server centralizzati, i nodi della rete devono coordinarsi tra loro per identificare gli altri nodi e determinare la collocazione delle informazioni. A tal fine, vengono impiegate due principali procedure:

- *discovery*: necessaria per conoscere gli altri nodi della rete
- *lookup*: usata per conoscere i contenuti dei nodi

Queste operazioni vengono gestite diversamente a seconda di come il sistema peer to peer è organizzato:

- *Peer-to-peer puro*: ogni nodo si occupa singolarmente di scoprire i nodi della rete, il loro contenuto e comunicare le proprie informazioni.
- *P2P con Discovery server*: è presente un server nella rete (o più, a seconda della dimensione) che si occupa di far conoscere tutti i peer della rete. Rimane però sempre a carico dei singoli peer la ricerca delle risorse nei vari nodi.
- *P2P con Discovery e LookUp server*: in questo caso il server, oltre a dare le informazioni sui nodi, trasmette anche la lista dei loro contenuti.

2.3 *Caratteristiche fondamentali*

Esistono differenti tipologie di blockchain, ciascuna con utilizzi diversi; tuttavia, è possibile individuare alcune caratteristiche comuni a tutti i sistemi.

a. Decentralizzazione: tutte le informazioni e i dati sono distribuiti su ogni nodo della rete. In questo modo si evitano manipolazioni da un'organizzazione centrale, i nodi hanno accesso a tutti i dati nella blockchain che vengono trasmessi tramite peer to peer.
b. Trasparenza: essendo i dati distribuiti e tutte le transazioni registrate, chiunque può vedere lo storico delle operazioni. A ciascun nodo viene inoltre assegnato uno pseudonimo alfanumerico, di oltre trenta caratteri, che consente l'autenticazione pur garantendo l'anonimato.

c. Immutabilità: una volta che un blocco viene inserito sulla catena non è più possibile rimuoverlo o modificarlo.
d. Sicurezza: la sicurezza sulle blockchain è garantita da algoritmi di crittografia e da meccanismi di validazione(algoritmi di consenso).

2.4 *Tipologie di Blockchain*

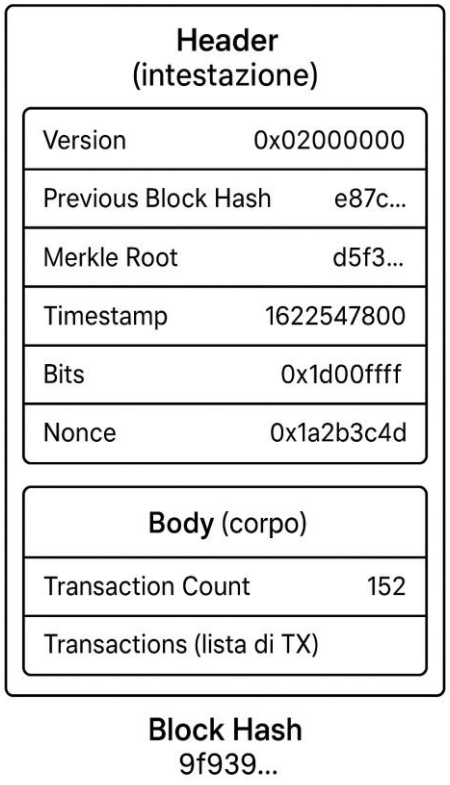
Esistono diverse tipologie di blockchain, distinte principalmente in base alle modalità di accesso. Di seguito vengono presentati i tre principali tipi:

- Public Blockchain: sono blockchain alle quali tutti i nodi della rete possono partecipare e validare le operazioni. Tra gli esempi più noti di questo tipo vi sono "Ethereum" e "Bitcoin". Non essendoci nessun genere di controllo su chi può partecipare alla rete si devono trovare dei metodi alternativi per garantire la sicurezza e impedire che un utente malevolo modifichi in modo incorretto la catena.
- Fully Private Blockchain: si tratta di blockchain in cui, nonostante i dati siano distribuiti e seguano la logica di accodamento nella chain, ogni modifica e validazione viene gestita in modo centralizzato.
- Consortium Blockchain: è la via di mezzo/una soluzione intermedia tra la blockchain pubblica ("opened") e la quella completamente privata ("fully private"). In questo tipo di rete c'è una parte di nodi preselezionati e garantiti che si occupano delle operazioni di validazione dei nuovi blocchi mentre tutti gli altri possono effettuare solamente la creazione. In tal modo la sicurezza è garantita dai granted node, poiché senza il loro consenso un'operazione, eventualmente fraudolenta, non viene accettata.

2.5 *L'anello della blockchain: il blocco*

Nella blockchain non è possibile eliminare o modificare un blocco già aggiunto, poiché essa costituisce una struttura dati di tipo *append-only*, nella quale i nuovi blocchi vengono esclusivamente aggiunti in coda alla catena. Tale caratteristica garantisce l'immutabilità dei dati e la sicurezza dell'intero registro distribuito. Una volta proposto, il blocco viene sottoposto al processo di validazione da parte della rete (secondo il meccanismo di consenso adottato). Solo dopo l'approvazione dei nodi partecipanti, il blocco viene effettivamente aggiunto alla blockchain, diventando parte permanente e immutabile del registro distribuito.
Ogni blocco è diviso in due sezioni: header e body.

Nel body si trovano le transazioni che vengono aggiunte alla catena mentre, l'header section contiene i metadati necessari per garantire l'integrità e la connessione del blocco con i precedenti. Tali componenti possono variare in funzione delle specifiche caratteristiche delle diverse implementazioni di blockchain (ad esempio, in base al meccanismo di consenso adottato); tuttavia, è possibile individuare alcuni elementi comuni presenti nella maggior parte delle architetture. In particolare, l'header include:



- *Timestamp* : data e ora della creazione del blocco.
- *Hash del blocco precedente*: garantisce il collegamento tra i blocchi, assicurando la continuità e l'immutabilità della catena. Ogni blocco è collegato al precedente attraverso un riferimento al suo hash, formando così una catena crittograficamente vincolata. Quando si desidera aggiungere un nuovo blocco, viene innanzitutto prelevato l'hash dell'ultimo blocco presente nella catena, che viene concatenato ai dati contenuti nel nuovo blocco. Successivamente, viene calcolato l'hash risultante, che rappresenta l'identificatore univoco del nuovo blocco e ne garantisce l'integrità.
- *Merkle Root*

2.6 Funzioni di Hashing

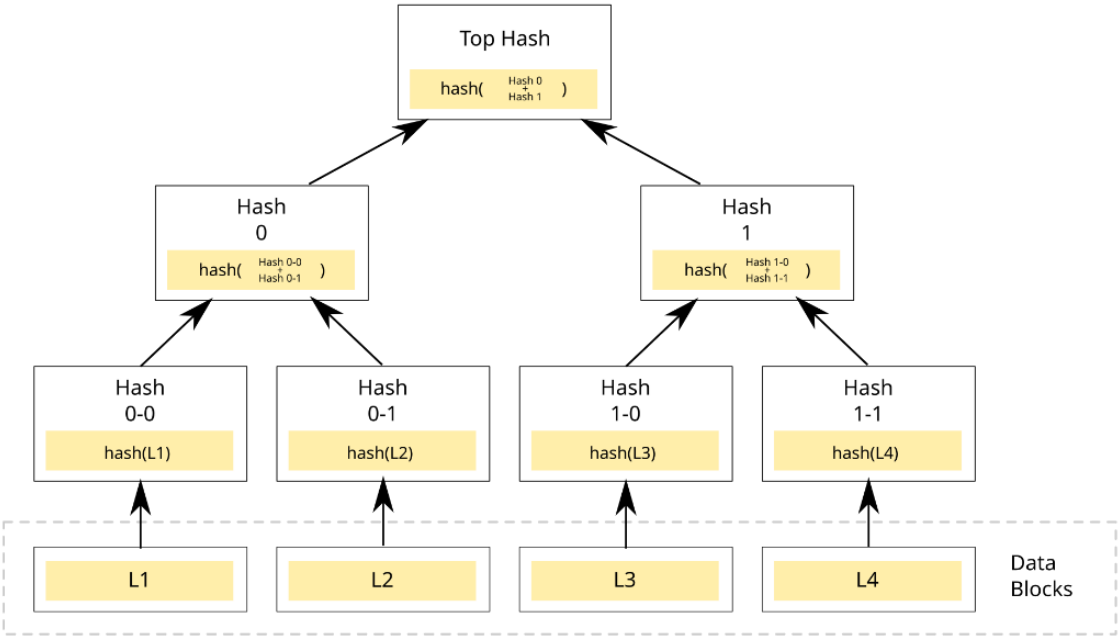
L'hashing è una tecnica di manipolazione delle stringhe che permette di ottenere da stringhe generiche una stringa sempre della stessa lunghezza. Questa è una tecnica fondamentale per il funzionamento delle blockchain.
Il risultato dell'hashing è chiamato "digest".
Perché una funzione di hashing sia considerata tale il suo digest risultante deve rispettare 5 regole:

a. Deterministica: per la stessa stringa in input si deve ottenere sempre lo stesso digest.

b. Resistenza alla pre-immagine: dato uno specifico digest non deve essere possibile recuperare la stringa di input che l'ha prodotto. La funzione di hash deve essere quindi irreversibile.
c. Resistenza alla seconda pre-immagine: data una stringa e il suo digest non deve essere possibile trovare un'altra stringa che genera lo stesso digest.
d. Resistenza alle collisioni: prese due stringhe casuali non deve essere possibile che, dopo averle elaborate con la funzione di hashing queste diano come risultato lo stesso digest.
e. Efficienza computazionale: generare l'hash non deve essere eccessivamente time e resource consuming.

2.7 Merkle Root

Per spiegare cos'è il merkle root è necessario introdurre il concetto di *merkle tree* e come questo si integra nelle blockchian.
Il *merkle tree* è una struttura dati ad albero in cui le foglie sono i digest di tutti i blocchi di dati che voglio rappresentare. Successivamente, ogni coppia di hash foglie viene concatenta e processata con funzione di hash, creando i nodi del layer superiore. Procedendo in modo iterativo, si arriva al *merkle root* ovvero il nodo radice dell'albero.



Nel blocco della blockchain ogni transazione viene sottoposta a funzione di hash diventano così le foglie dell'albero di merkle. Il merkle root viene dunque salvato negli header del blocco.
Questa struttura è fondamentale per verificare in modo rapido la validità delle transazioni nella catena. La *Merkle proof* permette di verificare con una complessità logaritmica tutti gli hash della tabella.

$$O(\log_2 N) \text{ dove } N \text{ sono il numero di transazioni.}$$

Il Merkle root è fondamentale per la verifica dell'immutabilità del blocco. Se infatti una qualsiasi transazione viene manomessa, il merkle root risultante sarà diverso e quindi rifiutato dagli altri nodi della rete. Tramite il merkle tree è possibile anche fare una verifica parziale dell'albero (Simplified Payment Verification) grazie alla sua struttura bilanciata. Questo è molto utile in un'ottica di performance e ottimizzazione della verifica.

2.8 Meccanismi di consenso

Essendo una rete distribuita, i vari nodi devono sincronizzarsi e concordare sulle differenze in modo che, alla fine, tutti possiedano la stessa versione. Per questo vengono utilizzati i consensus algorithm: questi permettono di validare un nuovo blocco in arrivo sulla blockchain. Qui di seguito è riportato il funzionamento dei due più famosi e utilizzati:

a) Proof of work

Questo algoritmo richiede che venga risolto un complesso problema di computazione. Coloro che operano per risolvere il problema si chiamano *miner*. Il primo *miner* che riesce a risolvere il problema lo comunica a tutti gli altri nodi che verificano la correttezza della soluzione. Se lo è allora, il *miner* si aggiudica la creazione del blocco,viene aggiunto alla catena e ottiene una ricompensa.
La risoluzione del problema comporta un gran lavoro computazionale (quindi consumo di energia elettrica) e molto time-consuming; questa complessità costituisce un efficace meccanismo di sicurezza. Un eventuale tentativo di inserire dati contraffatti all'interno della catena comporterebbe un ingente spesa di denaro e verrebbe immediatamente rifiutato. L'unico modo, infatti, per riuscire a far passare un blocco contraffatto è quello di controllare il 51% dei *miners*. Questo può diventare un problema se la rete blockchain non è sufficientemente ampia, poiché in tal caso ottenere il controllo del 51% delle risorse di validazione potrebbe risultare più facile.
Nonostante questo sia uno degli algoritmi più usati, come già anticipato, ha un impatto ambientale non indifferente per il grosso consumo di energia elettrica necessario. È poi un algoritmo che richiede parecchio tempo per convalidare le transazioni: la blockchain

di bitcoin, ad esempio, è attualmente in grado di elaborare 7 tps. Questo si ripercuote sulla scalabilità della catena (più cresce il numero di transazioni più queste vengono convalidate in ritardo fino a far perdere la fiducia nel sistema).

b) Proof of stake:

Questo algoritmo si basa sulla presenza e dimensione degli *stakes*. In una blockchain basata sul POS ogni utente che possiede dei token può decidere di metterli in *stake*, ovvero di bloccarli e lasciarli a disposizione della rete. Più è grande lo *stake* più si ha probabilità di aggiudicarsi la creazione o la validazione del blocco. In questo caso, se qualcuno cerca di inserire dei blocchi errati, la posta in gioco è lo *stake* messo in palio: viene decurtato se non tolto completamente e ridistribuito ai nodi che si sono accorti dell'errore.
Non essendo più necessario risolvere un problema complesso il consumo di energia elettrica è ridotto e anche il tempo di elaborazione è inferiore. Tuttavia, si può creare un problema di centralizzazione della rete perché, chi possiede più token ha più probabilità di essere selezionato e quindi di aumentare ancora di più il suo stake con le fee.

2.9 Applicazioni reali

La prima implementazione blockchain fu proposta da Satoshi Nakamoto nel 2009 con la creazione di Bitcoin con l'obbiettivo di fungere da "libro mastro" della valuta. Dopo questa prima applicazione si intuì la versatilità di questa tecnologia e di come poteva essere utile in molti altri campi, ad esempio, nella tracciabilità di prodotti, nell'identificazione digitale. Molti di questi ambiti sono in fase sperimentale ma i risultati sono promettenti, e inducono a pensare che possano essere presto utilizzate in maniera più strutturata.
Di seguito vengono riportati alcuni esempi concreti di applicazione diversa dalla finanza decentralizzata.

- SITA: società di telecomunicazioni che ha portato avanti una sperimentazione con la collaborazione della British Airways sulla sincronizzazione dei dati dei passeggeri e delle manutenzioni aeree tramite blockchain.
- MedRec: è un progetto sviluppato dal MIT di Boston per gestire gli accessi dei pazienti alle loro informazioni mediche. Vengono mantenute nella blockchain solamente gli accessi dei pazienti garantendo così la privacy dei dati sensibili.
- Estonia: l'Estonia è un paese che ha puntato fortemente sulla tecnologia della blockchain. Gran parte dei servizi dell'amministrazione pubblica sono gestiti tramite questa.

2.10 Vantaggi e limiti

Come abbiamo visto quindi, questa tecnologia ha una serie di caratteristiche molto utili tra cui *l'immutabilità dei dati*, la *trasparenza* e la *tracciabilità*.

Dall'altro canto però presenta alcune limitazione da tenere in considerazione:

- Scalabilità: dovendo gestire la sincronizzazione e la validazione dei dati tra tutti i nodi decentralizzati, la capacità di transazioni al secondo è relativamente bassa, molto più bassa di un sistema centralizzato.
- Irreversibilità: una volta che una transazione è salvata sulla blockchain questa non può essere più modificata o revocata, nemmeno in caso di errore o di frode. È quindi richiesta una particolare attenzione nell'inserimento delle transazioni dunque è necessario adottare solidi sistemi di sicurezza per prevenirli prima della registrazione.
- Privacy: poiché tutti i dati sulla blockchain sono trasparenti questo può essere un problema nei casi in cui sono richieste particolari regole di privacy.
- Consumo energetico: specialmente nei sistemi che utilizzano blockchain con validazione POW è richiesto un alto consumo energetico per il loro funzionamento.

3. *Bitcoin*

Bitcoin è la prima blockchain mai creata e rappresenta il fondamento del settore crypto. È una rete decentralizzata, open-source e pubblica.
L'idea di Bitcoin nasce anche in relazione alla crisi finanziaria del 2007 con uno scopo ben preciso: fornire un sistema monetario alternativo, resistente alla censura e indipendente da governi e istituzioni.

Bitcoin non ha un fondatore "ufficiale": l'identità di Satoshi Nakamoto è sconosciuta. Dal 2010 non ha più fatto nessuna dichiarazione, lasciando la rete nelle mani della comunità e dello sviluppo open-source.

3.1 Tappe fondamentali

- Nel 2008: Satoshi Nakamoto pubblica il celebre whitepaper "Bitcoin: A Peer-to-Peer Electronic Cash System".
- 3 gennaio 2009: viene minato il genesis block, il primo blocco della blockchain.
- 2010: Bitcoin Pizza Day. Avviene il primo pagamento in Bitcoin: due pizze vengono acquistate per 10.000 BTC
- 2012: arriva il primo halving, un evento che dimezza la ricompensa per blocco da 50 a 25 BTC.
- 2013–2017: Bitcoin esplode in popolarità e il mondo inizia a conoscerlo. Nel 2017, le discussioni sulla scalabilità portano a una divisione nella comunità, da cui nascerà Bitcoin Cash (BCH).
- 2017: viene introdotto Segregated Witness (SegWit), un aggiornamento che migliora l'efficienza delle transazioni e apre la strada a nuove soluzioni, come il Lightning Network.
- 2021: El Salvador diventa il primo Paese al mondo ad adottarlo come moneta legale.
- 2024: avviene il quarto halving, che riduce la ricompensa per blocco a 3,125 BTC.
- 6 Ottobre 2025: Bitcoin raggiunge un nuovo massimo storico di 126,080 dollari.

3.2 Funzionamento di Bitcoin

Bitcoin utilizza il meccanismo di consenso Proof of Work.
Quando le transazioni arrivano sulla blockchain vengono condivise a tutti i miner e salvate nella mempool in attesa di essere inserite in un blocco. La mempool è la memoria dove risiedono tutte le transazioni non ancora validate. Ad ogni transazione viene associata una commissione che è decisa da chi la effettua. I miner tendono a dar precedenza alle transazioni con le commissioni più alte.
I miners raccolgono le transazioni e le inseriscono in un blocco.
Per validarlo devono trovare il nonce corretto per ottenere un hash con una determinata struttura (con un certo numeri di 0 all'inizio). Questo processo ha un alto costo computazionale ed è chiamato "mining". Gli altri nodi verificano la correttezza dell'hash e se è corretto lo aggiungono alla blockchain. Una volta creato un blocco valido il mier ottiene la ricompensa. Questa è data dalle commissioni delle transazioni (decise dai committenti) e dal "block reward", una piccola quantità di bitcoin minata appositamente dal miner per se stesso e inclusa nel blocco creato.
Ogni blocco solitamente contiene più di una transazione (in media 1800 transazioni per blocco). Ogni transazione contiene :

- l'indirizzo del mittente,
- l'indirizzo del destinatario,
- l'importo trasferito,
- la firma crittografica di chi la effettuata che ne attesta la validità.

Il processo di mining garantisce che i nodi siano incentivati a creare blocchi validi: infatti, mentre la creazione è complicata e costosa, la verifica di validità è molto rapida. In questo modo un blocco contraffatto viene riconosciuto subito e scartato.

3.3 La visione di "Internet del denaro"

Bitcoin nasce con l'obiettivo di rivoluzionare il concetto stesso di denaro. Il sistema fissa un massimo di 21 milioni di BTC, introducendo la prima forma di bene digitale scarso, simile all'oro. I principi fondamentali:

- Decentralizzazione: nessuna banca centrale può controllarne l'emissione.
- Resistenza alla censura: chiunque può inviare e ricevere valore.
- Trasparenza: tutte le transazioni sono pubbliche e verificabili.
- Sicurezza crittografica: basato su firme digitali e decentralizzato

4. Ethereum

Ethereum è una blockchain opensource che permette di creare dei programmi, gli smart-contract, con i quali è possibile interagire tramite transazioni. Nasce con l'idea di essere un "World computer" ovvero un computer distribuito su nodi sparsi in tutto il mondo dove chiunque possa sviluppare programmi, applicazioni e vi possa interagire da qualsiasi parte del mondo.
Nelle prime versioni era basata sull'algoritmo di POW; solo nel settembre del 2022 con l'aggiornamento "Bellatrix" avviene il "The Merge", ovvero il passaggio a POS.

4.1 Funzionamento

Nella versione odierna Etrhereum utilizza l'algoritmo di consenso Proof-of-stake. Quando un utente intende effettuare un'operazione sulla rete Ethereum (ad esempio il trasferimento di token ad un altro utente) deve, oltre a definire le informazioni fondamentali per l'operazione, definire anche un prezzo massimo che è disposto a pagare per quella transazione: il cosiddetto *gas*.
Il *gas* lo si può definire come "il carburante" che fa andare avanti la rete Ethereum; serve per preservare l'utilizzo eccessivo delle risorse e di conseguenza come sicurezza aggiuntiva (evita attacchi spam o di consumo infinito di risorse). Ogni operazione ha quindi un suo costo che è dato dalla formula

Gas used×Gas price = Costo in ETH

- Gas used: è il gas necessario per l'operazione che si vuole effettuare.
- Gas price: viene misurato in gwei (gigawei) con un rapporto in Ethereum di 1 ETH = 10^9 gwei .

Il gas è diviso in due parti e può essere deciso dall'utente (o suggerito per tipo di operazione).
La prima parte è la fee della rete (*Base fee*) ovvero la quantità minima necessaria per eseguire l'operazione. La base fee viene regolata in automatico dal protocollo EIP-1559 e varia a seconda della congestione della rete. Questa parte, alla fine della transazione, viene bruciata.
La seconda parte invece, è la ricompensa per il validatore (*Priority fee*): questa può essere variabile e decisa dall'utente. Più la Priority fee è alta più è probabile che un validatore la validi per prima (logicamente i validatori sono interessati a validare le transazioni con ricompense più elevate). Una volta che la transazione è registrata sulla

blockchain il rimanente dell'allocato viene restituito all'utente inziale.
Quindi, una volta definita la variabile, la transazione entra nel mempool dei vari nodi. Il mempoll è lo stake dove le transazioni non ancora validate rimangono in attesa di essere messe nella blochchain. Il validatore scorre poi le varie transazioni da inserire e decide quali processare per prima (le più redditizie). Inizia così a preparare il nuovo blocco per poi proporlo.
Gli altri nodi validano il nodo proposto.
Il prezzo del gas è quindi molto influenzato dalla congestione della rete: in parte per il funzionamento del protocollo EIP-1559 ma anche perché se vengono validate prima le transazioni con una ricompensa più alta gli utenti cercheranno sempre di alzare il prezzo della ricompensa, in modo da avere subito le loro transazioni validate. Questo porta a un'inflazione del costo delle operazioni e di conseguenza un problema di scalabilità.

4.2 Come avviene la validazione

Sulla catena Ethereum il tempo viene suddiviso in slot di 12 secondi e in epoch, composte da 32 slot.
Un nodo validatore per essere tale deve eseguire 3 diverse client: un client di esecuzione, uno di consenso e il client di validazione. Il sistema POS di Ethereum richiede che ogni validatore metta "a rischio" una quantità di 32 ETH.
Il funzionamento è il seguente: l'obbiettivo è di proporre un blocco per ogni slot. Ad ogni slot vengono selezionati un nodo proposer e una serie di committee (gruppo di nodi che validano la creazione del blocco). Il numero di membri per ogni committee può variare ma è generalmente basato su un numero target (128 validatori per committee). Ogni validatore può partecipare ad un solo committee per slot.
La scelta dei blocchi validator e proposer viene effettuata tramite un seme casuale RANDAO; questa elezione viene fornita con un lookahead, ovvero i nodi vengono eletti in anticipo per gli epoch futuri.
Ogni nodo nel committee, oltre a validare il blocco proposto, ha il compito di produrre attestazioni, ovvero votare:

- il blocco head della catena
- Il blocco source e il blocco target

Questo voto consente di identificare quale blocco rappresenta la testa della catena dal quale partire per appendere i nuovi blocchi. Il protocollo che regola questo è LMD-GHOST.
Ogni validatore produce quindi un'attestazione che propaga a tutti gli altri membri della committee; a livello di committee vengono raccolti i voti e distribuiti al resto della rete. Per limitare la congestione sulla rete, ogni validatore condivide le attestazioni prima con la sua subnet di committee che successivamente vengono aggregate e diffuse a tutti i nodi della rete. Ogni validatore mantiene localmente le ultime attestazioni ricevute per

i calcoli futuri.
Il nodo proposer utilizza le attestazioni salvate in locale e sempre tramite LMD-GHOST calcola l'head corrispondente. Quest'ultimo viene poi utilizzato come padre del nuovo blocco inserendolo nell'header.
Se la rete è particolarmente sotto stress possono verificarsi ritardi nell'arrivo delle attestazioni. Può capitare quindi che i validatori calcolino nella stessa epoch head diversi: in questo caso verranno create delle sotto catene temporanee discordanti tra loro ma che vengono risolte con le successive iterazioni del LMD-GHOST.
Nelle attestazioni sono espressi anche i nodi *source* e *target*: questi servono per determinare la struttura definitiva della catena e selezionare i nodi checkpoint che non possono più essere cancellati o riorganizzati.
Infatti i nodi *source* e *target* si riferiscono rispettivamente all'ultimo blocco justified (ultimo blocco votato come potenziale finalizzato) e al blocco proposto per diventare justified. Questi blocchi sono scelti a livello di blocchi checkpoint, ovvero il primo nodo di un'epoch.
Se almeno 2/3 dei nodi concorda su questi nodi per due epoch consecutivi diventano rispettivamente *finalized* (immutabile) e *justified*.
Il protocollo che gestisce questo è il Casper-FFG.

4.3 Blocco di Ethereum

Un blocco di Ethereum è diviso in due parti: header e body.

Di seguito si riporta un estratto dalla documentazione ufficiale di Ehtereum.

Campo	**Descrizione**
slot	lo slot a cui appartiene il blocco
indice_proponente	l'ID del validatore che propone il blocco
parent_root	l'hash del blocco precedente
state_root	l'hash radice dell'oggetto di stato
Body	un oggetto contenente più campi, come definito di seguito

Body:

Campo	**Descrizione**
randao_reveal	un valore utilizzato per selezionare il prossimo proponente di blocchi

Campo	Descrizione
et1_data	informazioni sul contratto di deposito
graffiti	dati arbitrari utilizzati per contrassegnare blocchi
proposer_slashings	elenco di validatori da tagliare
taglio_attestatori	elenco di attestatori da tagliare
attestazioni	elenco di attestazioni a favore del blocco corrente
depositi	elenco dei nuovi depositi nel contratto di deposito
uscite_volontarie	elenco di validatori che escono dalla rete
sync_aggregate	sottoinsieme di validatori, utilizzato per servire i client leggeri
execution_payload	transazioni passate dal client di esecuzione

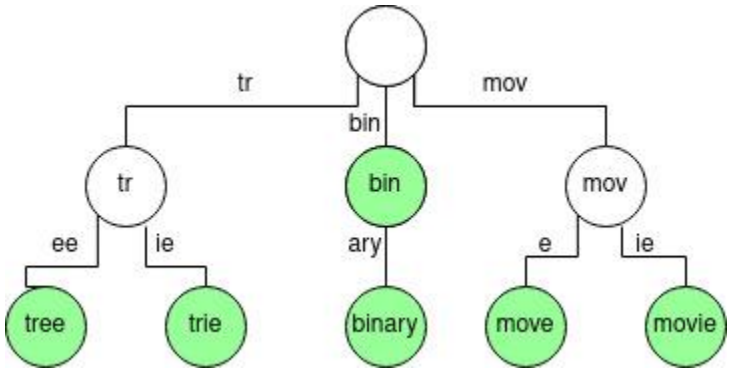
Il campo attestations contiene un elenco di tutte le attestazioni nel blocco. Le attestazioni hanno il proprio tipo di dati, contenente diversi pezzi di dati. Ogni attestazione contiene:

Campo	Descrizione
aggregation_bits	un elenco dei validatori che hanno partecipato a questa attestazione
dati	un contenitore con diversi campi secondari
firma	firma aggregata di tutti i validatori attestanti

Il campo data nell'attestation contiene quanto segue:

Campo	Descrizione
slot	lo slot cui si riferisce l'attestazione
indice	indici per l'attestazione dei validatori
beacon_block_root	l'hash radice del blocco Beacon contenente questo oggetto
fonte	l'ultimo punto di controllo giustificato
obiettivo	il blocco di confine dell'ultima epoca

Le transazioni in un blocco Ethereum possono dover contenere informazioni più dettagliate rispetto a Bitcoin. Per questo il Merkle Tree utilizzato in Ethereum è il "Merkle Patricia Tree", che unisce le proprietà di un albero di prefissi (trie) con quelle di un Merkle Tree classico.



4.4 Storia di Ethereum

- Nel 2014 viene presentato per la prima volta Ethereum nel WhitePaper pubblicato da Vitalik Buterin all'età di 19 anni. Buterin è un programmatore russo naturalizzato canadese fondatore anche del famoso periodico Bitcoin Magazine. La sua idea era di espandere le funzionalità di Bitcoin, limitate al solo scopo economico.
  "So like, think of the difference between something like a plot key calculator and a smartphone, where a plot key calculator does one thing and it does one thing well, but really people want to do all these other things. And if you have a smartphone then on the smartphone you have a plot key calculator as an app. You have playing music as an app. […]So basically taking that same kind of idea of

increasing the power of the system by making it more general purpose and applying it to blockchains." [*Vitalik Buterin, intervista a Business Insider, 13 Febbraio 2019]*

A seguito della pubblicazione ricevette un primo investimento dalla Peter Thiel Fellowship per inziare a lavorare sulla piattaforma.

- Tra luglio e settembre del 2014, per promuovere il progetto e recuperare fondi,viene organizzata la prima ICO (Initial Coin Offering).
- Nel luglio del 2015, vengono lanciata la prima versione della blockchain e il suo token nativo (Ether).
- Nel giugno del 2016, avviene un importante evento per la storia e lo sviluppo della rete: l'hacking di TheDAO. A seguito di questo attacco fu effettuato un hard fork della blockChain creandone così due diverse:
    - Ethereum: è la rete Ehtereum conosciuta come tale dove sono state rimosse tutte le transazioni fraudolente.
    - Ethereum Classic: la rete Ethereum con all'interno anche le transazioni dell'hacking.
- Nel 2020 Vitalik Buterin pubblica un cambio della road map di Ethereum e introduce il Layer2.
- Il 15 Settembre del 2022 avviene il Merge. Questa è una data storica per Ethereum perché avviene il passaggio da POW a POS, riducendo anche l'impatto ambientale dell'inquinamento prodotto del 99%.
- Nel marzo 2024 riceve l'aggiornamento Dencum, che con l'aggiunta del'EIP-4844 riduce di molto il costo del gas. Con questo aggiornamento vengono introdotti dei blob che accorpano le transazioni perché ne vengano elaborate di più insieme.

4.5 WEB 3.0

Il cofondatore di Ethereum, Gavin Wood, propose l'idea di *web 3.0* alla quale Ethereum dovrebbe contribuire a realizzare. Per spiegare cosa sia il web 3.0 bisogna introdurre 3 fasi del web: lettura, scrittura , possesso.

Web 1.0 – Lettura: questa fase si inserisce in un periodo che spazia dal 1990 al 2004 e che coincide con la prima versione di internet pubblica. Le varie risorse venivano caricate direttamente su computer collegati ai quali era possibile accedervi tramite internet da ogni parte del mondo. Si poteva immaginare internet come una grande biblioteca che ognuno poteva consultare. I contenuti erano però statici e prodotti solitamente da enti (università, aziende) o appassionati, anche perché la loro creazione non era così immediata.

Web 2.0 - Lettura e Scrittura: è il periodo che arriva dal 2004 fino ai nostri giorni. Da questo momento gli utenti iniziano a interagire in modo attivo e non più come semplici fruitori di contenuti. Con l'introduzione di nuovi strumenti come i social, si può creare, condividere e caricare materiale online. Tuttavia, questo materiale e la gestione degli strumenti che lo permettono è in mano a grandi colossi della tecnologia, che lo possiedono e possono regolamentarlo con regole arbitrarie e scelte da loro.

Web3.0 - Lettura, Scrittura, Possesso: è un'idea di web dove si è indipendenti dalle regole e dalle restrizioni di grosse terze parti. Ethereum si pone come fondamento per la sua realizzazione.
Le idee fondamentali del web 3.0 sono:

- **Decentralizzazione**: internet deve essere distribuito e non riservato ad aree gestite da grosse compagnie. Ad oggi, la maggior parte dei dati su internet è centralizzata in grossi datacenter posseduti da colossi tecnologici (Google, Amazon, Microsoft).
- **Non basato sulla fiducia**: non è necessario affidarsi a terze parti ma è possibile effettuare ogni operazione su internet (e quindi sulla catena) senza intermediari. Tramite gli smart contract è possibile automatizzare tutti i processi di gestione delle operazioni sulla rete senza bisogno di un agente terzo che se ne occupi.
- **Sovranità dei Dati**: attualmente, come abbiamo già detto, per poter usufruire dei servizi in internet bisogna appoggiarsi a grosse compagine IT che salvano e custodiscono i dati degli utenti. Questi hanno la possibilità di rivenderli a terzi senza un nostro controllo (la legislazione che limita queste operazioni è poco regolamentata). Tramite il web 3.0 sono gli utenti ad avere direttamente il possesso dei dati e decidere attivamente chi può accedervi e chi no.
- **Accessibilità globale**: per sua natura il web 3.0 è open source ed accessibile da chiunque tramite connessione internet.
- **Interoperabilità**: le applicazioni sul web 3.0 sono interoperabili, perché non sono gestite da organizzazioni separate ma dagli smart contract, molto più semplice da fare interagire tra loro.
- **Tokenizzazione**: nel web 3.0 entra il concetto di tokenizzazione. Ogni utente può possedere dei token della piattaforma internet che possono essere usati per interagire con le app sul web ed eventualmente anche usati come ricompensa. L'utilizzo di token quindi può incentivare l'utilizzo della tecnologia rendendola facilmente adottabile.

4.6 DAO

Nel paragrafo “Storia di Ethereum” abbiamo già trovato un esempio di DAO. L’acronimo DAO sta per “Decentralized Automated Organization” che indicano delle organizzazioni distribuite.
Prendendo in esempio un’azienda tradizionale, questa ha dei finanziatori che investono ma la struttura dell’organizzazione ha una gestione gerarchica differente. Ci sono quindi un Ceo, un consiglio di amministrazione e più in generale una struttura amministrativa che prende le decisioni. Tutti questi rispondono poi agli investitori ma in modo indiretto.
In una DAO invece, può non essere presente questa struttura amministrativa; le decisioni sono prese direttamente dagli investitori che possono proporre delle azioni o delle idee che vengono poi votate dal resto dei membri. Questo grazie all’acquisto dei suoi token. I token della DAO rappresentano quindi un controllo diretto sull’organizzazione e più si possiedono token più il proprio voto avrà un peso. Chi ha un gran numero di token potrà influenzare maggiormente le votazioni.
Tutte le azioni approvate dalla votazione vengono poi automaticamente gestite tramite gli smart contracts della DAO.

4.7 Attacco a “THE DAO”

“The Dao” fu una delle prime Decentralized automated organization che vennero create: si trattava di un fondo di investimento che avrebbe dovuto finanziare futuri progetti commerciali e no-profit.
Sfortunatamente erano ancora agli albori e gli smart contract non erano ancora stati testati sufficientemente soprattutto nell’ambito della sicurezza. Nello smart contract in questione, era infatti presente un bug nell’operazione di prelievo: quando l’operazione di prelievo veniva invocata, prima veniva effettuato il passaggio di denaro e poi il saldo veniva aggiornato.
L’attacco si verificò attraverso la seguente modalità. L’operazione prelievo veniva richiamata ricorsivamente: la funzione whitrdawal() richiamava all’interno se stessa. Poiché il saldo veniva aggiornato solo una volta che la chiamata a withdrawal() ritornava correttamente, le continue chiamate permettevano di tenere bloccata l’applicazione al livello di passaggio dei soldi e non dell’aggiornamento saldo. Quindi, le chiamate ricorsive continuavano a prelevare soldi ma il saldo rimaneva sempre lo stesso. Il ciclo continuava finché c’erano soldi sul fondo o finché non finiva il “gas” per fare effettuare le transazioni.
Gli hackers riuscirono a rubare al fondo 3.6 milioni di ETH (60 milioni di dollari). A seguito dell’attacco, la comunità Ethereum si spaccò in due: chi voleva riportare lo stato della chain Ethereum prima del furto e chi invece sosteneva che dovesse rimanere così. Alla fine, a seguito di una votazione, si decise per tornare indietro. Avvenne quindi

l'hard fork della catena creando così due diverse blockchain.
La prima, "Ethereum", era quella senza le transazioni fraudolente ed è la più conosciuta.
"Ethereum Classic", invece, è la blockchain originale senza il rollback.
A rimedio del bug, venne introdotto il pattern Checks-Effects-Interactions che segue 3 passaggi in questo ordine:

1° - Verifica le condizioni

2° - Aggiorna lo stato

3° - Interagisce con gli altri contratti

4.8 Smart Contract e DApp

Il concetto di *smart contract* venne teorizzato da Nick Szabo nel 1994, ancora prima della creazione delle blockchain.
"*A smart contract is a* ***computerized transaction protocol that executes the terms of a contract****." [Nick Szabo, 1997]*
In un contesto dove l'utilizzo della tecnologia era in forte crescita in quasi tutti gli ambiti, teorizzò l'idea di contratti che potessero essere automaticamente elaborati e validati da una macchina senza la necessità di un ente esterno (ad esempio un notaio). Il tutto si basava su contratti stipulati da due o più parti che avessero delle condizioni chiare con delle conseguenze altrettanto chiare (meccanismo if than- else): se una delle condizioni si verificava in automatico la macchina la riconosceva e applicava la rispettiva clausola.
Nel concreto quindi, lo smart contract è codice che viene elaborato dal sistema Ethereum con delle funzionalità e logiche definite che. può interagire con altri smart contract e su cui si basano le DApp presenti su Ethereum (applicazione decentralizzate).
I primi smart contract erano relativamente semplici: vennero utilizzati per attività di ICO (Initial coin offering) ovvero un crowfunding dove avveniva una vendita di token a chiunque volesse finanziare il progetto. Sostanzialmente erano necessarie due azioni: la creazione del token con tutte le sue informazioni e la gestione del crowfunding (quanti token assegnare all'investitore, tasso di cambio, controlli di sicurezza per eventuali vendite nulle).
Una volta capita la potenzialità degli smart contract si iniziò a sperimentare un'interazione tra di loro: nascono così le prime Distributed App.
La prima tipologia di DApp sviluppate offrivano servizi finanziari decentralizzati. Adesso invece si trovano DApp per ogni genere di utilizzo che trovano applicazione in molti campi come social network, gaming, logistica ecc..
Poiché è possibile creare DApp per qualsiasi genere di operazione e qualunque sviluppatore con le necessarie competenze può crearne una, è indispensabile che ci sia

un controllo sulla sicurezza al fine di garantire affidabilità all'ecosistema delle DApp. Anche perché, una volta che lo smart contract o la Dapp viene caricata sulla chain non può essere più modificata (deve esserne creata una nuova). Nascono così le figure degli *auditor*: società riconosciute che si occupano di verificare che gli smart contract siano sicuri e rispettino le norme di sicurezza per ogni settore a cui si applicano. Questo serve non solo per evitare che vengano caricate DApp con delle vulnerabilità ma anche per dare fiducia negli investitori.
Sostanzialmente il funzionamento di un Audit è il seguente: una volta che la DApp è pronta, viene richiesto l'Audit da una delle compagnie che offrono il servizio. Quest'ultima revisiona il codice (il tempo solitamente varia da qualche giorno a qualche settimana a seconda della dimensione del progetto) e stila una lista di eventuali criticità e azioni correttive. Solitamente, una volta attuate le operazioni consigliate si effettua un'ulteriore verifica di follow-up per verificare che siano state applicate correttamente. Una volta che la Dapp è completata e revisionata viene deployata sulla chain tramite una transazione.
Solidity è il linguaggio di programmazione alla base degli smart contract su Ethereum concettualizzato nel 2014 da Gawin Wood. È un linguaggio ad oggetti basato su C++, Python e Javascript. Il suo scopo è quello di permettere la creazione di logiche decentralizzate sulla Ethereum Virtual Machine. Il deploy del codice avviene tramite una transazione sulla blockchain con il codice compilato.
Uno dei pilastri alla base di Ethereum è che sia open-source. Ci possono essere situazioni in cui alcuni smart contract non vengano resi pubblici, ovvero non venga diffusa la pubblicazione della documentazione o la condivisione del code repository. In ogni caso però il bytecode rimane disponibile sulla blockchain.

4.9 Layer 2

Il grande successo di Ehtereum e la sua applicabilità in molti campi ha portato presto a un problema di scalabilità: la quantità di transazioni che doveva essere gestita per tutte le Dapp su Ethereum inziava a diventare insostenibile con grossi costi di gas.
È stato necessario quindi pensare a una soluzione che ovviasse al problema: creare dei layer superiori alla blochChain Ethereum che si preoccupassero di assorbire delle operazioni affidate altrimenti alla rete Ehtereum e di alleggerire quindi quest'ultima. I layer 2 servono quindi per migliorare la scalabilità, i costi e la velocità delle operazioni sulla blockchain Ethereum.
Esistono due principali categorie di Layer 2:

a. Roll up: questa tipologia è quella più usata come soluzione. Un scaling-system di Roll up comporta un sistema a livello più alto che effettua un gran numero di transazioni off-chain che poi vengono registrate in un

unico blocco sulla blockchain. Il problema di effettuare operazioni off-chain può comportare problemi nella validità delle operazioni effettuate. Questo viene gestito in due modi differenti.

- Optimistic Roll-up: questi layer operano considerando tutte le transazioni come valide (ottimisticamente appunto) ma mettono a disposizione una prova di frode. Prima quindi che vengano registrate le transazioni off-chain il sistema attende un determinato tempo (solitamente qualche giorno) in modo che qualsiasi validatore possa ricalcolare la prova di frode e richiedere eventualmente l'invalidazione del blocco.
- ZK-Roll-up: questi funzionano nel modo opposto agli Optimistic. Considerano infatti che tutte le operazioni siano false fino a che non viene fornita una prova di zero conoscenza (Zero Knowledge). Quindi per ogni blocco di transazione viene fornita anche la prova matematica della validità del blocco. Questi, rispetto ai precedenti sono più sicuri e più veloci ma anche molto più complessi da implementare.

b. Sidechain: si utilizzano delle blockchain separate collegate alla blockchain Ethereum tramite brige. Quando un utente decide di spostarsi dalla Blockcahin Ethereum alla sidechain contatta lo smart contract che si occupa di fare il bridge tra le chain. Blocca quindi il quantitativo di asset che l'utente intende spostare sulla second chain e successivamente si occupa di creare (mint) lo stesso valore nell'asset corrispondente alla nuova chain. Si crea quindi una copia dei token sulla second catena. Sulla nuova chain le operazioni avvengono in modo completamente scollegato da Ethereum. Una volta che si intende ritornare sulla chain Ethereum si ripasserà dal bridge per sbloccare l'asset e bruciare la copia sulla side chain.
Qui, a differenza dei roll-up, dove le transazioni venivano salvate sulla blockchain Ethereum, le due catene non sono sincronizzate. Ethereum non conosce le attività della sidechain tranne per le operazioni sui bridge. Il vantaggio è che sono molto flessibili ma perdono la sicurezza e la trasparenza di Ethereum, essendo truth-less con la catena originale. Devono quindi creare loro un sistema di validazione e di tracciabilità.

4.10 Protocollo ERC-20

ERC-20 è il protocollo per gli smart contract che devono gestire token fungibili.
I metodi necessari da implementare perché rispetti lo standard sono:

- **totalSupply**(): recupera il numero totale di token
- **balanceOf** (address _owner): ritorna il numero di token posseduti da adress _owner
- **transfer** (address _to, uint256 _value. Se il chiamante non ha sufficienti token ritorna un'eccezione
- **transferFrom** (address _from, address _to, uint256 _value): metodo che trasferisce l'ammontare di token _value all'indirizzo _to da parte di un utente terzo _from
- **approve** (address _spender, uint256 _value): metodo che autorizza a spendere fino a _value token. É utilizzato come meccanismo di sicurezza della per la transferFrom
- **allowance** (address _owner, address _spender): metodo utilizzato per verificare lo stato delle approve() concesse

Gli eventi sono operazioni che vengono attivate in automatico (trigger) quando viene invocato un metodo o a seguito di una particolare operazione:

- Transfer (address indexed _from, address indexed _to, uint256 _value)
- Approval (address indexed _owner, address indexed _spender, uint256 _value)

5. Non Fungible Token

I Non Fungible Token o NFT, si differenziano dai classici token per la loro unicità sulla blockchain.
Normalmente, un fungible token non ha caratteristiche differenti rispetto agli altri e può quindi essere considerato indistinguibile e interscambiale con gli altri token. Un tipico esempio di fungible token sono le criptovalute.
Al contrario, gli NFT sono entità digitali univoche con delle caratteristiche uniche e un valore proprio. Per questa loro natura possono, in linea teorica, rappresentare il prodotto digitale degli oggetti fisici e trovano l'abbinamento perfetto con il mondo dell'arte. Gli NFT infatti vengono utilizzati per rappresentare delle opere artistiche in formato digitale. Questa cosa però non è esente da rischi e problemi che verranno affrontati in seguito.
Da un punto di vista tecnico gli NFT hanno dei protocolli diversi dai fungible token con metodi e funzionalità specifiche per gestire la loro univocità.

5.1 Protocollo ERC-721

Il protocollo ERC-721 viene proposto nel 2018 da William Entriken, Dieter Shirely, Jacob Evans e Nastassia Sach ed è lo standard per gli NFT in Ethereum. Questo protocollo definisce le regole per poter gestire l'univocità e le caratteristiche intrinseche dell'NFT.
I metodi necessari da implementare perché possa essere rispettare lo standard sono:

- **balanceOf** (address _owner) ritorna il numero di token posseduti dall'indirizzo passato
- **ownerOf** (uint256 _tokenId) ritorna l'indirizzo del proprietario per uno specifico tokenId (address)
- **safeTransferFrom** (address _from, address _to, uint256 _tokenId, bytes data) external payable: trasferisce in maniera "safe" (verifica in automatico se il ricevente è in grado di ricevere l'nft) la proprietà di un NFT(tokenId) da un proprietario(_from) a un altro(_to). Tramite il parametro aggiuntivo data è possibile passare dati ulteriori che senza un formato specifico
- **safeTransferFrom** (address _from, address _to, uint256 _tokenId): trasferisce in maniera "safe" la proprietà di un NFT(tokenId) da un proprietario(_from) a un altro(_to).
- **transferFrom** (address _from, address _to, uint256 _tokenId): trasferisce la proprietà di un NFT(tokenId) da un proprietario(_from) a un altro(_to). Non essendo "safe" chi la utilizza è responsabile di confermare che il ricevente sia in grado di ricevere l'nft altrimenti questo viene perso.

- **approve** (address _approved, uint256 _tokenId) permette di approvare un altor indirizzo (_approved) a trasferire uno specifico token. Se il chiamante non è il proprietario del token, deve restituire un eccezione;
- **setApprovalForAll** (address _operator, bool _approved) permette di approvare un altro indirizzo (_approved) a trasferire i token. Se il chiamante non è il proprietario del token, deve restituire un eccezione;
- **getApproved** (uint256 _tokenId): restituisce gli indirizzi approvati per uno specifico token.
- **isApprovedForAll** (address _owner, address _operator)

Oltre a questi metodi deve poter memorizzare:

1. I tokenId degli NFT collegati allo smart-contract
2. Gli address proprietari dei singoli NFT
3. I metadati degli NFT

Per quanto riguarda i metadati degli NFT, ovvero le informazioni che caratterizzano i singoli NFT, si evita di salvarle direttamente sullo smart contract. Questo perché solitamente sono di grosse dimensioni e gestirle direttamente sulla blockchain può richiedere grossi costi per le transazioni. Si preferisce quindi salvare i metadati off-chain e salvare sulla blockchain solamente i lori riferimenti.

5.2 Differenza con ERC-20

La differenza tra ERC-20 ed ERC-721 è la rispettiva gestione di token fungibili e non fungibili: per il primo l'importanza è sulla quantità di token assegnati a uno specifico address mentre per l'altro è fondamentale il mapping tra singolo tokenId a specifico address.
Un'ulteriore differenza è la possibilità di aggiungere nel protocollo ERC-721 delle proprietà metadata che caratterizzano il token stesso, generalmente sottoforma di json object.

5.3 Protocollo ERC-1155

ERC-1155 è un protocollo creato per poter gestire contemporaneamente sia i token fungibili che quelli non fungibili. Questo è un protocollo multi-asset, cioè gestisce più tipi di token in un singolo contratto. Tramite questo protocollo sia i token fungibili che non fungibili sono definiti con un *id*: l'id dei token fungibili è unico per tutti. Con gli smart contract che seguono questo protocollo è possibile trasferire più tipi di token in una unica transazione (non possibile con i precedenti due che richiedono una transazione per tipo) risultando molto più efficiente.

Una caratteristica particolare di questo protocollo è che introduce anche la possibilità di creare *Token semi-fungibili*: quando questi token vengono creati si comportano come token fungibili (indistinguibili tra loro) e quindi intercambiabili con altri token semi fungibili. Può però essere trasformato in token NFT e diventare non fungibile. Una volta che diventa non fungibile acquisisce delle caratteristiche uniche e non può più ritornare ad essere fungibile.

5.4 Applicazioni pratiche degli NFT

La tecnologia NFT ha ricevuto, sin dalla sua creazione, un forte seguito dovuto in parte dalla tendenza di mercato e in parte dall'interesse che questa nuova tecnologia ha suscitato in molti. Ha trovato quindi utilizzo in molti campi.
Il primo e forse più famoso è quello dell'arte digitale. Un artista crea un'opera e ha la possibilità di venderla in formato digitale tramite la creazione di uno o più NFT. In questo modo può garantire l'unicità o la rarità della sua opera. Un fattore interessante di questa applicazione è che l'autore può decidere di vendere l'NFT all'effettivo proprietario dell'opera mentre l'opera reale può essere esposta in un museo ed essere fruibile da tutti.
Gli NFT sono anche usati nel mondo del gaming, specialmente nel mondo del gaming online per creare degli asset unici. Questo porta a meccanismi di compravendita tra i giocatori direttamente nel gioco, offrendo così un'sperienza ancora più immersiva: l'oggetto della transazione è effettivamente un oggetto reale, unico e tracciabile, non duplicabile dagli sviluppatori o dall'azienda che lo gestisce.
Nel settore musicale, molti artisti hanno deciso di seguire la moda degli NFT e di associare ai loro album o ad alcuni pezzi musicali un NFT. Anche il mondo della moda utilizza questa tecnologia in abbinamento a nuove collezioni. Tra i tanti esempi troviamo marce di moda importante come la "Nike" con le "Nike CryptoKicks" ovvero scarpe da ginnastica completamente digitali personalizzabili; "Dolce & Gabbana" con la collezione "Genesi", una collezione di vestiti in parte fisici e in parte digitali collegati tramite NFTs. Non da meno è il settore dell'automotive, in particolare in auto di lusso. Nel 2022 fu venduto al prezzo di oltre 1,6 milioni di dollari, un modello di Lamborghini Ventador insieme a un NFT creato dalla collaborazione tra Krista Kim, Steve Aoki e INVNT GROUP.

5.5 Problematiche degli NFT

Nonostante il potenziale offerto dalle possibilità di creare degli oggetti unici e distinguibili tra loro nel mondo digitale questo non è esente da rischi e problematiche. In primo luogo, essendo un mercato recente, complesso dal punto di vista del funzionamento e non molto conosciuto, ha attirato molti truffatori. Si possono quindi trovare finti NFT spacciati come se fossero creati da artisti famosi oppure bolle speculative che cercano di far salire il prezzo il più in fretta possibile senza che ci sia un valore dietro l'opera (ad esempio NFT senza validi diritti d'autore). La maggior parte dei siti che vende NFT non si preoccupa di verificare l'autenticità delle opere principalmente a causa della sua complessità.
Di seguito sono elencate le principali problematiche che si possono individuare e i relativi esempi pratici:

1. Mancanza di una verifica sull'autenticità
   L'NFT è un oggetto unico e distinguibile ed è possibile identificarlo grazie alla blockchain. Questa affermazione, tuttavia, non è applicabile al suo autore: chi genera un NFT potrebbe spacciarsi per un artista o un cantante famoso, creando un falso. È il caso di Bansky (street artist americano) nel 2021: un truffatore ha creato un NFT spacciandolo per un'opera dell'artista che ha poi rivenduto per più di 300.00 dollari. Anche senza spacciarsi per un artista è facile, tramite l'anonimato, escogitare molte tipologie di truffe.
   Nel 2022, per esempio, venne annunciata una collezione di NFT chiamati "Frosties", che garantiva delle grosse offerte e benefici ai loro possessori. Questo generò una grossa aspettativa sul progetto e in molti iniziarono ad acquistare i suoi NFT. Una volta raggiunta la cifra di oltre un milione, il progetto fu completamente abbandonato, tutte le pagine web, i social media e ogni elemento che riguardasse il progetto vennero chiusi e gli autori scomparirono con i soldi. La scarsa regolamentazione e anche la molta inesperienza tecnica di chi compra e si interessa negli NFT comporta una serie di ulteriori truffe come phishing, furto di NFT, ecc..

2. Valore degli NFT
   Il reale valore dell'NFT è molto difficile da calcolare, perché si può paragonare ad un'opera d'arte. Il suo valore equivale a quanto qualcuno è disposto a spendere, raramente ha un corrispettivo reale. Questo può portare a delle grosse bolle speculative o anche al fenomeno del "Wash Trading": in questo caso compratori e venditori continuano a rivendere e comprare lo stesso token aumentandone sempre di più il prezzo così da poi rivenderlo a una parte terza ad un prezzo gonfiato.

3. Diritti di acquisto sull'opera
   Con l'acquisto dell'NFT spesso si può creare confusione su che cosa effettivamente si stia acquistando. Gli NFT possiedono standard tecnici che regolano le funzionalità del token, ma non hanno uno standard definito di cosa debbano rappresentare. Se non specificatamente concordato nel contratto d'acquisto, comprando un NFT legato a un opera non si sta effettivamente comprando i diritti d'autore dell'opera ma solamente un certificato digitale unico. Possedere un NFT non significa quindi automaticamente possedere l'elemento a cui è legato. Questo può diventare un problema se si vuole utilizzare l'opera dell'NFT per attività coperte da copyright.

4. Dipendenza da link esterni alla blockchain
   Solitamente ad un NFT viene collegato un elemento, la maggior parte delle volte è una risorsa (un'immagine o un documento) che viene storato su db non all'interno del meccanismo di tracciabilità della blockchain. La risorsa sul DB con l'NFT solitamente è collegata tramite un link URL, che porta alla risorsa sul DB. Si consideri, a titolo esemplificativo, un NFT che sia collegato ad un'immagine storata su un DB che si trova su un server privato e che quindi l'NFT abbia l'url dell'immagine al suo interno. Qui, la blockchain dà la certezza che il link nell'NFT non cambierà mai, ma non si può dire lo stesso su come il Db possa essere modificato. Se l'immagine viene sostituita con un'altra o persino cancellata, il link presente sull'NFT anche se non è stato modificato, punterà a qualcosa di diverso.

5. Vuoti normativi:
   Gli NFT, così come in generale la tecnologia delle blockchain, sono relativamente recenti. Di conseguenza, la loro creazione e la loro compravendita non è regolata correttamente. Un'ulteriore problematica è che i diversi paesi nel mondo possono avere regole eterogenee sulla loro identificazione e gestione, comportando difficoltà e disuguaglianze soprattutto nella compravendita internazionale.

6. Obsolescenza della blockchain
   Una volta acquistato un NFT, mantiene un valore finché c'è interesse nel determinato NFT e contemporaneamente finché la blockchain su cui risiede ha una community che la utilizza. Se questa blockchain inizia a diventare menoutilizzata per svariati motivi come la nascita di nuove blockchain o semplicemente perché l'interesse in quella determinata tecnologia va scomparendo, l'NFT rimarrà senza valore poiché intrinsecamente legato all'interesse della blockchain. Quindi, il rischio è di pagare molto per un elemento il cui valore può rapidamente decrescere.

5.6 Risvolti legali NFT

Associare un'opera ad un NFT prevede una serie di considerazioni a livello legale che è importante quanto meno accennare senza entrare nei dettagli più tecnici. Associare un'opera ad un NFT non garantisce automaticamente di avere tutti i diritti d'autore sull'opera stessa.
Possiamo evidenziare 4 diritti riconosciuti dalla legge a livello europeo che possono influenzare il valore dell'NFT stesso.

- Diritto di riproduzione: il diritto di copiare l'opera
- Diritto di elaborazione: il diritto di autorizzare modifiche, adattamenti o opere derivate.
- Diritto di messa a disposizione del pubblico: diritto di rendere accessibile l'opera al pubblico
- Diritto di distribuzione: diritto di mettere in commercio l'opera.

È importante quindi quando si acquista un NFT, comprendere bene cosa si sta acquistando. Soprattutto se è legata ad un'opera fisica. Per il progetto in questione questi risvolti sono toccati marginalmente, ma è comunque parte integrante dell'obbiettivo finale. Come vedremo nella soluzione proposta è prevista la presenza di un'azienda terza che opera anche da intermediario tra creatore ed acquirente. Il costituire un mercato di NFT gestito da questa da un'azienda può rendere più facile il normare questo genere di problemi. È possibile infatti specificare nel contratto di vendita dell'NFT quali diritti si stanno acquistando e questo può dipendere da opera ad opera

5.7 Sviluppo del proof of concept

Lo scopo del progetto consiste quindi nel costituire una piattaforma di compravendita di NFT affidabile che possa essere un punto di riferimento per artisti o esperti. Per fare ciò s'intende costituire un ecosistema di tecnologie che permetta di superare alcune criticità elencate nei capitoli precedenti.
Gli obbiettivi principali sono:

a) Gestione dell'autenticazione dell'opera da legare all'NFT esterno alla blockchain
b) L'implementazione di un meccanismo di autenticazione documentale per garantire l'originalità dell'opera
c) Creazione dell'NFT tramite smartcontract.
d) Rendere persistenti le opere digitali esterne alla blockchain
e) Abilitazione alla multichain

Le opere da associare ad un NFT possono essere di due tipologie:

a) **Completamente digitali**: in questo caso un artista intende proporre un opera completamente digitale (come un poster digitale).
b) **Con un corrispettivo fisico**: in alternativa c'era la possibilità di creare un NFT con un corrispettivo fisico. Ad esempio un antiquario può voler riprodurre un oggetto digitale che rappresenti un'oggetto (ad esempio una mappa antica) per essere poi venduta insieme all'opera o separatamente.

Una volta individuata la categoria dell'NFT si procede in due maniere distinte. Nel caso di un'opera digitale, questa viene caricata su un sistema di storage separato dalla blockchain, con la possibilità di essere salvato in maniera distribuita o centralizzata. I riferimenti a questo vengono inseriti all'interno dei metadati dell' NFT al momento del minting.
Nel caso di opere fisiche invece, è richiesto un passaggio in più. L'opera deve essere autenticata da un ente qualificato, come una galleria d'arte o un antiquario: questo può emettere le certificazioni di autenticità dell'opera, digitalizzarle e firmarle con una firma digitale. A questo punto tutta la documentazione di veridicità dell'opera viene caricata su uno storage, nelle stesse modalità scelte per l'opera digitale e mintato l'NFT. In questo caso il passaggio in più richiede per l'ente qualificato di possedere una firma digitale per poter firmare la documentazione che attesta la veridicità dell'opera.
Il sistema di storage potrebbe essere deciso all'utente al momento della creazione dell'NFT, consentendo la scelta tra un sistema centralizzato o distribuito. In ogni caso di storage decentralizzato, la permanenza dei dati esterni alla blockchain Ethereum verrebbe garantita dall'utilizzo di un'altra tecnologia blockchain: Filecoin.
Si intendono anche proporre delle possibili soluzioni per la possibilità di implementare il passaggio da una blockchain a un'altra con l'utilizzo della blockchain Polkadot.
Nei seguenti capitoli verrà per prima introdotta la firma digitale, necessaria per il secondo caso in esame. Successivamente vengono trattate le tecnologie e i metodi per poter garantire la persistenza dei dati e il passaggio cross-chain.
Infine viene proposto un esempio di smart-contract sviluppato in Solidity che integri le funzionalità descritte.

6.0 Firma Digitale

La firma digitale è un'applicazione utilizzata per garantire l'autenticità della firma in formato digitale. Il concetto è solitamente utilizzato per i documenti ma la tecnologia su cui si basa e che verrà trattata in questo paragrafo, viene utilizzata in molti ambiti della crittografia e si può applicare con efficacia anche in questo caso di studio.

6.1 Crittografia

Quando si parla di crittografia bisogna tenere a mente tre elementi:

4. Il "messaggio" che si vuole crittografare
5. La chiave segreta che viene utilizzata per codificare l'elemento
6. L'algoritmo che definisce il modo in cui la chiave segreta e il messaggio vengono combinati ottenendo il risultato crittografato

Una volta che il messaggio è crittato tramite la chiave segreta e l'algoritmo, è possibile decifrarlo, recuperando il messaggio originale. È quindi importante che la chiave segreta rimanga tale perché la cifratura sia valida, infatti chiunque sia in possesso della chiave segreta può decifrare il messaggio.
Esistono due categorie di crittografia: *simmetrica* e *asimmetrica*. La differenza tra le due sta nell'utilizzo della chiave segreta.
Crittografia simmetrica: la crittografia simmetrica si basa sull'utilizzo di un'unica chiave che chi codifica e decodifica devono condividere. È il metodo perfetto per codifiche locali, come la cifratura di dischi rigidi, ma viene anche utilizzato per le comunicazioni cifrate (spesso in combinazione con la crittografia asimmetrica). L'algoritmo più utilizzato per questo tipo di codifica è l'AES.
L'utilizzo di una singola chiave lo rendono il metodo più semplice e veloce, tuttavia vi è il problema di come condividere la chiave in modo sicuro. Per ovviare a questo problema, si sono studiate delle soluzioni come il protocollo Diffie-Hellman che consente la generazione di una chiave partendo da valori comuni pubblici. In questo modo, due utenti possono concordare la chiave tramite canali non sicuri ma senza mai condividere la chiave segreta.
La crittografia simmetrica presenta un ulteriore problema quando la comunicazione deve avvenire tra più utenti. Infatti, se ci sono n utenti serviranno n(n-1)/2 chiavi per poter garantire la comunicazione sicura tra tutti quanti (una chiave diversa per ogni coppia).
Crittografia asimmetrica: la crittografia asimmetrica si basa invece su due chiavi distinte, una privata e una pubblica. Queste due chiavi sono generate tramite metodi matematici in modo tale che le due siano complementari e quindi ciò che viene codificato con la chiave privata può essere decodificato solo dalla chiave pubblica, e viceversa. In questo modo è necessario che solo la chiave privata rimanga segreta mentre la chiave pubblica viene solitamente resa disponibile a tutti gli utenti. Non avviene

quindi mai il passaggio della chiave segreta rendendo il metodo molto più sicuro della crittografia simmetrica. Di contro, la generazione delle chiavi e la codifica/decodifica sono più lenti e complessi.
I due algoritmi più utilizzati per la crittografia sono i seguenti.

a) Rivest–Shamir–Adleman (RSA): alla base vi è l'utilizzo dei numeri primi e la difficoltà nel fattorizzarli.

Esempio di meccnaismo di cifratura e decifratura RSA

1. Generazione delle chiavi

Presi due numeri primi:

p = 5

q = 11

Calcoli:

*1. n = p * q = 5 * 11 = 55*

*2. φ(n) = (p-1)(q-1) = 4 * 10 = 40*

3. Preso e = 3, che è coprimo con 40

*4. Si trova d tale che 3 * d ≡ 1 (mod 40).*

*→ d = 27, perché 3 * 27 = 81 ≡ 1 (mod 40)*

Chiavi finali:

- Chiave pubblica: (n, e) = (55, 3)

- Chiave privata: (n, d) = (55, 27)

2. Cifratura

Messaggio: m = 12 (con 0 ≤ m < n)

Calcoli: c = m^e mod n = 12^3 mod 55

- 12^2 = 144 ≡ 34 (mod 55)

*- 12^3 = 34 * 12 = 408 ≡ 23 (mod 55)*

Risultato: messaggio cifrato c = 23

3. Decifratura

m = c^d mod n = 23^27 mod 55

Si riducono i calcoli:

- 23^2 = 529 ≡ 34 (mod 55)

- 23^4 = 34^2 = 1156 ≡ 1 (mod 55)

*- 27 = 4 * 6 + 3 → 23^27 ≡ (23^4)^6 * 23^3 ≡ 1^6 * 23^3 (mod 55)*

- 23^3 = 782 ≡ 12 (mod 55)

Risultato: messaggio decifrato m = 12

4. Conclusione

- Messaggio originale: m = 12

- Cifrato: c = 23

- Decifrato: m = 12 (coincide con l'originale)

Legenda:

φ(n) – funzione di Eulero

mod n – modulo di n

≡ – congruente

b) Elliptic Curve Cryptography (ECC): basato su operazioni sul campo ellittico. Più efficiente del RSA.
   Questo tipo di crittografia offre diverse applicazioni che variano in base alla direzione in cui viene utilizzata la codifica, ovvero se si cifra con la chiave pubblica o con la chiave privata.

Inoltre, l'utilizzo della crittografia asimmetrica consente di ottenere diverse proprietà fondamentali per la sicurezza delle comunicazioni digitali.
Confidenzialità: serve per passare un'informazione segreta al possessore della chiave privata. La codifica avviene con la chiave pubblica e la decodifica con la chiave privata, così si ha la certezza che solo il destinatario la possa leggere. Nei casi di comunicazione bidirezionale è necessario che mittente e destinatario possiedano due coppie di chiavi pubbliche e private, così che rispettivamente possano codificare il messaggio con una chiave pubblica ed essere sicuro che solo l'altro interlocutore la potrà leggere.
Autenticità: questa applicazione viene chiamata "firma digitale". La crittografia avviene con la chiave privata quindi, chiunque in possesso della chiave pubblica può effettivamente essere sicuro che un certo messaggio sia stato cifrato proprio dal possessore della chiave privata. Questo metodo, oltre a garantire che la provenienza sia l'utente possessore della chiave privata (solo lui può averla codificata) può anche garantire l'integrità del messaggio e il non ripudio.
Ovvero:

- Autenticità: solo il possessore della chiave privata può aver firmato il messaggio.
- Integrità: il destinatario non può modificare il messaggio firmato in nessun modo.
- Non ripudio: il mittente non può ripudiare un messaggio firmato perché lui è l'unico possessore della chiave.

Quest'ultimo utilizzo della crittografia asimmetrica è un elemento importante che è possibile applicare anche nel sistema di autenticità degli NFT.

6.2 Certificati Digitali

I certificati digitali sono dei documenti elettronici necessari per associare una chiave pubblica a una certa entità reale. Questo è necessario perché la crittografia con chiave pubblica e privata non è sufficiente per garantire l'identità del firmatario. Il meccanismo di crittografia asimmetrica garantisce che solo il proprietario della chiave privata abbia firmato il documento ma non ci dà informazioni sulla sua vera identità.
I certificati digitali fanno parte di un sistema più ampio chiamato PKI (Public Key Infrastacture), ovvero l'infrastruttura che permette la gestione e l'utilizzo dei certificati stessi e, più in generale, di comunicare tra utenti e macchine in modo sicuro su Internet, verificandone l'identità.
Al centro della PKI ci sono le Certification Authority. Queste sono organizzazioni terze e fidate che rilasciano i certificati digitali per le coppie di chiavi, garantendo così l'identità del titolare e l'affidabilità del sistema. Inoltre, le CA sono responsabili di validare l'autenticità di un certificato.
Il funzionamento è il seguente: la CA valida l'identità del richiedente. La validazione dipende dal livello di sicurezza del certificato richiesto. Una volta verificata l'identità, si procede con l'assegnazione della chiave pubblica e privata. Questo può avvenire in due modalità: il richiedente fornisce la propria chiave pubblica dimostrando di possedere la chiave privata; oppure è la CA stessa a generare chiave pubblica e chiave privata. Completata questa parte, la CA emette effettivamente il certificato e lo firma digitalmente con la sua chiave. Procede quindi con la pubblicazione del certificato nelle repository pubbliche, rendendolo così accessibile a chi deve verificarne la validità. La CA ha anche la possibilità di revocare dei certificati nel caso questi non siano più validi. Esistono due tipi di CA:

a. *Root CA*: sono le CA di partenza per tutto il sistema PKI, il loro certificato è auto firmato perché non esistono CA superiori. Sono il fulcro della fiducia di tutto il sistema dei certificati. Per questo motivo, devono essere molto ben protette: solitamente non vengono esposte online e sono custodite in ambienti protetti. Non emettono i certificati direttamente agli utenti finali ma li emettono solamente per altre CA così da creare un layer di mezzo e aumentare la sicurezza.

b. *CA intermedie*: sono le CA che fanno parte del layer intermedio. Il loro certificato viene rilasciato dalle Root CA e si occupano di emettere i certificati per i soggetti richiedenti. Queste, a loro volta, possono avere sotto di sé altre CA intermedie, aumentando così i layer e incrementando i livelli di sicurezza del PKI. Se viene compromessa una CA intermedia infatti è sufficiente revocare il suo certificato (e di conseguenza tagliare via il ramo dei certificati generati da questa CA) senza compromettere l'intera rete. Creare delle CA intermedie permette anche di categorizzarle: spesso nella pratica si usano CA separate per scopi specifici, ad esempio, CA dedicate ai certificati per la firma digitale e altre per i certificati usati nel protocollo TLS.

6.3 Applicazione della firma nel caso di NFT

Applicare una firma digitale ad un NFT equivale alla sottoscrizione di un documento conferendogli di conseguenza le caratteristiche di autenticità, non ripudio e integrità. Se l'integrità era già fornita dal sistema delle blockchain, lo stesso non si può dire per le prime due proprietà.

Di seguito vengono illustrati i metodi possibili per firma digitale su NFT.

a) **Firma tramite metadati**: il creatore può allegare una firma digitale nel campo dei metadati dell'NFT, come una prova della creazione dell'opera. La firma può essere una rappresentazione crittografica che dimostra che la chiave privata del creatore è stata utilizzata per creare l'NFT.
b) **Firma tramite smart contract**: un altro approccio potrebbe essere l'uso di uno smart contract personalizzato per "firmare" l'NFT. In questo caso, l'NFT potrebbe includere una funzione che certifica che il creatore ha approvato la creazione e l'emissione dell'NFT tramite una firma digitale legata al suo wallet.
c) **Creazione di un hash dell'oggetto fisico**: se un NFT è legato a un oggetto fisico, come una scultura o un'opera d'arte, il creatore potrebbe creare un hash univoco dell'oggetto (ad esempio, un'immagine dell'oggetto o una descrizione digitale) e firmarlo digitalmente con la sua chiave privata. Questo hash verrebbe poi utilizzato come riferimento nell'NFT.

6.4 Wallet

La crittografia asimmetrica è anche alla base della tecnologia blockchain. Infatti, tutte le operazioni crittografiche (firma di un blocco, validazione dei blocchi presenti, effettuare una transazione ecc…), vengono effettuate tramite l'utilizzo delle chiavi private e delle chiavi pubbliche. Non solo i nodi validatori, ma qualsiasi utente della blockchain ha necessità di avere un wallet. Questi contengono la chiave privata con cui

ci si autentica sulla blockchain e sono alla base di tutte le operazioni.
È importante quindi, specificare che i wallet contengono unicamente le chiavi per autenticarsi e non la criptovaluta, i token o gli NFT che rimangono sempre e solamente salvati sulla blockchain.
Esistono due categorie di wallet:

1. *Wallet custoditi*: sono wallet forniti da servizi terzi che conservano le credenziali dell'utente, alle quali è possibile accedere tramite internet e utilizzarle come un servizio web. Ci si collega quindi al sito del custodial provider tramite il quale si può interagire con la blockchaine.
2. *Non custodial wallet*: questi invece, non sono custoditi da una terza parte e sono totale responsabilità dell'utente. Possono essere di varie tipologie, come applicazioni desktop o supporti fisici come chiavette USB. Questi permettono di avere una maggiore privacy perché non vengono mai condivisi con nessuno ma tutta la gestione della loro sicurezza e segretezza è demandata all'utente.

7. Persistenza degli NFT

Com' è stato detto precedentemente, una volta che un NFT viene caricato sulla catena questo rimane immutabile. Per efficienza della blockchain stessa, è sconsigliato caricare l'opera digitale direttamente sulla blockchain ma le best practices prevedono di caricare solamente i metadata riferiti all'opera e di mantenere fisicamente quest'ultima in uno store separato, raggiungibile tramite un indirizzo sull'NFT. Questo mantenimento esterno alla catena però non garantisce che lo store non venga modificato.

A titolo esemplificativo, si consideri un dipinto con la sua cornice. Nell'universo blockchain, l'NFT rappresenta la cornice: le caratteristiche di immutabilità, tracciabilità, e possesso rimangono sulla cornice e non vengono trasferite al dipinto stesso. Questo significa che la tela potrebbe venire sostituita senza che le cronice venga modificata. Si è reso necessario quindi pensare a una soluzione che garantisse la corretta associazione e integrità dell'opera digitale rispetto al relativo NFT.

Da quanto detto, emergono quindi due distinti problemi: l'immutabilità dell'opera sullo store esterno e la gestione dello store. Quest'ultima è un problema non indifferente poiché il valore dell'NFT è legato indistricabilmente all'opera associata. La mancanza di garanzie sullo store che lo mantiene può incidere sul valore e sulla fiducia dell'NFT stesso. A fronte di questa problematica, può essere utilizzato come soluzione alternativa un sistema di storage distribuito.

7.1 IPFS

InterPlanetary File System o IPFS, è un protocollo peer to peer strutturato per la gestione e condivisione dei dati in modalità decentralizzata e sicura. Solitamente si accede alla maggior parte delle risorse in internet tramite protocollo HTTP che è basato sulla posizione delle risorse. In HTTP si arriva alla risorsa tramite un URL che rappresenta il percorso gerarchico della struttura della rete.

Per esempio: **http://host/dir1/dir2/resource**

Qui, si può leggere dal path la posizione della **resource** all'interno di HOST (**host** contiene **dir1** che contiene a sua volta **dir2** e **resource** si trova all'interno di questa). Il path è unico perché con quell'indirizzo si potrà recuperare solamente quella risorsa. Il protocollo http è quindi centralizzato perché ha necessità di avere una posizione specifica nella rete (host) per poter sviluppare il path gerarchico. Se l'host non è raggiungibile per dei malfunzionamenti della rete la risorsa diventa inaccessibile. Con l'IPFS invece, si abbandona il concetto di posizione-centrico passando a un modello

contenuto-centrico. Per recuperare una risorsa non è importante dove si trova ma cosa si cerca.

7.1.1 Caratteristiche dell'IPFS

Tra le principali caratteristiche di IPFS vi sono:

a) Verificabilità: IPFS utilizza funzioni di hash per rappresentare i dati. In questo modo è possibile verificare l'autenticità e l'integrità dei dati salvati, proteggendoli da manipolazioni malevole.
b) Resilienza: come riportato nell'introduzione, in caso di malfunzionamento della rete la risorsa può diventare inaccessibile. Se il nodo della rete che possiede fisicamente la risorsa ha un disservizio non è più possibile raggiungerla. Questo è il caso di un *single point of failure*: l'accessibilità alla risorsa è completamente dipendente dalla salute dell'host che la possiede. In IPFS si supera questo problema: il malfunzionamento di uno o più nodi della rete non impatta sul funzionamento della rete intera. Se anche un singolo nodo sull'intera rete possiede la risorsa è ancora possibile recuperarla.
c) Decentralizzazione: i dati sono distribuiti in maniera decentralizzata su tutti i nodi della rete riducendo l'utilizzo dei server centralizzati (principale problema dei single failure point).

7.1.2 Funzionamento

Quando i dati vengono caricati su IPFS, la risorsa viene suddivisa in blocchi, ogni blocco viene "hashato" e gli viene assegnato un Content IDentifier (CID). Il CID è unico ed è ricavato solitamente combinando l'hash del file e il suo codec, ovvero l'etichetta che ne descrive la struttura dati.
Il protocollo IPFS si appoggia al modello IPLD (InterPlanetary Linked Data) ovvero un ecosistema di specifiche e strutture dati che permettono la decentralizzazione delle informazioni e la loro tracciabilità. Secondo questa struttura, i dati vengono separati in blocchi organizzati in grafi aciclici e collegati tra di loro tramite i CID. I grafi prodotti dal modello sono dei Merkle DAG con uno specifico CID root. Il CID root sostituisce quindi il path dell'http e diventa l'identificativo univoco della risorsa nel sistema.

Il Merkle DAG possiede 3 caratteristiche:

a) È direzionato: i vari nodi del grafico sono collegati tra loro tramite archi orientati che indicano una direzione nella dipendenza tra nodi. Nel caso d'uso IPFS vi è

una gerarchia tra i nodi ma in generale non è necessaria (esempio di due nodi con lo stesso figlio).

b) È aciclico: non presenta cicli, ovvero un nodo non può mai direzionare a un suo predecessore.
c) Utilizza funzioni Hash: ogni nodo è elaborato tramite a una funzione di hash, ottenendo quindi per ciascuno un identificativo (CID). Tramite il CID ogni nodo è riconoscibile in modo univoco ed immutabile.

È necessario fare distinzione tra il concetto di Merkle DAG e Merkle tree introdotto precedentemente: entrambi condividono l'utilizzo delle funzioni di hash sui nodi ma differiscono nella forma e negli usi pratici.
Il Merkle tree è una struttura ad albero binario in cui ogni nodo interno contiene l'hash dei suoi figli e tutti i dati si trovano nei nodi foglia.
Il Merkle DAG, invece, è un grafo diretto aciclico dove i nodi possono avere più genitori e le strutture non devono rispettare vincoli di bilanciamento o binarietà. Ogni nodo è identificato da un hash del suo contenuto (inclusi i link ai figli) e può essere condiviso da più percorsi nel grafo. Quindi, il Merkle tree è una sottocategoria del Merkle DAG con vincoli strutturali più rigidi.

Il funzionamento vero e proprio del protocollo si divide in 3 fasi: content addressing, providing e retriving.
Content Adressing: questa fase avviene a livello locale sul primo nodo che effettua il caricamento. In questo step, il file viene diviso in più blocchi, ognuno dei quali viene "hashato" ottenendo un CID che sarà univoco per ogni blocco. L'esatto procedimento di separazione dipende dal formato del file caricato.
Providing: in questa operazione ogni nodo indica agli altri nodi della rete che possiede un determinato CID (una determinata risorsa). Per fare questo, aggiorna la DHT (Distributed Hash Table) in cui vengono salvate le informazioni che indicano per ogni peer i CID che conserva. Questa operazione deve essere fatta periodicamente perché ogni elemento nella DHT ha un "Time To Live" e dopo questo tempo viene scartato. Ogni nodo ha una porzione della DHT.
Retrieving: questa è l'operazione che avviene quando un nodo vuole recuperare un file dalla rete. Si appoggia alla DHT per trovare quale peer possiede i CID a cui è interessato. La DHT nei nodi, non copre solitamente l'intera rete quindi, se i nodi contattati non possiedono il CID richiesto inoltrano la richiesta ai nodi della propria DHT. A questo punto, contatta tutti i peer recuperati chiedendo i CID corrispondenti in maniera iterativa (protocollo BitSwap). Se li possiedono lo inviano, altrimenti rispondono in maniera negativa. È possibile ricevere anche risposte negative perché le informazioni sulla DHT possono essere vecchie (il peer non possiede più quel CID) o un peer che lo possiede può essere offline. Il peer richiedente può così crearsi un mapping di quali nodi

invieranno precisi CID. Il nodo utilizza il Merkle DAG per sapere quali CID compongono una risorsa e come mapparli.
Quando iniziano ad arrivare i vari blocchi, il peer li ricostruisce tramite il Merkle DAG. I blocchi corretti vengono aggiunti correttamente, quelli che non rispettano il DAG (modificati o corrotti) vengono scartati.

7.2 Filecoin

Com'è già stato anticipato, una risorsa rimane nella rete finché almeno un nodo lo possiede localmente ("pinned"). Se però tutti gli utenti la eliminano viene perso. Riservare uno spazio locale per salvare i dati comporta dei costi, inoltre non si ha la certezza che tutti gli utenti intendano mantenere la risorsa salvata. Di conseguenza il protocollo IPFS da solo non garantisce la persistenza dei dati sulla rete ma si occupa solo della loro gestione e distribuzione in modalità peer to peer.
Viene quindi introdotto *Filecoin*. Sviluppato anch'esso da Protocol Labs, è una blockchain che utilizza le stesse tecnologie distribuite alla base di IPFS, come l'indirizzamento tramite hash e la struttura dati IPLD, ma aggiunge un sistema di ricompense per gli utenti che conservano i dati nel tempo.
Lo scopo di Filecoin è quindi ricompensare chi sta effettivamente conservando i file caricati nel sistema. Deve ottenere la prova che il file sia effettivamente conservato da un utente e che questo stia conservando esattamente il numero di copie che dichiara di conservare (non quindi duplicare i dati localmente al momento della verifica). Segue un esempio di processo con l'utilizzo di IPFS.
Quando un client desidera conservare una risorsa in maniera persistente all'interno dell'ecosistema IPFS/Filecoin, il processo segue una serie di passaggi ben definiti. In primo luogo, il file viene caricato su IPFS, generando così un CID che rappresenta in modo univoco il contenuto digitale. Successivamente, il client stipula un accordo di archiviazione (deal) con un miner, tramite il protocollo P2P di Filecoin. A questo punto, il file viene trasmesso direttamente al miner attraverso diversi canali, come IPFS, Bitswap o HTTP. Una volta ricevuto, il miner procede a replicare il file e a sigillarlo mediante il protocollo di Proof-of-Replication (PoRep). Infine, il miner invia una prova crittografica on - chain che dimostra di averlo replicato correttamente.
È importante notare che tutto il traffico dei dati di grandi dimensioni avviene off-chain, direttamente tra i nodi della rete, mentre la blockchain memorizza esclusivamente le prove crittografiche e i metadati, senza contenere i dati veri e propri. Questo approccio consente di mantenere la sicurezza e l'integrità dei contenuti, ottimizzando l'efficienza del sistema.

7.2.1 Funzionamento

I dati distribuiti sulla rete Filecoin vengono divisi in "sector" ovvero le unità base di archiviazione su Filecoin.
Hanno una dimensione fissa di 32/64 GB e un tempo di vita limitato e prefissato. Il tempo di vita equivale alla promessa degli storage minter di conservare il contenuto sul proprio spazio di archiviazione per uno specifico periodo: inizialmente questo è di 18 mesi ma può essere rinnovato dal minter alla sua scadenza.
È quindi necessario una tipologia di validazione differente rispetto a quelle viste precedentemente (Proof-of-work e proof of-stake).
Filecoin utilizza due diversi algoritmi per validare in modo efficace che gli utenti stiano mantendo i dati sullo storage.
Il primo è *Proof-of replication*: si applica quando il miner riceve i dati da salvare sul proprio storage. Quando riceve un sector lo sigilla tramite un algoritmo di PoRep (il protocollo di Filecoin prevede lo Stacked DRG PoRep ma è previsto il passaggio al Narrow Stacked Expander PoRep con miglioramenti delle prestazioni e una riduzione dei costi). Un algoritmo di PoRep è un processo computazionale intensivo che permette la creazione di un sector unico. I dati vengono cifrati e compressi più volte utilizzando una "randomness", creando una progressione di livelli di codifica dipendenti dalla causalità stessa e dal contenuto codificato nei livelli precedenti. Durante questo processo, chiamato sealing sector, vengono costruiti due Merkle tree: uno per i dati originali e uno per i dati codificati.
La "randomness" è un elemento fondamentale per assicurare l'unicità del sector: questa è imprevedibile, così che il minter non possa creare in anticipo la prova. In questo modo rende il sector non manipolabile. È però necessario che sia anche verificabile. Filecoin prevede due diverse sorgenti di randomness: DRAND e VRF.
Una volta completato il processo di sealing, il minter genera la PoRep proof, denominata SNARK. Nello SNARK vengono passati:

1. SectorId: l'identificativo del sector sigillato
2. CommD: è il Merkle Root dei dati archiviati prima del processo di sigillatura
3. CommC: è il Merkle root costruito da ogni livello di codifica del processo
4. CommRLast: è il Merkle root dell'ultimo livello di codifica
5. CommR: è la concatenazione del commC e del CommR

Tramite lo SNARK la rete può validare rapidamente che il sector sia stato sigillato correttamente e che quindi il minter stia effettivamente riservando dello spazio fisico per quel sector.

Il secondo algoritmo è *Proof-of-Spacetime*. Come abbiamo detto Filecoin ricompensa gli utenti che mantengono le risorse salvate nel tempo. Quindi oltre a provare che

effettivemante i minter abbiano riservato dello spazio univoco per le risorse, devono dimostrare che lo continuano a mantenere anche nel futuro. Per dimostrarlo devono superare due differenti prove:

1. *WinningPoSt*: ad ogni epoch l'algoritmo di validazione sceglie un numero ristretto di candidati per la creazione dei nuovi blocchi. I minter scelti però, prima di potersi aggiudicare la creazione, devono fornire una prova che stanno effettivamente archiviando i sector che intendono proporre nel blocco. La finestra temporale nella quale devono vincere la sdifa è molto ristretta così da non avere il tempo di recuperare i dati da un altro nodo (data retrieval attack).
2. *WindowPoSt*: tutti i storage miners devono periodicamente presentare prove che continuano a conservare tutti i settori promessi (ad esempio ogni 24 ore). Il periodo è suddiviso in più "deadline" (es. ogni ~30 minuti) entro le quali occorre rispondere. In caso di mancata prova si incorre in penalità.

Expected Consensus è l'algoritmo di validazione utilizzato da Filecoin. L'algoritmo ogni 30 secondi ("epoch"), effettua un'elezione segreta di alcuni nodi partecipanti alla creazione del nodo successivo (stabilito a 5). La lunghezza della epoch e il numero di nodi eletti è una stima basata sulla probabilità che in quel lasso di tempo ci saranno un certo numero di nodi che intendono proporre un blocco alla rete. L'elezione dei miner avviene attraverso l'*ElectionProof*: i miner generano un ticket per provare che sono stati eletti in quell'epoch. La generazione del ticket sfrutta le due sorgenti di "randomness" di Filecoin: DRAND e VRF.
DRAND è un protocollo pubblico e distribuito per la generazione di valori casuali verificabili e imprevedibili. I minter recuperano un random beacon generato dal DRAND. Questo beacon sarà uguale per tutti per tutta l'epoch ed è necessario per la verifica una volta eletti i nodi.
A questo punto passano il randomness, insieme alla loro chiave privata, come input della sorgente di casualità VRF, eventualmente passando anche il ticket precedente così da creare una correlazione tra i ticket.
Anche il VRF è una funzione verificabile di generazione di valori randomici basata sulla dualità chiave pubblica e chiave privata. Viene utilizzato il VRF per generare un ulteriore valore casuale passando come seed il randomness fornito dal DRAND. Tramite la chiave pubblica tutti gli altri nodi possono verificare che non ci siano state manipolazioni. Il valore estratto sarà un numero compreso tra 0 e 1 denominato "ticket". Più basso è il "ticket" e più probabilità si avrà di vincere. Il risultato del VRF infatti viene messo a confronto con la probabilità di vincita del nodo. Quest'ultima è data dalla formula $\mathbf{1-P[X=0]}$ dove X è la distribuzione di Poisson con λ

$$\lambda = \textbf{(potenza del nodo)/(potenza totale)*(numero di nodi estratti per epoch)}$$

Come specificato sopra il numero di nodi estratti per “epoch” è di 5 per default. La potenza del nodo invece è calcolata in percentuale in base a quanto spazio di archiviazione ha messo a disposizione per la rete.
Se il numero estratto è più piccolo della probabilità di vincita allora si aggiudica la creazione del blocco.
Poichè Filecoin si basa su un sistema probabilistico è soggetto ad attacco Sybil. Un’attacco Sybil consiste nel dividere la propria potenza tra più sottoentità: distribuire le risorse su più miner fittizi garantirebbe più probabilità di vittoria. La potenza sulla rete è data dalla percentuale di spazio di archiviazione messo a disposizione.
Il WinCount, infatti, tiene conto di possibili vittorie multiple per un singolo nodo rendendo inutile la divisione del peso su più entità. In questo modo il singolo nodo può aggiudicarsi più vittorie per un singolo epoch. Infatti, se il numero estratto è molto basso è possibile che sia inferiore a

$$1-P[X=0]-P[X=1]$$

che equivale alla probabilità che 1 nodo vinca due estrazioni. La probabilità può andare avanti con sempre un maggior numero di vittorie: in questo caso il Wincount sarà sempre maggiore.
Una volta che il nodo ha vinto il round, gli altri nodi verificano con facilità che non abbia barato: recuperano il random beacon specifico di quell’epoch, verificano che il ticket generato da questo sia corretto tramite chiave pubblica e verificano anche il suo WinCount.
A questo punto, tutti i nodi eletti, prima di proporre un blocco, devono vincere la sfida WinningPost che è già stata specificata.

I blocchi proposti vengono raggruppati in “Tipset” e propagati nella rete. Poiché la propagazione non è immediata, i nodi possono calcolare i blocchi successivi partendo da tipset diversi. Il risultato è la creazione di catene divergenti. È necessario che ad un certo punto i vari tipset vengano risolti in un'unica catena valida. I vari nodi quindi quando ricevono un nuovo tipset lo confrontano con quello salvato in locale, scegliendo quello più pesante. Il peso si calcola come:

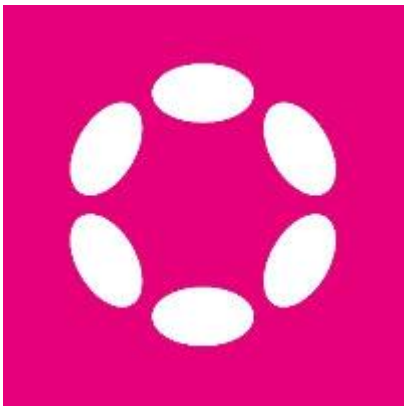
Peso del tipset = ParentWeight + + (wPowerFactor[r+1] + wBlocksFactor[r+1]) * 2^8

dove:

- **ParentWeight**: è il peso cumulativo del tipset genitore.
- **wPowerFactor[r+1]**: basato sulla potenza di archiviazione. È una variabile che cresce proporzionalmente in base logaritmica.
- **wBlocksFactor[r+1]**: basato sul numero di blocchi prodotti nel round corrente.
- **2^8**: utilizzato per evitare perdita di numeri decimali nelle operazioni

Se ci sono due tipset con lo stesso peso allora vengono considerati i ticket dei blocchi con punteggio più basso; se i primi due sono anch'essi uguali, si passa al secondo più basso e così via.
In Filecoin esistono due tipologie di nodi che possono eventualmente sovrapporsi: i miner, che sono coloro che propongono i blocchi della catena e che partecipano al meccanismo di consenso, e i retrival node.
Questi sono necessari perché il sistema di Filecoin è pensato per premiare la conservazione dei dati ma il recupero di questi può essere anche molto lento.

8. Polkadot



Polkadot è una blockchain che nasce con l'obbiettivo di permettere l'interoperabilità tra diverse blockchain in maniera semplice. Attraverso il suo network, infatti, è possibile sviluppare blockchain che permettono l'integrazione con quelle già esistenti.

Si basa su 3 elementi fondamentali:

1. La Relay Chain: è la rete blockchain strutturale di Polkadot. Su questo layer vengono finalizzate le transazioni e avvengono tutte le operazioni per garantire sicurezza e validità dei blocchi.
   Poiché Polkadot ha, per sua struttura, necessità di essere molto veloce, la fase di convalida delle transazioni aggiunte non avviene nel momento in cui il blocco viene creato ma in un secondo momento. Questo permette di ridurre di molto il tempo di creazione dei blocchi (a seguito dei risultati di alcuni test di performance nel 2020 ha dimostrato di poter elaborare più di 1000 transazioni al secondo).
2. Le para-chain: sono le blockchain che vengono sviluppate a partire dalla Relay-Chain. Permettono di aggiungere funzionalità e soprattutto le compatibilità necessarie per integrarsi con le altre blockchain esterne all'ecosistema Polkadot. Mantengono però la sicurezza, garantita dalla Relay-Chain.
3. Bridge: questi sono gli elementi fondamentali per l'integrazione di diverse blockchain. Permettono l'integrazione di transazioni tra le varie blockchain e il passaggio di "valore" dall'una all'altra

Anche Polkadot utilizza il sistema di valiidazione POS. Inoltre, il fatto di possedere dei DOT (token di Polkadot), permette di spenderli per poter prendere decisioni sulla rete.

Vi sono 3 tipologie di utenti.
I detentori di DOT: chiunque possieda dei DOT può spenderli per accettare o rifiutare una modifica proposta sulla rete. Possono proporre loro stessi delle modifiche alla rete (anche se il prezzo per farlo è elevato).
Il Consiglio: sono coloro che propongono le modifiche alla rete Polkadot. Vengono votati dai detentori di DOT e una loro proposta di modifiche è meno costosa.
Il Comitato tecnico: sono i team che sviluppano attivamente codice su Polkadot. Possono fare richieste speciali in caso di emergenza e sono nominati dal Consiglio.
Attraverso Polkadot è quindi possibile fare uscire un NFT dall'ecosistema Ethereum. Per farlo si può utilizzare la parachain Moonbeam che è sviluppata appositamente per essere compatibile con gli smartcontract di Ehtereum.

9. Sviluppo effettivo

Quello che il trattato vuole proporre è quindi un'ipotesi di sviluppo di un Marketplace di NFT gestito da un'azienda che si fa da garante della gestione dei servizi e dell'intermediazione tra i committenti degli NFT e i validatori delle opere. L'obiettivo principale è creare un sistema sicuro, trasparente e verificabile dove la fiducia non è legata all'azienda stessa ma è garantita dall'integrazione delle tecnologie sopra descritte. L'azienda può eventualmente offrire servizi in supporto alle soluzioni eccessivamente complicate che il creatore dell'NFT non intende adoperare (ad esempio lo storage decentralizzato). In questo modo possono venire offerti servizi su misura per le diverse richieste.

Di seguito viene riportato il codice dello smart-contract che implementa le funzionalità necessarie e vengono analizzati i vari passaggi funzionali. Successivamente vengono evidenziati alcuni punti aperti o criticità e proposte le opportune soluzioni tecniche.

9.1 Alcune informazioni per una corretta lettura del codice

Event e Function:

In Solidity ci sono due tipologie di metodi: Event e Functions. È importante distinguerle per capire bene il loro utilizzo.
Le *function* servono per effettuare le logiche principali negli smart contract e sono le uniche che possono apportare modifiche alla struttura della catena (aggiungere transazioni, modificare variabili etc…)
Gli *event* invece, sono operazioni utilizzate principalmente per la comunicazione tra lo smart contract e gli enti esterni alla catena e sono richiamabili solamente dallo smart contract che le dichiara. Solitamente sono triggerate all'interno di una determinata *function*. Un esempio pratico può essere la funzione "mint" dove viene creato un token seguita dall'event notifyMint, che invia la comunicazione della creazione ad altri agenti.

Data Location:

Solidity permette 3 diversi livelli di data location per dare la possibilità allo sviluppatore di gestire la memoria in maniera più efficace, anche perché l'utilizzo di memoria che richiede transazioni sulla blockchain ha un costo.

- Storage: questa allocazione di memoria è un'allocazione permanente sulla blockchain.
- Memory: questo tipo di allocazione è temporaneo a livello di funzione. Viene allocato lo spazio solo per il periodo di elaborazione della funzione e poi viene liberato.

- Calldata: ha la stessa funzione temporanea di “Memory” ma è uno spazio di sola lettura. È utilizzato per allocare i parametri delle funzioni invocate.

9.2 Codice

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.17;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/utils/Counters.sol";
import "@openzeppelin/contracts/utils/cryptography/ECDSA.sol";


contract ContrattoPoCNFT is ERC721, Ownable {
    using Counters for Counters.Counter;
    using ECDSA for bytes32;


    Counters.Counter private contatoreTokenId;

    struct InfoNFT {
        string riferimentoAsset;
        string riferimentoDoc;
        bool operaFisica;
        address certificatore;
        uint256 timestampCreazione;
  string tipoStorage;
    }

    mapping(uint256 => InfoNFT) public opereDigitali;
    mapping(address => bool) public isCertificatore;
    mapping(address => uint256) public nonceCert;

    constructor(string memory name_, string memory symbol_) ERC721(name_,
symbol_) { }

    function aggiungiCertificatore(address certificatore) external onlyOwner {
        require(certificatore!= address(0), "certificatore zero");
        isCertificatore[certificatore] = true;
    }

    function rimuoviCertificatore(address certificatore) external onlyOwner {
        isCertificatore[certificatore] = false;
    }
event NFTMintato(
        uint256 indexed tokenId,
```

```
        address indexed to,
        bool operaFisica,
        address indexed certificatore,
        string riferimentoAsset,
        string riferimentoDoc,
        string tipoStorage
    );

function mintOperaDigitale(
        address to,
        string memory riferimentoAsset,
        string memory tipoStorage
    ) external returns (uint256) {
        require(bytes(riferimentoAsset).length > 0, "riferimentoAsset
richiesto");

        contatoreTokenId.increment();
        uint256 tid = contatoreTokenId.current();
        _safeMint(to, tid);

        opereDigitali[tid] = InfoNFT({
            riferimentoAsset: riferimentoAsset,
            riferimentoDoc: "",
            operaFisica: false,
            certificatore: address(0),
            timestampCreazione: block.timestamp,
            tipoStorage: tipoStorage
        });


        emit NFTMintato(tid, to, false, address(0), riferimentoAsset, "",
tipoStorage);
        return tid;
    }
function verificaCertificatore(
        address to,
        string memory riferimentoAsset,
        string memory riferimentoDoc,
        bool operaFisica,
        uint256 nonce
    ) internal view returns (bytes32) {
        return keccak256(
            abi.encodePacked(
                "\x19Ethereum Signed Message:\n32",
                keccak256(abi.encodePacked(to, riferimentoAsset,
riferimentoDoc, operaFisica, nonce))
            )
        );
    }
```

```
function mintOperaFisica(
        address to,
        string memory riferimentoAsset,
        string memory riferimentoDoc,
        string memory tipoStorage,
        address certificatore,
        bytes memory signature
    ) external returns (uint256) {
    // check dei requirement necessari
        require(bytes(riferimentoAsset).length > 0, "riferimentoAsset
richiesto");
        require(bytes(riferimentoDoc).length > 0, "riferimentoDoc richiesto
per opere fisiche");
        require(certificatore != address(0), "certificatore zero");
        require(isCertificatore[certificatore], "certificatore non
autorizzato");

        uint256 nonce = nonceCert[certificatore];
        bytes32 hash = verificaCertificatore(to, riferimentoAsset,
riferimentoDoc, true, nonce);
        address recovered = hash.recover(signature);
        require(recovered == certificatore, "firma non valida");

        // increment nonce per il certificatore che firma
        nonceCert[certificatore] = nonce + 1;

        contatoreTokenId.increment();
        uint256 tid = contatoreTokenId.current();
        _safeMint(to, tid);

        opereDigitali[tid] = InfoNFT({
            riferimentoAsset: riferimentoAsset,
            riferimentoDoc: riferimentoDoc,
            operaFisica: true,
            certificatore: certificatore,
            timestampCreazione: block.timestamp,
            tipoStorage: tipoStorage
        });

        emit NFTMintato(tid, to, true, certificatore, riferimentoAsset,
riferimentoDoc, tipoStorage);
        return tid;
    }

event RichiestoTrasferimentoBridge(
        uint256 indexed tokenId,
        address indexed owner,
        string blockchainTarget,
```

```
        string indirizzoTarget
    );

    function burnForBridge(uint256 tokenId, string memory blockchainTarget,
string memory indirizzoTarget) external {
        require(_isApprovedOrOwner(msg.sender, tokenId), "non owner o
approved");

        address owner = ownerOf(tokenId);
        emit RichiestoTrasferimentoBridge (tokenId, owner, blockchainTarget,
indirizzoTarget);

        delete opereDigitali[tokenId];
        _burn(tokenId);
    }

    function mintFromBridge(
        address to,
        string memory riferimentoAsset,
        string memory riferimentoDoc,
        bool operaFisica,
        address certificatore,
        string memory tipoStorage
    ) external onlyOwner returns (uint256) {

        contatoreTokenId.increment();
        uint256 tid = contatoreTokenId.current();
        _safeMint(to, tid);

        opereDigitali[tid] = InfoNFT({
            riferimentoAsset: riferimentoAsset,
            riferimentoDoc: riferimentoDoc,
            operaFisica: operaFisica,
            certificatore: certificatore,
            timestampCreazione: block.timestamp,
            tipoStorage: tipoStorage
        });

        emit NFTMintato(tid, to, operaFisica, certificatore, riferimentoAsset,
riferimentoDoc, tipoStorage);
        return tid;
    }
function tokenArtwork(uint256 tokenId) external view returns (InfoNFT memory)
{
        require(_exists(tokenId), "token inesistente");
        return opereDigitali[tokenId];
    }
```

```
    function _baseURI() internal view virtual override returns (string memory)
{
        return "";
    }


    function tokenURI(uint256 tokenId) public view virtual override returns
(string memory) {
        require(_exists(tokenId), "token inesistente");
        return opereDigitali[tokenId].riferimentoAsset;
    }
}
```

9.3 Analisi Codice

```
pragma solidity ^0.8.17;


import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/utils/Counters.sol";
import "@openzeppelin/contracts/utils/cryptography/ECDSA.sol";
```

L'utilizzo delle librerie OpenZeppelin permette di importare molte funzionalità utili, sicure e già testate per la gestione dei token. Nel dettaglio le librerie indicate servono per:

- ERC721.sol: contiene i metodi necessari ad implementare il protocollo ERC-721
- Ownable.sol: permette di limitare determinate funzionalità riservate solamente al livello di autorizzazione all'owner
- Counters.sol: offre funzionalità utili per gestire contatori incrementali
- ECDSA.sol: utilizzato per la verifica delle signatures

```
contract ContrattoPoCNFT is ERC721, Ownable {
    using Counters for Counters.Counter;
    using ECDSA for bytes32;


    Counters.Counter private contatoreTokenId;

    struct InfoNFT {
        string riferimentoAsset;
        string riferimentoDoc;
        bool operaFisica;
        address certificatore;
        uint256 timestampCreazione;
  string tipoStorage;
    }
```

```
    mapping(uint256 => InfoNFT) public opereDigitali;
    mapping(address => bool) public isCertificatore;
    mapping(address => uint256) public nonceCert;
```

In questa sezione è definita la struttura dell'NFT con le informazioni utili per gestire le logiche future.

- riferimentoAsset: indirizzo all'opera rappresentata. Come abbiamo detto l'idea è di lasciare a chi compra la possibilità di decidere dove tenerlo ed eventualmente se utilizzare una tipologia di storage centralizzato o distribuito. Potremmo quindi avere un URL o anche un link IPFS.
- riferimentoDoc: questo elemento è facoltativo. In caso di opere d'arte fisiche potrebbe essere necessario accompagnare un attestato di veridicità (ad esempio un certificato). Questo contiene l'indirizzo alla documentazione specifica.
- operaFisica: booleano che indica se l'opera rappresentata è solamente digitale o ha un corrispettivo fisico.
- certificatore: se è presente la documentazione che attesta la veridicità dell'opera viene fornito l'indirizzo dell'ente certificatore di modo che si possa effettuare la verifica on-chain della firma di quest'ultimo.

I due mapping invece sono la struttura dati dove vengono mappati rispettivamente:

- gli owner con l'NFT
- gli address degli autenticatori per l'opera rappresentata

```
    constructor(string memory name_, string memory symbol_) ERC721(name_,
symbol_) { }
```

Costruttore utilizzato al momento della creazione del contratto (deploy sulla blockchain). Viene passato un nome e un simbolo per specificare il nome della collection NFT. Richiama a sua volta il costruttore del contratto ERC721 per implementare tutte le funzionalità del protocollo.

```
    function aggiungiCertificatore(address certificatore) external onlyOwner {
        require(certificatore!= address(0), "certificatore zero");
        isCertificatore[certificatore] = true;
    }

    function rimuoviCertificatore(address certificatore) external onlyOwner {
        isCertificatore[certificatore] = false;
```

```
    }
```

Sono le 2 funzioni per poter aggiungere o rimuovere i certificatori dalla lista di quelli abilitati. Sono operazioni permesse solamente al proprietario dello smart contract (onlyOwner). In questo modo solamente chi ha mintato lo smart contract può gestire i certificatori. Può essere una logica utile da avere nel caso venga aggiunta una verifica ulteriore di validità o eventualmente se qualche certificatore non sia più valido.

```
event NFTMintato(
        uint256 indexed tokenId,
        address indexed to,
        bool operaFisica,
        address indexed certificatore,
        string riferimentoAsset,
        string riferimentoDoc,
        string tipoStorage
    );

function mintOperaDigitale(
        address to,
        string memory riferimentoAsset,
        string memory tipoStorage
    ) external returns (uint256) {
        require(bytes(riferimentoAsset).length > 0, "riferimentoAsset
richiesto");

        contatoreTokenId.increment();
        uint256 tid = contatoreTokenId.current();
        _safeMint(to, tid);

        opereDigitali[tid] = InfoNFT({
            riferimentoAsset: riferimentoAsset,
            riferimentoDoc: "",
            operaFisica: false,
            certificatore: address(0),
            timestampCreazione: block.timestamp,
            tipoStorage: tipoStorage
        });


        emit NFTMintato(tid, to, false, address(0), riferimentoAsset, "",
tipoStorage);
        return tid;
    }
```

Questa è la funzione per mintare un NFT senza corrispettivo fisico.

9.3.1 Creazione NFT e firma in caso di opera fisica:

```
function verificaCertificatore(
        address to,
        string memory riferimentoAsset,
        string memory riferimentoDoc,
        bool operaFisica,
        uint256 nonce
    ) internal view returns (bytes32) {
        return keccak256(
            abi.encodePacked(
                "\x19Ethereum Signed Message:\n32",
                keccak256(abi.encodePacked(to, riferimentoAsset,
riferimentoDoc, operaFisica, nonce))
            )
        );
    }

function mintOperaFisica(
        address to,
        string memory riferimentoAsset,
        string memory riferimentoDoc,
        string memory tipoStorage,
        address certificatore,
        bytes memory signature
    ) external returns (uint256) {
    // check dei requirement necessari
        require(bytes(riferimentoAsset).length > 0, "riferimentoAsset
richiesto");
        require(bytes(riferimentoDoc).length > 0, "riferimentoDoc richiesto
per opere fisiche");
        require(certificatore != address(0), "certificatore zero");
        require(isCertificatore[certificatore], "certificatore non
autorizzato");

        uint256 nonce = nonceCert[certificatore];
        bytes32 hash = verificaCertificatore(to, riferimentoAsset,
riferimentoDoc, true, nonce);
        address recovered = hash.recover(signature);
        require(recovered == certificatore, "firma non valida");

        // increment nonce per il certificatore che firma
        nonceCert[certificatore] = nonce + 1;

        contatoreTokenId.increment();
        uint256 tid = contatoreTokenId.current();
        _safeMint(to, tid);
```

```
        opereDigitali[tid] = InfoNFT({
            riferimentoAsset: riferimentoAsset,
            riferimentoDoc: riferimentoDoc,
            operaFisica: true,
            certificatore: certificatore,
            timestampCreazione: block.timestamp,
            tipoStorage: tipoStorage
        });

        emit NFTMintato(tid, to, true, certificatore, riferimentoAsset,
riferimentoDoc, tipoStorage);
        return tid;
    }
```

Questa funzione serve per mintare un NFT collegato a un'opera fisica e gestire la firma da parte dell'ente certificatore.
La firma dell'opera avviene nei seguenti passaggi:

1. L'ente certificatore produce la documentazione ufficiale che ne attesta la veridicità e la sottoscrive con una firma qualificata fornita da una CA autorizzata. Questo passaggio avviene off-chain ed è necessario per dare una valenza legale all'operazione.
2. La documentazione viene salvata su uno storage e si referenzia con l'indirizzo nella `riferimentoDoc` dell'NFT.
3. Sempre l'ente certificatore a questo punto, genera un hash utilizzando il suo indirizzo, l'indirizzo dell'asset, l'indirizzo della riferimentoDoc e un nonce; lo firma con la chiave privata del suo wallet. Questo è necessario per validare ,anche sulla blockchain, che ciò che verrà salvato sull'NFT è quello che l'ente certificatore ha firmato. Il nonce è un elemento importante per la sicurezza della firma: il suo utilizzo permette di evitare l'attacco multi-sign aggiungendo un elemento "casuale" ed unico così da non poter replicare la stessa firma successivamente. Il nonce viene salvato nel mapping `nonceCert`.
4. L'NFT, a questo punto, può essere "mintato" tramite la function `mintOperaFisica`. La funzione verificaCertificatore verifica al momento del mint che i dati passati siano effettivamente ciò che l'ente validatore ha certificato. Se dall'hash estrae la sua chiave pubblica allora è corretto e si può procedere, altrimenti fallisce. Il doppio passaggio sotto la funzione keccak256 serve per preparare l'hash nel formato corretto per Ethereum. In alternativa, si può usare la libreria ECDSA.sol sempre fornita da OpenZeppelin. Naturalmente, le stesse operazioni per generare l'hash devono essere fatte anche lato ente validatore.

9.3.2 Possibilità di passaggio crossChain

```
event RichiestoTrasferimentoBridge(
        uint256 indexed tokenId,
        address indexed owner,
        string blockchainTarget,
        string indirizzoTarget
    );

    function burnForBridge(uint256 tokenId, string memory blockchainTarget,
string memory indirizzoTarget) external {
        require(_isApprovedOrOwner(msg.sender, tokenId), "non owner o
approved");

        address owner = ownerOf(tokenId);
        emit RichiestoTrasferimentoBridge (tokenId, owner, blockchainTarget,
indirizzoTarget);

        delete opereDigitali[tokenId];
        _burn(tokenId);
    }

    function mintFromBridge(
        address to,
        string memory riferimentoAsset,
        string memory riferimentoDoc,
        bool operaFisica,
        address certificatore,
        string memory tipoStorage
    ) external onlyOwner returns (uint256) {

        contatoreTokenId.increment();
        uint256 tid = contatoreTokenId.current();
        _safeMint(to, tid);

        opereDigitali[tid] = InfoNFT({
            riferimentoAsset: riferimentoAsset,
            riferimentoDoc: riferimentoDoc,
            operaFisica: operaFisica,
            certificatore: certificatore,
            timestampCreazione: block.timestamp,
            tipoStorage: tipoStorage
        })
        emit NFTMintato(tid, to, operaFisica, certificatore, riferimentoAsset,
riferimentoDoc, tipoStorage);
        return tid;
    }
```

In questa sezione è implementato il codice per gestire il passaggio su una catena esterna e per immetterlo nuovamente su Ethereum.
`burnForBridge` deve essere richiamata dal possessore del token altrimenti fallisce. Questa emette l'event verso il bridge e cancella le informazioni dell'NFT e il tokenId. La `mintFromBridge` invece viene richiamata per ri-immettere nella chain il token dalla blockchain esterna.

9.4 Possibili problemi e modifiche

Di seguito vengono riportati alcuni punti critici nel codice sopra mostrato ed eventuali risoluzioni.
In primo luogo, periste il problema che la firma qualificata non è garantita on-chain. Una possibile soluzione consisterebbe nel pubblicare il documento della firma qualificata su un database e inserire l'indirizzo sull'NFT come campo aggiuntivo.
Un ulteriore aspetto da considerare riguarda la struttura dello smart contract utilizzato, il quale al momento tratta unicamente la casistica di NFT single asset. È possibile aggiungere la funzionalità di gestire NFT multiasset modificando riferimentoAsset in un array. In caso di opere fisiche bisognerebbe anche gestire il mapping delle riferimentoDoc con lo specifico riferimentoAsset. Per questa opzione però sarebbe corretto passare al protocollo ERC-1155, già pensato per gestire gli NFT multiasset.
Un'altra problematica tecnica è la condivisione del "nonce" tra la catena e il certificatore. Poiché è necessario che venga utilizzato lo stesso nonce per la verifica della documentazione, si potrebbe ipotizzare un web service fornito dall'azienda. Potrebbe condividere in modo sicuro al certificatore le informazioni usate dalla funzione verificaCertificatore o direttamente l'hash da firmare. Il certificatore, quindi, dovrebbe solamente accedere al sito, firmare l'hash in locale con la sua chiave e rinviare il tutto all'applicazione che si occuperebbe di inserirlo nell'NFT.

Si evidenzia inoltre un'ulteriore criticità relativa al passaggio dell'NFT su una chain esterna. Nel caso di trasferimento ad una blockchain diversa da Ethereum, lo smart contract effettua uno burn del token una volta emesso l'evento verso il bridge di destinazione. Questo passaggio però, è molto rischioso perché lo effettua senza che il bridge abbia dato conferma dell'operazione. Può accadere che il token venga "bruciato" ma che l'operazione di passaggio sulla seconda chain fallisca. In questo modo il token sarebbe perso per sempre. Ci possono essere due soluzioni adottabili:

a. Two-Step-Commit: qui l'emissione dell'evento non è seguita automaticamente dal burn del token bensì rimane in attesa della conferma del mint lato bridge prima di effettuarla.
b. Lock and Mint: in questo caso invece, non avviene mai il burn del token. Quando viene inviato l'event per il passaggio a una chain esterna, il token viene "congelato" sulla catena sorgente (si può implementare la logica sullo smart

contract in modo da non rendere possibile nessuna operazione su quell'NFT specifico). Se il passaggio sulla catena target fallisce, oppure se viene immesso nuovamente nella catena sorgente, viene rilasciato.

Il primo metodo però comporta anche lui una criticità: nel periodo tra l'invio dell'evento e la ricezione della risposta dal bridge, il token è ancora attivo sulla chain sorgente. Può quindi esser trasferito e venduto senza limitazioni. Dunque, è consigliabile utilizzare il secondo metodo dove non avviene mai il burn della risorsa. MoonBeam implementa il Lock and Mint.
Per la funzione di immissione del token poi è specificata solamente come onlyOwner. In questo caso potrebbe essere una buona idea introdurre un ruolo bridge e permettere direttamente quest'ultimo di richiamare la funzione.

Infine la tipologia di storage è stata lasciata libera. Nella sezione è stata indicata genericamente come indirizzo o riferimento. Si intende lasciare la decisione di utilizzare uno storage centralizzata o decentralizzato al committente dell'NFT. In questo caso si possono ipotizzare due possibili scenari. In caso di scelta centralizzata potrebbe venire utilizzato un cloud datacenter, eventualmente gestito dall'azienda stessa. In questo caso la fiducia riguardo la permanenza e l'immutabilità degli allegati viene affidata completamente all'azienda gestrice dello storage.

Conclusioni

Nella seguente dissertazione sono state trattate le tecnologie alla base degli NFT, in particolare su Ethereum. Si è analizzata la loro struttura, il funzionamento e in che modo viene resa sicura e trasparente. È stato evidenziato anche come il sistema non manchi di punti critici: l'architettura di un market di NFT gestito deve quindi integrare una serie di soluzioni atte a risolverli. Nel corso dell'analisi, infatti, è emerso come il valore degli NFT non risieda tanto nell'oggetto digitale rappresentato, quanto nel sistema di garanzie e trasparenza che la blockchain conferisce. Questo vale specialmente per l'identità degli autori e delle opere. Il fatto che un'azienda reale ben identificabile faccia da intermediario tra creatori e acquirenti, è già di per sé una misura di sicurezza. Tuttavia, per aderire alla filosofia di Ethereum, che mira a separare il più possibile le logiche di processo dalla fiducia in un'entità centralizzata, si è ricercata una soluzione aggiuntiva. La proposta di utilizzare una firma digitale riconosciuta a livello legale permette di separare l'affidabilità dell'azienda da quella del certificatore. La soluzione proposta consente inoltre che il validatore e il creatore dell'opera siano soggetti esterni all'azienda, mantenendo così un sistema realmente decentralizzato. La loro interazione con l'azienda avviene esclusivamente nella fase di creazione dell'NFT, quando i dati e le certificazioni vengono integrati nella blockchain.

Allo stesso tempo, si è evidenziato che la permanenza e la sicurezza delle opere dipendono fortemente dalle modalità di conservazione. L'utilizzo di un sistema decentralizzato garantisce che l'NFT venga caricato in maniera immutabile sulla rete e non lasciato quindi su store centralizzati, che potrebbero essere modificati dai proprietari del datacenter stesso (anche se poco probabile). Il sistema Filecoin in più, permette che i dati salvati siano garantiti e mantenuti nel tempo.

Non si è voluto tralasciare il punto di vista pratico. Infatti, implementare e integrare le diverse soluzioni proposte ha necessariamente dei costi maggiori. Nello sviluppo di un mercato di NFT di questo genere, si è voluto ragionare non solo come esercizio teorico ma con l'ottica di un'azienda che intende creare un profitto da questa attività. Obbligare l'utilizzo di tutto il sistema "di contorno" rischia di rendere il bacino di utenti che sono disposti ad affrontare il costo elevato, relativamente basso. Per evitare quindi di creare un mercato d'élite si è volutamente deciso di lasciare la possibilità di sfruttare anche altre soluzioni che rendono di l'NFT meno costoso. Nello specifico, infatti, il codice proposto dà la possibilità di utilizzare un sistema centralizzato, come ad esempio un cloud, per "storare" il corrispettivo digitale dell'NFT. In questo modo si evitano i costi di integrare il sistema Filecoin. L'altro elemento più evidente è la possibilità lasciata agli utenti di "mintare" degli NFT senza un corrispettivo fisico. In questo modo anche artisti minori o semplici utenti possono accedere al servizio con costi ridotti senza l'operato di un ente validatore.

Tuttavia, l'utilizzo pratico di strumenti analizzati non sempre è alla portata di tutti gli utenti e può risultare complesso per gli utenti meno esperti. Per questo motivo, l'azienda

può offrire servizi di supporto aggiuntivi, come piattaforme di firma digitale o interfacce semplificate per la creazione e la gestione degli NFT, con l'obiettivo di rendere il sistema accessibile senza compromettere i principi di trasparenza e decentralizzazione che lo caratterizzano.

Nel complesso, il lavoro ha mostrato come sia possibile progettare un marketplace NFT che coniughi trasparenza, verificabilità e sostenibilità operativa, integrando tecnologie decentralizzate con soluzioni aziendali orientate all'usabilità e alla fiducia.

Bibliografia

[1] Roberto Garavaglia, *TUTTO SUGLI NFT: Crypto art, token, blockchain e loro applicazioni,* Milano, Hoepli, 2022

[2] Boiardi Luca, *INVESTIRE IN BITCOIN E CRIPTOVALUTE Lo studio dei fondamentali, le strategie d'investimento e i segreti della finanza decentralizzata*, Milano, Hoepli, 2022

[3] Paolo Maria Gangi, *DIRITTO DEI NON-FUNGIBLE-TOKEN: Disciplina generale, proprietà intellettuale, aspetti regolamentari*, Lavis, Giappichelli, 2024

[4] Satoshi Nakamoto, *BITCOIN: A Peer-to-Peer Electronic Cash System*, 2008 *[p. 8,16]*

[5] Vitalik Buterin, *ETHEREUM: A Next-Generation Smart Contract and Decentralized Application Platform,* 2014 *[p. 22]*

[6] Nick Szabo, *SMART CONTRACTS: Building Blocks for Digital Markets*, 1997 [p. 26]

[7] Berti R., Spoto F., Zumerle F., "*NFT: che cosa sono, come funzionano, come investire sui non fungible token."*
URL *www.agendadigitale.eu/documenti/nft-che-cosa-sono-come-funzionano-come-investire-sui-non-fungible-token/ [p 32, 33]*

[8] *Borsa Italiana*
URL *https://www.borsaitaliana.it/notizie/sotto-la-lente/quali-sono-i-rischi-legati-agli-nft.htm [p. 33, 34]*

[9] *Standard token non fungibile ERC-721*
URL *https://ethereum.org/it/developers/docs/standards/tokens/erc-721 [p. 30, 31]*

[10] *Standard token non fungibile ERC-20*
URL *https://ethereum.org/it/developers/docs/standards/tokens/erc-20 [p. 29]*

[11] *Standard Multi-Token ERC-1155*
URL *https://ethereum.org/it/developers/docs/standards/tokens/erc-1155 [p. 31]*

[12] *Bitpanda - Il problema della scalabilità nella rete Bitcoin*
URL *https://www.bitpanda.com/academy [p 13]*

[13] *Ethereum Improvement Proposals*
URL *https://eips.ethereum.org/EIPS/eip-721 [p. 30,31]*

[14] *Cointelegraph Italia*
URL *https://it.cointelegraph.com/*

[15] *Coinbase – "Qual è la differenza tra Optimistic Rollups e ZK-Rollups?"*
URL *https://www.coinbase.com/it/learn/tips-and-tutorials/what-is-the-difference-between-optimistic-rollups-and-zk-rollups [p. 28]*

[16] *Bitcoin Developer*
URL *https://developer.bitcoin.org/ [p. 17]*

[17] *Blockchain*
URL *it.wikipedia.org/wiki/Blockchain [p. 10, 11, 12]*

[18] *Binance Academy*
URL *https://www.binance.com/en/academy*

[19] *Entrust - Securing a world in motion*
URL *https://www.entrust.com/resources/learn/ [p. 37,40]*

[20] *Filecoin Docs*
URL *https://docs.filecoin.io/ [p. 46,47,48,49]*

[21] *IPFS*
URL *https://docs.ipfs.tech/ [p. 44,45]*

[22] *Opensea Developers - Metadata Standard*
How to add rich metadata to your ERC721 or ERC1155 NFTs
URL *docs.opensea.io/docs/metadata-standards [p. 31, 32]*

[23] *IPLD – Documentation*
URL *https://ipld.io/docs/ [p. 44]*

[24] *MIT Media Lab*
URL *https://www.media.mit.edu/ [p. 14]*

[25] {Solidity} – *Documentation*
URL *https://docs.soliditylang.org/en/v0.8.30/ [p.53]*

[26] *Teleborsa – "Allarme educazione finanziaria: poco più di 1 italiano su 10 ha competenze finanziarie accettabili"*
URL *https://www.teleborsa.it/News/2025/09/02/allarme-educazione-finanziaria-poco-piu-di-1-italiano-su-10-ha-competenze-finanziarie-accettabili-143.html [p. 6]*

[27] *Young Platform*
URL *https://youngplatform.com/ [p. 6,7]*

[28] *Polkadot*
URL https://docs.polkadot.com/ [p. 51]

[29] *ZeroUno - "Così la blockchain può essere applicata al comparto aereo"*
URL https://www.zerounoweb.it/blockchain/cosi-la-blockchain-puo-essere-applicata-al-comparto-aereo/ [p. 14]

[30] *KBA - "Patricia trie: a predestined blockchain thing"*
URL https://kba.ai/6771-2/ [p. 14]