LUISS

Dipartimento di Economia e Finanza

Cattedra Teoria e Gestione del Portafoglio

Blockchain e Smart Contract

Prof. Nicola Borri
RELATORE

Prof. Federico Carlini
CORRELATORE

Emanuele Rampioni Matr.710581
CANDIDATO

Anno Accademico 2019/2020

Indice

Introduzione

La blockchain è una nuova tecnologia di cui spesso si sente parlare in ambiti diversi, completamente eterogenei tra loro, che finora ha trovato prevalente applicazione nel mondo Bitcoin. Tuttavia, associarla unicamente alla criptovaluta più famosa al mondo è erroneo e riduttivo.

La blockchain infatti a livello mondiale si sta affermando in diversi settori. Il primo tra questi è quello finanziario, a causa della sua capacità di trasferimento di denaro oltre i confini internazionali senza passare per un intermediario, con basse commissioni ed in maniera quasi istantanea. Un altro settore è quello della cybersecurity poiché la blockchain utilizza chiavi crittografiche che nascondono e non alterano il messaggio fino a quando quest'ultimo non arrivi all'effettivo destinatario. Infine, anche il voto elettorale beneficia molto dell'introduzione della blockchain perché permette l'autenticazione degli elettori, la conservazione dei registri di voto in maniera sicura e un'attività di spoglio precisa e trasparente.

Oltre a questi tre ambiti, ce ne sono molti altri coinvolti dalla blockchain, principalmente a causa del concetto di fiducia che quest'ultima riesce ad assicurare; per questo motivo ritengo che, prima di poter decidere dove la blockchain rappresenti un effettivo vantaggio e non un costo dettato dall'utilizzo di una tecnologia nuova ma non necessaria (magari per scopi prevalentemente pubblicitari), sia necessario conoscere in modo preciso il suo funzionamento e quindi entrare in campo informatico.

L'obiettivo della prima parte del mio elaborato è quello di esaminare in maniera dettagliata le fasi di trasmissione delle transazioni da un blocco all'altro, in modo da cercare di far capire il meccanismo informatico della blockchain e le innovazioni da essa apportate (userò quella di Bitcoin giudicata da tutti come l'interpretazione del concetto massimo di blockchain).

La seconda parte dell'elaborato riguarda la definizione e la compilazione di alcuni smart contracts. Essi vengono definiti "protocolli di transazione informatizzati, che eseguono i termini di un contratto", dato che, come dice Nick Szabo (colui a cui si fa risalire la creazione degli smart contracts) "traducendo le clausole contrattuali in codici e incorporandoli in hardware o software in grado di auto applicarle, è possibile ridurre al minimo la necessità di intermediari tra le parti ed eccezioni dannose o accidentali". Tuttavia, nonostante la famosa nomea che hanno raggiunto, rimane ancora una certa difficoltà nel capire dove e come i contratti intelligenti possano essere scritti e poi fatti funzionare.

Per questo motivo mi sono posto l'obiettivo di comprendere come si creassero a livello informatico, se fossero realmente funzionanti tramite applicazioni reali e come si lanciassero su una vera blockchain; il tutto è stato fatto in un'ottica finanziaria ovvero ho cercato il più possibile di collegare il mondo blockchain e smart contracts a quello della finanza. Se si vuole investire in tecnologie innovative supportate da blockchain e smart contracts, bisogna conoscere in cosa si sta investendo e per farlo bisogna avere delle nozioni informatiche che ci permettano di comprendere il loro funzionamento.

Infatti, nella parte relativa agli smart contracts prima ho dato alcune nozioni base per programmare su Solidity, il linguaggio di programmazione comunemente usato per la compilazione dei contratti intelligenti, e poi li ho usati in campo finanziario.

Per esempio, ho creato uno smart contract che serve per gestire un conto corrente bancario. Ho ideato tre funzioni: una per vedere il bilancio, una per ritirare soldi e una per depositarli.

Un secondo smart contract che serve per scrivere per sempre messaggi e codici in blockchain. Attraverso la precisione, l'immutabilità e la crittografia della blockchain, le catene di distribuzione beneficerebbero dell'introduzione dei contratti intelligenti.

Un terzo smart contract che serve per la creazione di un token ERC-20 che potrebbe essere lanciato sulla blockchain di Ethereum. Infatti, molte start up e non solo stanno decidendo di non finanziarsi più mediante IPO o con normali mezzi di crowdfunding, ma piuttosto mediante ICO (Initial Coin Offering) in cui creano un loro token privato e lo vendono ai possibili investitori. Ho scritto il codice per lanciare il token e ho ipotizzato che anche la LUISS possa ideare un proprio token per coinvolgere attivamente gli studenti nelle decisioni universitarie.

In sintesi, la tesi cerca di avvicinare blockchain e smart contracts al mondo della finanza, utilizzando tuttavia un approccio informatico che considero imprescindibile se si vuole capire realmente queste due importanti innovazioni che rivoluzioneranno il nostro futuro.

In che direzione stiamo andando?

Il mondo sta attualmente vivendo un periodo di importanti cambiamenti sotto il profilo Tech, tali per cui una fase di rivoluzione tecnologica è probabilmente in corso d'opera.

Alcuni dati: nel 2020 sono presenti al mondo 7,4 miliardi di persone, ma i dispositivi connessi ad Internet sono 30 miliardi; viene stimata una media di 6,58 devices connessi per consumatore e sono 20 i miliardi di euro da spendere entro il 2020 all'interno dell'UE in investimenti per l'Intelligenza Artificiale.

Viviamo quindi in una fase storica in cui ci sono più dispositivi connessi che persone, in cui cresce il numero di individui con accesso ad Internet h24 e in cui il 44% dei bambini di età inferiore ad un anno hanno già usato dispositivi intelligenti, come uno smartphone o un tablet. Infatti, secondo uno studio presentato dal Pediatric Academic Societies, condotto su un campione di 370 genitori di bambini piccoli di età compresa tra 6 mesi e 4 anni, il 97% delle famiglie possiede varie televisioni distribuite per casa, l'83% ha tablets, il 77% ha smartphones e il 59% ha una connessione ad Internet. Per quanto riguardo i figli con età inferiore ad un anno, il 52% ha guardato la TV, il 36% ha toccato o switchato uno schermo di un dispositivo, il 24% ha chiamato qualcuno e il 15% ha usato un'app. Insomma, basta pensare a soli 20 anni fa per capire che questo scenario sarebbe stato impensabile.

Quale è il background che ha portato a questa situazione?

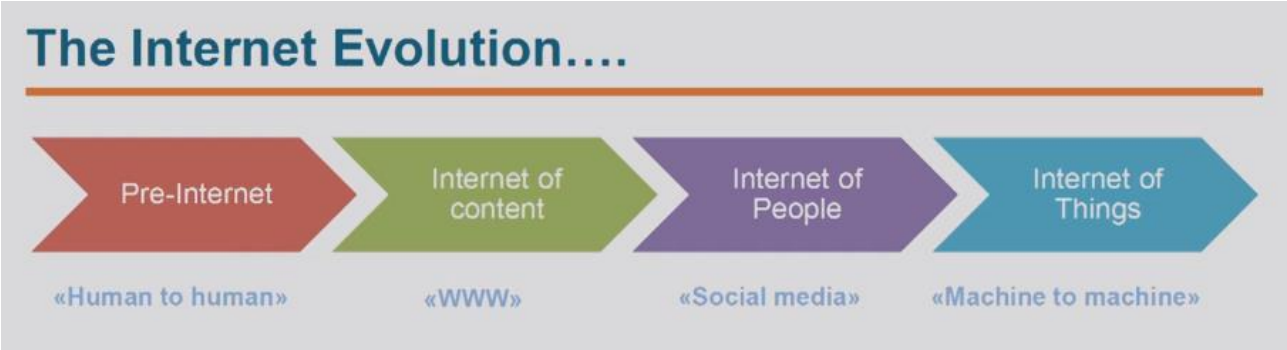


Figura 0-1- L'evoluzione di Internet – Giorgio Angiolini – Blockchain Week Rome 2019

Si possono suddividere le fasi dello sviluppo tecnologico moderno prendendo come riferimento una delle più grandi innovazioni della storia dell'umanità: Internet.

La prima fase può essere definita come "Human to human" ed è quella in cui l'uomo per comunicare usava solo la voce o il telefono e si dava maggiormente importanza al rapporto interpersonale, in particolare alle opinioni, ai valori ed alle emozioni (come dice Bryan Kramer nel suo libro Human to Human: #H2H "I prodotti non hanno emozioni. Le persone le hanno. Le persone vogliono sentire qualcosa. E le persone commettono errori")

La seconda fase è quella dell'Internet of content ("WWW") e riguarda il "Boom" di Internet, con l'entrata di miliardi di persone all'interno della rete, la creazione di siti web e l'evoluzione della piattaforma come mezzo imprescindibile per la società.

La terza fase riguarda l'Internet of people con lo sviluppo dei social media, con tutti i pro e i contro che hanno portato e stanno portando ("I social media danno diritto di parola a legioni di imbecilli che prima parlavano

solo al bar dopo un bicchiere di rosso, senza danneggiare la collettività. Venivano subito messi a tacere, mentre ora hanno lo stesso diritto di parola di un Premio Nobel“ Umberto Eco, 2015)

La quarta fase, non ancora pienamente raggiunta, ma verso la quale ci stiamo dirigendo, è l'Internet of Things ("Machine to Machine").

Internet of Things (IoT) descrive un modello di tecnologia digitale che sfrutta Internet con lo scopo di migliorare la vita delle persone. Infatti, l'IoT si basa sulla connessione ad Internet di miliardi di smart devices e sensori, i quali collezionano e condividono milioni di informazioni che possono essere usate da parte di governi, società, ospedali ed individui. L'IoT è nato e si sta sviluppando in seguito all'invenzione di processori a basso costo e connessioni wireless: per ora si stima che in futuro un terzo dei dispositivi intelligenti saranno computers, smartphones e tablets mentre i restanti due terzi saranno invece "oggetti" che ad oggi non sono "smart" come ad esempio porte del garage, lampade, frigoriferi, vestiti sportivi, semafori e altri apparecchi che verranno inventati, il cui limite oggi è rappresentato solamente dalla nostra immaginazione.[1]

Questo mondo è più vicino di quanto si possa pensare: i ricercatori stimano che ogni nuovo mese oltre 3 milioni di nuovi strumenti saranno connessi ad Internet e che nei prossimi 4 anni ci saranno molto più di 30 miliardi di nuovi devices connessi alla rete.

Come funziona un sistema IoT?

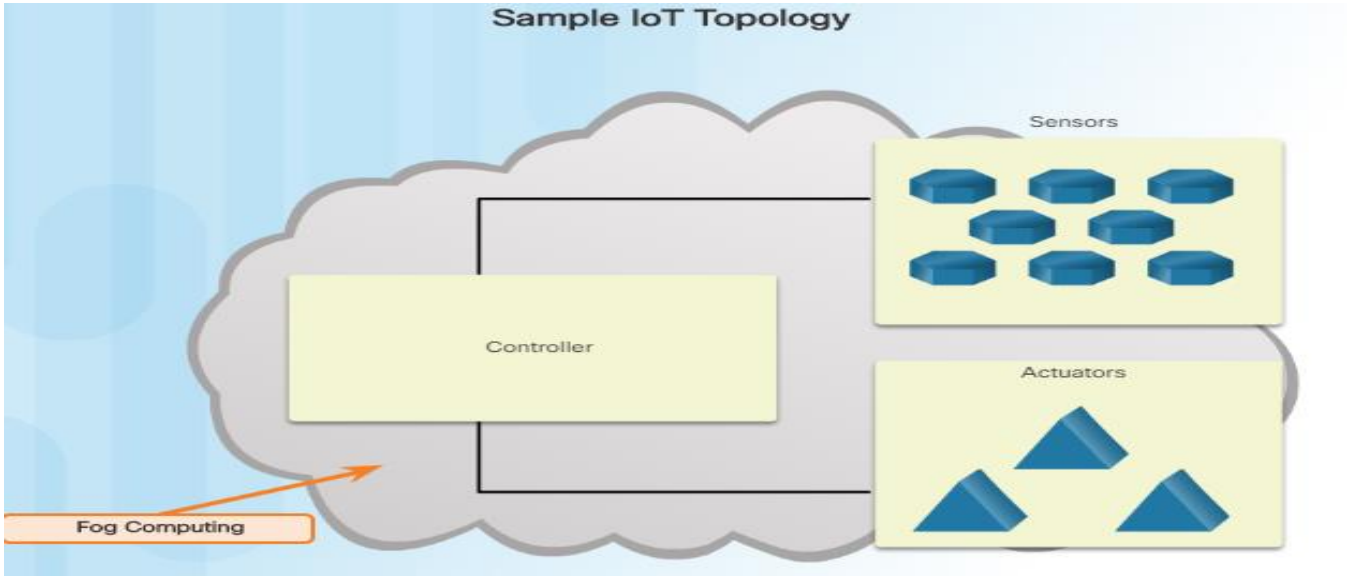


Figura 2 – Esempio di sistema IoT – Intro IoT – Cisco Networking Academy

Un semplice esempio di tecnologia IoT potrebbe essere rappresentato da questo schema: i sensori che raccolgono informazioni nel mondo esterno dovrebbero essere connessi ad un network cosicché le informazioni immagazzinate possano essere trasferite attraverso una rete wireless ad un controller. Il controller, una volta ricevuta l'informazione dai sensori, dovrebbe essere in grado di prendere un'immediata decisione o trasferire l'informazione ad un apparecchio più potente per un'ulteriore analisi. I sensori spesso

[1] Giorgio Angiolini - Blockchain Week Rome 2019 - 2019

lavorano insieme ad un actuator, che è uno strumento che riceve un input elettrico e lo trasforma in un'azione fisica. Un esempio: se un sensore rileva un livello di calore eccessivo in una stanza, manda l'informazione al controller; il controller, a sua volta, gira l'informazione ad un actuator che dovrebbe accendere il condizionatore.

Attraverso un programma di Cisco chiamato Packet Tracer potrebbe essere interessante progettare la propria smart home, smart city, smart factory ecc…

Figura 3 – Packet Tracer – Intro IoT – Cisco Networking Academy

Nel 2020 l'importanza della programmazione è assolutamente innegabile, soprattutto per la formazione di un mondo IoT. In particolare, il comportamento di uno smart device può essere rappresentato da un diagramma di flusso. I diagrammi di flusso sono usati in molti campi, come l'ingegneria, la fisica e ovviamente la programmazione. Il concetto base del diagramma di flusso è che per ciascun input che viene dato al diagramma corrisponde un certo risultato come output che dipende dal SI o dal NO. I programmatori normalmente creano prima un diagramma di flusso che permette loro di capire la logica dietro ogni progetto, indipendente da qualsiasi tipo di linguaggio di programmazione, per poi trasformare il tutto su Python, Java, C++ ecc.. La Figura 4 ne è un esempio:

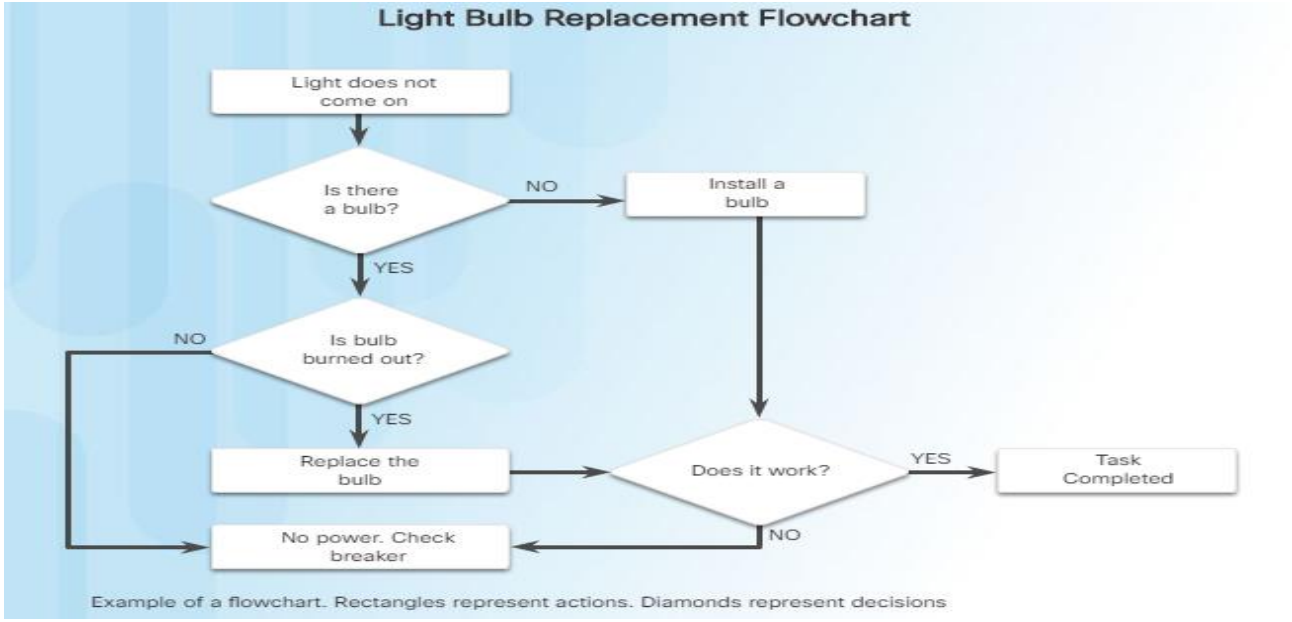


Figura 4 – Esempio di diagramma di flusso – Intro IoT – Cisco Networking Academy

Come detto, il concetto espresso dal diagramma di flusso deve essere trasferito in linguaggio di programmazione. Un linguaggio di programmazione è un linguaggio formale che specifica un insieme di istruzioni non ambigue per produrre un particolare output a partire da un dato input. Un esempio di utilizzo base di un linguaggio di programmazione, in questo caso Python, è rappresentato in Figura 5: questo "mini programma", per un dato input numerico, dice se questo è un anno bisestile come output.

```
year = int(input("Enter a year to check if it is a leap year\n"))
if (year % 4) == 0:
   if (year % 100) == 0:
       if (year % 400) == 0:
           print("{0} is a leap year".format(year))
       else:
           print("{0} is not a leap year".format(year))
   else:
       print("{0} is a leap year".format(year))
else:
   print("{0} is not a leap year".format(year))
```

Figura 5 – Programmazione con Python – Intro IoT – Cisco Networking Academy

Generalmente quando il linguaggio di programmazione è stato scelto e il diagramma di flusso è stato costruito, si può iniziare la creazione di un programma.

Altre innovazioni della rivoluzione tecnologica

Un altro importante aspetto riguarda i Big Data. I Big Data sono un'immensa quantità di informazioni trasmesse dai sensori istallati in differenti oggetti e locations. Non esiste un volume preciso di dati tale per cui, oltre quella soglia, quei dati possano essere classificati come Big Data, ma essi presentano tre caratteristiche: volume, velocità e varietà.

Le aziende non per forza devono generare i loro Big Data, in particolar modo le aziende più piccole potrebbero non avere i sensori, il volume di clienti, o la capacità di generarli; in ogni caso esistono informazioni pronte ad essere usate da tutte le aziende che li stiano cercando indipendentemente dalla loro dimensione.

Quali sono le sfide dei Big Data? Molteplici, ma in particolar modo si concentrano su: - management, poiché i dati provengono da differenti settori aziendali - sicurezza, in quanto i dati raccolti devono essere al sicuro e non divulgati a malintenzionati - analisi, dato che le informazioni arrivano in differenti modi come ad esempio tabelle, grafici ma anche video, email e foto (più difficili da analizzare) - accesso, per cui le informazioni devono essere consultabili dai proprietari in qualsiasi luogo e a qualsiasi ora. Tanto per dare un riferimento numerico riguardante il mondo attuale, IBM stima che ogni minuto in tutto il mondo vengono mandati oltre 3,5 milioni di messaggi. Il volume di dati che si può quotidianamente ricavare è immenso.

I Big Data, oltre che dai sensori, principalmente provengono dai Social Network (Facebook, Instagram, YouTube ecc..), pagine Web, motori di ricerca (Google, Yahoo…), informazioni presenti negli archivi pubblici e privati, Metadata, ovvero informazioni collegate ad esempio alla mail, documenti trasmessi ecc… i quali, una volta raccolti, attraverso il processo di Data Mining, vengono trasformati in informazioni significative per le aziende.

Figura 6 – Big Data – Intro IoT – Cisco Networking Academy

Ma possono le macchine pensare? L'automazione, che è quel processo condotto autonomamente dalle macchine che riduce, e in certi casi elimina, l'intervento umano, in certi casi potrebbe far rispondere positivamente a questa domanda.

L'automazione nasce dal settore manifatturiero, in particolar modo dalla catena di montaggio, quando i processi di assemblaggio dei pezzi di una automobile venivano prima svolti dall'uomo, il quale poi progressivamente è stato sostituito dalle macchine; queste infatti erano eccellenti nel ripetere in maniera continuativa lo stesso task senza incappare in errori o cedere alla fatica. Il risultato è un lavoro di 24 ore, senza

interruzioni, per un prodotto più uniforme. La sua nuova frontiera riguarda la capacità delle macchine di cambiare il proprio comportamento sulla base di agenti esterni, proprio come fanno le persone.
L'automazione tuttavia sta uscendo dall'ambito dell'industria manifatturiera e dal mondo della robotica, dove era stata confinata in questi anni, per entrare nella realtà di tutti i giorni come ad esempio nelle case, macchine, edifici ecc..

Altri due aspetti particolarmente disruptive che si stanno facendo spazio sono l'Intelligenza Artificiale e il Machine Learning.
L'Intelligenza Artificiale (AI) è l'intelligenza dimostrata dalle macchine. Questa ovviamente si pone in contrasto con l'intelligenza naturale che è quella degli organismi viventi. L'AI può cercare di recepire gli input che le derivano dall'ambiente in cui si trova e prendere decisioni per massimizzare la probabilità di raggiungere un obiettivo. Il Machine Learning invece è un sottosettore dell'AI che usa tecniche statistiche per fornire ai computer l'abilità di imparare dal loro ambiente di operatività. Questo gli consente di migliorare in un determinato task senza essere specificatamente programmato per esso. Ciò risulta particolarmente utile quando è difficile o impossibile programmare aprioristicamente specifici algoritmi.

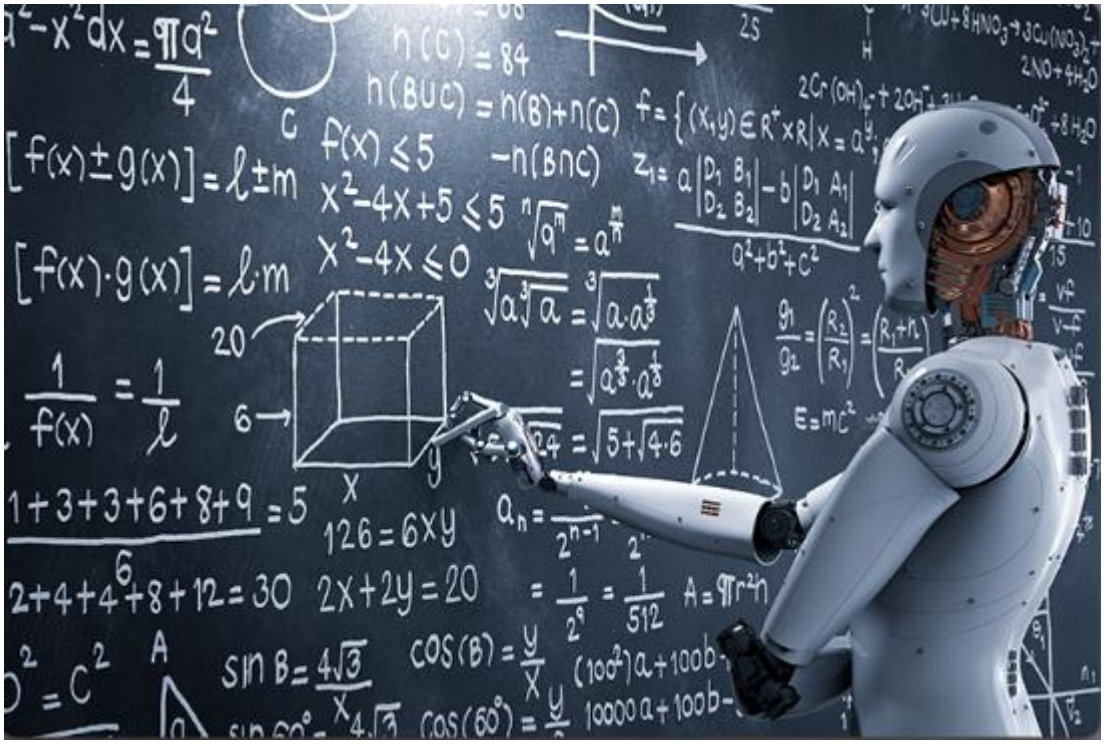


Figura 7 – Intelligenza Artificiale – Intro IoT – Cisco Networking Academy

Quali sono i rischi che si sono manifestati in seguito ad un sistema basato su Internet, in particolare attraverso questa ricerca smaniosa di appropriarsi di data?
Un rischio importante è l'hackeraggio da parte di black hat hackers di informazioni personali dei clienti di qualche azienda. I dati di società o istituzioni governative ad esempio possono contenere informazioni sensibili riguardanti l'azienda, brevetti o addirittura informazioni per la sicurezza nazionale. A causa dello sviluppo di un sistema sempre più basato sulla digitalizzazione, si necessita di "extra" sicurezza per proteggere i dati. Per capire l'importanza della privacy delle informazioni online si possono menzionare alcuni crash: l'attacco

(purtroppo riuscito!) da parte di black hat hakers al sistema di Yahoo e Gmail: gli hackers hanno rubato le credenziali di accesso alla mail ad oltre un milione di clienti per poi rivenderle nel Dark Web. L'attacco hacker a Equifax, una delle più grandi agenzie di credito, che ha comportato la perdita di informazioni personali di 145 milioni di persone: per paura di una ricaduta disastrosa nel mercato, l'attacco subìto è stato rivelato dalla società solamente due mesi dopo. MyFitnessPal, una delle più famose applicazioni per la palestra, si è fatta rubare username, e-mail e password di 150 milioni di utenti e infine Uber che, a causa di una falla nel sistema di sicurezza, ha subìto il furto di 57 milioni di dati di utenti, tenendo per oltre un anno la notizia segreta, dopo il pagamento di 100000 dollari di riscatto.

I pericoli connessi alla sicurezza, per il mondo IoT, derivano in particolar modo da tre situazioni: - il numero crescente di devices connessi: più sono gli apparecchi connessi, più aumenta la possibilità di attacchi informatici, soprattutto perché i devices tendono ad essere tutti diversi, con differenti linguaggi di programmazione, sistemi operativi, CPU ecc.. – la loro location, come ad esempio frigoriferi, lampade, tronchi di albero o addirittura dentro il nostro corpo; questo a volte determina una certa difficoltà nel proteggerli – l'impossibilità di fare upgrade; difatti i sensori potrebbero essere collocati in luoghi inaccessibili dove l'intervento umano potrebbe diventare impossibile oppure potrebbero essere progettati in modo da non essere programmati per effettuare un upgrade: questo comporta che se un device non ha la possibilità di aggiornarsi, rimarrà vulnerabile per tutta la vita. [2]

Come si collocano Blockchain e Smart Contracts in questo contesto?

Blockchain e Smart Contracts si adattano perfettamente all'IoT, AI, Machine Learning, Big Data e CryptoCurrencies e possono essere considerati un solido punto di partenza per lo sviluppo della rivoluzione tecnologica attuale.

La blockchain, ritenuta da molti la vera innovazione portata da Bitcoin, da un punto di vista ideologico risolve uno dei più grandi problemi presenti in un rapporto di scambio: la fiducia. Infatti, all'interno di un rapporto compratore-venditore si è sempre inserita una terza parte (il cosiddetto "Middle Man"), un'entità che si mette di mezzo, che garantisce che una transazione su Internet venga effettuata correttamente, senza il problema del double-spending. Questo concetto è abbastanza radicato nella nostra storia; la banca ne rappresenta un classico esempio. Tuttavia, la società si sta accorgendo che la terza parte spesso non è un valore aggiunto per la transazione, quanto piuttosto un costo sia in termini monetari, sia in termini di fiducia. Si fa strada infatti il concetto di disintermediazione, intesa come il fenomeno di riduzione dei flussi intermediati, ovvero la perdita di importanza del Middle Man, con l'annullamento della sua capacità di intercedere presso i due attori sociali per il raggiungimento dell'accordo. Questo concetto di disintermediazione nel rapporto di scambio si è verificato anche nel settore dei servizi: si pensi ad esempio ad Uber. Uber è un'azienda che fondamentalmente offre un servizio di trasporto senza possedere mezzi di trasporto. Lo stesso avviene per quanto riguarda Airbnb:

[2] Intro IoT - Cisco Networking Academy - 2020

offre servizi di locazione senza avere luoghi fisici di proprietà. Eppure, Uber e Airbnb sono due aziende che hanno "spaccato" il mercato facendo, qualcuno dice anche in maniera non propriamente legale a causa dei vantaggi fiscali, profitti enormi. Potrebbero esistere Uber e Airbnb senza che il cliente avesse pienamente fiducia nel funzionamento di queste due applicazioni?

In Airbnb, ad esempio, il cliente non è chiamato a fidarsi di una società (nel caso di Airbnb possiamo prendere come riferimento una catena di hotel) come è sempre stato finora, bensì di un'altra persona, un qualsiasi privato, che sta mettendo a disposizione la sua casa. Il cliente, a sua volta, potrebbe addirittura passare dall'altra parte del "rapporto", essendo lui in futuro a mettere a disposizione la propria casa (logicamente la stessa cosa non è possibile con un hotel).

Un simile sistema senza la fiducia di ambedue le parti non potrebbe esistere.

Rimanendo su questo concetto, si analizzi come la fiducia stia evolvendo, come mostra la Figura 8.

Figura 8 – L'evoluzione della fiducia – Giorgio Angiolini – Blockchain Week Rome 2019

Il primo esempio è la fiducia che è nata a livello locale, nei piccoli villaggi, all'epoca del baratto. Con l'avvento della moneta e gli scambi in grande quantità, le persone iniziano a non fidarsi più le une delle altre cosicchè nascono gli intermediari, la terza parte, come ad esempio le banche e i governi che garantiscono che le transazioni vengano effettuate nella maniera corretta. Successivamente si passa a quella che viene chiamata la fiducia distribuita, ovvero la fiducia che passa attraverso Social Network, Sharing Economy e viene distribuita tra gli utilizzatori. Si pensi nuovamente ad Airbnb: con il concetto di reputazione e il meccanismo di recensioni, il cliente tende maggiormente a fidarsi di altri clienti che hanno già usufruito del servizio piuttosto che del soggetto venditore e proprietario dello stesso. Addirittura, in Airbnb si è arrivati alla condizione tale per cui, per paura di recensioni negative che gli impediscano in futuro di usare l'applicazione, l'utilizzatore stia molto più attento al suo comportamento rispetto che in un hotel, in cui fondamentalmente una volta pagato il prezzo della stanza si sente maggiormente libero di fare quello che vuole, perché non c'è nessuna recensione futura sul suo comportamento. La quarta fase, a cui ancora non si è arrivati, riguarda una fiducia di carattere algoritmico (possiamo considerarci a metà tra la fiducia distribuita e quella algoritmica) che viene introdotta proprio da blockchain e smart contract.

Blockchain e Smart Contracts sono in grado di garantire la fiducia anche in assenza di terze parti. Nel 2015 il giornale "The Economist" definiva la blockchain "Macchina della fiducia" sostenendo che: "La blockchain consente alle persone che non hanno una particolare fiducia reciproca di collaborare senza dover passare attraverso un'autorità centrale neutrale."

La teoria degli equilibri punteggiati di Stephen Jay Gould del 1972 asserisce che i cambiamenti evolutivi si verificano, in brevi periodi di tempo sotto l'impulso di determinanti forze selettive, mentre i periodi di stabilità evolutiva sono più lunghi e frequenti. Così è anche per la fiducia. [3]

Figura 9 – I salti della fiducia – Giorgio Angiolini – Blockchain Week Rome 2019

La fiducia procede a salti e, in ogni salto, c'è una sorta di timore nell'approcciarsi verso il nuovo sistema, per il rischio di incappare in frodi. Si pensi all'e-commerce: quando una decina di anni fa ci fu il boom degli acquisti su Internet con la nascita e lo sviluppo di Ebay e Amazon, c'era comunque un po' di preoccupazione in ciascuno di noi per paura che, dopo aver eseguito un ordine, il prodotto non arrivasse e quindi avessimo perso i nostri soldi. Oggi invece nessuno ha il minimo dubbio che il prodotto verrà consegnato. Questo ad esempio dice che è stato fatto il salto da carte di credito a e-commerce. Saranno le criptovalute il prossimo step? [4] [5]

[3] Who can you trust - Rachel Botsman - 2017

[4] Big Data: a revolution that will transform how we live, work and think – Kenneth Cukier, Viktor Mayer Schonberger – 2013

[5] Learning with Big Data: the future of education – Viktor Mayer Schonberher - 2014

L'altra faccia della medaglia

In questa breve introduzione sulla rivoluzione digitale in atto, si è cercato di spiegare quale direzione stia prendendo il mondo dal punto di vista tecnologico, anche se a non tutti piace. Per spiegare questo concetto si rifletta per esempio sul fatto che quella a destra nella Figura 10 sia un'edicola.

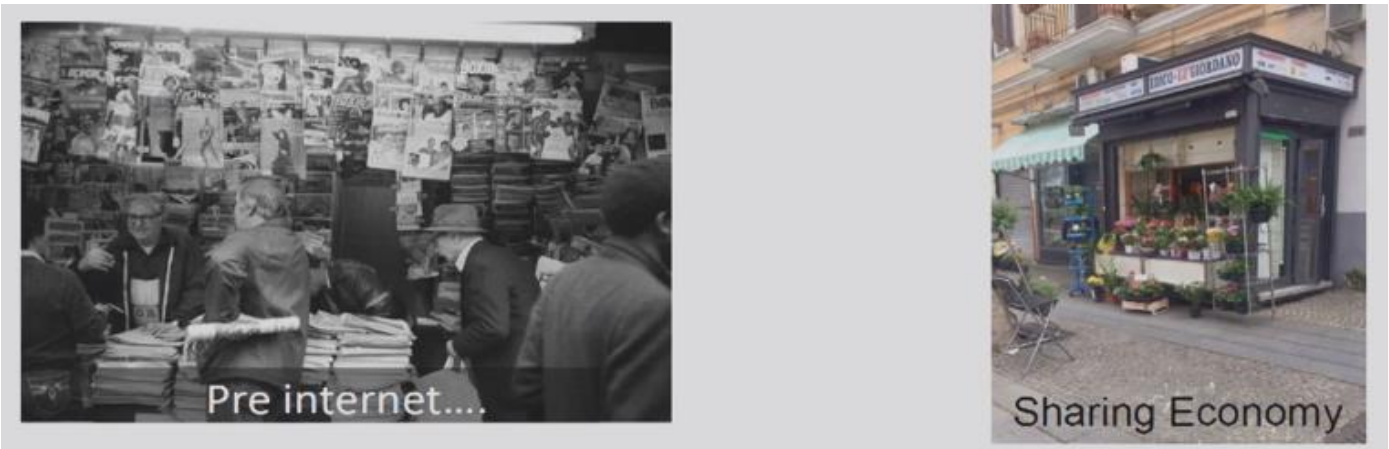


Figura 10 – La società moderna – Giorgio Angiolini – Blockchain Week Rome 2019

Come si può vedere in Figura 10, gli edicolanti, a causa di Internet e della digitalizzazione della carta, sono costretti spesso a reinventarsi come fiorai. Non solo, dal punto di vista del consumatore, veniva considerato un piacere andare in edicola, oltre che per comprare il giornale, anche come momento di relazione sociale, per la possibilità di scambiare qualche parola con edicolante e clienti. Questo è solo un esempio, ma sono molti i casi di trasformazione di un'attività economica.

Un altro esempio è quello della guida di una macchina. A causa di Intelligenza Artificiale e Machine Learning si sta arrivando ad una macchina intelligente in cui il ruolo del guidatore verrà azzerato, magari a vantaggio della sicurezza stradale. Quante persone però trovano divertente e rilassante scalare le marce e girare il volante?

Di situazioni come queste ce ne sono moltissime e il rischio, considerata la direzione in cui si sta andando, è quello di una estrema digitalizzazione di qualsiasi attività umana, in cui l'uomo assume un ruolo secondario e non è più padrone del proprio mondo. Potrebbe essere l'uomo a seguire gli ordini che gli vengono impartiti dalla macchina e questo potrebbe portare addirittura ad una limitazione della libertà, senza che ce ne stessimo rendendo conto.

1. Un mezzo per la diffusione di fiducia: la Blockchain

Alex Tapscott e Don Tapscott, due dei primi esperti in materia di digital economy, nel loro libro "Blockchain Revolution", definiscono la blockchain "Protocollo della fiducia", in contrapposizione alle modalità con cui sono avvenute fino ad oggi le transazioni su Internet. Infatti, gli scambi in rete avvengono grazie ad un intermediario che si mette in mezzo nella transazione, la certifica ed in cambio guadagna sia una commissione, sia i dati di acquirente e venditore. Pertanto, solo l'intermediario ha la conoscenza dei dati personali delle due parti, mentre quest'ultime non solo non si conoscono, ma non hanno neanche la possibilità di verificare la loro identità reale. I servizi più importanti su Internet (ad esempio Amazon, o società di pagamento come PayPal) si basano su questa caratteristica del sistema, permettendo ad un terzo di svolgere il ruolo di middle man, di terza parte fidata, a cui vengono fornite le informazioni con il compito di arbitrare un determinato scambio.
La blockchain si pone l'obiettivo di superare tale imposizione, creando un protocollo che consenta, tramite una serie di tecnologie e di regole, di assicurare l'origine e l'integrità nella trasmissione dei dati senza ricorrere ad una terza parte fidata, ma basandosi su un sistema peer-to peer.
Questo era il tassello mancante nel sistema delle transazioni online. In effetti, alcuni strumenti erano stati già inventati per consentire gli scambi diretti tra soggetti a distanza. I sistemi di firma elettronica a crittografia asimettrica sono stati messi a punto proprio allo scopo di garantire la paternità dei documenti e l'integrità degli stessi, attraverso il meccanismo della doppia chiave (pubblica/privata) appartenente allo stesso soggetto. Così anche i sistemi di timestamp, nonché quelli per la trasmissione sicura dei documenti informatici, hanno tutti il medesimo obiettivo, ovvero di garantire la fiducia tra soggetti che interagiscono a distanza. Il problema però, è che fino all'avvento della blockchain questi sistemi non erano stati integrati, ossia venivano visti come singole soluzioni a specifici problemi, senza però unificarsi in un unico protocollo che garantisse, in maniera facile e sicura, dette caratteristiche.
L'idea espressa nel paper di Satoshi Nakamoto, pseudonimo del soggetto (o del gruppo di soggetti) che ha teorizzato la blockchain per la creazione di una valuta virtuale distribuita (Bitcoin), ha il merito di mettere insieme tali tecnologie, per arrivare ad un vero e proprio nuovo modello di trasmissione dati, che garantisca l'originalità, l'integrità, la tracciabilità, la sicurezza delle transazioni e la loro esatta collocazione nel tempo.
Per tali motivi, per le persone che operano su Internet, è possibile effettuare scambi di informazioni (che siano criptovalute, contratti o altri atti negoziali) in maniera fidata, ossia sapendo che colui che avvia la transazione è effettivamente il proprietario dell'informazione trasferita.
La fiducia, in questo senso, viene garantita introducendo un principio di scarsità nel mondo digitale.
I dati tipicamente sono replicabili senza costi aggiuntivi e senza limitazioni (una serie di bit è un dato che può sempre essere replicato). Attraverso l'identificazione univoca di un dato e la sua memorizzazione in una serie di blocchi, in modo da farlo diventare permanentemente tracciabile, si è ottenuta la possibilità di rendere non più spendibile un elemento che sia stato già oggetto di transazione. Nel momento in cui sulla blockchain viene registrato, in maniera irrevocabile, tale trasferimento, colui che ne era proprietario non potrà più cederlo ad

altri, garantendo così un meccanismo di sicurezza in merito alle risorse che ciascuno possiede sulla rete. La circostanza che la blockchain sia pubblica, immodificabile e distribuita sulla rete dei nodi che la compongono, fa venir meno la necessità di un intermediario. Ed infatti, la pubblicità della blockchain permette a tutti i partecipanti alla rete di conoscere, in ogni momento, le transazioni eseguite, rendendo possibile sapere esattamente chi sia l'attuale proprietario del dato. Tale trasparenza, collegata alla certezza della titolarità dell'informazione, consente ai partecipanti di avere fiducia sul fatto che il soggetto che trasferisce l'informazione ne sia l'effettivo proprietario. Tutto ciò introduce un elemento estremamente innovativo nell'ambito delle comunicazioni elettroniche, che rappresenta un nuovo paradigma per le persone e gli enti, i quali possono rapportarsi direttamente con la rete, senza necessità di un intermediario ed attraverso un protocollo di fiducia distribuita che si rafforza al crescere dei componenti del network. [6]

1.1. Distributed Ledger Technology (DLT)

Prima di definire le varie tipologie di blockchain bisogna partire dalla definizione di ledger. Il ledger è uno strumento che serve per archiviare le informazioni di un determinato progetto, settore, azienda ecc..
Tendenzialmente questo registro delle informazioni, questo libro mastro, è stato sempre redatto e aggiornato dall'autorità centrale, titolare del progetto, che riceveva i dati da tutti i partecipanti che avevano rapporti economici con esso. Questo modello, usato finora da tutte le società, può essere definito Centralized Ledger, come mostra la Figura 11.

Figura 11- Registro Centralizzato - Distributed Ledger Technology (DLT) and Blockchain – World Bank Group

Tuttavia, a causa delle riforme tecnologiche nel reparto FinTech, il settore finanziario sta attraversando un momento di profonda trasformazione digitale. In particolar modo, vengono coinvolti in primo piano i sistemi di pagamento, i prestiti, le assicurazioni, il venture capital e le criptovalute.

[6] Blockchain Education Network (BEN) - 2019

Distributed Ledger Technology (DLT) e blockchain sono due attori di primaria importanza nel comparto FinTech.

Il DLT si riferisce ad una tecnologia nuova e in fase di rapido sviluppo che si basa sull'archiviazione e la condivisione di dati tra i vari partecipanti al network (chiamati nodi).

La blockchain può essere vista come un particolare tipo di DLT per l'archiviazione e la trasmissione di informazioni in strutture chiamate blocchi, connessi l'uno con l'altro in una catena digitale. La blockchain funziona con metodi algoritmici e crittografici per memorizzare e sincronizzare i dati in maniera immutabile tra gli utenti del network.

E' molto importante sottolineare che non tutti i DLT hanno come metodo di funzionamento una tecnologia blockchain.

Due caratteristiche fondamentali del DLT sono:

- l'abilità di raccogliere, memorizzare e scambiare informazioni in maniera digitale con differenti soggetti senza il bisogno di un'autorità centrale e senza la necessità di doversi fidare di qualche controparte.
- l'impossibilità di double-spending, ovvero che lo stesso token possa essere speso due volte.

La tecnologia blockchain per la prima volta è stata usata nel protocollo Bitcoin, ma i suoi sviluppi vanno ben oltre il solo settore delle criptovalute; per esempio, la blockchain ha già avuto applicazioni nel settore dei pagamenti, del security market e del mercato dei collateral. In più i margini di utilizzo di DLT e blockchain non sono confinati al solo settore finanziario ma vanno ben oltre. La DLT è stata utilizzata per sviluppare i servizi di identità digitale, archiviazione decentralizzata di flusso di beni e materiali all'interno di una supply chain ecc..

Gli utilizzatori della tecnologia DLT ne sottolineano il numero elevato di vantaggi rispetto ai tradizionali Centralized Ledger, come ad esempio la decentralizzazione e la disintermediazione, la più grande trasparenza, il più facile processo di verifica, i vantaggi in termini di velocità ed efficienza, la riduzione dei costi, l'automazione e la programmabilità.

I sistemi Distributed Ledger possono essere di due tipi: Permissionless e Permissioned.

Nel sistema Permissionless non c'è un proprietario centrale che controlla l'accesso al network. Tutto quello che serve per unirti alla rete e aggiungere transazioni è un computer con il software rilevante. Nel sistema Permissioned invece i membri del network sono preselezionati da un proprietario o da un amministratore del ledger che controlla l'accesso alla rete e fa rispettare le regole (si veda la Figura 12)

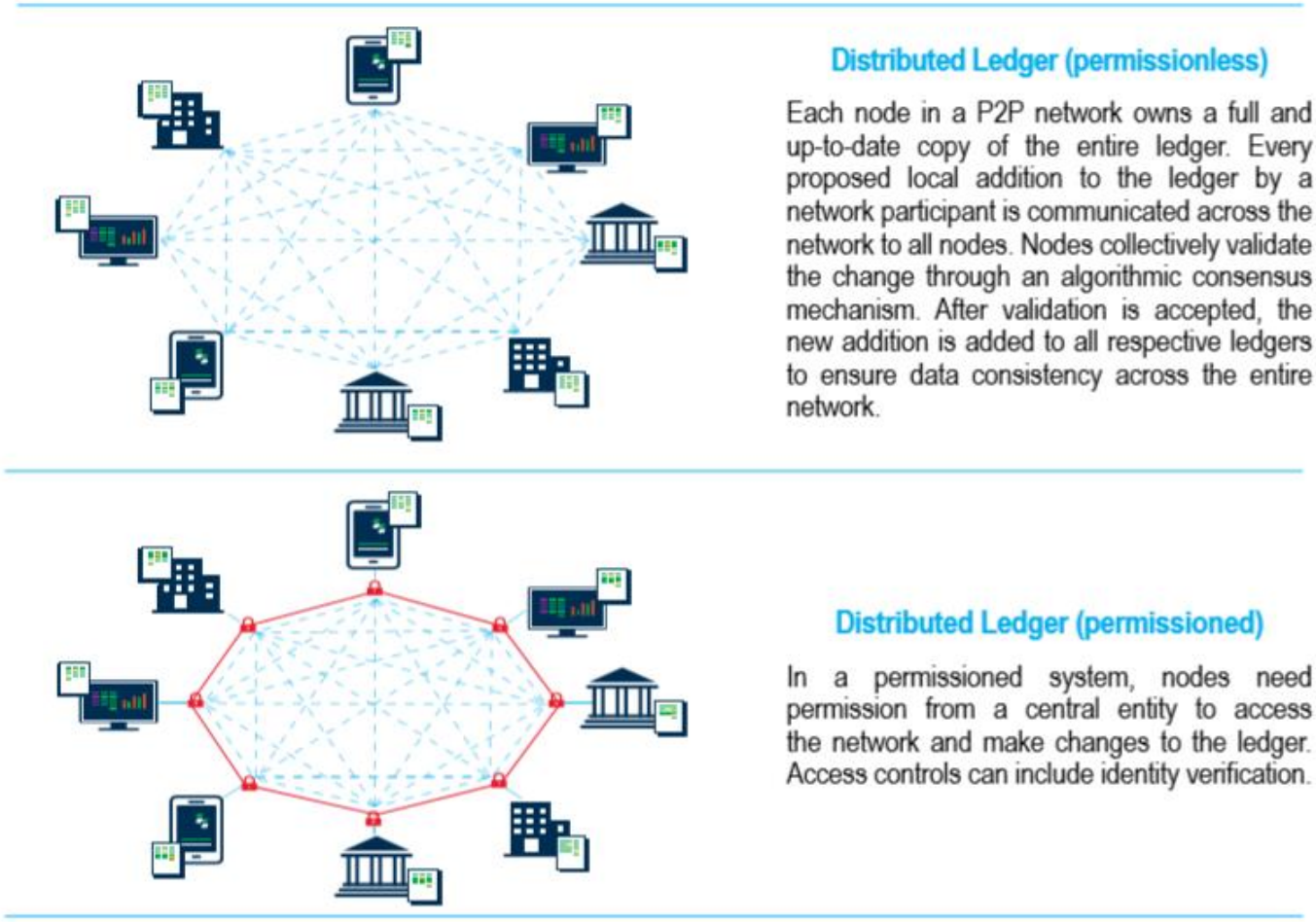


Figura 12 – Registro Permissionless e Permissioned - Distributed Ledger Technology (DLT) and Blockchain – World Bank Group

Ci sono vantaggi e svantaggi in entrambi i tipi di DLT. Per esempio, i sistemi Permissioned sono migliori per risolvere problemi riguardanti la verifica dell'identità e la privacy delle informazioni, ma hanno bisogno di un'autorità centrale che regoli gli accessi, autorità che potrebbe identificare un soggetto ideale per gli attacchi hacker. Inoltre, i sistemi Permissioned, il cui accesso al network comunque è sempre regolato, non richiedono una potenza computazionale di calcolo particolarmente alta per la verifica delle transazioni, ma fanno riferimento a differenti algoritmi per stabilire il consenso tra i membri. In aggiunta il sistema Permissioned può anche potenzialmente adattarsi più facilmente alla legislazione vigente e ai cambiamenti legislativi. Tuttavia, questo sistema riduce per certi versi quelle che sono le più importanti innovazioni portate dalla DLT. L'integrità del sistema e la sicurezza vengono raggiunte invece dal sistema Permissionless attraverso crittografia e algoritmi, che assicurano che i partecipanti al network siano incentivati a far rispettare la correttezza di funzionamento del ledger, senza bisogno di barriere all'entrata o fiducia tra i partecipanti. Tuttavia, a causa della possibilità per tutti i nodi di validare le transazioni, tendenzialmente la potenza di calcolo dei computer deve essere maggiore. Infatti, mentre nei sistemi Permissioned un amministratore si prende la responsabilità di assicurare che tutti i nodi siano affidabili, nel sistema Permissionless qualsiasi nodo può proporre l'aggiunta di una transazione, la quale viene replicata anche da altri partecipanti al sistema, potenzialmente anche senza nessun meccanismo di consenso.

Per capire se e quale tecnologia DLT fa al caso di un'azienda si può seguire la Figura 13:

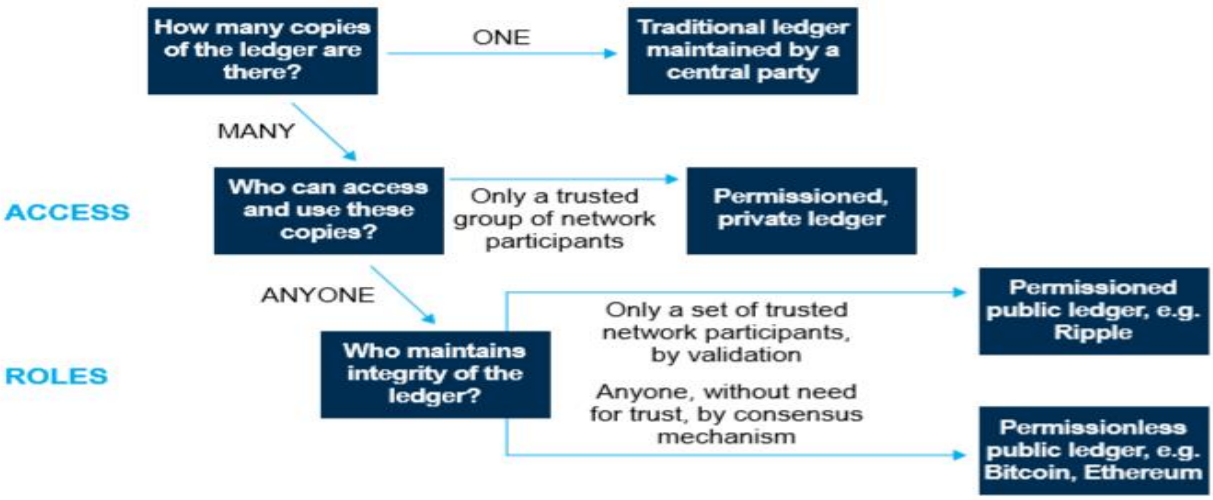


Figura 13 – Come usare i Ledger - Distributed Ledger Technology (DLT) and Blockchain – World Bank Group

La stessa divisione che è stata fatta per i sistemi DLT può essere fatta per la blockchain. Esistono infatti le blockchain permissioned e permissionless. Anche le caratteristiche sono più o meno quelle già elencate per le tecnologie DLT, ma si analizzino più nel dettaglio. [7]

1.2. Blockchain Permissioned (Private)

Una blockchain permissioned, che solitamente coincide con una blockchain privata, è quel tipo di blockchain in cui l'autorizzazione a "scrivere" i blocchi viene mantenuta centralizzata presso un'unica organizzazione. E' quindi una blockchain di tipo verticale, dove uno o più soggetti sono in grado di validare le transazioni che vengono registrate sulla catena. Le autorizzazioni per la consultazione della blockchain possono essere pubbliche o limitate discrezionalmente. Quindi questo implica che essa sia ristretta ad una cerchia conosciuta di entità: il termine "permissioned" riflette il fatto che l'autorità centrale possa introdurre delle policies per censurare le transazioni e quindi potenzialmente restringere l'uso della blockchain a sviluppatori e utilizzatori finali, i quali possono solamente fare affidamento su un'interfaccia fornita dagli operatori della blockchain per leggere e inviare le transazioni.

Le blockchain permissioned sono più attrattive per le istituzioni: queste blockchain potrebbero creare un ambiente più controllato e prevedibile rispetto a quelle permissionless. A differenza delle criptovalute, le blockchain permissioned generalmente non hanno token nativi. Infatti, i token nativi sono necessari nelle criptocurrencies per fornire incentivi a validare le transazioni; nelle blockchain permissioned invece i metodi per confermare le transazioni sono altri e tendenzialmente meno complicati. Nel più semplice dei casi, creare i blocchi su una blockchain permissioned non coinvolge calcoli associati con la proof of work.

[7] Distributed Ledger Technology (DLT) and Blockchain - World Bank Group - 2017

1.3. Blockchian Permissionless (Pubbliche)

Una blockchain permissionless è liberamente accessibile a chiunque. Non vi sono restrizioni circa la disponibilità di consultazione degli scambi, l'effettuazione degli stessi e la possibilità di partecipazione al meccanismo di consenso. Si tratta del modello su cui è stata realizzata la blockchain Bitcoin, pensata per disintermediare i meccanismi di fiducia reciproca tra i partecipanti. Essa non pone requisiti specifici per la partecipazione al network, incentivando ed auspicando anzi la sua espansione tramite meccanismi che remunerano coloro che mettono a disposizione la potenza computazionale dei loro hardware. Tipicamente i partecipanti a questo tipo di blockchain vengono denominati "miners" in quanto il loro compito è quello di impiegare delle risorse per consentire la creazione dei blocchi.
Il termine "permissionless" riflette proprio il fatto che non ci sono ampie policies restrittive riguardo l'uso della blockchain perché viene data liberamente la possibilità a utenti finali e sviluppatori di entrare e uscire dal protocollo.

Mentre le blockchain permissioned meglio si adeguano alle strutture legislative in corso e sono molto più attrattive per un'introduzione del sistema blockchain all'interno di un ledger già esistente nel breve-medio termine, tale tipologia limita uno degli aspetti fondamentali di questa tecnologia: la necessità di non doversi fidare della terza parte. Difatti uno degli obiettivi della blockchain è quello di rimuovere il fattore umano (middle man) dalle transazioni processate, rimpiazzandolo con un protocollo rigoroso e pubblicamente disponibile, che viene reso funzionante da un network di computer indipendenti.
Se la blockchain non è consultabile e poco trasparente per l'utilizzatore finale, l'aspetto della fiducia viene ampiamente ridimensionato. Addirittura, l'utilizzatore finale potrebbe arrivare a dubitare dell'effettivo utilizzo di un sistema blockchain, dato che non c'è possibilità di consultarla né c'è libertà di accesso alle informazioni scritte. Il fattore umano rimarrà una vulnerabilità nelle blockchain private fino a quando non si riuscirà a ridurlo al minimo possibile. Inoltre, questa condizione di difficile consultazione, le fa diventare scarsamente accessibili a persone esterne al progetto; invece i codici open source e standardizzati delle blockchain pubbliche le rendono un ambiente di sviluppo per ricercatori informatici. In questo senso le blockchain pubbliche sono simili ai protocolli di Internet IP, TCP, HTTP mentre una blockchain privata è designata per essere simile ad un'iniziativa personale su Internet. Il proprietario di questa blockchain privata potrebbe contenere vulnerabilità di sicurezza che rimangono nascoste e sfruttate per un lungo periodo di tempo, mentre una blockchain pubblica può sempre essere consultata e, nel caso di problemi di sicurezza, scoperta. Questo diventa addirittura un incentivo nelle blockchain permissionless, poiché gli utilizzatori hanno un guadagno immediato nello scoprire le vulnerabilità potendo lucrare sugli eventuali bug.
Al contrario il limitato set di nodi nelle blockchain permissioned suscita preoccupazioni perché:

- L’autenticazione dei validatori introduce una vulnerabilità nel sistema: mentre, nel caso di permissionless blockchain, il processo di validazione delle transazioni è per natura neutra ed equidistante tra tutti i nodi, nelle permissioned blockchain ci sono pochi fidati operatori. Questo lascia un’ampia possibilità di corruzione e manipolazione.
- Il concetto della blockchain si basa su un’economia incorporata, nel senso che i blocchi sono tutti collegati tra di loro. Ogni blockchain forma il proprio ecosistema economico; una blockchain privata è perciò un’economia centralizzata controllata, con tutto quello che comporta e che si voleva superare.
- C’è il rischio che i nodi colludano al fine di creare differenti copie della blockchain e/o alterare pezzi di essa.
- Non è chiaro cosa succederebbe nel caso in cui i nodi diventassero disinteressati al suo mantenimento o come si potrebbe recuperare nel caso di un attacco hacker andato a buon fine (la permissionless blockchain offre come soluzione una self-organization, quindi tutti i nodi prendono comunemente una decisione)

Questi rischi portano a dire che nell’ambito delle blockchain private bisogna stare attenti a distinguere i progetti che effettivamente usano la blockchain come un utile mezzo di archiviazione e trasmissione di dati da quelli che invece la usano con scarsi vantaggi pratici e solamente come mezzo pubblicitario in quanto tecnologia attualmente imperante. [8]

A tal proposito sono stati ideati dei modelli che cercano di evidenziare se ci sia la necessità e quale tipologia di blockchain si debba adottare per un determinato progetto.

Suichies Model

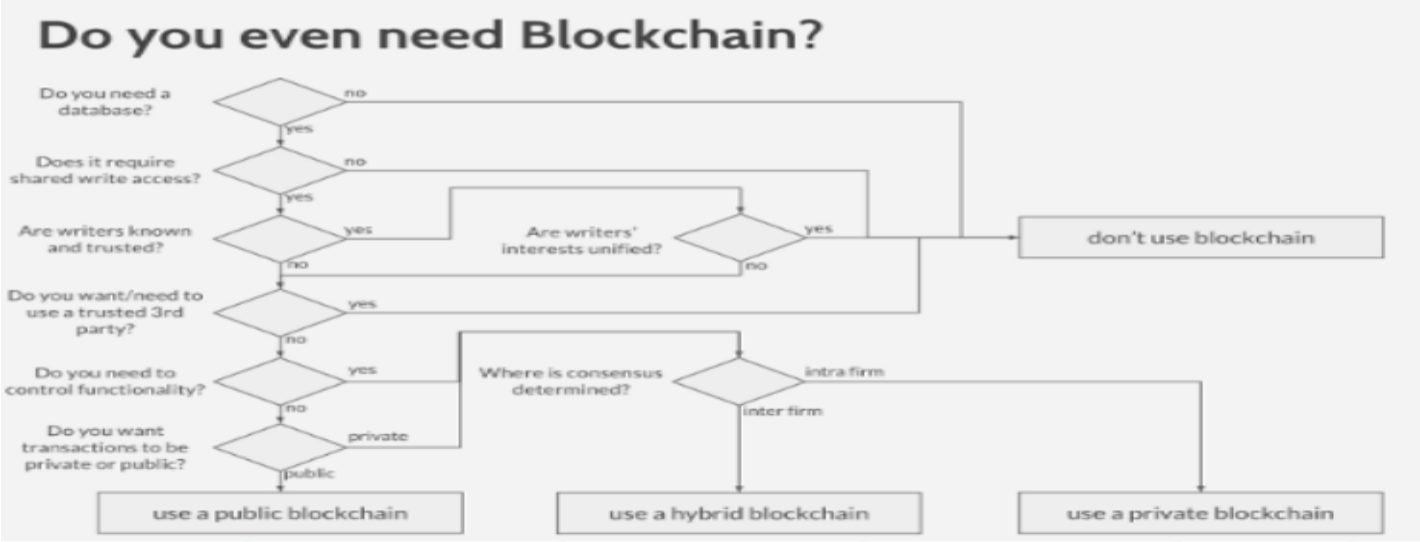


Figura 14 - Suichies Model - Medium

[8] Public versus Private Blockchain - BitFury Group -2015

Questo modello, attraverso un diagramma di flusso basato su semplici domande, permette di capire se e quale tipo di blockchain necessiti un'iniziativa economica.

Karl Wustl e Arthur Gervais

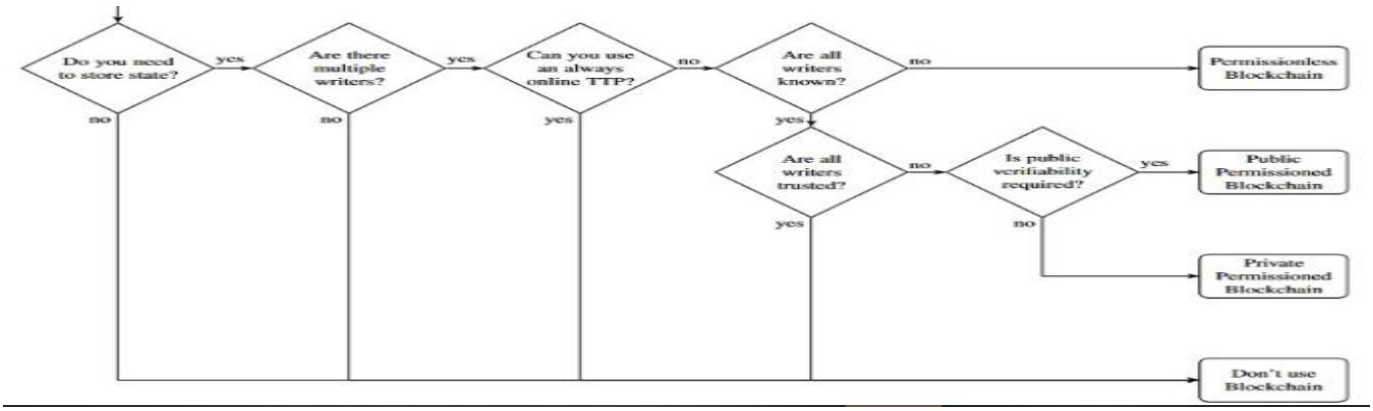


Figura 15 – Wustl e Gervais Model - Medium

Il modello di Wustl e Gervais è un semplice diagramma di flusso che indirizza verso la scelta più logica per un progetto.

DHS Model

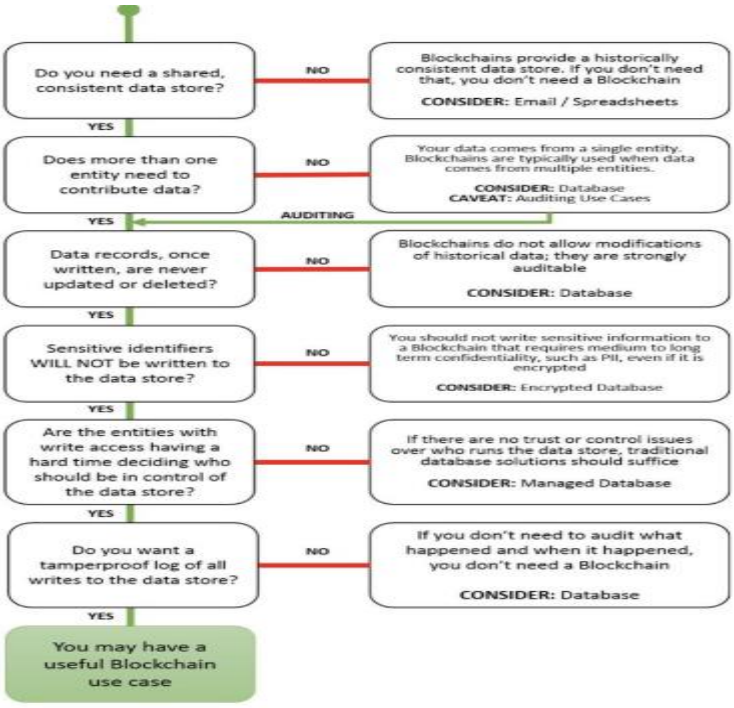


Figura 16 – DHS Model – Medium

Un ultimo diagramma che potrebbe essere utilizzato è il modello DHS, che non dice la tipologia di blockchain da utilizzare, ma semplicemente indica se sia conveniente o meno utilizzarla, fornendo in alternativa indicazioni sul mezzo di archiviazione dati più adatto alle necessità del caso. [9]

[9] When do you need blockchain? Decision model - Sebastien Meunier - 2019

2. La blockchain di Bitcoin

2.1. Una breve introduzione

Per capire all'atto pratico il funzionamento della blockchain, si prenda come riferimento la blockchain di Bitcoin, giudicata da tutti come quella maggiormente completa, decentralizzata e portatrice del vero significato di Distributed Ledger Technology.

Preliminarmente si deve considerare che poiché molti concetti alla base di Bitcoin sono nuovi e tecnicamente difficili, una comune lamentela è stata quella secondo cui le criptovalute sono troppe complicate per un utilizzo su base giornaliera. Tuttavia, si consideri che lo stesso valeva per Internet e più in generale per l'inizio di ogni nuova era tecnologica, in cui i dettagli tecnici di "che cos'è" e "come funziona" sono sempre stati di interesse popolare. Tale situazione inoltre, non rappresenta una vera e propria barriera all'entrata: infatti per l'utente medio per esempio non è necessario sapere come funziona il TCP/IP per poter inviare un'e-mail. Le nuove tecnologie vengono utilizzate senza tenere conto dei dettagli tecnici, a patto che vengano sviluppati front-end appropriati e affidabili (nel senso che abbiano una interfaccia grafica capibile per il comune cliente): non tutti gli utilizzatori hanno bisogno di vedere (e tanto meno di digitare manualmente) un indirizzo pubblico alfanumerico di 32 caratteri come accade nella blockchain.

Poiché Bitcoin e i wallets di criptocurrencies sono legati al denaro, c'è un'attenzione maggiore nelle applicazioni che verranno rilasciate per gli utenti finali e nella fiducia che deve essere sedimentata nei consumatori. Ci sono molti problemi di sicurezza da affrontare nel mondo cripto per formare un pubblico di fruitori digitalizzati, capaci per esempio di eseguire il backup del denaro, di sapere cosa fare laddove perdessero la chiave privata e cosa laddove ricevessero una moneta illegale (cioè, precedentemente rubata) in una transazione. Questi problemi vengono costantemente studiati dall'industria blockchain e le criptovalute possono essere viste come un altro tassello nella continua evoluzione della tecnologia finanziaria (fintech) che include gli ATM, l'online banking e Apple Pay.

L'adozione di applicazioni per criptovalute potrebbe essere semplice con front-end facili e affidabili, ma l'utilizzo mainstream delle app blockchain va al di là delle monete digitali. Ad esempio, i servizi notarili virtuali sembrano facilmente implementabili anche al giorno d'oggi a causa della registrazione facile, a basso costo, sicura, permanente e rintracciabile di IP, contratti, testamenti e documenti simili. Potrebbe tuttavia succedere che le persone preferiscano interagire con un notaio su certe questioni (forse per la consulenza umana o la "psicoanalisi" che i notai possono fornire) e, per questo, l'adozione di tecnologie basate esclusivamente sull'efficienza informatica potrebbe vacillare.

Nel complesso, tuttavia, se Bitcoin e l'industria blockchain dovessero maturare, molto probabilmente ciò avverrà per fasi, similmente al modello di adozione di Internet. Infatti, quest'ultimo inizialmente ha risolto i problemi di un ridotto gruppo di persone, come ricercatori accademici e militari, poi, sono arrivati i giocatori online e gli appassionati e, alla fine, tutto il mondo. Nel caso di Bitcoin, la prima ad adottare la nuova

tecnologia è stata la ristretta cerchia di persone preoccupate per il funzionamento, a volte limitante, della valuta fiat; tuttavia, i prossimi passi per un suo utilizzo diffuso, potrebbero essere compiuti grazie alle modalità con cui la blockchain risolverà i problemi pratici per altri grandi gruppi di professionisti. Per esempio, essa risolve un problema importante come quello della censura di Internet nei regimi politici repressivi, dove invece i servizi DNS (domain name system) decentralizzati su blockchain, potrebbero fare una grande differenza. Allo stesso modo, nel mercato della proprietà intellettuale, la blockchain potrebbe essere impiegata per registrare il susseguirsi di creazioni di brevetti e rivoluzionare il contenzioso in materia di proprietà intellettuale nei settori della custodia, dell'accesso e dell'attribuzione dei beni.[10]

Nel suo whitepaper, Satoshi Nakamoto definisce Bitcoin "A Peer-to-Peer Electronic Cash System".
La tecnologia peer-to-peer è stata un'invenzione disruptive, che ha rivoluzionato diversi settori.
Si prenda come esempio il primo caso di tecnologia peer-to-peer in ambito musicale che si verificò con l'applicazione chiamata Napster, una creazione di Sean Parker. Attraverso Napster, i file musicali potevano essere condivisi tra gli amici (peers) senza dover andare in qualche negozio musicale per comprare l'intero CD dell'artista.
Prima di Napster, un soggetto comprava l'album, lo masterizzava e lo portava a casa di un amico. Questo meccanismo non solamente coinvolgeva un middle man (negozio di CD), ma scomodava anche il soggetto stesso, impegnandolo a compiere qualcosa di fastidioso (masterizzazione e trasporto del CD). Napster tagliava fuori il middle man e permetteva di condividere con gli amici l'album musicale preferito dal comfort di casa propria. Ovviamente i middle man non erano troppo felici dell'invenzione di Mr. Parker e dopo poco tempo lanciarono una serie infinita di azioni legali contro Napster, reclamando i loro diritti e provocandone la chiusura.
Napster tuttavia aveva ormai fatto in tempo a rivoluzionare l'industria musicale; molti lo considerano infatti il precursore di iTunes.
Quando si pensa ad un servizio di file-sharing, bisogna considerare che Bitcoin non sia troppo differente da Napster, dato che i files che vengono condivisi sono unità di valore piuttosto che brani musicali. Se si riuscisse a trovare un mercato che accettasse musica come mezzo di pagamento in cambio di cibo, allora Napster sarebbe potuta diventare la prima valuta digitale della storia. La sicurezza della tecnologia di Bitcoin è ciò che la rende più adatta rispetto a Napster per essere una valuta: il cuore di Bitcoin infatti è la blockchain. Questo ledger impone che ogni scambio venga regolato sotto forma di bitcoin; dal momento in cui ogni bitcoin viene creato, ogni suo movimento viene registrato in blockchain e questo certifica che i bitcoin non possano essere falsificati.
Per creare la blockchain, ogni 10 minuti circa il software Bitcoin raccoglie le transazioni verificatesi dentro un blocco. Quest'ultimo contiene la "storia" del blocco precedente e delle transazioni che sono già accadute. Quando tutti i blocchi sono collegati insieme, si forma una catena, la blockchain.

[10] Blockchain: Blueprint for a new economy - Melany Swan - 2015

La sicurezza di Bitcoin dipende dal processo di collegamento di tutte le transazioni.
Si immagini che una banconota da un dollaro venisse tracciata ogni volta che venga utilizzata, dal momento della sua emissione fino ad un suo eventuale ritiro dal mercato. Ogni pacchetto di gomme, fiori o giocattoli che venisse comprato con quel dollaro, verrebbe registrato. Se un falsario avesse fatto una copia fisica di questa banconota, il sistema avrebbe posseduto un registro con il nome del legittimo proprietario della banconota e, qualora il falsario avesse provato a spendere la copia fasulla, il sistema di sicurezza non avrebbe permesso la transazione.
La soluzione che propone Bitcoin per superare il problema della duplicazione della valuta, che in ambito digitale viene chiamato double-spending, è la combinazione tra blockchain e miners.
Quando le transazioni vengono aggiunte alla blockchain, diventa impossibile cambiare quelle precedenti. I miners infatti si occupano di verificare che le transazioni siano vere e non ci sia un tentativo di falsificazione della blockchain; la loro azione si risolve in una complessa equazione matematica che loro stessi sono chiamati a risolvere mediante la potenza computazionale dei loro computer. La risposta all'equazione contiene un codice che verifica tutte le transazioni precedenti e, se tale codice non si adatta ad esse, i miners si accorgono che la blockchain è stata falsificata. In termini molto semplici, se A deve mandare dei bitcoin a B, deve comunicarlo al network Bitcoin. I miners "ascoltano" quel messaggio crittografato e utilizzano potenti computer per assicurarsi che A sia il legittimo proprietario dei bitcoin. Una volta che la titolarità dei bitcoin è stata provata, i miners verificano la transazione e la registrano in blockchain. Per questo lavoro, come ricompensa, ricevono dei bitcoin.

2.2. L'origine di Bitcoin

Bitcoin è stato creato dallo sviluppatore Satoshi Nakamoto nel 2009. Il concetto dietro a Bitcoin è piuttosto semplice: durante la crisi finanziaria del 2008, le persone hanno subito ingenti danni economici e le valute fiat non si sono dimostrate all'altezza della situazione. Poiché il sistema finanziario era sull'orlo del collasso, molte banche centrali hanno adottato politiche monetarie espansive, ovvero in parole povere, hanno stampato moneta. Le banche hanno inondato i mercati con liquidità e abbassato i tassi di interesse a livelli prossimi allo zero per prevenire il verificarsi di un'epoca simile alla Grande Depressione del 1930. L'effetto di questi movimenti è stata una fluttuazione su larga scala del valore delle valute fiat; questa situazione è stata chiamata "currency wars" – cioè una gara a chi svalutava di più la propria moneta allo scopo di fare diventare la propria economia la più sostenibile, con un'offerta di beni e servizi meno costosi di quelli dei vicini e dei competitors globali.
La risposta delle banche centrali è stata sempre la stessa: i governi dovevano salvare le banche in difficoltà mentre veniva stampata moneta extra, che andava a svalutare il valore della moneta già in circolazione. Nel salvataggio delle banche, c'era un trasferimento netto dei debiti verso il bilancio dello stato; ciò comportava

un aumento degli interessi futuri sul debito che sarebbero stati ripagati attraverso un aumento della tassazione. Questo ha creato una sorta di ingiustizia sociale causata dagli errori del settore finanziario.
Quello che sembrava chiaro è che i banchieri centrali, apparentemente indipendenti dai governi, stavano portando l'economia verso settori sconosciuti e si stavano preparando a svalutare le valute fiat a loro piacimento solo per far girare il sistema finanziario come prima del 2007. Nel farlo, salvavano le stesse istituzioni responsabili della crisi e i banchieri responsabili dei crimini.
Qui risiede l'origine di Bitcoin: un sistema finanziario decentralizzato che esula dal controllo di un ristretto numero di decision-makers. Satoshi Nakamoto aveva deciso di creare un sistema totalmente nuovo, che avesse la forza di sovvertire il sistema finanziario di allora. Inoltre, voleva formare un protocollo totalmente open source, in modo che chiunque fosse libero di consultarlo e migliorarlo, dotato di una tecnologia importante come la blockchain.
L'incontro tra finanza e tecnologia può creare molti problemi, ma tutti sono stati colpiti dalla crisi del 2007 e molti Stati stanno ancora combattendo per cercare di recuperare le perdite verificatesi durante quel periodo. Nakamoto è stato vittima come tutti della cattiva gestione durante la crisi da parte delle banche centrali e ha cercato di creare una soluzione alternativa. L'infrastruttura finanziaria era imperfetta e un qualcosa di nuovo era richiesto e bene accetto. Se questa alternativa possa essere Bitcoin rimane da scoprirlo.

2.3. Bitcoin come moneta

Quando le persone parlano di bitcoin, una delle prime cose che tendono a menzionare è il prezzo attualmente in corso della criptovaluta. Il prezzo viene liberamente determinato sulla base dell'incontro tra domanda e offerta. Anche se l'offerta di Bitcoin è limitata ad un massimo di 21 milioni di bitcoin totali - destinato ad essere raggiunto più o meno nel 2140 - non esiste una così grande domanda finora.
Per fare in modo che bitcoin possa essere usato come la moneta di tutti i giorni, è necessario che sempre più persone ne facciano un utilizzo ricorrente. Come si può immaginare, è difficile convincere alcune aziende ad accettare bitcoin come mezzo di pagamento e ancora più difficile è coinvolgere i consumatori a pagare i servizi ricevuti utilizzando questa valuta digitale.
I vantaggi per le aziende potrebbero essere molti: Bitcoin taglia le commissioni e altri costi, ma se nessun acquirente sta pagando in bitcoin, non c'è vantaggio nell'accettarlo come mezzo di pagamento. Quindi sta più al consumatore far partire l'ingranaggio.
Per fare in modo che il pagamento tramite bitcoin possa diventare più facile, anche per piccoli importi, un buon metodo potrebbe essere quello di utilizzare carte prepagate e carte di credito collegate al proprio bitcoin wallet. Attraverso questo metodo ad esempio si potrebbe iniziare un sistema di scambi in bitcoin anche per piccoli importi o beni di uso comune (non si tenga conto al momento del problema della scalabilità).

2.4. Bitcoin come mezzo di investimento

Nonostante Bitcoin nasca come mezzo di pagamento, come dice il whitepaper di Satoshi Nakamoto, finora l'investitore tradizionale lo ha considerato o un asset speculativo, cercando di prevederne i movimenti per speculare sul suo valore futuro, o uno strumento di diversificazione di portafoglio, considerata la scarsa correlazione con i titoli tradizionali e l'assenza di controllo da parte di qualche entità super partes, o entrambi.

Figura 17 – Andamento del prezzo di Bitcoin – Coinmarketcap

2.5. Il problema dei generali Bizantini

Risolvere per primi un problema può portare non solo ad accumulare molti soldi, ma anche ad avere una notorietà in precedenza inimmaginabile. A questo proposito balza subito alla mente il caso di Mark Zuckerberg che, trovandosi di fronte alla difficoltà di comunicare con gli altri studenti del suo corso, creò una piattaforma adatta a questo scopo, Facebook, in seguito aperta a tutto il mondo.

Il problema che Bitcoin ha risolto viene chiamato Problema dei Generali Bizantini e viene presentato in due differenti versioni.

Una versione è stata proposta nel 1982 dagli informatici Leslie Lamport, Robert Shostak e Marshall Pease: si immagini che molte divisioni dell'esercito Bizantino stiano campeggiando davanti ad una città nemica e ogni divisione sia comandata da un generale. I generali possono comunicare tra di loro solamente mediante dei messaggeri al fine di stabilire, dopo aver osservato il nemico, un comune piano di azione.

Tuttavia, alcuni generali potrebbero essere traditori, cercando di impedire a quelli leali di raggiungere un accordo. I generali leali devono possedere un algoritmo che gli assicuri di decidere tutti lo stesso ragionevole piano di azione.

I generali leali non dovrebbero solamente raggiungere un patto comune, ma dovrebbero anche assicurarsi che i generali traditori non modifichino l'accordo fatto in un piano d'attacco sbagliato.

Una seconda versione del problema, che in realtà sarebbe anche quella originaria del 1975, chiamata semplicemente Il Problema dei Due Generali, è stata proposta da A. Akkoyunlu, K. Ekanadham, e R. V. Huber.
Il Problema dei Due Generali inizia con due eserciti che vogliono attaccare una città e saccheggiarne le ricchezze. La città si trova in una valle tra due colline e può essere conquistata solo se entrambe le armate attaccano nello stesso momento. I generali decidono di comunicare il momento dell'attacco una volta che hanno avuto la possibilità di esaminare la città e nel frattempo posizionano le loro truppe sulle due colline opposte.
Una volta che i generali arrivano sulle rispettive colline, l'unico modo per comunicare è inviare un messaggero che passi attraverso la valle, il quale rischia di essere catturato oppure addirittura catturato e sostituito con uno falso, portatore di un messaggio ingannevole.
Riassumendo, il Problema dei Due Generali dice che è necessario comunicare il momento esatto di un attacco sincronizzato, mandando un messaggero attraverso la valle insicura.
La metafora aiuta a comprendere ciò che può verificarsi su una rete di computer se si passano informazioni preziose tra i nodi. Ogni computer in rete è un nodo e rappresenta un generale. L'unico modo che hanno i computer per comunicare è mediante una ragnatela non sicura di linee telefoniche, cavi in fibra ottica e Internet.
Internet, nel Problema dei Due Generali, è la valle attraverso la quale deve passare il messaggio. Ogni nodo sulla rete ha bisogno di un modo per capire se il messaggio che ha ricevuto sia legittimo. Come ha dimostrato la National Security Agency degli Stati Uniti, i messaggi inviati su Internet possono essere intercettati e ciò è problematico quando si tenta di inviare qualcosa di valore.
Il problema che deve essere risolto è come si possa comunicare, per esempio, il messaggio "attacchiamo alle nove in punto" in modo tale che entrambi i generali possano lanciare l'offensiva in quel momento esatto. Questo può sembrare semplice, ma la complessità sta proprio nella sua sottigliezza.
Una volta che il primo generale ha inviato il messaggero, non può sapere se quest'ultimo ha attraversato la valle dei nemici. Inoltre, il generale che ha ricevuto il messaggio non può essere certo che il messaggero che arriva al suo accampamento sia quello alleato e non invece una spia dei nemici: attraversare la valle può comportare infatti la cattura e la sostituzione del messaggero ufficiale con uno nemico.
A prima vista, si potrebbe concludere che la soluzione migliore sia quella di inviare molti messaggeri, poiché è improbabile che vengano tutti catturati: alcuni di loro riusciranno sicuramente a superare l'accampamento nemico con i messaggi corretti.
Tuttavia, ci si può rendere conto che, indipendentemente da quanti messaggeri vengano inviati, non vi è comunque alcuna garanzia che il messaggero arrivato stia portando il messaggio corretto.
Si può comunque pensare che la maggior parte dei messaggi ricevuti dica: "Attacchiamo alle nove" convincendo il secondo generale. Tuttavia, nessuno dei due può essere certo che il messaggio sia stato effettivamente ricevuto o che la maggior parte dei messaggeri arrivati non siano traditori.

Naturalmente, ogni generale potrebbe mandare una conferma che il messaggio è stato ricevuto. Sfortunatamente sorge lo stesso problema. Nessuno dei due può essere certo che la conferma sia valida, anche se vengono spediti un numero infinito di messaggi.
Inoltre, se uno dei due generali esita a causa dell'incertezza, l'attacco fallirà.
Questo è stato il problema che gli informatici hanno dovuto affrontare dal 1975; gli autori conclusero che il Problema dei Due Generali era impossibile da risolvere. La scienza informatica accettò la conclusione come un dato di fatto, fino a quando Satoshi Nakamoto non la ribaltò. Quando Leslie Lamport, Robert Shostak e Marshall Pease proposero il Problema dei Generali Bizantini (BGP), questo altro non era che un'estensione del Problema dei Due Generali. Aggiungendo anche altri generali, il problema diventava ancora più complicato e difficile da risolvere.
Internet è una rete di computer non sicura: è la valle attraverso la quale devono essere inviati dei messaggi. Quando Alice invia un'e-mail a Bob, la e-mail deve attraversare una valle insidiosa e non vi è alcuna garanzia che il messaggio ricevuto sia lo stesso messaggio che è stato inviato. Una semplice soluzione per proteggere un'e-mail era quella di crittografare il messaggio, ma ciò non offriva ancora la sicurezza necessaria per il trasferimento di qualcosa di valore. È comunque troppo facile per un malintenzionato inviare un messaggio crittografato, ma falso.
Se Alice e Bob parlano solo di un'eventuale riunione a cui probabilmente nessuno dei due vuole partecipare, non è così importante mantenere una rete perfettamente protetta. Se viene semplicemente manomesso l'orario della riunione facendo ritardare Bob, Alice può chiamare Bob e dirgli di correre nella sala conferenze. Tuttavia, cosa succederebbe se il messaggio inviato contenesse tutte le transazioni completate con la carta di credito?
Quindi, nel mondo reale, il BGP può avere effetti devastanti quando qualcosa di valore viene trasferito su una rete non sicura.

2.6. Come Bitcoin risolve il Problema dei Generali Bizantini (BGP)

Risolvere il BGP comportava la soluzione di vari aspetti: un metodo per proteggere il contenuto del messaggio, un modo per ridurre il numero di messaggi inviati, un modo per rilevare un messaggio falsificato e uno per remunerare tutto questo meccanismo.
Bitcoin risolve il Problema dei Generali Bizantini crittografando il messaggio, imponendo un costo per decodificarlo, fornendo un modo per verificare che il messaggio sia stato legittimamente decodificato e attribuendo un incentivo ai "generali" onesti. Quando viene generato un messaggio, Bitcoin utilizza la crittografia per trasformare lo stesso in bit qualsiasi sia la dimensione, attraverso un algoritmo noto come SHA256 ovvero Secure Hash Algorithm.
Mediante lo SHA256, un messaggio lungo due frasi o due paragrafi sarà ridotto ad una quantità fissata di caratteri alfanumerici casuali. Una volta trasformato, il messaggio diventa irriconoscibile e può essere

decodificato solo risolvendo una complessa equazione matematica che è la prova che si è lavorato sulla soluzione.

L'equazione deve essere abbastanza difficile da permettere ai computer di impiegare circa lo stesso tempo per trovare una risposta. Era quindi necessario che fosse predisposto un periodo di tempo standardizzato per fare in modo che la difficoltà di risoluzione del problema non fosse troppo facile o troppo difficile. Una difficoltà troppo alta di risoluzione del problema rallenterebbe eccessivamente il network e ne comprometterebbe la scalabilità, mentre un problema troppo facile metterebbe a rischio la sicurezza.

Bitcoin specifica che la soluzione del problema deve richiedere mediamente 10 minuti e ogni due settimane viene regolata la difficoltà di calcolo della rete in modo che il tempo medio di risoluzione rimanga fissato a 10 minuti. La potenza di calcolo e l'energia usata per risolvere il problema matematico servono come costo contro l'invio di "messaggi falsi".

Usando il protocollo Bitcoin, tornando all'esempio dei BGP, se un generale traditore avesse voluto inviare un messaggio falso, avrebbe dovuto spendere dei soldi per un computer veloce e pagare l'elettricità necessaria per frodare il sistema.

Se un generale avesse voluto trasmettere un messaggio falso senza fare il lavoro necessario per risolvere l'equazione matematica, gli altri generali avrebbero potuto semplicemente guardare quanta potenza di calcolo stesse spendendo il generale traditore. Se si fossero accorti di un utilizzo ridotto o nullo della potenza di calcolo, i generali avrebbero potuto immediatamente supporre che il messaggio fosse falso. Affinché si possa verificare che un messaggio sia stato legittimamente decodificato, ogni generale deve dimostrare di aver impiegato 10 minuti circa per risolvere il problema. I generali lo fanno osservando la quantità totale di potenza di calcolo sulla rete. Se la rete totale impiega 10 minuti per risolvere il problema matematico, i generali possono presumere che i messaggi trasmessi siano stati decodificati legittimamente.

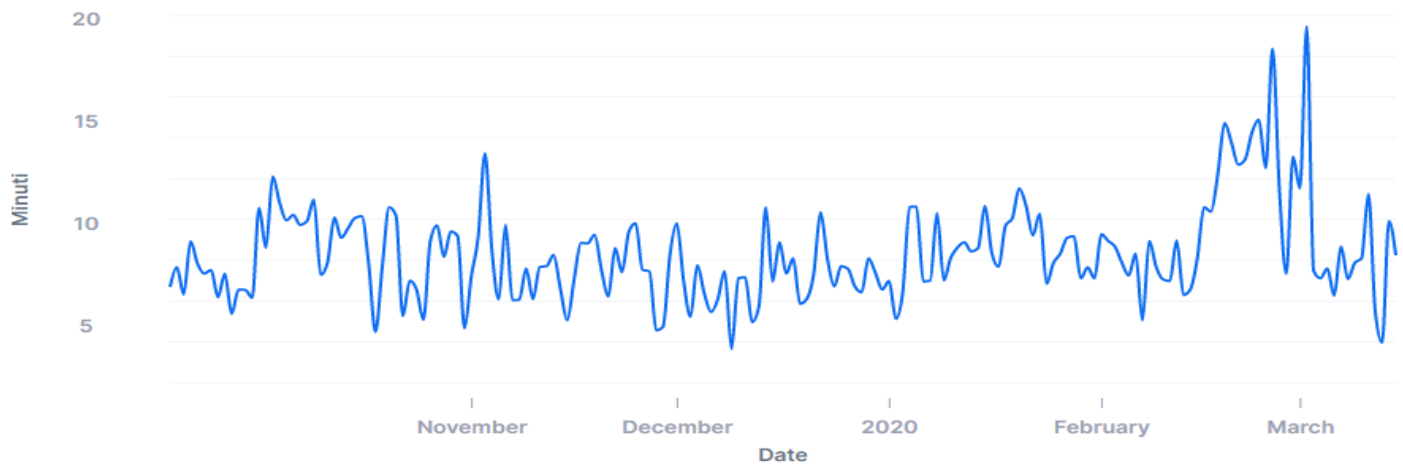


Figura 18 – Tempo medio di conferma di una transazione negli ultimi 180 giorni – Blockchain.info

Inoltre, Bitcoin richiede che più di un generale confermi di aver ricevuto lo stesso messaggio. Attraverso la cosiddetta "proof of work" (prova di lavoro) svolta per risolvere l'equazione matematica, tutti i generali possono essere tranquilli che il messaggio trasmesso sia quello vero.

Infine, il protocollo Bitcoin fornisce un incentivo per essere i primi a decodificare legittimamente il messaggio. Il primo generale che risolve il problema e trasmette il messaggio valido all'interno della rete riceve un

risarcimento sotto forma di criptovaluta. Man mano che il valore della valuta cresce, aumenta anche l'incentivo ad essere un generale onesto. In questo modo, Bitcoin fornisce un meccanismo che si autorinforza all'aumentare dei partecipanti.

La criptovaluta offerta come ricompensa è bitcoin e può essere visto come la remunerazione per un pezzo di informazione trasmessa in maniera legittima. Questa informazione è vera perché il protocollo Bitcoin la traccia fin dalla sua origine: ogni informazione (bitcoin) sulla rete viene registrata e seguita fin dal principio per verificarne la legittimità.

Bitcoin utilizza la crittografia e il proof of work per risolvere il BGP. I generali bizantini non possono fidarsi aprioristicamente che il messaggio ricevuto sia il messaggio legittimo (si ipotizzi per un momento che il messaggio potrebbe essere "attacco" o "ritirata").

Il protocollo Bitcoin avvolge il messaggio in un'equazione matematica incredibilmente difficile e la soluzione al problema è il messaggio legittimo ("attacco" o "ritirata").

Un ulteriore livello di sicurezza è costituito dall'equazione crittografica stessa. Se la soluzione può essere scoperta, non può comunque essere decodificata. Il messaggio trasformato non codificato è un mistero fino a quando l'hash crittografico non viene risolto. Per risolvere il BGP, Bitcoin invia il messaggio a tutti i generali contemporaneamente. Quando questi ultimi lo ricevono, iniziano a lavorare per risolvere il problema matematico. Il primo che risolve il problema trasmette la risposta agli altri generali e una volta che anche gli altri hanno risolto il problema, possono verificare di aver ricevuto tutti la stessa soluzione confrontandola con quella degli altri.

La potenza computazionale usata per risolvere il problema è la prova che i generali hanno effettivamente fatto il lavoro per risolvere il problema ed è nota anche come proof of work (prova di lavoro). I generali traditori possono trasmettere un messaggio falso, ma quando anche gli altri generali risolveranno il problema, non otterranno la stessa soluzione trasmessa dal traditore.

Quando il 51% dei generali riceve la stessa risposta, il messaggio viene considerato vero. Tale meccanismo è simile al modo in cui le scoperte scientifiche vengono presentate alla comunità per essere spiegate e rese poi note. Ad esempio, se qualcuno sperimenta una nuova invenzione, anche gli altri scienziati sono chiamati a replicare lo stesso esperimento. Se la maggioranza degli scienziati è in grado di riprodurre i risultati dell'esperimento originale, allora viene dichiarata una nuova scoperta.

Ogni volta che l'esperimento viene replicato con successo, diventa sempre più difficile per un hacker o per uno scienziato sleale tornare indietro e modificare il risultato di ogni esperimento. Ecco come Bitcoin diventa più forte e più sicuro man mano che cresce.

Infine, i generali leali vengono premiati con bitcoin per aver legittimamente trasmesso il messaggio corretto; questo allinea gli incentivi dell'individuo agli incentivi del gruppo. [11]

[11] The Bitcoin Big Bang - Brian Kelly - 2014

2.7. Introduzione alla Blockchain

La blockchain è un libro mastro sicuro, trasparente e decentralizzato.
Sicuro non significa che sia possibile nascondere le informazioni, ma significa semplicemente che nessuno può manomettere la blockchain senza provocare dei "campanelli d'allarme" che verrebbero rapidamente notati dagli sviluppatori. Anche se qualcuno tenta di alterare fraudolentemente la blockchain, i dati originali resterebbero comunque validi e si scoprirebbe il tentativo di manomissione confrontando le informazioni sui vari ledger distribuiti tra i nodi.
La blockchain è progettata per utilizzare un hash crittografico e il timestamp (marca temporale) per rendere il registro inalterabile. Per gli esperti di blockchain, questo permette di ispezionare i dati per determinare le caratteristiche di una transazione o per rilevare tentativi di manomissione. La blockchain di Bitcoin assicura che nessuna parte in uno scambio dovrà fidarsi di una singola terza parte che tiene per loro il denaro.
La tecnologia dietro la blockchain può essere utilizzata per creare un efficace sistema di deposito a garanzia automatizzato in cui il sistema non scomparirà con il denaro o negherà mai l'accesso ad ogni conto personale. Questo dà a Bitcoin la capacità di essere una valuta che può attraversare i confini internazionali, creare wallets Bitcoin ed evitare di essere controllata da qualsiasi autorità centralizzata. Il mantenimento della blockchain si basa sul funzionamento di più nodi in grado di memorizzare i dati delle transazioni e su processori capaci di convalidare gli scambi. Si pensi ai nodi come a dei server uguali che si aggiornano regolarmente l'un l'altro e che permettono ai clients autenticati di connettersi ad essi. In caso di malfunzionamento di un nodo, il personale IT può lavorare con un esperto di blockchain per isolare il server e risolvere i problemi determinando cosa sia andato storto. I malfunzionamenti di un nodo di solito comportano la mancata trasmissione di informazioni valide, come ad esempio dati che non hanno senso, oppure il rifiuto di trasmettere blocchi in rete. Il nodo potrebbe "andare fuori strada", creare la propria versione della blockchain e inventare informazioni che gli altri nodi non possono utilizzare in alcun modo significativo. In casi come questo, un esperto di blockchain avvertirà il personale IT di isolare il server dal resto della rete fino a quando non sarà in grado di individuare e risolvere il problema.
Se un nodo funziona male e deve essere isolato a causa di una manutenzione all'interno di un'azienda che si basa sull'accesso ai dati in tempo reale, non ci sarà una perdita di tempo e di ricavi perché i nodi rimanenti prenderanno il suo posto e i clienti verranno reindirizzati e potranno ricollegarsi rapidamente ad un nodo funzionante. Questa è un'ottima soluzione per un imprenditore che non vuole spiegare ai clienti perché il server di una rete centralizzata sia andato fuori uso.
La blockchain è stata originariamente progettata per essere un sistema decentralizzato che tiene traccia degli addebiti e degli accrediti: l'esistenza di migliaia di nodi Bitcoin in sei continenti diversi dimostra la sua capacità di poter in futuro diventare effettivamente il "World Wide Web" della finanza.

2.8. Crittografia in Bitcoin

La crittografia è una parte importantissima di Bitcoin, senza la quale la criptovaluta non sarebbe potuta esistere.

La crittografia è la scienza che permette di comunicare in maniera sicura in presenza di avversari che possono ascoltare e persino controllare il canale di comunicazione (come si vedeva nel Problema dei Due Generali).

La crittografia classica (simmetrica) si occupa della conversione di un messaggio in testo cifrato. Un testo cifrato è senza senso per l'avversario in ascolto sul canale di comunicazione, ma non il destinatario che sa come tradurlo nel messaggio originale. I crittografi raccomandano che l'algoritmo di cifratura sia reso pubblico e che solo la chiave di cifratura venga tenuta segreta: questo è conosciuto come il Principio di Kerckhoffs.

La logica alla base di questo principio è che sia facile realizzare un algoritmo di cifratura che non possa essere "rotto" dal suo creatore, ma sia molto difficile realizzare un algoritmo che non possa essere "rotto" da nessuno al mondo. Non importa quanto sia intelligente il creatore dell'algoritmo, c'è comunque un'alta probabilità che ci sia qualcuno più intelligente al mondo, per cui rendere pubblico un algoritmo di cifratura e sottoporlo all'esame della comunità crittografica è una buona idea. Il fulcro del Principio di Kerckhoffs è rendere pubblico l'algoritmo crittografico, ma mantenere private le chiavi di crittografia.

La maggior parte dei lavori iniziali di crittografia sono stati fatti su crittografia simmetrica. L'obiettivo della crittografia simmetrica è quello di criptare un messaggio utilizzando la chiave segreta, in modo che il messaggio originale possa essere recuperato solo se la chiave segreta è conosciuta. Una buona crittografia simmetrica non deve far trapelare alcuna informazione né sul messaggio né sulla chiave segreta.

Bitcoin utilizza tre tipi di differenti codici crittografici:

- Crittografia a chiave pubblica, utilizzata da Bitcoin per gestire le transazioni.
- Funzioni Hash, usata da Bitcoin per proteggere le informazioni nella blockchain.
- Crittografia a chiave simmetrica, usata da Bitcoin per proteggere le chiavi private nel portafoglio di un utente. Infatti, per evitare che utilizzatori non autorizzati possano accedere ai fondi contenuti in un portafoglio, le chiavi private memorizzate su un dispositivo sono di solito criptate. Quando l'utente ha bisogno di accedervi, ad esempio quando firma una transazione, fornisce al dispositivo la password. Il dispositivo decripta temporaneamente le chiavi private, le utilizza per firmare la transazione e pulisce la memoria dove sono state memorizzate le chiavi non criptate. Per crittografare le chiavi private di un portafoglio, Bitcoin utilizza la crittografia a chiave simmetrica. Il suo scopo è quello di "aggrovigliare" la transazione con la chiave fornita in modo tale che senza di questa sia impossibile districarla. La crittografia a chiave simmetrica utilizza la stessa chiave per la cifratura e la decifratura.

2.8.1. Crittografia a chiave pubblica

La crittografia a chiave pubblica è stata sviluppata negli anni Settanta da Diffie, Hellman e Merkle. Bitcoin non utilizza algoritmi di crittografia a chiave pubblica, ma un parente stretto chiamato firma digitale.
La crittografia a chiave pubblica è stata sviluppata come risposta ad un'importante debolezza della crittografia simmetrica: la distribuzione delle chiavi (si noti che bitcoin usa la crittografia simmetrica non per gestire le transazioni ma solo per la protezione delle chiavi private). Quando due persone utilizzano la crittografia simmetrica, devono prima assicurarsi di condividere la stessa chiave: devono scambiarsi le chiavi attraverso un canale sicuro prima di utilizzare il sistema di crittografia simmetrica. Tuttavia, ci sono molte situazioni in cui questa condizione preliminare è di fatto impossibile, come ad esempio l'e-commerce. Internet è un canale insicuro: il traffico può essere intercettato e persino modificato durante il transito. Pertanto, è impossibile stabilire una connessione sicura attraverso Internet utilizzando solo la crittografia simmetrica ed infatti la crittografia a chiave pubblica è stata sviluppata per superare questo problema.
Può essere fatta un'analogia tra la crittografia a chiave simmetrica e una cassaforte con una sola chiave. Questa chiave (simmetrica) può essere usata sia per bloccare (cifratura) che per sbloccare la cassaforte (decifratura). Invece un'analogia simile per la cifratura a chiave pubblica è quella con una cassaforte con una coppia di chiavi. Una delle chiavi, la chiave pubblica, può essere usata solo per bloccare la cassaforte, mentre l'altra chiave, la chiave privata, può essere usata solo per sbloccare la cassaforte.
Nella crittografia a chiave pubblica, un punto importante è che solo la chiave privata (la chiave che apre la cassaforte) deve essere tenuta segreta. È perfettamente sicuro pubblicare la chiave pubblica (la chiave che chiude la cassaforte), perché più ampia è la sua distribuzione, più facile è per una controparte averne accesso e utilizzarla per comunicare.

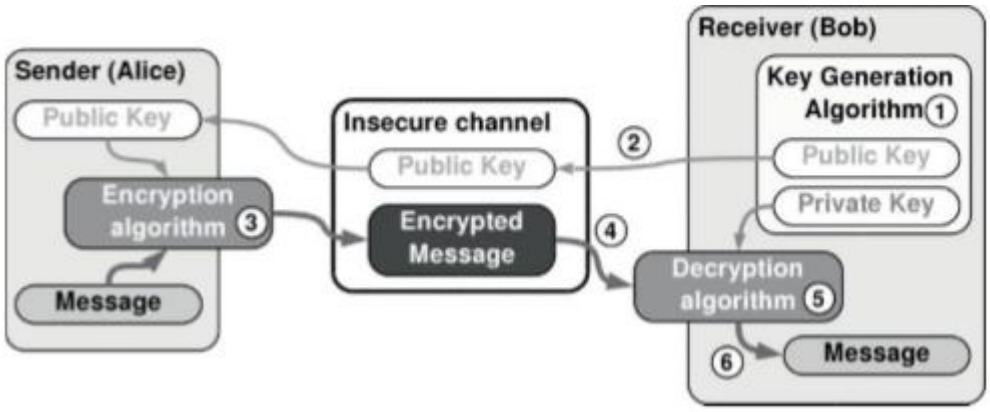


Figura 19 – Crittografia a chiave pubblica – Understanding Bitcoin – Pedro Franco

La Figura 19 mostra la crittografia a chiave pubblica applicata ad un messaggio. Per prima cosa, il destinatario del messaggio crittografato (Bob) genera una coppia di chiavi pubbliche e private, eseguendo un algoritmo di generazione delle chiavi. Le chiavi pubbliche e private sono chiamate “public-pair keypair” e sono collegate matematicamente. Ogni protocollo a chiave pubblica ha il proprio algoritmo di generazione delle chiavi. Il destinatario (Bob) invia la sua chiave pubblica al mittente (Alice), ma tiene la sua chiave privata al sicuro.

Dopo aver ricevuto la chiave pubblica di Bob, Alice procede a criptare il messaggio utilizzando la chiave pubblica di Bob. Il risultato è il messaggio criptato (ciphertext), che però è stato inviato su un canale che potrebbe essere insicuro. Un aggressore che intercetta la connessione può entrare in possesso del messaggio cifrato, ma non può decifrarlo. Solo Bob, che ha la chiave privata corrispondente alla chiave pubblica, è in grado di decifrare il messaggio cifrato utilizzando l'algoritmo di decodifica, ottenendo quindi il messaggio originale.

2.8.2. Firme Digitali

Una seconda applicazione della crittografia a chiave pubblica è quella delle firme digitali. L'obiettivo delle firme digitali è simile a quello delle firme scritte a mano: assicurano che un messaggio generato dal firmatario non sia stato manomesso e che la firma non sia contestabile (il firmatario di un messaggio non dovrebbe poter negare di averlo firmato).
Le firme digitali sono utilizzate nel protocollo Bitcoin.
Gli indirizzi Bitcoin sono fondamentalmente chiavi pubbliche. Esiste una chiave privata corrispondente ad ogni chiave pubblica che coincide quindi con ogni indirizzo Bitcoin. Le chiavi pubbliche possono essere interpretate come i numeri di un conto bancario, mentre le chiavi private possono essere interpretate come le firme che sbloccano quei conti. Per spendere i bitcoin di un indirizzo, una transazione che autorizza la spesa deve essere firmata con la chiave privata. Il software del portafoglio crea un indirizzo Bitcoin eseguendo l'algoritmo di generazione delle chiavi.

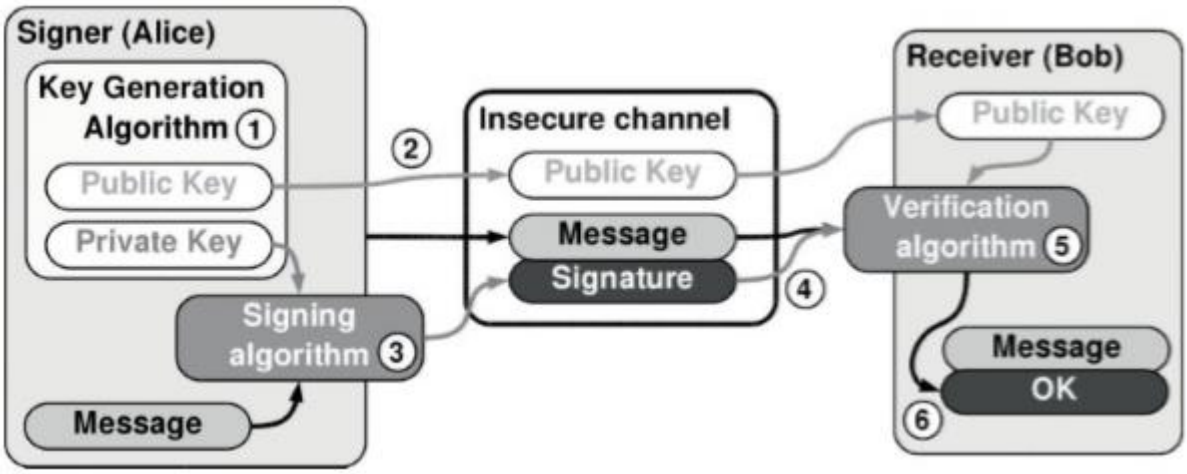


Figura 20 – Firme Digitali - Understanding Bitcoin – Pedro Franco

La Figura 20 mostra il processo di firma. In primo luogo, il firmatario (Alice) genera una coppia di chiavi pubbliche e private, utilizzando l'algoritmo di generazione delle chiavi. Lei invia la sua chiave pubblica attraverso il canale di comunicazione e successivamente usa la chiave privata per firmare digitalmente il messaggio. È importante che tenga la chiave privata per sé, non rivelandola a nessuno. Una volta firmato il messaggio, sia il messaggio che la firma vengono inviati al destinatario (Bob). Si noti che il messaggio non è

criptato, ma solo autenticato. Bob verifica la firma, utilizzando la chiave pubblica di Alice. Se la verifica dà un risultato positivo, sa che il messaggio è stato spedito da lei.
Nella crittografia a chiave pubblica, la firma vincola il mittente al messaggio. Solo il firmatario (Alice) ha una copia della chiave privata, quindi solo lei può firmare un messaggio con quella chiave. Inoltre, per dimostrare che un messaggio è stato effettivamente firmato da Alice, Bob deve solo mostrare ad un'ipotetica terza parte il messaggio, la chiave pubblica e la firma. In sintesi, la crittografia a chiave pubblica risolve tutti i problemi e consente una firma digitale sicura e vincolante.
Un punto tecnico riguardante le firme digitali è rappresentato dal fatto che il messaggio potrebbe essere di lunghezza arbitraria. Questo può essere un problema perché gli algoritmi utilizzati dalla crittografia a chiave pubblica sono piuttosto lenti. La soluzione a questo problema è prendere prima l'hash del messaggio (arbitrariamente lungo) e solo dopo firmarlo. L'output di una funzione hash è della stessa lunghezza indipendentemente dalle dimensioni dell'input: usando questo stratagemma, i messaggi di qualsiasi lunghezza possono essere firmati. Un protocollo di firma digitale è la combinazione di un algoritmo a chiave pubblica con uno schema di firma digitale. L'algoritmo a chiave pubblica fornisce il metodo matematico asimmetrico sottostante, mentre lo schema della firma digitale propone un modo per utilizzare questo algoritmo asimmetrico al fine di ottenere una firma digitale funzionante. [12]

Essendo la blockchain e gli smart contract l'argomento centrale della tesi, non penso sia necessario addentrarsi ulteriormente all'interno degli schemi crittografici di Bitcoin riguardanti chiave pubblica e chiave privata. Basti sapere che Bitcoin usa una curva ellittica crittografica per le firme digitali, che può essere vista come una grande successione di punti.
Il protocollo della curva ellittica parte da un punto noto, chiamato generatore. Una chiave pubblica è un punto della curva ellittica. Una chiave privata è il numero di step dal generatore che devono essere attraversati per arrivare al punto della chiave pubblica. Il calcolo della chiave pubblica, data la chiave privata, è molto veloce, ma l'inverso, dato che la chiave pubblica trova la chiave privata, è praticamente impossibile. Questo metodo è noto come il problema del logaritmo discreto. L'algoritmo di forza bruta per risolvere il problema del logaritmo discreto attraverserebbe i punti della curva ellittica uno alla volta partendo dal generatore fino ad arrivare al punto desiderato. Fortunatamente questo algoritmo è computazionalmente impossibile, dato che impiegherebbe un numero irragionevole di anni per essere portato a termine con i classici computer.

[12] La sicurezza dietro Bitcoin: Crittografia Asimmetrica – Danilo Giudice - 2018

2.9. Transazioni

Le transazioni sono composte da una lista di input (TxIn) e una lista di output (TxOut). Ogni output di transazione (TxOut) contiene due dati: un importo e l'indirizzo del destinatario. L'indirizzo è derivato dalla chiave pubblica: solo il proprietario della chiave privata può sbloccare i fondi memorizzati nel TxOut. Per farlo, il proprietario della chiave privata deve firmare una transazione in cui invia i fondi ad un nuovo indirizzo Bitcoin. Un input di transazione (TxIn) contiene un riferimento ad un precedente output di transazione e una firma che prova che i fondi nel precedente TxOut a cui fa riferimento possono essere spesi. Questa firma deve essere fatta con la chiave privata associata alla chiave pubblica nell'indirizzo Bitcoin. Se la firma non corrisponde, la transazione è considerata non valida e viene eliminata dalla rete. Una transazione raggruppa diversi TxIn e TxOut (almeno uno per ciascuno). Lo scopo di una transazione è quello di far passare i fondi dall'input all'output. Gli input in una transazione si riferiscono agli output delle transazioni precedenti. Questi output non devono essere già stati spesi, altrimenti la transazione non è valida. Affinché la transazione sia corretta, la somma degli importi degli input deve essere maggiore o uguale alla somma degli output. La differenza tra gli input e gli output, se esiste, è la commissione della transazione. Questa fee è percepita dai minatori che includono la transazione in un blocco.

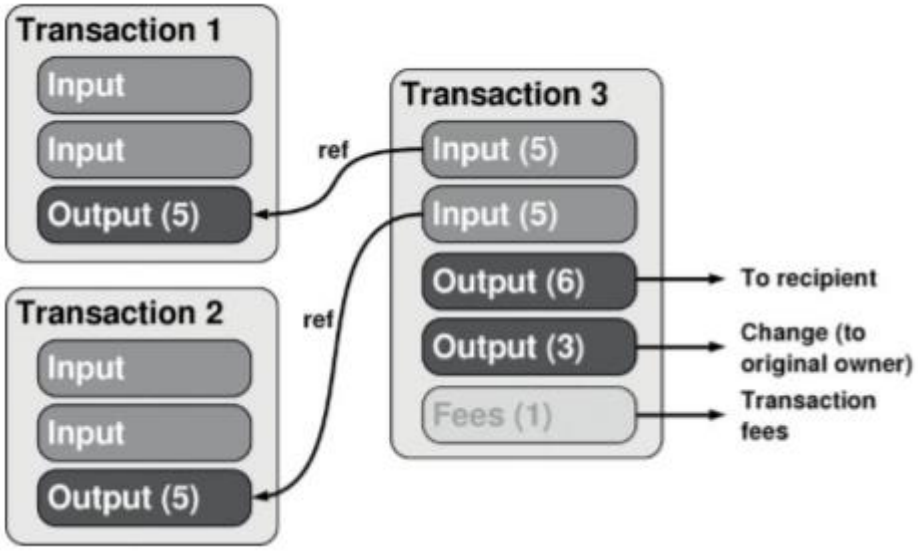


Figura 21 – Transazione - Understanding Bitcoin – Pedro Franco

Nella Figura 21, il mittente vuole inviare 6 bitcoin al destinatario. Tuttavia, il mittente non ha a sua disposizione alcun output con un saldo di esattamente 6 bitcoin, dato che controlla solo due output con un saldo di 5 bitcoin ciascuno. Così crea una transazione che raggruppa questi due output e invia 6 bitcoin al destinatario. Per farlo inserisce un output verso l'indirizzo che voleva e riceve come resto 3 bitcoin lasciandone 1 come fee per i minatori. Prima di inviare la transazione nel network, deve firmare i due input, per dimostrare di controllare gli indirizzi indicati. La transazione viene poi inviata alla rete. Il primo nodo della rete che la riceve, la verifica per vedere se si tratti di una transazione valida. Se è corretta, il nodo la ritrasmette ad altri nodi della rete.

Per verificare che una transazione sia valida, un nodo segue questi passi:

- Controlla che gli output precedenti, a cui fa riferimento la transazione, esistano e che non siano stati già spesi. Il nodo esegue questo controllo consultando la cache degli output delle transazioni non spese (UTXO).
- Controlla che la somma dei valori degli input sia maggiore o uguale alla somma degli output. Vale a dire, controlla che la transazione non stia spendendo più di quello che è disponibile. La differenza tra la somma del valore degli output e la somma del valore degli input è considerata la commissione per il miners ed è inclusa nella transazione coinbase (si vedrà nel dettaglio dopo).
- Verifica che le firme per ciascuno degli input siano valide, cioè che ciascuno degli input sia firmato con la chiave privata corrispondente alla chiave pubblica associata all'indirizzo cui si riferisce.

Satoshi ha imposto che l'output di una transazione sia speso interamente, perché è computazionalmente più efficiente. Il software di Bitcoin mantiene una cache degli output delle transazioni non spese (UTXO). L'UTXO è un database molto utile perché può essere usato per controllare rapidamente se le nuove transazioni siano valide. Quando arriva una nuova transazione, i suoi input vengono cercati nell'UTXO: se vengono trovati, allora essi corrispondono agli output già precedentemente validati e la transazione continua ad essere analizzata. Se uno qualsiasi degli input non viene trovato nell'UTXO, la transazione non è considerata valida e viene scartata.

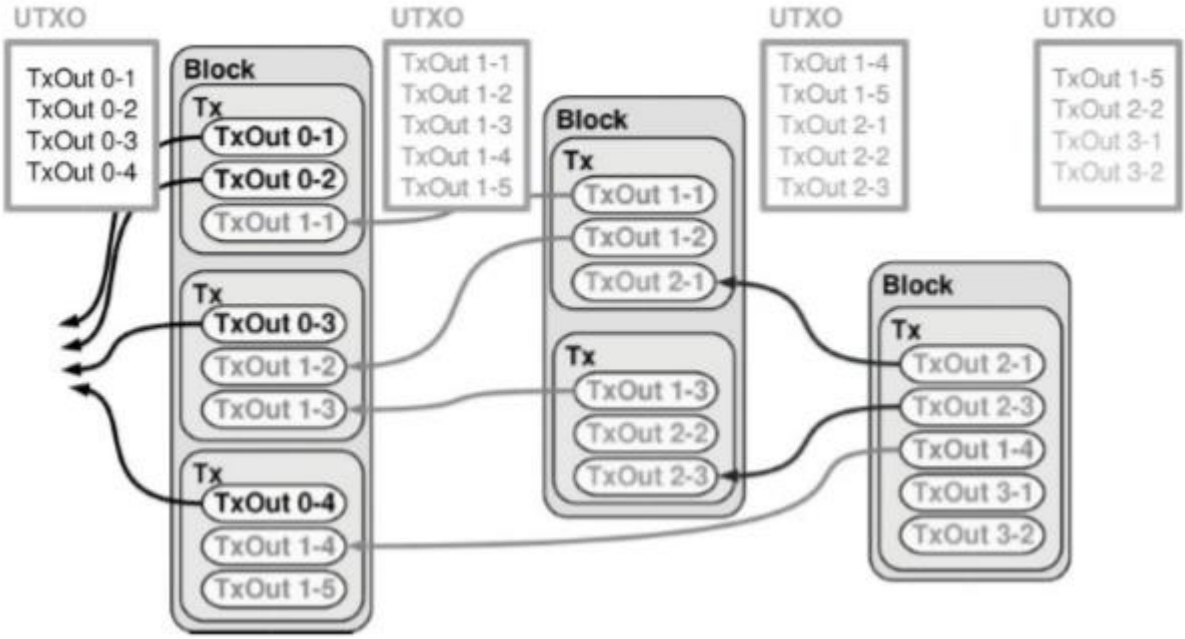


Figura 22 – UTXO - Understanding Bitcoin – Pedro Franco

La Figura 22 fa vedere il funzionamento dell'UTXO. Inizialmente ci sono quattro transazioni non spese nella rete (da 0-1 a 0-4) e l'UTXO certifica queste quattro TxOut.

Nella fase successiva viene creato un blocco che include tre transazioni. Le tre transazioni spendono i quattro TxOut dell'UTXO, rimuovendoli. Tuttavia, le nuove transazioni nel blocco introducono anche nuove uscite (da 1-1 a 1-5) e queste uscite sono incluse nell'UTXO. Il blocco successivo spende tre dei cinque TxOut nell'UTXO (da 1-1 a 1-3) e aggiunge tre nuovi TxOut (da 2-1 a 2-3) e così via. Con l'arrivo di ogni nuovo blocco, le uscite che vengono spese vengono rimosse, ma le nuove uscite create dalle transazioni nel blocco vengono aggiunte all'UTXO. Il vantaggio di mantenere un UTXO è che è molto più piccolo dell'intero database

delle transazioni (la blockchain). Questo permette ai nodi di mantenere l'UTXO in RAM, il che velocizza notevolmente il controllo della validità delle nuove transazioni. Dove si trovano intanto i bitcoin? Si potrebbe dire che i bitcoin risiedono negli output non spesi delle transazioni nella blockchain.

2.9.1. Transaction Scripts

Ogni output crea un puzzle matematico che per essere speso, deve essere risolto. Il puzzle per sbloccare i fondi e la soluzione sono rappresentati da due script. Lo script che crea il puzzle si chiama <scriptPubKey>, perché è la parte dello script che contiene la chiave pubblica, mentre il puzzle che risolve la <scriptPubKey>, sbloccando così i fondi, si chiama <scriptSig>, perché è la parte dello script che contiene la firma.

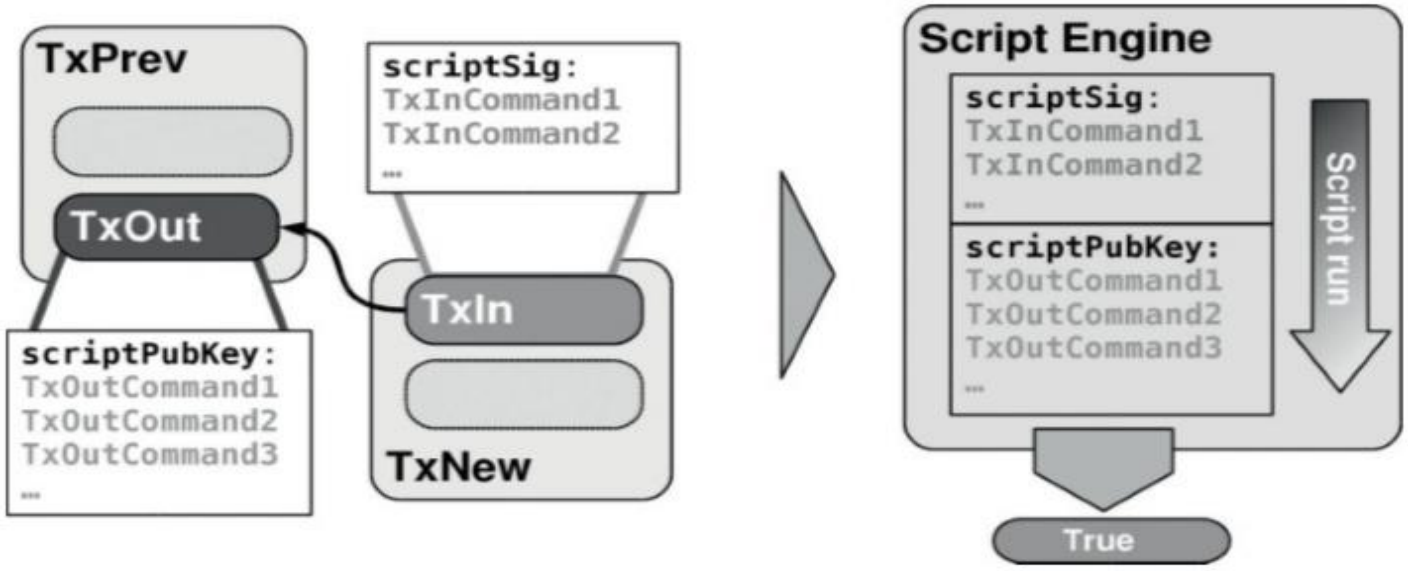


Figura 23 – Spesa di un output - Understanding Bitcoin – Pedro Franco

La Figura 23 presenta il processo di spesa di un output. Un output (TxOut) crea un <scriptPubKey> che deve essere risolto per spendere i fondi contenuti. L'input (TxIn) che tenta di spendere quell'output definisce un <scriptSig>. Il protocollo verifica che il <scriptSig> risolva il puzzle matematico creato dal <scriptPubKey>. Per fare questo, il protocollo crea uno script completo concatenando il <scriptSig> con il <scriptPubKey> e lo esegue: se il risultato finale valuta il calcolo "true", allora l'input è considerato valido. Se lo script fallisce a metà o se il risultato finale non valuta "true", l'input non è valido e l'intera transazione viene considerata falsa e abbandonata.

Il linguaggio di scripting in Bitcoin è abbastanza flessibile e potente, ma non è Turing-completo (un linguaggio Turing-completo può essere usato per istruire un computer a risolvere qualsiasi problema di calcolo). È stato deciso in questo modo per evitare attacchi alla rete. Se il linguaggio di scripting fosse Turing-completo, un attaccante potrebbe creare un <scriptPubKey> che entra in un loop infinito. Ciò causerebbe lo stallo dei nodi che valutano questo script, causando il crollo della rete: per questo motivo si è deciso che il linguaggio non contenga loop.

2.10. Funzioni hash

Una funzione hash è un algoritmo che prende dati di lunghezza arbitraria come input ed emette una stringa di bit di lunghezza fissa, chiamata hash value, che è sempre la stessa se si hanno gli stessi dati di ingresso, mentre, con piccole differenze di input, si producono grandi differenze nel risultato.

Un requisito comune è che il calcolo dell'hash sia veloce. Le funzioni hash value hanno una dimensione inferiore a quelle dei dati di ingresso e sono ampiamente utilizzate nel calcolo, per esempio per localizzare rapidamente i dati usando tabelle. Una buona funzione di hash dovrebbe distribuire i valori di input agli hash value in maniera proporzionale, in modo che ogni hash value sia collegato all'incirca allo stesso numero di possibili valori di input. Un modo per raggiungere questa proporzionalità è far sì che l'hash value "si comporti" nel modo più casuale possibile. Si noti che, sebbene l'hash si "comporti" in modo apparentemente casuale, è comunque deterministico: dato un input, il suo valore in hash sarà sempre lo stesso.

Bitcoin utilizza funzioni di hash crittografico per eseguire la proof of work. Le funzioni di hash crittografico (chiamate anche funzioni di hash sicuro) impongono dei requisiti aggiuntivi rispetto a quelli delle normali funzioni di hash:

- Unidirezionalità (resistenza di pre-immagine). Dato il valore di hash, deve essere computazionalmente impossibile trovare i dati dell'input. Questa è una proprietà chiave per l'applicazione della proof of work.
- Debole resistenza alle collisioni. Dato un input, è computazionalmente impossibile trovare un altro input con lo stesso hash.
- Forte resistenza alla collisione. E' computazionalmente impossibile trovare due punti di input che risultino nello stesso valore di hash.

Bitcoin utilizza SHA256^2 come funzione per la proof of work. SHA256^2 è l'applicazione della funzione hash SHA256 applicata due volte, il cui output è lungo 256 bit. La Figura 24 mostra un esempio di funzione hash SHA256^2. Il valore di input nella parte superiore della Figura 24 è la stringa "This is the message #1". Il valore hash in esadecimale è "7800e9f07...". Nella parte inferiore della Figura 24, il valore di input è la stringa "This is the message #2" e il valore hash è "263cec7db...". Si noti come una modifica di una sola cifra nel messaggio porta ad un valore di hash completamente diverso. Una buona funzione di hash si comporta come una trasformazione casuale dal valore di input al valore di hash.

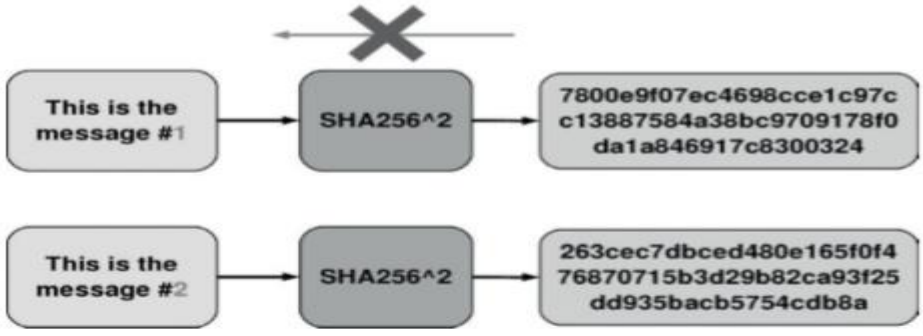


Figura 24 – Funzione Hash - Understanding Bitcoin – Pedro Franco

SHA256 soddisfa il requisito della resistenza di pre-immagine: dato l'hash, è computazionalmente impossibile recuperare il messaggio che lo ha generato. Computazionalmente impossibile significa che non esiste un algoritmo conosciuto che possa recuperare il messaggio entro un tempo che sia correlato algebricamente alla dimensione dell'input. Quindi in pratica gli algoritmi più noti per rompere una funzione di hash, cioè recuperare il messaggio dato l'hash, sono algoritmi di forza bruta che richiedono un tempo (esponenziale) enorme. Molte funzioni crittografiche di hash, tra cui SHA256, sono costruite a partire da una funzione primitiva più semplice chiamata "funzione di compressione". Una funzione di compressione opera su un input di lunghezza fissata e produce un output della stessa lunghezza. Lo scopo di una funzione di compressione è quello di rimescolare i bit dell'input in modo deterministico ma complicato per arrivare all'output. Questo si ottiene inserendo il messaggio originale attraverso una serie di operazioni di data-shifting e mixing con costanti di tipo casuale.

La costruzione Merkle-Damgård è una tecnica per costruire funzioni di hash crittografico che accettano dati di input di lunghezza arbitraria usando una funzione di compressione come blocco di costruzione. Merkle e Damgård hanno dimostrato che, se la funzione di compressione è resistente alle collisioni, allora anche l'intera costruzione è resistente alle collisioni. La Figura 25 sotto mostra la costruzione generale. Nel caso di SHA256, la funzione di compressione (contrassegnata con f nella Figura 25) opera su 256 bit di dati. La funzione di compressione accetta due ingressi: un intermedio hash value e un blocco di input. Per calcolare l'hash SHA256, l'input (chiamato anche messaggio) viene prima suddiviso in blocchi di 256 bit, mentre la fine è riempita con zeri e la lunghezza del messaggio. Il valore intermedio dell'hash viene inizializzato partendo dall'IHV (Initial Hash Value). Poi la funzione di compressione viene applicata ad ogni blocco del messaggio, utilizzando l'hash value dell'ultimo step come valore intermedio dell'hash nella fase successiva. Il valore di hash prodotto dall'ultimo step è l'hash SHA256 dell'intero messaggio.

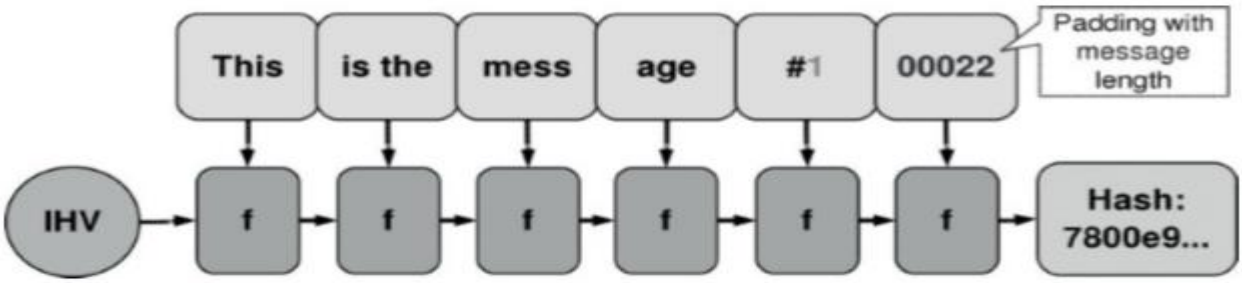


Figura 25 – Funzione Merkle-Damgård - Understanding Bitcoin – Pedro Franco

SHA256, come la maggior parte delle funzioni di hash crittografico, è stata progettata per essere veloce su hardware di uso generale. Inoltre, poiché le funzioni di compressione utilizzano solo operazioni bitwise (nel senso che semplificano alcune operazioni logiche con un solo simbolo) e aggiunte di registri a 32 bit, è adatto per un'efficiente implementazione su hardware.

La costruzione Merkle-Damgård è vulnerabile agli attacchi di "length extension".

Si supponga che un aggressore non conosca il messaggio m, ma conosca il suo hash H(m). Potrebbe calcolare l'hash di un messaggio m|m´ (dove | denota la concatenazione) usando H(m) come IHV all'algoritmo dell'hash e poi lo implementa con m´, ottenendo così H(m|m') come risultato.

La soluzione al problema di "length extension" viene proposta facendo l'hash due volte. Vale a dire, applicare di nuovo la funzione hash all'hash del messaggio. Infatti, è proprio per questo motivo che si ritiene che Satoshi Nakamoto abbia scelto SHA256^2 invece del semplice SHA256 come protezione contro gli attacchi di length extension. Non è chiaro come sfruttare il length extention contro la proof of work, ma sembra che Satoshi avesse deciso di andare sul sicuro. Inoltre, il principale svantaggio del doppio hash, ovvero l'aumento del tempo di calcolo computazionale, è irrilevante in Bitcoin, perché lo scopo della proof of work è quello di rendere computazionalmente costosa la ricerca del nonce (codice numerico casuale utilizzato una volta sola, che viene cercato dai miners come soluzione dei calcoli per verificare le transazioni e ricevere la ricompensa in bitcoin).[13] [14]

[13] Cryptography hash functions – Tutorialspoint - 2018

[14] Cryptographic hash function – Tim Fisher - 2020

2.11. Timestamp

Un timbro (timestamp) digitale è analogo a un timbro fisico, come il timbro postale su una lettera o un francobollo di un organo ufficiale.

Un timbro digitale dimostra che certe informazioni, come ad esempio un documento digitale, esistevano in un determinato momento. Esso ha molte applicazioni, come per esempio documentare che sia stato stipulato un contratto tra due parti, che si sia concretizzata una transazione in un sito web o che si sia svolto un insieme di scambi in valuta digitale. L'informazione inclusa in un timestamp digitale è di solito l'hash dei dati da proteggere e il suo utilizzo ha diversi vantaggi. In primo luogo, le informazioni che devono essere contrassegnate con un timestamp possono essere mantenute private e separate dal mezzo utilizzato per la protezione del timestamp. In secondo luogo, un hash è di solito significativamente più piccolo delle informazioni che lo hanno generato, riducendo i costi di archiviazione. In terzo luogo, le firme digitali di solito funzionano meglio su dati di dimensioni predeterminate.

Ci sono diversi modi per garantire un timestamp digitale. Un metodo che va contro i principi blockchain consiste nell'inviare ad una persona di fiducia una copia delle informazioni affinchè lei le memorizzi in un luogo (si spera) sicuro, insieme all'ora di ricezione. Questo metodo è soggetto alla possibilità che la terza parte di fiducia perda il database o venga hackerata. Un secondo metodo si affida ancora ad una terza parte di fiducia, chiamata TimeStamping Authority (TSA) (chiamata anche notaio digitale) che firma con la sua chiave privata una combinazione di dati da proteggere insieme all'ora in cui questi dati le sono stati comunicati; questa firma viene rispedita successivamente al proprietario originale dei dati. La sicurezza di questo sistema si basa sull'integrità della TSA. Se quest'ultima decidesse di collaborare con un impostore, potrebbe apporre retroattivamente la data e l'ora di un documento. Un terzo metodo per garantire un timestamp digitale sarebbe quello di pubblicare l'hash dei dati in un luogo pubblico, ad esempio un giornale (come mostrato in Figura 26). Un aggressore, che volesse modificare una qualsiasi delle transazioni, dovrebbe trovare un'incongruenza con l'hash pubblicato o, in alternativa, dovrebbe modificare tutti i giornali con una copia dell'hash pubblicato.

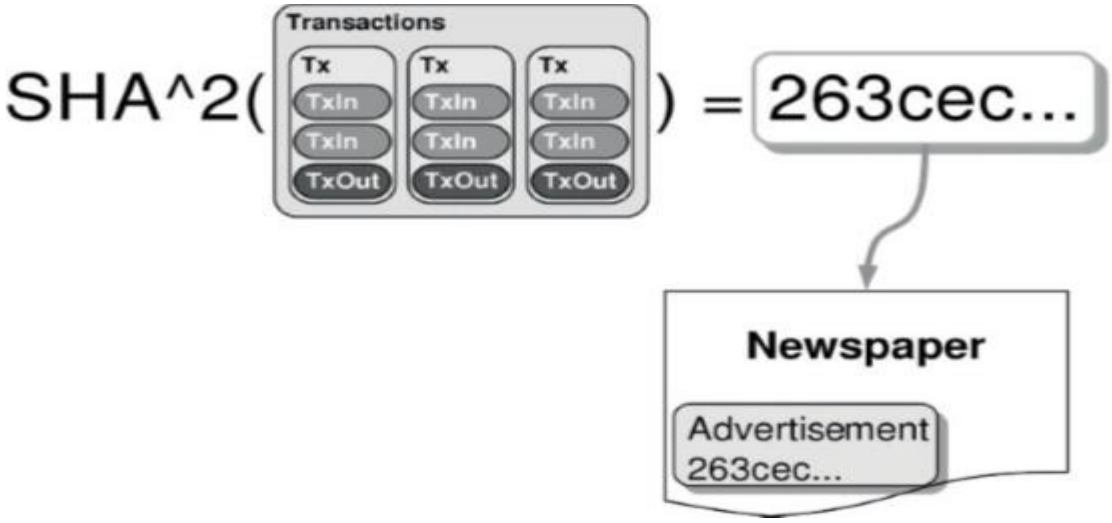


Figura 26 – Timestamp pubblicato in un giornale - Understanding Bitcoin – Pedro Franco

La Figura 27 introduce un altro perfezionamento usato per rendere più sicuri i timestamps digitali: i linked timestamps (timestamps collegati). L'intuizione principale è che se l'hash da pubblicare è già collegato all'hash precedentemente pubblicato, questo aiuterà a rendere più sicuro l'hash più vecchio. Infatti, un aggressore che desidera modificare alcuni dati del vecchio hash dovrebbe fare un doppio cambiamento: uno per il vecchio hash e un altro per quello nuovo che è stato appena pubblicato. Dopo un altro passo nella catena, l'aggressore dovrebbe fare una tripla modifica. In questo modo, la catena di hash aumenta in modo esponenziale anche la sicurezza sui vecchi hash.

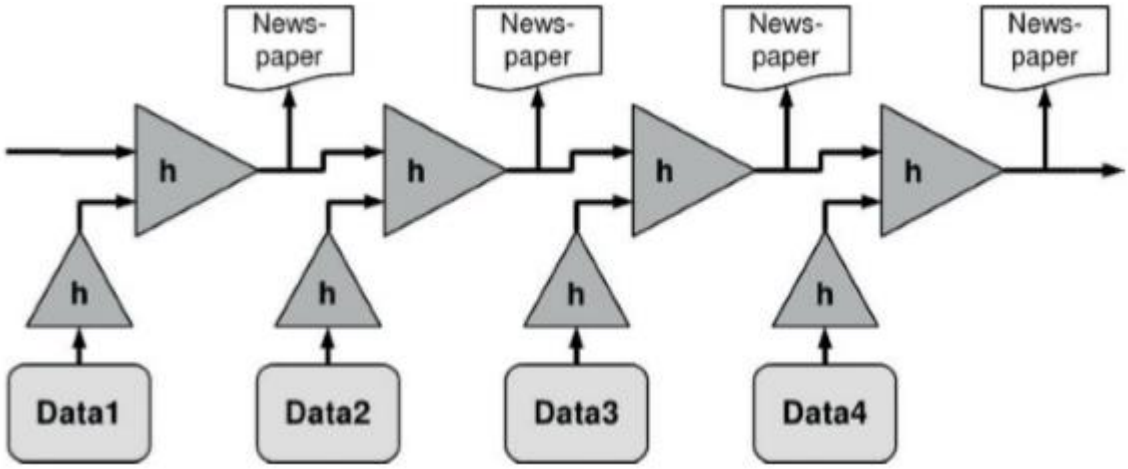


Figura 27 – Timestamp collegati - Understanding Bitcoin – Pedro Franco

Anche se nella Figura 27 l'hash viene pubblicato in ogni momento, gli hash potrebbero essere promulgati con una periodicità più lunga per risparmiare sui costi.

In sintesi, una TimeStamping Authority potrebbe funzionare nel seguente modo: in primo luogo, i dati da timbrare dai clienti vengono raccolti durante un certo periodo. Al termine del periodo di raccolta, i dati vengono crittografati in hash con l'ausilio, ad esempio, di un albero di Merkle (si vedrà in seguito). L'hash risultante viene messo in hash insieme all'hash finale del periodo precedente e infine viene pubblicato. Una valuta digitale potrebbe utilizzare questo metodo per garantire le transazioni, ma ciò richiedebbe una controparte centralizzata che raccolga le transazioni e le pubblichi. La stessa avrebbe anche bisogno di un mezzo pubblico, come la pubblicità sui giornali, per pubblicare gli hash, con la controparte centrale che sostiene presumibilmente il costo della pubblicazione. Questa controparte, che gestisce la valuta, costituisce un punto centrale di fiducia nel sistema, ma il principio di progettazione principale di Bitcoin è la fiducia distribuita e la proof of work è l'ultimo pezzo del puzzle per realizzarla. [15] [16]

[15] What is Bitcoin Timestamp? Can it be changed? – Dotwallet.com - 2019

[16] Bitcoin's Block Timestamp Protection Rules – Bitmex.com - 2019

2.12. Proof of work (prova di lavoro)

I servizi digitali sono soggetti a diversi tipi di attacchi. Uno di questi è per esempio il denial of service (DoS) che si verifica quando un server viene inondato di richieste fasulle in cambio dei suoi servizi. Se il numero di richieste fasulle è superiore al numero di richieste che il server è in grado di soddisfare, gli utenti legittimi non potranno usare il sito o subiranno ritardi. Un denial of service interrompe il regolare andamento di un sistema e si dovrebbe quindi cercare di evitarlo in tutti i modi: un attacco DoS è per esempio lo spamming via e-mail, in cui un account di posta elettronica privato viene inondato di e-mail non richieste, che di solito contengono pubblicità o malware.

Gli attacchi DoS sono diventati particolarmente famosi quando gli hackers di Anonymous li hanno lanciati verso diversi siti web.

Un attacco DoS coinvolge solitamente molti partecipanti, situati in luoghi fisici differenti.

Talvolta un aggressore può arruolare un botnet (rete di computer infettati da software dannosi in modo da poter essere controllati da remoto. I computer vengono forzati a inviare spam, diffondere virus o lanciare attacchi DDoS senza che i veri proprietari ne siano consapevoli) per lanciare i cosiddetti attacchi DDoS (Distributed Denial of Service); questi sono più difficili da contrastare poiché l'intrusione nel sito web proviene da numerosi indirizzi IP; ciò che rende più difficile distinguere gli aggressori dagli utenti legittimi.

Una possibile difesa contro i DDoS consiste nell'esigere che il cliente che richiede il servizio abbia fatto un certo lavoro, cioè una proof of work (prova di lavoro). La prova di lavoro potrebbe essere un problema di calcolo, di memoria, di intervento utente e così via. La proof of work deve essere moderatamente difficile da risolvere ma facile (computazionalmente veloce) da verificare.

Un fornitore potrebbe per esempio porre il problema a chiunque richiede il suo servizio, concedendolo solo a quegli utenti che svolgono il calcolo. Se la prova di lavoro è ben progettata, questo costo sarà un piccolo inconveniente (come un breve ritardo) per gli utenti legittimi, ma un forte deterrente per gli aggressori.

I sistemi proof of work possono essere implementati seguendo due protocolli:

- Challenge-Response: questo protocollo presuppone una comunicazione continuativa tra il client e il server. Prima il client richiede il servizio, poi il server sceglie una prova di lavoro e sfida il client. Il client deve risolvere la prova del lavoro e inviare la risposta al server che verificherà se questa sia stata eseguita correttamente per ricevere la concessione dell'accesso al servizio. Il vantaggio di questo protocollo è che il server può adattare la difficoltà della proof of work alle caratteristiche del client.
- Solution-Verification: questo protocollo è asincrono: la soluzione e la verifica possono essere effettuate in tempi diversi. Non è richiesta una comunicazione continua tra il server e il client. In primo luogo, il client crea un problema di proof of work e lo risolve: questo problema dovrebbe essere diverso ogni volta e scelto da un algoritmo. Poi il client invia la soluzione al server, che la verifica e procede di conseguenza.

Per rendere sicura la blockchain, Bitcoin richiede che la proof of work sia eseguita su blocchi di transazioni che seguono il protocollo Solution-Verification. Bitcoin utilizza l'inversione parziale dell'hash come funzione di proof of work che richiede che l'hash di un blocco di transazioni corrisponda ad un certo modello, ovvero che l'hash inizi con un certo numero di zeri.

È importante che la funzione di hash sia preimage-resistant (resistenza pre-immagine), ovvero dia la sicurezza di essere una funzione difficile da invertire; cioè, dato un elemento nel range di una funzione hash, dovrebbe essere computazionalmente impossibile trovare un input che permetta di risalire a quell'elemento. Quindi, per tutti gli output predefiniti, è impossibile trovare un input che riconduca a quell'output; pertanto, dato y, è difficile trovare una x tale che h(x) = y in cui h è una funzione in hash, altrimenti non sarebbe computazionalmente difficile trovare un'inversione parziale dell'hash, vanificando lo scopo della proof of work.

La Figura 28 mostra l'inversione parziale dell'hash applicata al messaggio "Questo è il messaggio #". La funzione hash utilizzata nella Figura 28 è SHA256^2, la funzione hash di Bitcoin. L'inversione parziale dell'hash in Figura 28 richiede che i primi 12 bit (3 caratteri esadecimali nella figura) siano zero. Il sistema di calcolo esadecimale è un sistema posizionale di base 16. Per comporre un numero, vengono usati 16 simboli: le cifre da 0 a 9 e le lettere dalla A alla F. Il modo per risolvere l'hash parziale è il nonce collegato al messaggio: infatti si deve trovare un nonce tale per cui esso corrisponda all'hash parziale, cioè che l'hash inizi con almeno tre caratteri zero. La Figura 28 incrementa il nonce fino a quando una soluzione non viene trovata. La soluzione si trova nel nonce 6193. Anche se questo esempio inizia con un nonce di valore 1 e lo incrementa su base unitaria, nulla impedisce di provare i nonce a caso. Per le funzioni hash ben programmate, il costo di calcolo di entrambi i metodi è equivalente.

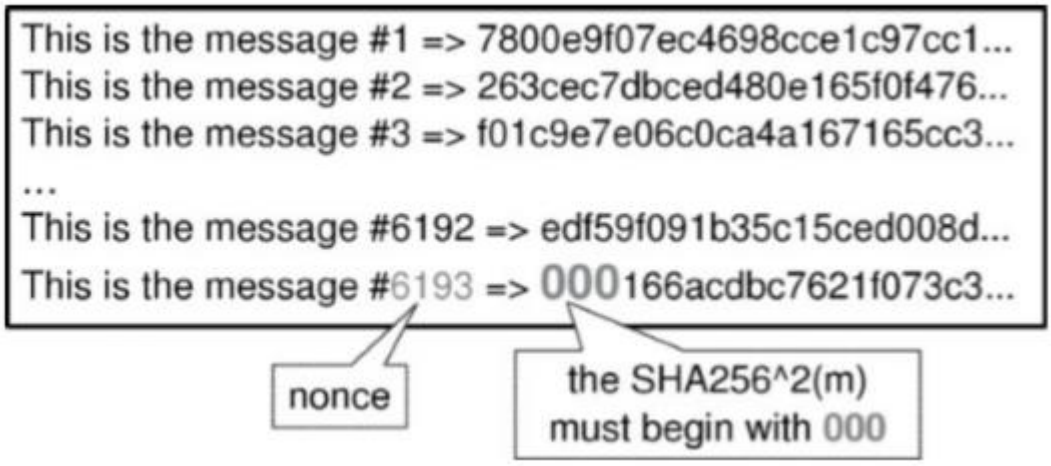


Figura 28 – Inversione parziale dell'hash - Understanding Bitcoin – Pedro Franco

È computazionalmente costoso risolvere l'inversione parziale dell'hash, perché sono stati provati molti nonce (6.193 in questo caso) prima di trovare la soluzione corretta, ma allo stesso tempo è anche computazionalmente poco costoso verificare che il lavoro sia stato fatto, dato che richiede solo la corretta forma alfanumerica dell'hash. Uno dei vantaggi dell'uso dell'inversione parziale dell'hash è che la difficoltà della proof of work può essere facilmente aggiustata cambiando il numero di 0 bit (zeri) con cui deve iniziare l'hash della soluzione (uno degli aspetti geniali di Satoshi Nakamoto che aveva capito che l'hashing power della rete Bitcoin poteva

subire ampie variazioni, per cui anche la proof of work si sarebbe dovuta adeguare di conseguenza, per mantenere la chiusura di un blocco ogni 10 minuti).

Bitcoin utilizza in particolare il sistema Hashcash, da cui ha preso in prestito l'inversione parziale dell'hash, come metodo per la prova di lavoro.

Hashcash è stato introdotto da Adam Back nel 1997 come meccanismo di limitazione per lo spamming via e-mail, che, a causa del costo nullo, era diventato pratica ricorrente. Gli spammer inviavano milioni di e-mail non richieste con pubblicità o malware. Anche se la probabilità che una singola e-mail avesse l'effetto desiderato sul destinatario (cioè che cliccasse su di essa) era molto bassa, per gli spammer era economicamente molto redditizia grazie all'esiguo costo per ogni e-mail inviata. Hashcash propone di aggiungere un token all'intestazione di un'e-mail, il quale include una proof of work che dice che è stato speso un certo tempo di calcolo per generarla. La proof of work usata in Hashcash è l'inversione parziale dell'hash, utilizzando SHA-1 come funzione di hash (poi sostituita in Bitcoin da SHA256^2).

Figura 29 – Hashcash - Understanding Bitcoin – Pedro Franco

La Figura 29 presenta un esempio di intestazione Hashcash che include:

- La versione del protocollo Hashcash, che in Figura 29 è la 1.
- La dimensione della pre-immagine, 20 bit o 5 zeri iniziali in linguaggio esadecimale. La dimensione della pre-immagine indica il numero di bit di zero iniziali per l'inversione parziale dell'hash.
- La data di invio dell'e-mail, 131130 o 30 novembre 2013. Il token Hashcash è valido solo per un certo periodo.
- L'indirizzo e-mail del destinatario.
- Un nonce (cZtALTI6/ddddxYN6) utilizzato solo per una e-mail. La logica alla base di questo nonce è quella di evitare che la stessa intestazione di Hashcash venga utilizzata per inviare molte e-mail. Il server di posta elettronica ricevente può memorizzare i nonce in una cache e, se arriva un'e-mail il cui nonce viene trovato nella cache, l'e-mail viene eliminata. Questo evita che un botnet possa riutilizzare la stessa intestazione Hashcash in più e-mail.
- Il contatore di Hashcash (00...004ZsG). Hashcash incrementa questo contatore fino a trovare un valore tale che l'Hashcash dell'intera intestazione corrisponda al requisito di pre-immagine.

L'hash SHA-1 dell'intestazione dell'Hashcash è mostrato in fondo alla Figura 29. I primi 20 bit (5 caratteri esadecimali) di questo hash sono zeri e quindi l'intestazione è valida. La quantità di lavoro per risolvere il problema dell'hash parziale è esponenziale al numero di zeri iniziali, raddoppiando ad ogni zero iniziale. [17] [18]

[17] What is Proof of Work – Ledger.com - 2019

[18] Proof of Work system – Gerardus Blokdyk - 2018

2.13. Funzionamento della blockchain

Bitcoin combina le idee alla base del timestamp e della proof of work in modalità Hashcash per arrivare al risultato di rendere sicura la blockchain: questa è la principale innovazione introdotta da Bitcoin. Ogni blocco contiene un gruppo di nuove transazioni e un collegamento al blocco precedente della catena. Si noti che le vecchie transazioni sono ancora scritte in blockchain: i vecchi blocchi non vengono mai rimossi, quindi la blockchain può solo aumentare in lunghezza. Ogni blocco è certificato con una proof of work di inversione parziale dell'hash, come rappresentato nella Figura 30.

In primo luogo, ogni blocco include un gruppo di transazioni (valide), l'hash del blocco precedente e un nonce. Il nonce in un blocco risolve il problema dell'inversione parziale dell'hash, cioè è un numero tale per cui l'hash dell'intero blocco (incluso il nonce) inizi con un certo numero di zeri. Nella Figura 30 l'hash del blocco 271.076 è 0x0000000000000006e1163.. iniziando con 61 zero bit (o 15 caratteri zero in rappresentazione esadecimale). È facile regolare la difficoltà del blocco aumentando il numero di zeri di partenza. Il protocollo Bitcoin regola questa difficoltà per raggiungere la durata media di 10 minuti tra un blocco e l'altro. Questa regolazione della difficoltà fa parte delle regole di Bitcoin ed è codificata per ogni client. La difficoltà del blocco viene regolata ogni 2.016 blocchi minati o approssimativamente ogni 2 settimane. La regolazione tiene conto della variazione della potenza di calcolo dell'intera rete (hashing power) dall'ultima regolazione, confrontando i timestamp di due blocchi distanti 2.016 posizioni l'uno dall'altro. Quando la potenza di calcolo cresce (quindi più miners in circolazione o sviluppi hardware maggiori), i blocchi saranno estratti più velocemente rispetto ai 10 minuti previsti. La difficoltà quindi verrà regolata più in alto, seguendo tuttavia il livello dell'hashing power raggiunto.

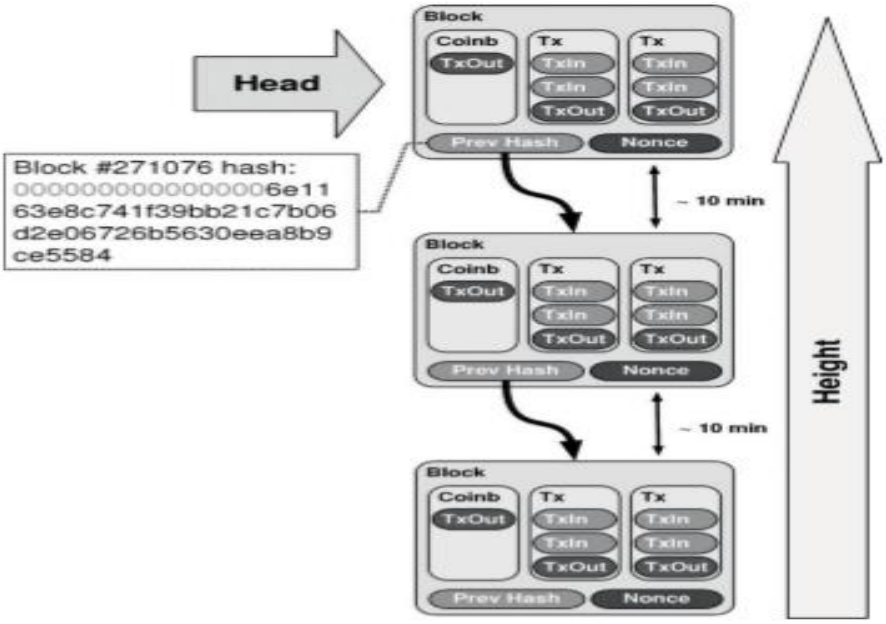


Figura 30 – Blockchain - Understanding Bitcoin – Pedro Franco

Se la correzione della difficoltà è superiore a 4 o inferiore a 1/4, viene bloccata a questi valori. Questo aiuta a proteggere da grandi oscillazioni nella difficoltà se la potenza della rete cambia troppo rapidamente. L'individuazione di una collisione parziale dell'hash (così viene definita la ricerca dell'hash da parte dei miners)

viene fatta attraverso un metodo di forza bruta (si va a tentativi casuali): provando con nonce randomici (o consecutivi) fino a quando uno di questi non crea la collisione parziale dell'hash.

Così il tempo necessario per risolvere un blocco è casuale.

La ricompensa per ogni blocco è la remunerazione che viene pagata ogni volta che un minatore risolve il problema dell'inversione parziale dell'hash. Così i nuovi bitcoin emessi vengono assegnati ai minatori che contribuiscono con la loro potenza di calcolo a certificare la blockchain.

Ogni blocco include una transazione speciale chiamata "Coinbase". Coinbase (nome anche di un famoso Exchange di criptovalute) è la prima transazione in un blocco. Essa ha un solo un input di una transazione (TxIn), che non è collegato a nessun precedente output di transazione (TxOut). Coinbase invece può avere molti output, la cui somma è uguale alla ricompensa del blocco, formata quindi dai bitcoin ottenuti per il mining più le commissioni associate al blocco.

I blocchi di solito includono molte transazioni oltre coinbase; tuttavia gli stessi possono essere considerati validi anche senza includere alcuna transazione eccetto coinbase, che invece è obbligatorio. In effetti, questo era molto comune all'inizio della rete Bitcoin, quando c'erano pochissime transazioni e questi blocchi "vuoti" aiutavano a formare la blockchain e premiavano i minatori con nuove emissioni di bitcoin.

I minatori possono scegliere quali transazioni convalidare nei blocchi che stanno minando e di solito le selezionano in base alla quantità di commissioni che pagano (decise da coloro che scambiano bitcoin). Il minatore che risolve il problema di calcolo usa coinbase come ricompensa per se stesso.

L'hash rate di un minatore è la sua potenza computazionale, misurata in hash/secondo e l'hash rate della rete è la somma dell'hash rate totale di tutti i minatori. L'estrazione di un blocco può essere paragonata ad una lotteria, dove le probabilità di un singolo minatore sono proporzionali alla sua quota di hash rate nella rete.

La ricompensa per il blocco è stata inizialmente fissata a 50 bitcoin per blocco. Ogni 210.000 blocchi, o all'incirca ogni 4 anni, la ricompensa viene dimezzata. La Figura 31 mostra il numero di bitcoin in circolazione assumendo che ogni blocco venga minato in 10 minuti di media.

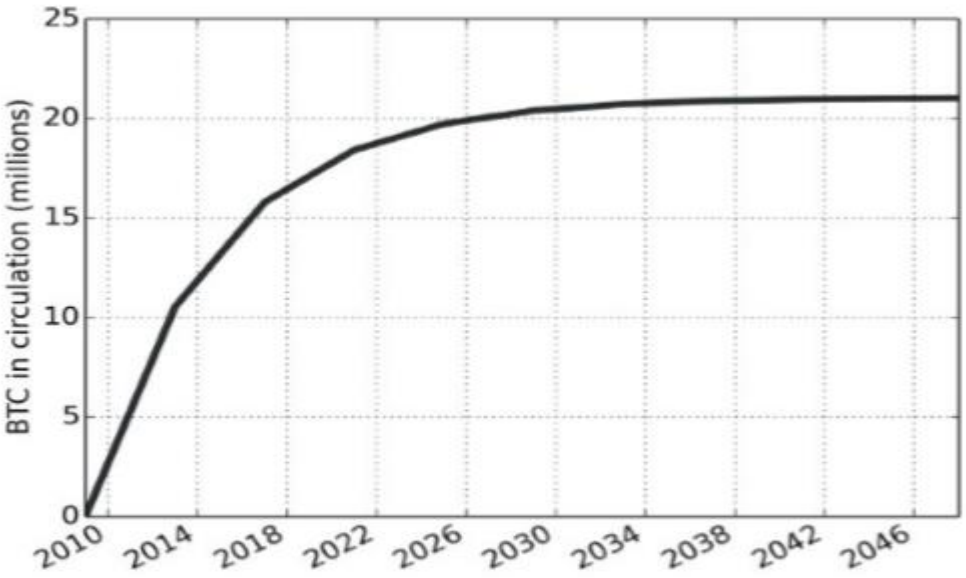


Figura 31 – Emissione Bitcoin - Understanding Bitcoin – Pedro Franco

Come previsto, il numero di bitcoin emessi diminuisce in modo esponenziale fino ad arrivare ad un totale massimo in circolazione di circa 21 milioni. Questo algoritmo di generazione della criptovaluta è considerato immutabile dalla comunità.

Storicamente, circa il 99% del compenso dei minatori proviene dalla ricompensa del blocco, e solo l'1% dalle commissioni di transazione; si prevede tuttavia che nel tempo una percentuale maggiore della retribuzione sarà dovuta alle commissioni dei soggetti coinvolti nello scambio.

Il processo di risoluzione dei blocchi è chiamato mining (estrazione mineraria) in analogia con l'estrazione dei metalli preziosi. Questa analogia, sebbene utile, può portare ad un fraintendimento: la ricompensa del blocco è fissata dal protocollo e non è influenzata dal numero di miners in circolazione. Contrariamente all'estrazione dell'oro, un aumento dei minatori non aumenta il numero di bitcoin in circolazione. Un ingresso di nuovi miners invece incrementa l'hashing power, riducendo il numero di minatori originali, ma mantenendo costante la ricompensa totale per ogni blocco.

Il blocco che ne precede un altro viene chiamato parent block (blocco padre). I blocchi fanno riferimento al loro parent block nella blockchain includendo il suo hash nella propria struttura; ciò consente alla blockchain di mantenerli in ordine cronologico. Il primo blocco della blockchain è chiamato genesis block (blocco genesi) ed è stato creato da Satoshi il 3 gennaio 2009. L'ordine di un blocco nella blockchain, a partire dal blocco genesi, è chiamato block height (altezza del blocco). L'ultimo blocco aggiunto è chiamato blockchain head (blocco di testa) e i nuovi blocchi vengono aggiunti ogni volta sopra di esso.

Un fork si verifica quando due miners provano a minare un nuovo blocco più o meno nello stesso momento. Si immagini ad esempio di aggiungere degli anelli a una catena; se qualcuno arriva e sceglie un anello al centro della catena, iniziando ad aggiungere i propri anelli incurante degli altri, la catena sta subendo un fork. Entrambi i blocchi risolvono il problema dell'inversione parziale dell'hash, ma solo uno di essi può far parte della blockchain.

La blockchain può essere forkata in modo simile a quanto descritto e, in realtà, è stata forkata per esempio con un aggiornamento difettoso nel 2013, ma la reazione repentina degli sviluppatori di Bitcoin ha dimostrato che è possibile riconoscere e isolare rapidamente una fork fraudolento. Perché qualcuno vorrebbe forkare fraudolentemente una blockchain?

Potrebbe farlo perché vuole modificare in modo per lui conveniente un'informazione che esiste già sulla catena, quindi forkarla appena prima del blocco che contiene quell'informazione. Questo può diventare una tattica ricorrente per chiunque voglia modificare fraudolentemente un contratto, un atto o un titolo, se i tentativi di forkare la catena non vengono rilevati e bloccati velocemente. La finalità alla base del fork è quella di prendere il controllo del sistema in toto; la versione della catena, che ha più potenza di elaborazione e può quindi creare la blockchain più lunga, è considerata quella valida. Bitcoin stabilisce regolarmente dei record mondiali per la quantità di potenza computazionale che gestisce, quindi la maggior parte delle persone considera che il costo di un attacco a Bitcoin sia maggiore del risultato dello stesso.

Un esperto di blockchain può isolare un nodo che non funziona correttamente o che mostra segni di manomissione e indagare sull'eventuale problema.

Magari il nodo sta scrivendo informazioni false e/o senza senso. Magari non sta affatto trasmettendo nuove informazioni. Magari ha creato la sua nuova versione della blockchain.
Questo è in realtà uno dei vantaggi dell'avere un sistema decentralizzato in grado di gestire più nodi in diverse posizioni.
È facile isolare un nodo "malvagio" per la risoluzione dei problemi, senza compromettere il resto del sistema. I nodi che si comportano in modo corretto possono continuare a lavorare, creando nuove registrazioni valide e rendendole disponibili a chiunque voglia ispezionarle o utilizzarle, mentre l'esperto di blockchain indaga su cosa sia andato storto con il nodo che si è comportato in modo scorretto. Poi, una volta chiarita la questione, il fork può essere risolto e il nodo riparato può essere nuovamente aggiunto alla rete.
Il blocco scartato è chiamato orphan block (blocco orfano). La decisione su quale sia il ramo della blockchain valido non viene presa da nessuna istituzione centralizzata ma la disputa si risolve da sola in modo normale. Un fork infatti può persistere per diversi blocchi, come mostra la Figura 32 e, quando questo accade, c'è una spaccatura nella rete, per cui alcuni minatori credono che un ramo sia la blockchain legittima, mentre gli altri seguono l'altro. Il protocollo allora determina che la blockchain corretta sia la più lunga; così i minatori hanno un incentivo per smettere di lavorare in un ramo non appena sarà chiaro che si arriverà al blocco orfano, perché il lavoro su di esso sarebbe sprecato. Pertanto, i fork si risolvono rapidamente, di solito in un solo blocco.
Il numero medio di fork è stato di circa il 2%, cioè in media ogni 50 blocchi c'è un fork nella blockchain. Il protocollo quindi evita di avere un'autorità centrale che decida quale sia il ramo corretto, in linea con la filosofia di decentralizzazione di Bitcoin.

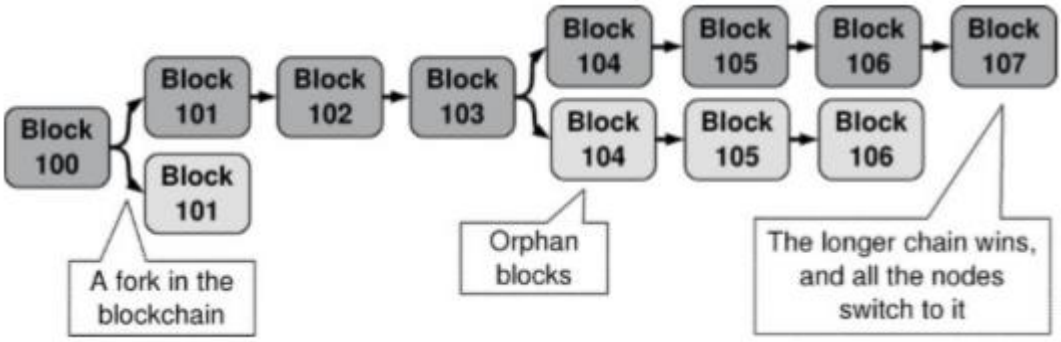


Figura 32 – Dinamica della blockchain e fork - Understanding Bitcoin – Pedro Franco

Le transazioni incluse nei blocchi di un fork non vanno perse: quando un fork viene risolto e un ramo della blockchain viene scartato, le transazioni di quel ramo vengono introdotte nuovamente nell'insieme di memoria delle transazioni non confermate, pronte per essere incluse nel blocco successivo estratto. Alcune di queste potrebbero già apparire in un blocco del ramo legittimo del fork e, in questo caso, tali transazioni verranno scartate ed escluse dal pool di memoria delle transazioni non confermate. Ogni risoluzione del fork produce vincitori (i minatori che hanno risolto i blocchi nel ramo accettato) e perdenti (i minatori i cui blocchi sono orfani).

Sono possibili all'interno del protocollo blockchain due tipi differenti di fork: l'hard fork e il soft fork.

L'hard fork è uno scenario in cui la rete cambia le regole di consenso: si chiama hard fork, perché dopo il fork, la rete non si ricongiunge su una singola catena, ma al contrario, le due catene evolveranno in modo indipendente. Gli hard fork si verificano quando una parte della rete opera secondo una serie di regole di consenso diverse rispetto al resto della rete: può avvenire a causa di un bug o di un cambiamento deliberato nell'implementazione delle regole di partecipazione.

Gli hard fork possono essere usati per cambiare le regole del consenso, ma richiedono un coordinamento tra tutti i partecipanti al sistema. I nodi che non passano alle nuove regole non possono partecipare al meccanismo di consenso e sono spinti in un'altra blockchain al momento dell'hard fork.

Si esamini la meccanica di un hard fork con un esempio specifico.

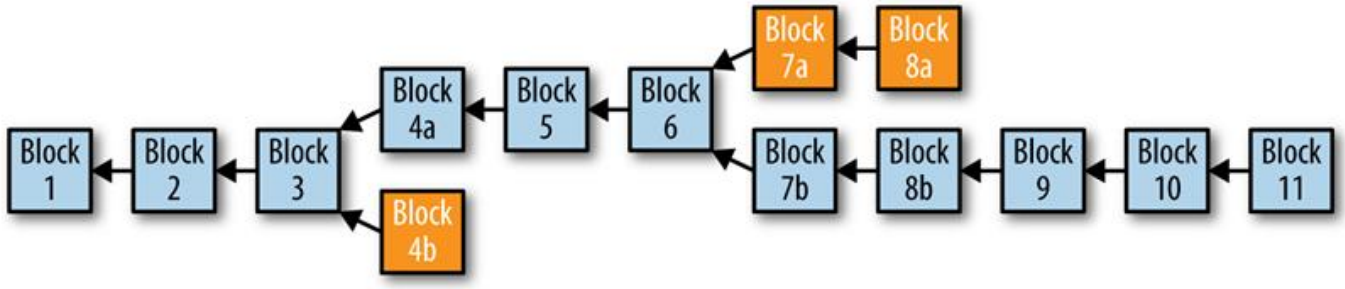


Figura 33 – Hard fork – Mastering Bitcoin - Antonopoulos

La Figura 33 mostra una blockchain con due biforcazioni. All'altezza del blocco 4, si verifica un fork. Con l'estrazione del blocco 5, la rete ritorna su una sola catena e il fork viene risolto.

Più tardi, però, all'altezza del blocco 6, si verifica un hard fork.

Si supponga che venga rilasciata una nuova implementazione della blockchain con un cambiamento delle regole di consenso. A partire dal blocco 7, i minatori che gestiscono questa nuova implementazione accetteranno un nuovo tipo di firma digitale, chiamiamola firma "X". Subito dopo, un nodo che esegue la nuova implementazione crea una transazione che contiene una firma "X" e un miner, con il software aggiornato, mina il blocco 7b contenente questa transazione.

Qualsiasi nodo o minatore che non abbia aggiornato il software per convalidare le firme X, non è ora in grado di elaborare il blocco 7b. Dal loro punto di vista, sia la transazione che conteneva una firma X, che il blocco 7b, contenente tale transazione, non sono valide, perché stanno valutando le transazioni sulla base delle vecchie regole di consenso. Questi nodi rifiuteranno la transazione e non propagheranno i blocchi; quindi i minatori che utilizzano le vecchie regole non accetteranno il blocco 7b e continueranno a estrarre un blocco il cui genitore è il blocco 6.

Infatti, i minatori che usano le vecchie regole possono anche non ricevere il blocco 7b se tutti i nodi a cui sono collegati obbediscono alle vecchie regole e quindi non trasmettono il blocco. Alla fine, saranno in grado di estrarre il blocco 7a, che è valido secondo le vecchie regole e non contiene alcuna transazione con firma X.

Le due blockchain continueranno per sempre a divergere. I minatori della catena "7b" continueranno ad accettare e a estrarre transazioni contenenti firme X, mentre i minatori della catena “7a" continueranno ad ignorare queste transazioni. Anche se il blocco 8b non contiene alcuna transazione con firma X, i minatori

della catena "a" non possono comunque processarla. A loro sembra essere un orphan block, in quanto il suo genitore "7b" non viene riconosciuto come blocco valido.

Per gli sviluppatori il termine “fork” ha un significato leggermente diverso. Nei software open source, un fork si verifica quando un gruppo di sviluppatori sceglie di seguire una diversa modalità di sviluppo del software e di avviare un'implementazione concorrente a quel progetto open source.
Si sono già discusse due circostanze che porteranno a un hard fork: un bug nelle regole del consenso e una modifica deliberata delle regole.
Nel caso di una modifica deliberata delle regole del consenso, un fork del software precede l'hard fork. Tuttavia, affinché questo tipo di hard fork si verifichi, è necessario prima sviluppare, adottare e lanciare una nuova implementazione software delle regole.
Esempi di software che hanno tentato di cambiare le regole di consenso sono Bitcoin XT, Bitcoin Classic, e più recentemente Bitcoin Unlimited. Tuttavia, nessuno di questi, ha dato luogo ad un hard fork.
Anche se un software fork è un prerequisito necessario, non è di per sé sufficiente affinché si verifichi un hard fork. Affinché quest’ultimo si verifichi, è necessario adottare un'implementazione concorrente e attivare nuove regole, da parte dei minatori, dei portafogli e dei nodi intermedi.
Al contrario, ci sono numerose implementazioni alternative di Bitcoin, e anche software fork, che non cambiano le regole di consenso e che, salvo un bug, possono coesistere sulla rete ed operare insieme senza causare un hard fork.
Le regole di consenso possono differire in modi ovvi ed espliciti, come nella validazione delle transazioni o dei blocchi, o in modi più sottili, come negli script bitcoin o nelle firme digitali, o anche in modi imprevisti, a causa di vincoli di consenso imposti da limitazioni del sistema o dettagli di implementazione.
Un esempio di quest'ultima situazione è rappresentata dall'imprevisto hard fork durante l'aggiornamento di Bitcoin Core dalla versione 0.7 alla 0.8, che è stato causato da una limitazione nell'implementazione del Berkley DB (un sistema per la creazione di database per software open source come Bitcoin) utilizzato per memorizzare i blocchi.
Concettualmente, si può pensare ad un hard fork come ad uno sviluppo in quattro fasi: un software fork, un network fork, un mining fork e un chain fork.
Il processo inizia quando un'implementazione alternativa, con regole di consenso modificate, viene creata dagli sviluppatori.
Quando questa implementazione viene lanciata nella rete, una certa percentuale di minatori, utenti del portafoglio e nodi intermedi possono adottare ed eseguire l’'"aggiornamento".
Il fork risultante dipenderà dal fatto che le nuove regole di consenso si applichino o meno ai blocchi, alle transazioni o a qualche altro aspetto del sistema. Se le nuove regole di consenso riguardano le transazioni, allora un portafoglio che crea una transazione secondo le nuove regole può modificare il network, seguito da un hard fork, quando la transazione viene estratta in un blocco. Se le nuove regole riguardano i blocchi, allora il processo di hard fork inizierà quando un blocco sarà estratto secondo i nuovi princìpi.

In primo luogo, la rete farà un fork. I nodi basati sull'implementazione originale rifiuteranno tutte le transazioni e i blocchi creati secondo le nuove regole, e si scollegheranno da tutti i nodi che invieranno loro queste transazioni e blocchi non validi.
Di conseguenza, la rete si dividerà in due: i vecchi nodi rimarranno connessi solo ai vecchi nodi e i nuovi nodi solo ai nuovi. Una singola transazione o blocco basati sulle nuove regole si diramerà attraverso la rete e porterà alla partizione del network.
Una volta che un minatore che utilizza le nuove regole estrae un blocco, anche la potenza di mining si diramerà. I nuovi minatori estrarranno sopra il nuovo blocco, mentre i vecchi minatori estrarranno una catena separata basata sulle vecchie regole.
La rete partizionata farà in modo che i minatori che operano secondo regole di consenso separate non ricevano i blocchi l'uno dell'altro, poiché ormai sono collegati a due reti differenti.
Quando i minatori si separano in due diverse blockchain a causa di un hard fork, l'hashing power della rete viene diviso in qualsiasi proporzione.
Si supponga, per esempio, una divisione dell'80%-20%, con la maggioranza dell'hashing power che utilizza le nuove regole di consenso. Si ipotizzi anche che il fork si verifichi immediatamente dopo un periodo di retargeting dell'hash rate, quindi che esso sia esattamente uguale a quello del periodo precedente.
Le nuove regole di consenso impiegano l'80% della potenza di mining precedentemente disponibile, quindi essa è improvvisamente diminuita del 20% rispetto al periodo precedente. Si troveranno blocchi in media ogni 12 minuti (sempre prendendo come riferimento Bitcoin), che rappresenta un calo del 20% del mining rispetto a prima. Questo tasso di emissione di blocchi continuerà per circa 24.192 minuti (a 12 minuti per blocco), o 16,8 giorni. Dopo 16,8 giorni, si verificherà un retarget e la difficoltà con cui si minano i blocchi si aggiusterà, in modo da produrne nuovamente ogni 10 minuti, sulla base della ridotta quantità di hashing power di questa blockchain.
La catena di minoranza, che estrae secondo le vecchie regole, ora con solo il 20% di hashing power, dovrà affrontare un compito molto più difficile. Su questa catena, i blocchi saranno ora estratti in media ogni 50 minuti. La difficoltà non sarà regolata per 100.800 minuti ovvero circa 10 settimane.

Non tutti i cambiamenti delle regole del consenso, ma solo i cambiamenti incompatibili con il sistema precedente, provocano un hard fork.
Se la modifica viene attuata in modo tale che un nodo che non aggiorna il sistema, veda ancora la transazione o il blocco come valido secondo le regole precedenti, la modifica è stata attuata senza un fork.
Il termine soft fork è stato introdotto per distinguere questo metodo di aggiornamento da un "hard fork".
In pratica, un soft fork non è affatto un fork. Un soft fork è un cambiamento compatibile con le regole di consenso che permette ai clienti non aggiornati di continuare ad operare in accordo con le nuove regole.
Un aspetto del soft fork è che gli aggiornamenti possono essere usati solo per limitare le regole del consenso, non per espanderle. Per essere compatibile, le transazioni e i blocchi creati secondo le nuove regole devono

essere validi anche secondo le vecchie regole, ma non viceversa. Le nuove regole possono solo limitare ciò che è valido; altrimenti, se respinte secondo le vecchie regole, daranno luogo ad un hard fork.

I soft fork possono essere implementati in diversi modi, infatti il soft fork non definisce un singolo metodo, ma piuttosto un insieme di metodi che hanno tutti in comune il non richiedere l'aggiornamento di tutti i nodi o l'espulsione dei nodi non aggiornati al di fuori dal network.

Per esempio, Segregated Witness (Segwit) era un cambiamento della struttura di uno scambio, che spostava lo script di sblocco (testimone) dall'interno della transazione ad una struttura dati esterna (segregazione). Segwit è stato inizialmente concepito come un aggiornamento hard fork, in quanto modificava una struttura fondamentale (transazione). Nel novembre 2015, uno sviluppatore che lavorava su Bitcoin Core ha proposto un meccanismo attraverso il quale Segwit poteva essere introdotto come soft fork. Il meccanismo utilizzato era una modifica dello script di blocco di UTXO creato secondo le regole di Segwit, in modo tale che i nodi non aggiornati vedevano lo script di blocco come recuperabile con qualsiasi script di sblocco. Di conseguenza, Segwit poteva essere introdotto senza che ogni nodo dovesse essere aggiornato o separato dalla blockchain: un soft fork.

Le critiche più comuni che vengono rivolte ai soft fork sono:

- Debito tecnico: poiché i soft fork sono tecnicamente più complessi degli hard fork, essi introducono quello che viene chiamato debito tecnico, ovvero un aumento del costo futuro della manutenzione del codice a causa dei tradeoff di progettazione fatti in passato. La complessità del codice causata da soft fork, a sua volta, aumenta la probabilità di bug e vulnerabilità di sicurezza.
- Validation relaxation: i nodi che non hanno aggiornato considerano valide le transazioni, senza valutare le regole di consenso modificate. In effetti, questi nodi convalidano non utilizzando l'intera gamma di regole di consenso, in quanto non vedono le nuove regole.
- Aggiornamenti irreversibili: poiché i soft fork creano transazioni con ulteriori vincoli di consenso, in pratica diventano aggiornamenti irreversibili. Se un upgrade soft fork dovesse essere riportato indietro dopo essere stato attivato, qualsiasi transazione creata secondo le nuove regole potrebbe comportare una perdita di fondi secondo le vecchie regole. Pertanto, i critici sostengono che un soft fork fallito che dovesse essere riportato alla condizione pre-aggiornamento a causa di un bug, porterebbe quasi certamente a una perdita di fondi.

Lo sviluppo del concetto di consenso e della modifica delle regole fa discutere molto e continua ad evolversi. Per sua stessa natura, Bitcoin pone un limite molto alto al coordinamento e al consenso per i cambiamenti.

In quanto sistema decentralizzato, non ha una "autorità" che possa imporre la sua volontà ai partecipanti della rete e il potere è diffuso tra più "attori" come i minatori, gli sviluppatori del Bitcoin Core, gli sviluppatori di wallet, gli exchange, i traders e gli utenti finali. Le decisioni non possono essere prese unilateralmente da nessuna di queste parti.

Per esempio, i minatori possono teoricamente cambiare le regole con una maggioranza semplice (51%), ma rimangono comunque vincolati dal pensiero delle altre categorie che operano nel progetto Bitcoin. Se agiscono unilateralmente, il resto dei partecipanti può semplicemente rifiutarsi di seguire le nuove regole da loro

imposte, mantenendo l'attività economica su una blockchain di minoranza. Senza attività economica (traders, portafogli, scambi), i minatori estrarranno una moneta senza valore con blocchi vuoti.
Questa diffusione del potere significa che tutti i partecipanti devono coordinarsi oppure non sarà possibile effettuare alcun cambiamento. Lo status quo di questo sistema è caratterizzato da poche modifiche possibili laddove ci sia il consenso da parte di una maggioranza molto ampia. La soglia del 95% per i soft fork riflette questa realtà.
È importante riconoscere che non esiste una soluzione perfetta per lo sviluppo del consenso. Sia gli hard fork che i soft fork comportano dei tradeoff. Per alcuni tipi di cambiamenti, i soft fork sono una scelta migliore, mentre per altri, lo sono gli hard fork. Non esiste una scelta perfetta: entrambi comportano dei rischi.
L'unica caratteristica costante dello sviluppo del consenso è che il cambiamento è difficile e per raggiungerlo spesso si deve arrivare al compromesso.
Alcuni vedono questo come un punto debole, altri come la più grande forza del sistema. [19] [20]

La rete Bitcoin è composta da nodi, ovvero computer collegati ad Internet, che eseguono e mantengono il software Bitcoin e una copia completa della blockchain.
Bitcoin è una rete peer-to-peer: tutti i nodi sono omogenei. I nodi ricevono transazioni e blocchi da altri nodi e li ritrasmettono alla rete attraverso un processo continuo.
Una transazione appena creata, ove non sia stata inclusa in nessun blocco, viene chiamata transazione non confermata. Una volta che una transazione viene inclusa, si dice che è stata confermata.
I nodi memorizzano le transazioni ricevute ma non confermate in un database chiamato "pool di memoria delle transazioni non confermate" (unconfirmed transactions' memory pool), spesso chiamato semplicemente “Mempool”. Non tutte le transazioni ricevute vengono aggiunte al mempool: se una transazione spende due volte un input di una transazione già presente, oppure non è standard, tale transazione verrà eliminata.
Una volta che un nodo riceve o estrae un nuovo blocco, il mempool viene aggiornato.
Quando una transazione viene creata, essa viene ritrasmessa alla rete Bitcoin attraverso una piccola quantità di nodi, che ricevono la nuova transazione, controllano che non sia già presente nel mempool e infine che sia valida. Se la transazione passa il controllo, viene ritrasmessa ad altri nodi della rete, altrimenti viene lasciata perdere.

[19] Mastering Bitcoin – Andreas M. Antonopoulos - 2014

[20] Cryptocurrencies simply explained – Julian Hosp - 2017

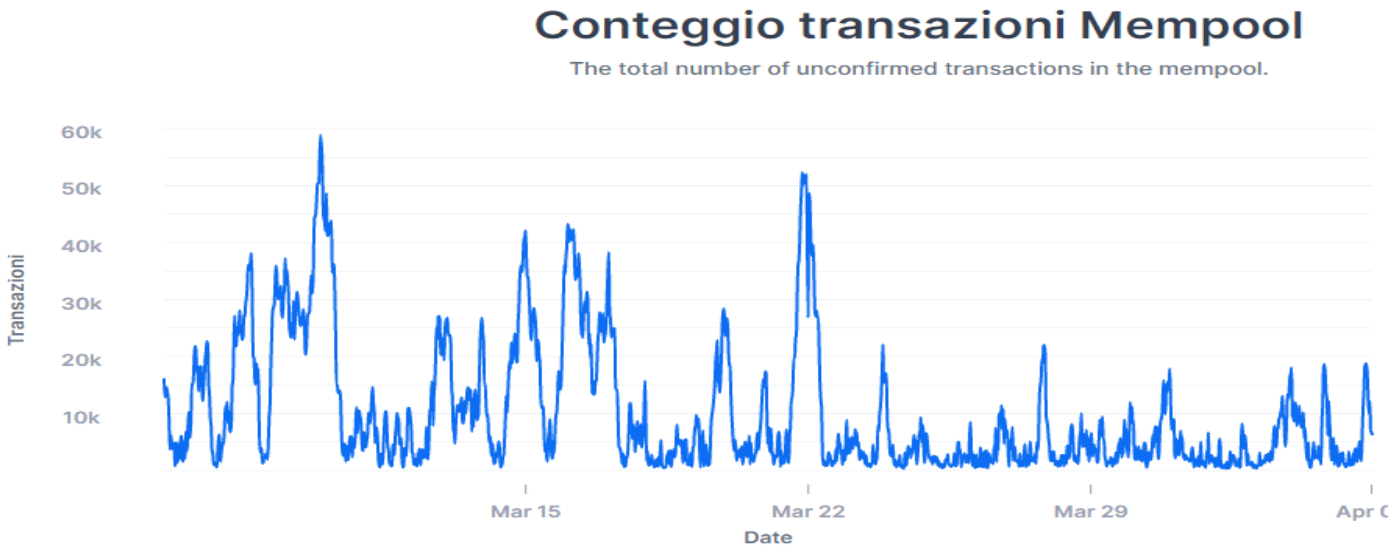


Figura 34 – Conteggio transazioni Mempool – Blockchain.info

In sintesi, oltre ad una copia completa della blockchain, un nodo mantiene anche strutture di dati aggiuntive, come la cache delle transazioni non spese o il pool di memoria delle transazioni non confermate, in modo da poter validare rapidamente le nuove transazioni ricevute e minare blocchi. Se la transazione o il blocco ricevuto è valido, il nodo aggiorna le sue strutture di dati e le ritrasmette ai nodi collegati.

È importante notare che un nodo non ha bisogno di fidarsi degli altri nodi, perché convalida in modo indipendente tutte le informazioni che riceve.

Quando un minatore trova un nuovo blocco, lo trasmette alla rete. Tutti i nodi riceventi verificano come prima cosa la validità del blocco, cioè che esso risolva il problema dell'inversione parziale dell'hash con la difficoltà richiesta. Successivamente aggiornano le loro strutture dati interne per inserire le nuove informazioni contenute:

- Aggiornano la cache degli output delle transazioni non spesi (UTXO)
- Aggiornano il pool di memoria delle transazioni non confermate. Ciò comporta l'esame dell'elenco delle transazioni e l'eliminazione di quelle che sono in conflitto, ovvero quelle che spendono gli stessi output nel nuovo blocco estratto

I nodi mantengono una serie di connessioni con altri nodi. Alcuni vorranno mantenere tante connessioni aperte quante lo permettono le risorse disponibili (CPU, larghezza di banda della rete). Per esempio, un nodo con un portafoglio Bitcoin potrebbe voler mantenere le connessioni con molti altri nodi, distribuiti il più possibile geograficamente, per rilevare e agire rapidamente in caso di double spending.

Allo stesso modo, un nodo che fa mining potrebbe volere il maggior numero possibile di connessioni aperte, in modo da ricevere una notifica tempestiva sui nuovi blocchi estratti. Una ricezione più rapida dei nuovi blocchi riduce al minimo il tempo sprecato nel tentativo di estrarre un blocco che diventerà orfano.

Per gli altri nodi, avere informazioni aggiornate non è così importante e quindi di solito si collegano solo ad un ristretto gruppo di nodi.

La propagazione di un blocco è rallentata sia dai soliti ritardi di rete, sia dal fatto che ogni nodo controlla la piena validità di un blocco prima di ritrasmetterlo. Molti ricercatori hanno scoperto che i nuovi blocchi impiegano circa 10 secondi per propagarsi attraverso la rete. Il ritardo di propagazione è proporzionale alle

dimensioni del blocco, quindi i blocchi di grandi dimensioni possono richiedere un tempo maggiore per diffondersi e la maggior parte dei fork della blockchain è causata proprio dal ritardo di propagazione dei blocchi.

Esistono due tipi di nodi: i nodi che fanno mining (detti anche nodi minerari) e i nodi passivi. I nodi minerari sono nodi che cercano attivamente di risolvere i blocchi, al fine di raccoglierne i guadagni annessi e quindi includono gli scambi, dal loro pool di memoria delle transazioni non confermate, nel blocco che stanno minando. I nodi passivi sono solitamente impiegati come portafogli Bitcoin, gestione di pagamenti, fornitori di dati di mercato e così via. Tutti i nodi mantengono un registro istantaneo aggiornato dello stato della rete, compresa una copia completa della blockchain, della cache UTXO e del mempool delle transazioni non confermate.

A titolo di esempio, i nodi che forniscono servizi di portafoglio svolgono questi compiti per conto dei loro clienti:

- Trasmettere alla rete le transazioni dei clienti
- Tenere traccia dello stato di una transazione. Cioè, notificare al cliente quando una transazione è stata inclusa in un blocco e quando sono stati aggiunti nuovi blocchi in cima a quello in cui la transazione è stata inclusa
- Inviare al cliente gli output della transazione non spesi appartenenti a determinati indirizzi. Ciò consente al cliente di tenere traccia dei fondi disponibili

I nodi che fanno mining non possono modificare le transazioni; possono solo decidere se includerle o meno nel blocco che stanno attualmente minando e se ritrasmetterle o meno al resto della rete. Il contenuto delle transazioni è protetto dalla crittografia a chiave pubblica e non può essere modificato dal nodo che elabora l'operazione.

Può sembrare che i nodi abbiano un incentivo a non trasmettere una transazione, soprattutto se assegni un compenso considerevole al minatore. Ci sono, tuttavia, vari fattori contro la non trasmissione delle transazioni:

- I clienti di solito si collegano a vari nodi quando trasmettono una nuova transazione, perché è nel loro interesse che la transazione sia ritrasmessa, in modo che sia inclusa nel blocco successivo estratto dalla rete. Un cliente può controllare se i nodi che sta usando stanno ritrasmettendo le sue transazioni e smettere di fare affidamento su quelli che non lo fanno. Questo è un risultato indesiderato per un nodo, che preferisce vedere le transazioni non appena vengono create.
- I nodi si collegano tra loro e possono controllare se qualcuno si sta astenendo dal trasmettere le transazioni. Se un "truffatore" viene scoperto, altri nodi possono punirlo, ad esempio con una strategia "tit-for-tat" (metodo con cui si risolve anche il dilemma del prigioniero) interrompendo con lui la trasmissione delle informazioni.
- Un nodo che non ritrasmette le transazioni può essere arruolato da un nodo malevolo per aiutare ad eseguire un attacco di double spending, alimentando una transazione che è stata già spesa da qualche altra parte nella rete. Questo può essere molto dannoso per il nodo, perché potrebbe significare perdere la ricompensa del blocco quando il nodo maligno raggiunge il suo obiettivo. Dato che le commissioni

di transazione rappresentano una piccola percentuale del compenso totale di un blocco, il mantenimento della soluzione di una soluzione di un blocco non vale di solito questo rischio.

Al contrario, i nodi che fanno mining hanno un incentivo a trasmettere i blocchi minati il più velocemente possibile per ridurre la probabilità che qualcun altro possa minare il blocco. [21]

[21] Bitcoin – Kerry Gan - 2019

2.14. Merkle Trees

Fino a questo punto, gli hash dei blocchi sono stati assunti come l'hash dell'intero blocco, che include sia quello del blocco precedente, sia tutte le transazioni. Ma si veda come si costruisce questo hash.
Una soluzione difficile che consiste nel concatenare il block header (documento di identità del blocco) con, a ruota, tutte le transazioni e fare alla fine l'hash dell'intera stringa di byte, ha diversi svantaggi:

- Se una transazione viene cambiata, la stringa di byte deve essere aggiornata e l'hash deve essere ricalcolato di nuovo. Questo costringe i nodi a tenere in memoria l'intera stringa di byte del blocco. Se una transazione al centro della stringa viene sostituita da una transazione più grande, la memoria deve essere allocata al centro della stringa di byte, il che rappresenta un'operazione costosa.
- Per verificare se una transazione appartenga ad un blocco, l'intero blocco deve essere disponibile. Solo allora l'hash può essere calcolato e verificato.

L'albero di Merkle (Merkle tree), proposto da Ralph Merkle nel 1982, è una struttura di dati che affronta questi problemi; Satoshi Nakamoto ha deciso di utilizzare tale sistema per timbrare un blocco di transazioni. Gli alberi di Merkle sono anche ampiamente utilizzati nelle applicazioni di file sharing, per tenere traccia dei blocchi di un file che sono stati correttamente scaricati. La Figura 35 mostra come un albero di Merkle viene usato per calcolare l'hash di un blocco. In primo luogo, si forma un albero binario con gli hash delle singole transazioni come "foglie". Questi sono indicati da Hash0, ... Hash3 nella Figura 35. Un albero binario è un grafico dove ogni genitore ha due figli. L'hash del nodo genitore è l'hash dei suoi due figli, a loro volta hash. Infatti, Hash01, che è l'hash genitore è = Hash(Hash0 || Hash1) e l'Hash è una funzione hash SHA256. Alla fine, si arriva a calcolare la radice di Merkle (Merkle root), con la quale si può definire l'intestazione del blocco. Il block header contiene l'hash del blocco precedente nella blockchain, la radice del Merkle tree e il nonce incluso dal minatore.

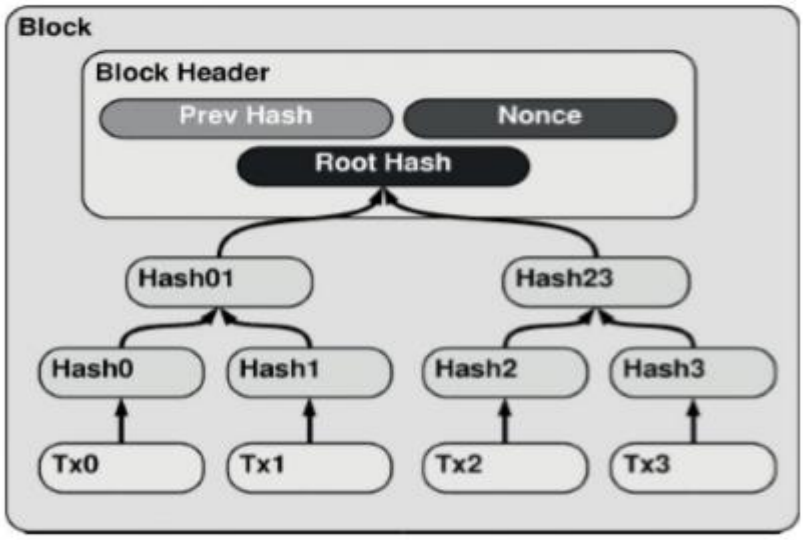


Figura 35 – Merkle tree - Understanding Bitcoin – Pedro Franco

L'hash del blocco è quindi solo l'hash dell'intestazione: le transazioni sono rappresentate attraverso la radice dell'albero di Merkle. Uno dei grandi vantaggi dell'utilizzo degli alberi di Merkle è la verifica delle transazioni.

Si supponga che un nodo voglia verificare che una transazione, per esempio Tx3, appartenga ad un blocco come nella Figura 36. Un nodo può eseguire questa operazione entro un tempo che è logaritmicamente correlato al numero di nodi dell'albero. Seguendo la Figura 36, il nodo deve solo calcolare Hash3, Hash23 e l'hash della radice e controllare il risultato rispetto all'hash della radice memorizzato nel blocco. Cioè, il nodo deve solo verificare il ramo di Merkle (Merkle branch), ovvero la parte dell'albero che permette di provare crittograficamente che una transazione è inclusa nell'albero. [22]

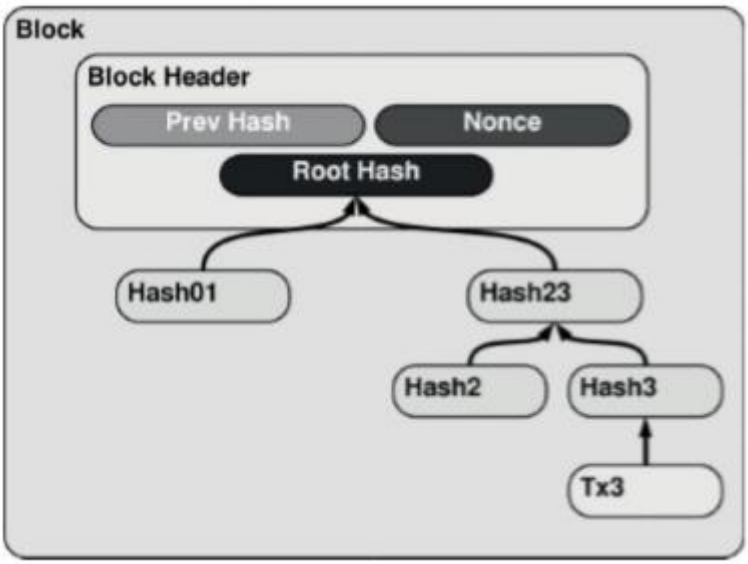


Figura 36 – Dinamica di un Merkle Tree - Understanding Bitcoin – Pedro Franco

[22] Merkle Root (Cryptocurrency) – Investopedia.com - 2020

2.15. Mining

Il mining è il processo di aggiunta di blocchi alla blockchain. I minatori (miners) contribuiscono con la loro potenza di calcolo a far crescere la blockchain risolvendo il problema dell'inversione parziale dell'hash e Bitcoin li remunera con le monete digitali più le tasse raccolte da tutte le transazioni incluse nel blocco.
Per trovare una soluzione, il software minerario di solito incrementa il nonce del blocco ed esegue l'algoritmo di proof of work per verificare se il nonce scelto generi un hash corretto (cioè un hash del blocco che soddisfa i requisiti di difficoltà). Una tipica ottimizzazione utilizzata dai minatori è quella di precalcolare l'hash della parte iniziale dell'intestazione del blocco che contiene l'hash del blocco precedente e la Merkle root.
Uno dei vantaggi del meccanismo di mining è che premia chi arriva prima: questo è stato molto importante soprattutto all'inizio, quando il Bitcoin si è messo in evidenza nei mercati finanziari, poichè non aveva una società alle spalle che lo potesse supportare. Per questo motivo, il marketing e la diffusione di un nuovo metodo di scambio digitale doveva essere fatto in maniera virale. Premiare i minatori pertanto era ed è un modo per farli entrare nel protocollo e creare il passaparola.
L'attività mineraria è simile ad un mercato in concorrenza perfetta: finché ci sarà un profitto da realizzare, i nuovi arrivati entreranno nel mercato e vi rimarranno fino a quando l'opportunità di guadagno non sarà esaurita.
La difficoltà di estrazione aumenta con l'ingresso di un maggior numero di minatori nella rete, ma la remunerazione totale del blocco rimane invariata. Alla creazione di Bitcoin, la ricompensa per ogni blocco era di 50 bitcoin. Questa ricompensa viene dimezzata ogni 210.000 blocchi, o all'incirca ogni 4 anni, per rispettare il ritmo di creazione del denaro stabilito da Nakamoto.
Si noti che l'emissione di nuovi bitcoin non è una progressione regolare, in quanto l'introduzione di nuovo hashing power fa aumentare temporaneamente il tasso di creazione di nuovi blocchi fino a quando il meccanismo di feedback e ricalcolo della potenza della rete non ha nuovamente luogo. Quindi, con un tasso di hash crescente, l'emissione di nuovi bitcoin accelera.
Il 28 novembre 2012 per esempio si è verificato un halving day (giorno in cui si dimezza la remunerazione di bitcoin per un blocco) più di un mese prima del previsto (la ricompensa per ogni blocco è stata dimezzata a 25 bitcoin).

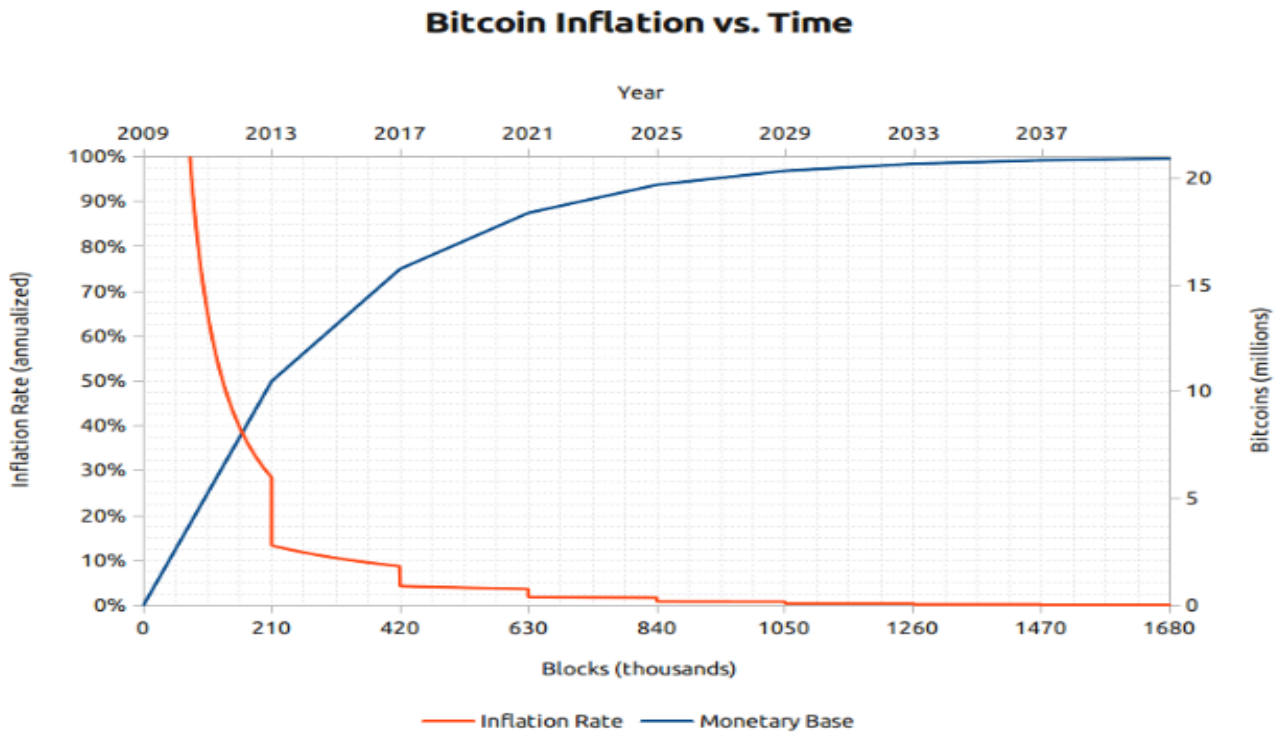


Figura 37 – Emissione di Bitcoin vs Inflazione - Kaskaloglu

Bitcoin è una rete peer-to-peer: chiunque può connettersi e iniziare a minare. I nuovi entranti non devono chiedere il permesso o aderire a una serie di regole prima di entrare nel sistema, né gli operatori storici possono colludere per impedire l'ingresso ai nuovi partecipanti. Così i nuovi investitori nel mining, se così si possono definire, parteciperanno alla ricerca del codice per ottenere la ricompensa del blocco.

In uno scenario di aumento del prezzo del bitcoin (o di aumento del progresso tecnologico), i minatori devono continuare ad aumentare il loro hash rate per ottenere la stessa ricompensa; questo processo continuerà fino a quando il costo marginale dell'ultimo minatore che entra sarà pari alla ricompensa prevista: a questo punto la rete ha raggiunto l'equilibrio, che può essere perturbato solo da qualche fattore esterno, come un ulteriore aumento del prezzo del bitcoin. Ci sono, tuttavia, alcune determinanti che potrebbero conferire un vantaggio per alcuni partecipanti, permettendo loro di godere di maggiori profitti:

- Vantaggio tecnologico: potrebbe derivare o da un'innovazione nell'implementazione dell'algoritmo di proof of work (SHA256^2) o da un minatore che controlla un miglior processo di creazione di chip, come ad esempio un grande produttore che entra nel business di mining.
- Copertura della volatilità dei Bitcoin: un minatore potrebbe ottenere un vantaggio se fosse in grado di coprire la volatilità del prezzo di Bitcoin in modo più efficace rispetto ai suoi concorrenti. Qualsiasi minatore potrebbe in linea di principio coprirla usando dei futures, anche se per le criptovalute non è ancora facilissimo. Questo vantaggio potrebbe essere particolarmente importante nei periodi in cui il prezzo del Bitcoin è basso e i concorrenti potrebbero essere costretti a vendere.
- Prezzi dell'elettricità più bassi: i minatori che sono in grado di avere bassi prezzi dell'elettricità hanno un vantaggio in termini di costi. L'estrazione mineraria di Bitcoin migrerebbe probabilmente verso luoghi con elettricità abbondante e a basso costo, come l'Islanda o la Russia. Questo potrebbe anche

far diminuire l'impatto ambientale del mining, poiché i luoghi con elettricità poco costosa sono di solito in grado di generarla in maniera sostenibile, come ad esempio da impianti idroelettrici.

In sintesi, i costi marginali di gestione delle attrezzature minerarie comprendono oltre al costo dell'elettricità, i costi di affitto del centro dati, i costi di refrigerazione, i costi per la manutenzione, il costo di ammortamento dell'attrezzatura stessa e il suo costo opportunità.

L'unica tecnologia attualmente valida - l'ASIC - non ha nessun altro utilizzo alternativo. Questi costi, insieme al ritardo nella produzione di attrezzature per l'estrazione mineraria (in risposta agli aumenti del prezzo di Bitcoin), potrebbero creare cicli di boom e bust nel sistema di mining.

La crescita più o meno continua dell'hash rate è dovuta a due tendenze:

- L'aumento del prezzo di Bitcoin dal giorno della sua creazione ad oggi, che ha attirato molti investimenti nel settore
- I progressi nella tecnologia, in quanto i produttori di attrezzature di mining hanno raggiunto un elevatissimo livello nella produzione di chip

Figura 38 – Ricavi del mining negli ultimi 3 anni – Blockchain.info

L'hardware minerario ha seguito una tendenza verso una sempre crescente specializzazione, in cui la maggioranza del chip è dedicato al calcolo dell'hash.

Ci sono state quattro fasi:

- CPU che sta per Central Processing Unit: è il chip principale all'interno di un computer e di altri dispositivi. È un hardware di uso generale: la sua potenza di calcolo può essere utilizzata per molti compiti, tra cui il mining di Bitcoin. La versione iniziale del Bitcoin Core (Bitcoin Core è un software libero e open-source che funge da nodo bitcoin (il cui insieme forma la rete bitcoin) e fornisce un portafoglio bitcoin che verifica completamente i pagamenti. Inizialmente, il software è stato pubblicato da Satoshi Nakamoto con il nome "Bitcoin" e successivamente è stato rinominato in "Bitcoin Core" per distinguerlo dalla rete) ha implementato il mining sulla CPU. Durante la prima fase (che si è svolta dal 2009 all'estate del 2010), il mining è stato eseguito solo utilizzando le CPU. Durante questa fase, la crescita dell'hash rate era dovuta solamente all'ingresso di nuovi appassionati.

- Le GPU che è l'acronimo di Graphics Processing Unit (unità di elaborazione grafica): il chip del computer utilizzato originariamente per l'accelerazione grafica. A partire dalla metà del 2010, le GPU sono state programmate per estrarre i Bitcoin, rendendo rapidamente antieconomica l'estrazione con CPU. Le GPU offrono un vantaggio rispetto alle CPU perché sono composte da centinaia o addirittura migliaia di unità di calcolo, rispetto alle poche decine di una tipica CPU. Le unità di calcolo di una GPU sono molto più limitate di quelle di una CPU, ma sufficienti per eseguire gli hash SHA256.
- FPGA sta per Field-Programmable Gate Array. Gli FPGA sono chip costruiti con blocchi logici che possono essere programmati e interconnessi per eseguire un particolare compito. Come suggerisce il nome, gli FPGA sono progettati per essere programmabili "sul campo", cioè dopo la spedizione. Gli FPGA sono stati introdotti nel settore minerario Bitcoin a metà del 2011 e per un certo periodo hanno gareggiato con le GPU.
- ASIC sta per Application-Specific Integrated Circuit. Gli ASIC sono chip costruiti per una specifica applicazione, a differenza delle CPU (o, in misura minore, delle GPU) che accettano software che eseguono molte possibili applicazioni. Le parti ASIC hanno la funzione SHA256 copiata quante volte l'area del chip lo consenta, al fine di eseguire il maggior numero possibile di tentativi di hash insieme. La potenza di un ASIC dipende dalla tecnologia del processo di produzione.

L'unica tecnologia redditizia attualmente per estrarre i Bitcoin è l'ASIC e, anche la maggior parte di queste macchine che si basano su vecchie tecnologie di processo, possono diventare progressivamente obsolete, facendo diventare il mining un processo addirittura antieconomico a causa dell'alto costo dell'elettricità e della difficoltà di competere per la ricerca del nonce con ASIC ormai superate.

A titolo esemplificativo, viene mostrata una schermata di mining in Windows attraverso il programma Cgminer con un BFL BitFORCE ASIC. [23]

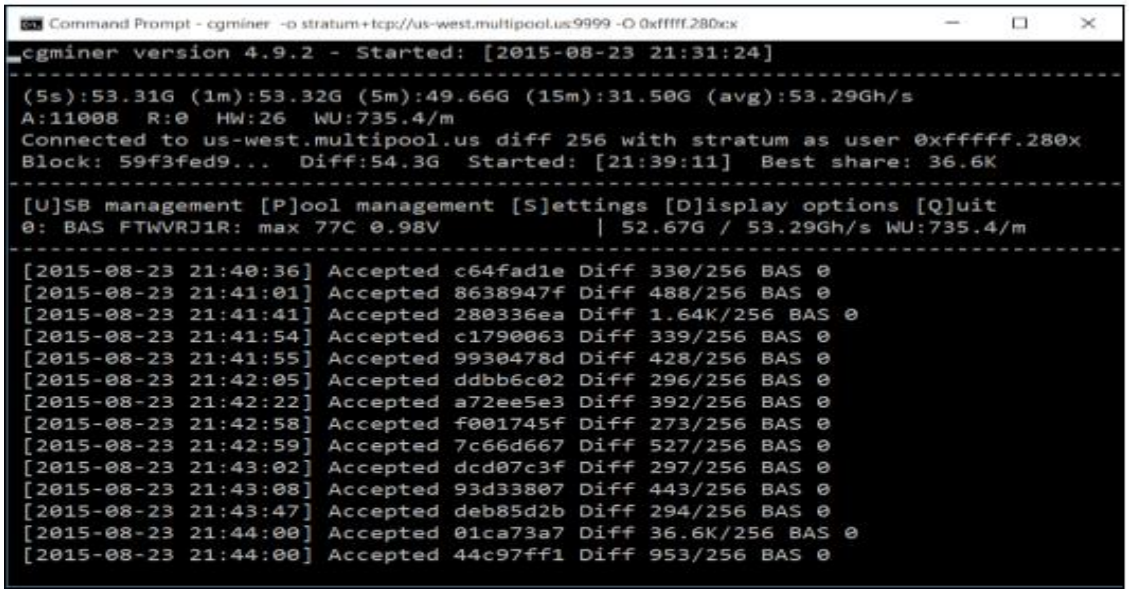


Figura 39 – Esempio di mining con ASIC – Bitcoin Essentials - Szmigielski

L'hash rate è anche un'indicazione della sicurezza della blockchain.

[23] Bitcoin Essentials – Albert Szmigielski - 2016

L'aumento del tasso di hash aumenta il livello di difficoltà per un aggressore che vuole effettuare un attacco alla blockchain, mentre al contrario, una diminuzione dell'hash rate sarebbe dannosa per la sicurezza.
Alcuni esperti hanno dimostrato una correlazione positiva tra il prezzo di Bitcoin e l'hash della rete.
Mentre è vero che gli investimenti in mining seguono il prezzo di Bitcoin, il contrario non è necessariamente vero. Dopo una diminuzione del prezzo del Bitcoin, una parte del mining power verrà scollegata e la sicurezza della blockchain si abbasserà relativamente. Questa dinamica dovrebbe influenzare il prezzo solo indirettamente, in quanto l'effetto di una diminuzione dell'hash rate, se presente, dovrebbe già essere incorporata nel nuovo prezzo di equilibrio.

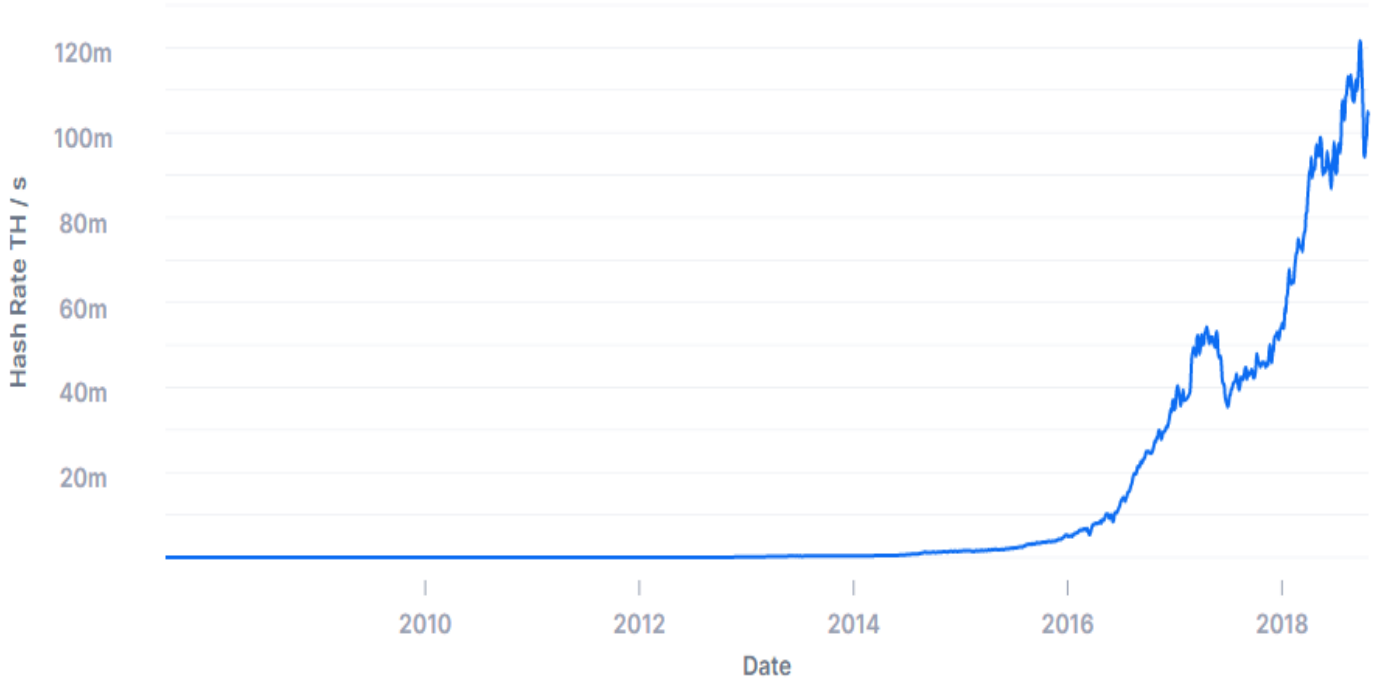


Figura 40 – Hash rate totale – Blockchain.info

Ci sono state alcune polemiche sull'impatto ambientale causato dal mining di Bitcoin; ciò è stato alimentato da valutazioni errate sul consumo totale di energia elettrica della rete, poiché alcuni giornalisti hanno citato una stima sul consumo di elettricità basato sul mining con CPU.
Dato che la tecnologia mineraria è andata avanti e l'unico strumento attualmente utilizzabile - l'ASIC - ha un consumo energetico molto più basso, queste cifre hanno sovrastimato il valore totale.
Man mano che la tecnologia mineraria migliora, i costi per minare saranno incentrati principalmente sul costo dell'elettricità e i miners si sposteranno probabilmente verso località dove l'energia sarà più economica, riducendo anche l'impatto ambientale.

Una mining pool è costituita da un insieme di miners che contribuiscono con il loro hashing power alla ricerca dell'hash del blocco e condividono i premi del mining. Formando un pool, i minatori possono avere un flusso di reddito molto più prevedibile da condividere. La ripartizione delle entrate in una mining pool è proporzionale all'hashing power fornito da ciascun minatore, meno una piccola commissione addebitata dall'operatore del pool, che di solito viene accreditata a scopo di lucro. Un ulteriore vantaggio è che i minatori

che partecipano a un pool non devono conservare una copia della blockchain o elaborare tutte le transazioni in entrata: è sufficiente che l'operatore del pool fornisca ai minatori una copia dell'intestazione del blocco da minare (block header).

Una mining pool in cui i partecipanti promettono di condividere tra loro il lavoro di calcolo e in cui un operatore promette di spartire onestamente le ricompense tra i suoi membri, è soggetto a conflitti di interesse. Sia i minatori, che gli operatori del pool sono incentivati a barare: i minatori, per esempio, tendono o a sovrastimare il loro tasso di hash, o a contribuire solo in parte con i loro ASIC, mentre minano da soli con il resto della loro potenza di calcolo.

Pertanto, il pool deve controllare il lavoro svolto dai minatori chiedendogli di presentare la proof of work, chiamata in questo caso share. L'operatore della mining pool deve misurare le shares ricevute dai suoi minatori e assegnare la ricompensa del blocco proporzionalmente.

Un altro approccio per controllare il lavoro svolto dai minatori è rappresentato dalle "metahashes".

Le metahashes sono gli hash di molti hash prodotti dal minatore. L'operatore del pool controlla la validità delle metahashes fornite, anche se tale controllo non è computazionalmente facile: per controllare tutte le metahashes di tutti i minatori, l'operatore dovrebbe rifare tutto il loro lavoro, il che vanifica lo scopo di un pool minerario.

Pertanto, esso le controlla solo periodicamente, di solito in modalità “Round-Robin” (In una metodologia Round-Robin, i processi vengono scelti in modo circolare tra quelli disponibili. La programmazione in ordine circolare potrebbe essere truffata se un minatore fosse in grado di indovinare la frequenza del controllo del metahash, quindi una certa casualità viene solitamente aggiunta al processo).

Poiché l'approccio metahash è molto più intenso dal punto di vista computazionale rispetto all'approccio share, esso è raramente utilizzato nella pratica.

I minatori sono incentivati a pubblicare da soli un nuovo blocco quando lo trovano, ma questo non può succedere se i minatori ricevono solo l'hash del block header dall'operatore del pool, che include un indirizzo, da lui controllato, nella transazione coinbase. In questo modo, un minatore non può cambiare l'indirizzo che gli sarà dato.

I gestori della mining pool hanno un incentivo a truffare i minatori. Quando viene minato un nuovo blocco, il gestore del pool ha un vantaggio economico nel non condividere la ricompensa o nel condividerla solo con il minatore che ha minato il blocco, lasciando il resto del pool all’oscuro di questa situazione.

Questo problema potrebbe essere risolto se i minatori chiedessero al gestore l'intera intestazione del blocco, non solo l'hash iniziale, in modo da poter monitorare la blockchain da soli. Un altro approccio è che i minatori ricevano un pagamento fisso per il loro lavoro, indipendentemente dal fatto che un blocco sia stato estratto dalla loro mining pool.

I blocchi hanno un limite di circa 1MB di dimensione totale. La dimensione di una transazione dipende dal numero di indirizzi da cui la transazione attinge i fondi e a cui i fondi vengono inviati.

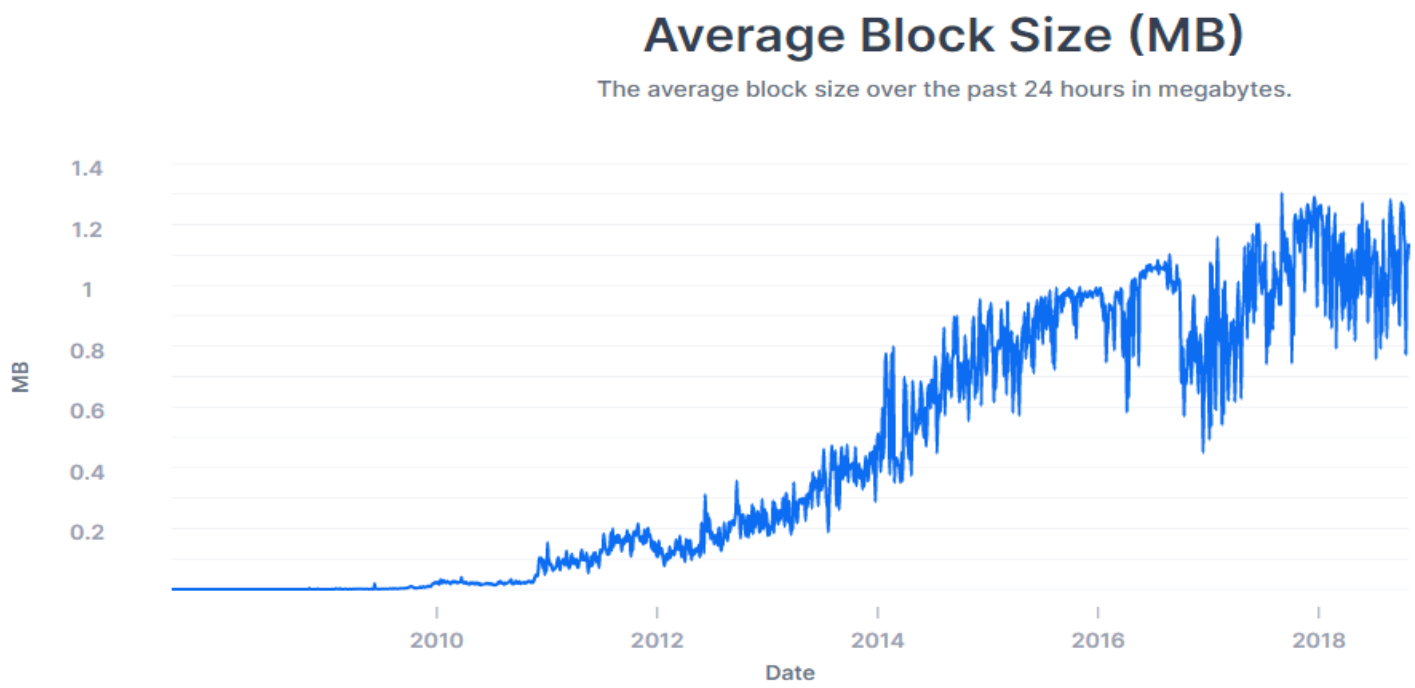


Figura 41 – Dimensione media dei blocchi – Blockchain.info

I minatori devono scegliere quali transazioni includere nel blocco che stanno estraendo dal loro mempool: questo è un problema di ottimizzazione rispetto alla dimensione del blocco.

Le transazioni sono ordinate in base al loro rapporto fee-per-kilobyte in ordine decrescente e scelte da questo elenco. Ci potrebbero essere scenari in cui i minatori si rifiutano di includere transazioni nei blocchi che estraggono a causa del problema "i blocchi più grandi richiedono più tempo per essere trasmessi"; includere più transazioni in un blocco, rende il blocco più grande e ciò implica la necessità di più tempo per la sua trasmissione attraverso la rete; questo fa aumentare il rischio che un altro minatore possa contemporaneamente trovare e trasmettere un blocco concorrente.

Ogni blocco riserva un certo spazio, chiamato priority block (blocco prioritario), per le transazioni prioritarie. Le transazioni con una priorità superiore a un determinato cut-off sono considerate principali e possono essere incluse nel priority block. Le stesse possono anche non includere alcuna commissione, mentre quelle non prioritarie devono forzatamente avere una commissione minima.

Tradizionalmente, per le commissioni minime c'è la possibilità per l'utente di modificarne il valore. Idealmente le commissioni di transazione sarebbero determinate da un meccanismo di mercato, per cui i minatori si comportano già come partecipanti razionali, scegliendo le transazioni che massimizzano i loro profitti.

Gli sviluppatori Bitcoin hanno lavorato intorno al concetto di smart fee o floating fee, cioè un algoritmo che stima la commissione in modo che la transazione abbia un'alta probabilità di essere inserita nel prossimo blocco estratto. Una transazione con una commissione bassa potrebbe rimanere non confermata per un certo periodo di tempo, specialmente quando ci sono molti scambi. I minatori possono avere un incentivo a includere le transazioni, anche quelle con commissioni molto basse, per liberare memoria dal loro mempool. D'altra parte, l'inclusione di molte transazioni rende il blocco più grande, con conseguenti ritardi di propagazione. Per

questo motivo, i minatori hanno talvolta scelto di limitare l'ammontare delle transazioni incluse nei loro blocchi. [24]

Figura 42 – Commissioni per transazione – Blockchain.info

[24] Cryptocurrency Mining – Devan Hansel - 2018

2.16. Attacchi alla blockchain

Una tipologia di attacco caratteristico per tutte criptovalute è quello del double spending, che si verifica quando due diverse transazioni cercano di spendere gli stessi soldi digitali. Il protocollo Bitcoin si difende da questo attacco decidendo che la transazione valida sia quella che per prima viene inserita in blockchain e certificata tramite la prova di lavoro. In questo modo Bitcoin risolve il problema del double-spending in modo decentralizzato, senza la necessità di un'autorità centrale che decida quale transazione sia quella valida.

2.16.1. Attacco al 51%

Si torni all'esempio dei BGP. La soluzione che propone Bitcoin per tale problema presenta un difetto: Bitcoin presuppone che i generali traditori non possano mai guadagnare più del 51% della potenza di calcolo sulla rete. Bitcoin ha un costo per l'invio di messaggi falsi, ma fornisce anche un incentivo per risolvere l'equazione matematica.
Dato che Bitcoin si basa sul consenso della potenza di calcolo maggioritaria, è vulnerabile se il 51% della rete cade sotto il controllo di alcuni malintenzionati.
Per capire l'attacco al 51%, si supponga di avere un gruppo di 10 generali che usano tutti il protocollo Bitcoin per decifrare un messaggio del tipo "attaccare" o "ritirarsi". In questo caso, 6 dei 10 generali avrebbero bisogno di confermare che il messaggio sia legittimo confrontandolo con altri 5 messaggi.
Fino a che tutti i 6 messaggi sono uguali, l'intero gruppo di generali accetta di seguire quel messaggio.
Il difetto si presenta quando si presume che tutti i generali lavorino in modo indipendente: se il gruppo include 6 generali traditori, è possibile che 6 messaggi falsi siano inviati e si vada verso una direzione sbagliata.
Tuttavia, la probabilità che ciò si verifichi in modo casuale è piuttosto piccola, ma non trascurabile, soprattutto perché Bitcoin tratta il trasferimento di qualcosa di valore. Si supponga che 6 generali abbiano deciso in anticipo di inviare un messaggio fasullo: se ciò dovesse accadere, il sistema fallirebbe. Perché dovrebbero far questo?
Usando il BGP, si torni alla ragione originale per cui i generali volevano attaccare la città: in caso di successo, ognuno avrebbe ricevuto la ricompensa.
Si assuma che la città da conquistare sia piena di indicibili ricchezze che potrebbero essere divise tra i conquistatori. Bitcoin assicura che il messaggio corretto sia stato ricevuto e in cambio fornisce moneta come ricompensa al primo generale onesto per aver risolto l'equazione matematica. Tuttavia, questo sistema va in down se il costo per risolvere il calcolo matematico diventasse più alto della ricchezza che un generale riceverebbe in caso di attacco riuscito. Inoltre, se l'incentivo (bitcoin) è significativamente più alto del costo per avere i computer, con la potenza di calcolo necessaria per risolvere il problema, si presenta comunque un altro tipo di problema.

Si analizzino le due situazioni: si assuma che ci sia un costo iniziale di 1.000 per acquistare un computer utile per risolvere l'equazione. Si supponga inoltre che il costo dell'elettricità per farlo funzionare sia di 100, per un investimento totale di 1.100. Se il generale riceve 500 per risolvere il problema e 700 per attaccare la città (che è il valore intrinseco di Bitcoin), allora guadagna 100 e ha un incentivo monetario per rimanere onesto. Tuttavia, se il denaro ricevuto per la risoluzione del problema di calcolo (l'attacco alla città) scende sotto 1.100, è improbabile che il generale partecipi e, per avere successo, tutti i generali sono chiamati a partecipare. Inoltre, cosa succederebbe se il valore dell'incentivo a risolvere l'equazione salisse significativamente, per esempio, a 20.000? In questo caso, il generale riceverà più soldi a risolvere l'equazione che ad essere un traditore, ma si verificherebbe un altro problema: l'incentivo è abbastanza alto da incoraggiare comportamenti nefasti.

Qualsiasi generale potrebbe acquistare sei computer e farli funzionare per un costo totale di 6.600 e, dato che questo generale sarebbe l'unico in grado di confermare 6 messaggi, potrebbe controllare il network. Inoltre, dominerebbe anche la maggior parte della potenza di calcolo sulla rete e potrebbe essere quasi sempre il primo a risolvere il problema matematico.

In questo caso, il generale riceve 20.000 per risolvere il problema, mentre spende solo 6.600 per la potenza computazionale e il profitto di 13.400 sarebbe un potente incentivo ad essere disonesto.

A livello tecnico invece, un aggressore che volesse cambiare la blockchain in un determinato blocco, dovrebbe minare nuovamente tutti i blocchi, da quello fino al blockchain head. Inoltre, poiché la rete Bitcoin procede senza interruzioni, l'attaccante dovrebbe non solo rifare tutto il lavoro passato, ma anche tenere il passo con la creazione di nuovi blocchi. L'unico modo per cui un attaccante potrebbe compiere una tale impresa è che sia in grado di comandare un hash rate grande quanto quello del resto della rete: questa situazione si chiama attacco al 51%.

Un attacco al 51% è anche possibile per un aggressore che controlla meno della metà dell'hash della rete. In questo caso la probabilità di successo dipende dalla percentuale di hashing power che l'aggressore possiede e dal numero di blocchi che deve minare. La probabilità di successo diminuisce in modo esponenziale in entrambi i casi. Solo quando l'aggressore controlla più del 50% dell'hash rate, la probabilità di successo è del 100%.

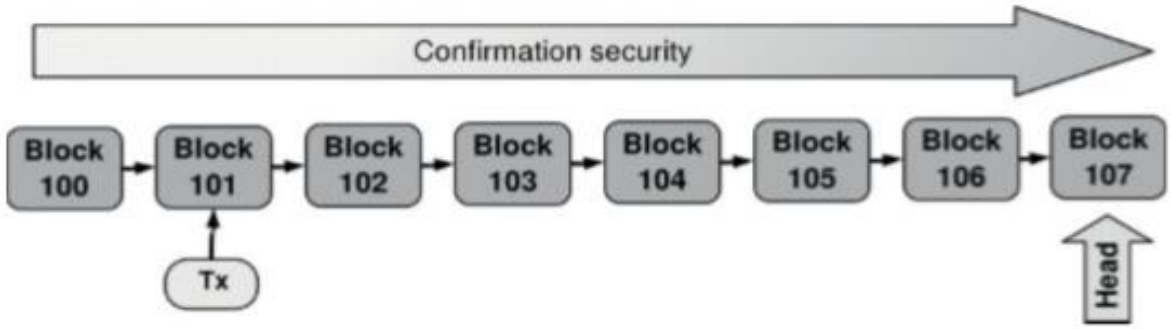


Figura 43 – Attacco alla blockchain - Understanding Bitcoin – Pedro Franco

È possibile per un gruppo di individui disonesti (non per una sola persona) acquistare una potenza di calcolo sufficiente per controllare l'intera rete. La potenza dei computer utilizzati per risolvere l'equazione matematica

sta aumentando ad un ritmo esponenziale e ciò potrebbe portare alcune mining pools ad avere una potenza di calcolo che potrebbe essere sufficiente per controllare l'intera rete Bitcoin.

2.16.2. Race attack

Quando un nodo riceve una nuova transazione non confermata che è in conflitto con una presente nella sua cache delle transazioni non confermate, cioè spende gli stessi output, la nuova transazione viene ignorata. Così i nodi conservano nella loro cache solo una copia della prima transazione ricevuta.
Si verifica un race attack quando un fornitore accetta il pagamento di una transazione non confermata controllando solo pochi nodi.
L'aggressore può inviare una transazione ai nodi vicini al venditore e un'altra a molti altri nodi della rete. In questo modo solo i nodi più vicini mostrano la transazione che invia i fondi al venditore, mentre il resto della rete include nel suo mempool la doppia spesa.
Per difendersi da questo attacco, un fornitore dovrebbe attendere che la transazione sia inclusa in almeno un blocco, perché un double spending è molto improbabile se la maggior parte dei nodi di mining della rete hanno già una transazione valida nel loro mempool. Un gestore di pagamenti o servizi di portafoglio può trarre vantaggio da questo fatto, mantenendo aperte le connessioni verso una grande porzione di nodi di mining e controllando che la transazione corretta sia nel pool di memoria delle loro transazioni non confermate; questo permette all'operatore del servizio di offrire una conferma della transazione molto veloce ma con un rischio limitato.
In una situazione ideale, tutte le transazioni in sospeso nella rete sarebbero incluse nel successivo blocco estratto e ciò presuppone che si propaghino nella rete senza ritardi, raggiungano tutti i minatori e vengano incluse nel blocco successivo.
In realtà, alcune transazioni possono subire ritardi prima di essere incluse in un blocco, ma possono anche essere lasciate cadere dalla rete e non arrivare mai in blockchain.
Può anche essere possibile che una transazione che, è stata dimenticata dalla rete, ricompaia dopo che un nodo inizi a trasmetterla di nuovo. Nei casi in cui una transazione non venga confermata, si dovrebbe forzare un double spending di quella transazione ad un indirizzo diverso. Questo ha il vantaggio di prevenire o almeno di rendere evidenti i problemi di malleabiliy di qualche transazione (magari a causa delle commissioni troppo basse).

2.16.3. Finney attack

Questo attacco è stato scoperto da Hal Finney da cui il nome dell’attacco. L'aggressore (che è anche un miner) mina segretamente un blocco, includendovi una transazione da uno dei suoi indirizzi a un altro e questa transazione non viene trasmessa alla rete, ma viene inclusa solo nel blocco che l'aggressore sta segretamente

minando. Poco prima di rilasciare il blocco, l'aggressore invia una doppia spesa del TxOut incluso nella sua transazione segreta, per cui quest'attacco riesce a bypassare la protezione di un utente-vittima che si limita a monitorare solo che la transazione si sia propagata attraverso la rete (ovvero osserva solo l'output e non il fatto che l'input sia stato spedito due volte).

Una volta che la vittima ha accettato il pagamento, l'aggressore rilascia il blocco estratto segretamente, effettuando così una doppia spesa. Questo attacco ha un costo opportunità associato, perché c'è un ritardo dal momento in cui il blocco viene estratto al momento in cui il blocco viene rilasciato nella rete. Questo ritardo si realizza a causa della chiusura dell'accordo con la vittima e dell'attesa che la transazione spesa due volte si propaghi attraverso la rete. Durante questo ritardo c'è la possibilità che qualche altro minatore trovi un blocco e quindi l'attacco con doppia spesa fallisca e l'aggressore perda la ricompensa.

Contrariamente all'attacco al 51%, un attaccante non deve controllare la maggior parte dell'hash rate. Un minatore con un hash rate basso potrebbe eseguire questo attacco, aspettando solo il momento in cui riuscirà ad estrarre un blocco nuovo.

2.16.4. Spamming di transazioni

Un aggressore potrebbe tentare di effettuare un denial of service (DoS) sulla rete Bitcoin, creando molte transazioni in cui invia i fondi a se stesso. Poiché lo spazio di archiviazione nei blocchi è limitato, questo attacco potrebbe impedire di validare le transazioni legittime.

Si consideri tuttavia che è improbabile che questo attacco funzioni a causa dei seguenti fattori:

- Il numero di transazioni "free" che un aggressore può inviare nella rete è limitato. I blocchi estratti hanno una dimensione massima per blocco di 50kB riservata alle transazioni prioritarie free. Le transazioni al di sopra di questo limite devono pagare delle commissioni. Quindi, alla fine, un aggressore dovrebbe pagare molte fees per effettuare questo attacco
- Le commissioni che l'aggressore dovrebbe pagare, dovrebbero competere con le commissioni delle transazioni vere. Con il passare del tempo, le commissioni per le transazioni presumibilmente aumenteranno, rendendo questo attacco più costoso da sostenere, poichè le commissioni che un aggressore deve inviare devono superare una certa soglia. Al di sotto di questa soglia una transazione è difficile che venga convalidata e decadrà nella rete. Pertanto, un aggressore dovrebbe possedere molti fondi in Bitcoin per poter effettuare l'attacco.

Una variante di questo attacco potrebbe essere l'invio di transazioni non valide ad alcuni nodi.

Una transazione non valida viene controllata dal nodo che la riceve e fatta cadere, ma il controllo della firma è un'operazione ad alta intensità di CPU per cui questo attacco potrebbe riuscire a rallentare i nodi che sono inondati di transazioni, anche se ora, con la tecnologia ASIC, la portata di questo tentativo di attacco si è ridimensionata.

Tuttavia, poiché i nodi inondati non trasmettono la transazione non valida, lo spamming cerca di rallentare il loro lavoro, con la conseguenza di creare un danno indiretto anche alla rete. I nodi pertanto devono proteggersi da questo attacco nello stesso modo in cui i normali server su Internet si proteggono da attacchi DoS. [25] [26] [27]

[25] Understanding Bitcoin – Pedro Franco – 2014

[26] Blockchain basics – Daniel Drescher – 2017

[27] Learning Bitcoin – Richard Caetano - 2015

2.17. Scalabilità

Con l'aumento dell'utilizzo, la grandezza della blockchain è cresciuta costantemente. La Figura 44 mostra la dimensione su scala normale.

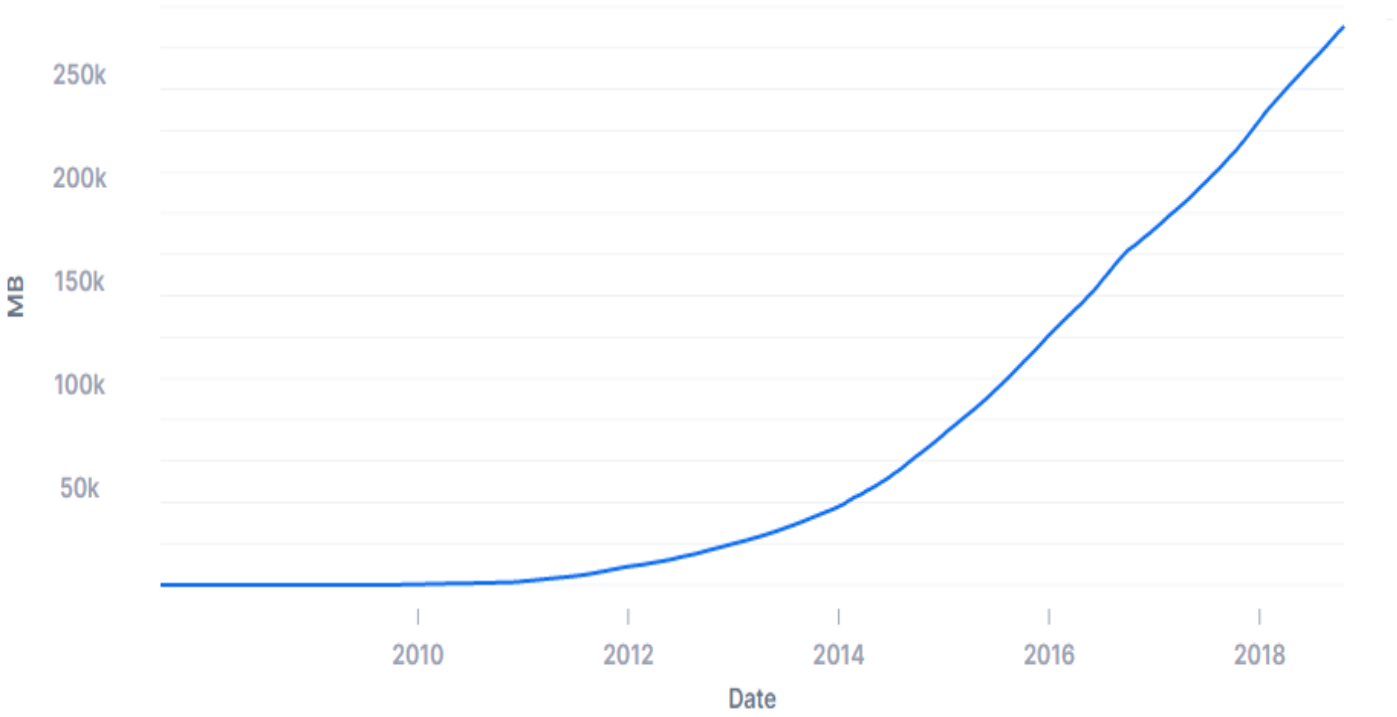


Figura 44 – Dimensioni della blockchain – Blockchain.info

Una delle critiche spesso sollevate contro Bitcoin è che non sia abbastanza scalabile per gestire un tasso di transazioni paragonabile a quello delle normali reti di elaborazione di pagamenti. Al momento della scrittura, i blocchi includono una media di 2100 transazioni.

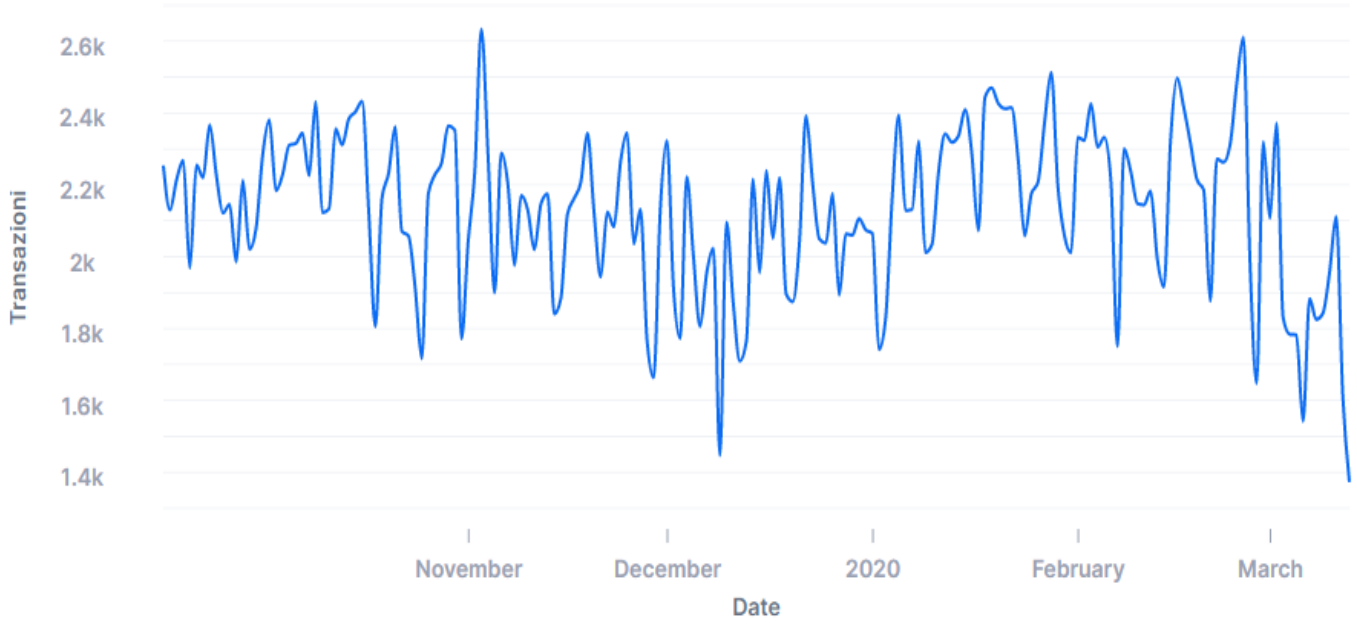


Figura 45 – Numero di transazioni nei blocchi – Blockchain.info

Infatti, la crescente adozione delle criptovalute ha sollevato molte preoccupazioni circa la loro capacità di scalare. Poiché Bitcoin è un sistema che si autoregola e che funziona estraendo i blocchi ad intervalli regolari, il suo crescente volume di transazioni si scontra con il limite massimo di dimensione di ciascun blocco rispetto all'intervallo in cui viene minato.

La tendenza crescente riguardante le dimensioni dei blocchi su Bitcoin, fa presagire un potenziale problema che si concretizzerebbe ove il sistema addirittura arrivasse a cancellare le transazioni a causa sia della scarsa velocità nel minare i blocchi sia della dimensione del blocco rispetto al numero di transazioni.

Di conseguenza, la comunità sta discutendo da tempo le tecniche per migliorare la scalabilità delle blockchain, in particolare di quella di Bitcoin. Questi dibattiti sono molto accesi e hanno portato a spaccature interne, senza aver delineato un chiaro percorso per affrontare il problema della scalabilità. La blockchain di Bitcoin impiega 10 minuti o più per confermare le transazioni, raggiungendo un flusso di massimo 7 transazioni al secondo (tps).

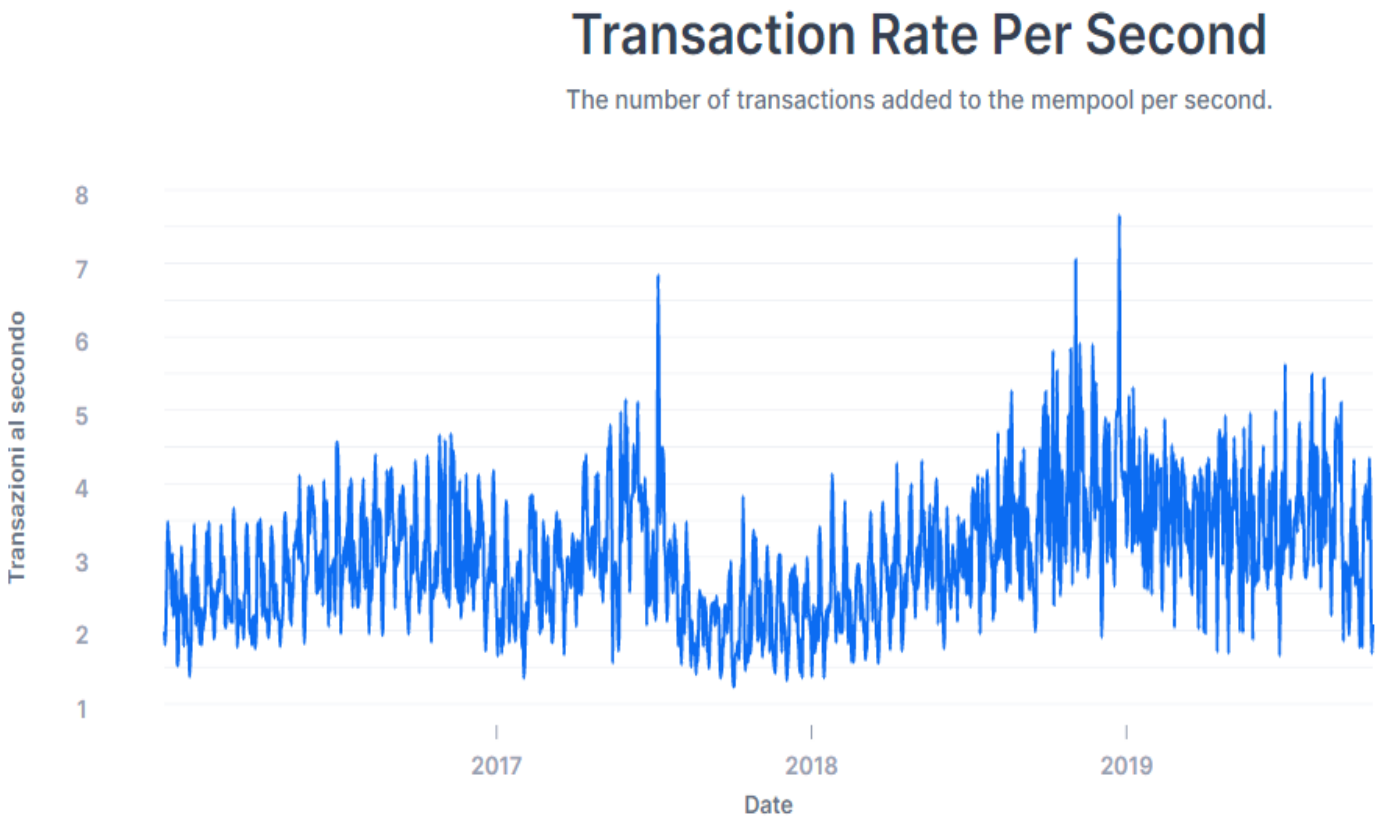


Figura 46 – Numero di transazioni al secondo – Blockchain.info

In confronto, un procedimento di pagamento tradizionale, come la carta di credito Visa, effettua una transazione in pochi secondi ed elabora in media 2000 transazioni al secondo, con un picco di 56.000 tps: esiste un divario troppo grande tra la situazione attuale di Bitcoin e la scalabilità dei sistemi di pagamento tradizionali.

Pertanto, le domande chiave sono: "Può la blockchain scalare come un sistema di pagamento tradizionale? Come bisogna fare per arrivarci?"

Per Bitcoin, la strategia migliore per avere un numero di tps come VISA è quello di spostare le transazioni fuori dalla blockchain.

Sarebbe utile e produttivo se Bitcoin supportasse un numero quasi illimitato di transazioni al secondo con tariffe estremamente basse per i micropagamenti, che possano essere inviati in sequenza tra due parti per consentire dei trasferimenti di denaro. Per potercela fare, sarebbe necessario ridurre drasticamente l'importo delle commissioni delle transazioni sulla blockchain di Bitcoin.

Affinchè Bitcoin abbia successo è necessario che, se dovesse diventare estremamente popolare e di uso comune per qualsiasi tipo di pagamento, i suoi attuali vantaggi derivanti dal decentramento continuino in futuro ad esistere, ma per fare in modo che le persone oggi possano credere che Bitcoin funzionerà domani, la criptovaluta deve risolvere il problema della dimensione dei blocchi.

1 MB rappresenta la dimensione del blocco della blockchain di Bitcoin: questa grandezza, stabilita da Satoshi Nakamoto, era stata decisa per prevenire attacchi DDOS da parte di hackers che avrebbero potuto creare blocchi di dimensioni enormi da diffondere nel network con la conseguenza di mandarlo in down.

La decisione ha dato vita nel lungo periodo ad un problema di capacità della rete stessa. Ogni transazione in bitcoin contiene molti dati personali, che occupano una quantità di spazio irrisorio nel valore di una transazione, ma quando gli scambi iniziano ad essere centinaia di migliaia al minuto, possono manifestarsi problemi. L'attuale limite di 1 MB, che riesce a sopportare dalle 3 alle 7 transazioni al secondo, è già poco sostenibile per il bacino di utenti.

Due diverse alternative si contesero la risoluzione del problema della scalabilità su Bitcoin: Bitcoin Unlimited e SegWit.

Bitcoin Unlimited provava ad abolire il concetto stesso di limite, presentando la possibilità di far creare ai minatori blocchi di dimensione arbitraria che sarebbero stati trasmessi alla rete, in competizione gli uni con gli altri per una posizione in blockchain. Ciò permetteva di aumentare il volume dei blocchi fino ad un massimo di 4 MB, sebbene molti esperti vicini allo sviluppo del progetto affermassero che probabilmente la grandezza dei blocchi si sarebbe assestata intorno ai 2 MB.

Alcuni utenti erano certi che l'abolizione del limite dei blocchi proposta da Bitcoin Unlimited avrebbe potuto far gonfiare a dismisura la blockchain. Questo avrebbe portato ad una maggiore centralizzazione di Bitcoin: solo le grosse mining pools sarebbero state infatti in grado di permettersi lo spazio, la potenza di calcolo e la banda necessarie a processare una tale quantità di dati. Gli operatori più piccoli sarebbero stati gradualmente eliminati dalla rete, contrariamente all'idea fondante del Bitcoin che lo voleva come moneta governata da ciascuno dei suoi utenti.

Neanche SegWit costituiva una soluzione perfettamente decentralizzata, dato che offriva un leggero incremento dello spazio, fino a 4 MB, collocando al tempo stesso alcuni dati non critici all'esterno dei blocchi. Infatti, di per sé, avrebbe aumentato la capacità dei blocchi a circa 2 MB nel breve termine, fino ad un massimo di 4 MB nel lungo periodo, a seconda della futura velocità di crescita della rete. Prima o poi, però, il limite sarebbe stato nuovamente raggiunto e la capacità aumentata.

Entrambi i tentativi sono falliti.

Quale potrebbe essere una soluzione? Un'idea potrebbe essere che, se solo due partecipanti si scambiano denaro in maniera ricorrente tutti i giorni, non è necessario che tutti gli altri nodi della rete Bitcoin sappiano

di quella transazione, poiché è invece preferibile avere solo il numero minimo indispensabile di informazioni sulla blockchain.

Utilizzando una rete di canali di micropagamento, Bitcoin può scalare fino a miliardi di transazioni al giorno con la potenza di calcolo disponibile su un moderno computer. L'invio di molti pagamenti, all'interno di un determinato canale di micropagamento, permette di inviare grandi quantità di fondi da una parte all'altra in modo decentralizzato. Questi canali non sono una rete di fiducia separata dal bitcoin, ma sono vere e proprie transazioni. I canali di micropagamento creano una relazione tra due parti per aggiornare continuamente i saldi, rinviando ciò che viene trasmesso alla blockchain in una singola transazione che compensa il saldo totale. I canali utilizzano transazioni bitcoin reali, scegliendo solo di spostare la trasmissione off chain in modo tale che entrambe le parti possano garantire il loro attuale saldo sulla blockchain; non si tratta di una rete di overlay poiché i pagamenti nei canali di micropagamento sono comunicati e scambiati con bitcoin reali off-chain.

Quindi, se entrambe le controparti concordano che il saldo corrente all'interno di un canale di scambio è di 0,07 BTC per Alice e 0,03 BTC per Bob, allora questo è il vero saldo. Tuttavia, senza la crittografia, si crea un possibile problema: se una controparte non è d'accordo sul saldo corrente dei fondi, allora è la parola di uno contro l'altro. Senza firme crittografiche, la blockchain non saprà chi possiede cosa.

Se il saldo nel canale è 0,05 BTC ad Alice e 0,05 BTC a Bob e il saldo dopo una transazione è 0,07 BTC ad Alice e 0,03 BTC a Bob, la rete deve sapere quale è il saldo corretto al momento. La blockchain risolve questo problema utilizzando il timestamping; sarebbe però auspicabile non utilizzare questo sistema se non in caso di assoluta necessità, in quanto lo stesso può comportare costi eccessivi per dei canali di micropagamento.

Invece, entrambe le parti possono impegnarsi a firmare una transazione e a non trasmetterla. Quindi, se Alice e Bob si impegnano a versare i fondi in un sistema multifirma 2 a 2 (dove è necessario il consenso di entrambe le parti per legittimare uno scambio), possono concordare il livello di fondi attuale. Alice e Bob possono decidere per esempio di riscambiarsi i bitcoin nuovamente rispetto alla transazione precedente nel sistema 2 a 2 e questo nuovo cambiamento non viene trascritto in blockchain.

Entrambe le parti possono farlo, ma possono anche scegliere di tenere per loro la transazione, sapendo che sono in grado di riscattare i fondi ogni volta che vogliono e rinviando la trasmissione di questa transazione, possono scegliere di scambiare bitcoin nuovamente in una data futura attraverso lo stesso canale.

Per modificare questo saldo, entrambe le parti creano un nuovo scambio attraverso l'indirizzo 2 a 2 multifirma, per esempio 0,08 ad Alice e 0,02 a Bob.

Pertanto, i canali di micropagamento creano solo un rapporto tra due parti.

Richiedere alla totalità dei membri del network di creare canali con tutti gli altri non risolve tuttavia il problema della scalabilità, che in Bitcoin può essere raggiunta solamente utilizzando una grande rete di canali di micropagamento. Se si presume una grande rete di canali e tutti gli utenti Bitcoin partecipano a questo sistema avendo almeno un canale aperto con una precisa controparte, è possibile creare una quantità quasi infinita di transazioni all'interno di questo network. Le uniche transazioni che vengono scritte sempre sulla blockchain Bitcoin sono quelle con controparti con cui non si ha un canale di pagamento diretto, svuotando almeno un

po' la blockchain. Inserendo nell'output delle transazioni Bitcoin un hashlock e un timelock, la controparte del canale diretto non sarà in nessun modo in grado di rubare i fondi e i Bitcoin potranno essere scambiati senza il pericolo di furto. [28]

Un progetto ambizioso, sviluppato da più parti tra cui Lightning Labs, ACINQ e Elements Projects, chiamato Lightning Network, è una delle soluzioni più efficaci possibili per risolvere il problema della scalabilità di Bitcoin.

Lightning Network è una proposta di implementazione dei contratti Hash Timelock con canali di pagamento bidirezionali, che permettono di instradare le transazioni in modo sicuro su più canali peer-to-peer. Lightning Network promette la consegna di transazioni Bitcoin scalabili, istantanee e senza commissioni, cercando di risolvere i problemi sopra descritti.

L'idea di fondo di Lightning Network nasce dal meccanismo dei Payment Channels (Canali di Pagamento), che sono un'applicazione di quanto è stato appena descritto. Esso è un sistema di transazioni off-chain, quindi non incluse direttamente all'interno della blockchain, che riduce significativamente i dati da registrare su di essa.

Il Payment Channels consistono nella creazione di un canale bidirezionale tra due utenti, che sono in grado di effettuare transazioni tra loro istantaneamente, liberamente e senza fiducia. Il canale tra gli utenti viene aperto con una transazione di finanziamento, che è inclusa nella blockchain, proprio come una normale transazione Bitcoin, richiedendo così una commissione per il mining.

Gli stessi utenti generano, firmano e scambiano anche una transazione di rimborso (refund transaction) che permetterà alle controparti di recuperare i loro fondi bloccati. Una volta aperto il canale descritto, gli utenti saranno in grado di effettuare transazioni tra di loro, entro i limiti della loro capacità di canale (l'importo bloccato) semplicemente aggiornando la transazione di rimborso, che sostituisce ogni volta la precedente. Nessuna transazione che si verifichi tra i due utenti finirà sulla blockchain, non necessitando quindi di tasse minerarie o di tempo di conferma.

Sarà infatti trasmessa una sola transazione, quella finale, che chiuderà il canale ogni volta che una delle controparti lo deciderà. In questo caso, l'utente dovrà solo inviare l'ultima transazione di rimborso e attendere che venga minata, per riprendere il controllo dei suoi bitcoin.

Il protocollo mantiene la stessa natura priva di fiducia di Bitcoin, implementando un meccanismo di penalizzazione: se una delle controparti (Bob) desidera spendere una vecchia transazione di rimborso, ritirando così più bitcoin di quelli che gli sono dovuti, la controparte lesa (l'utente dall'altra parte del canale) potrebbe spendere l'ultima transazione di rimborso valida. Il canale verrebbe quindi chiuso, dando tutti i bitcoin alla controparte danneggiata. Questo meccanismo disincentiva il tentativo di frodi all'interno dei canali di pagamento.

[28] The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments – Joseph Poon, Thaddeus Dryja - 2016

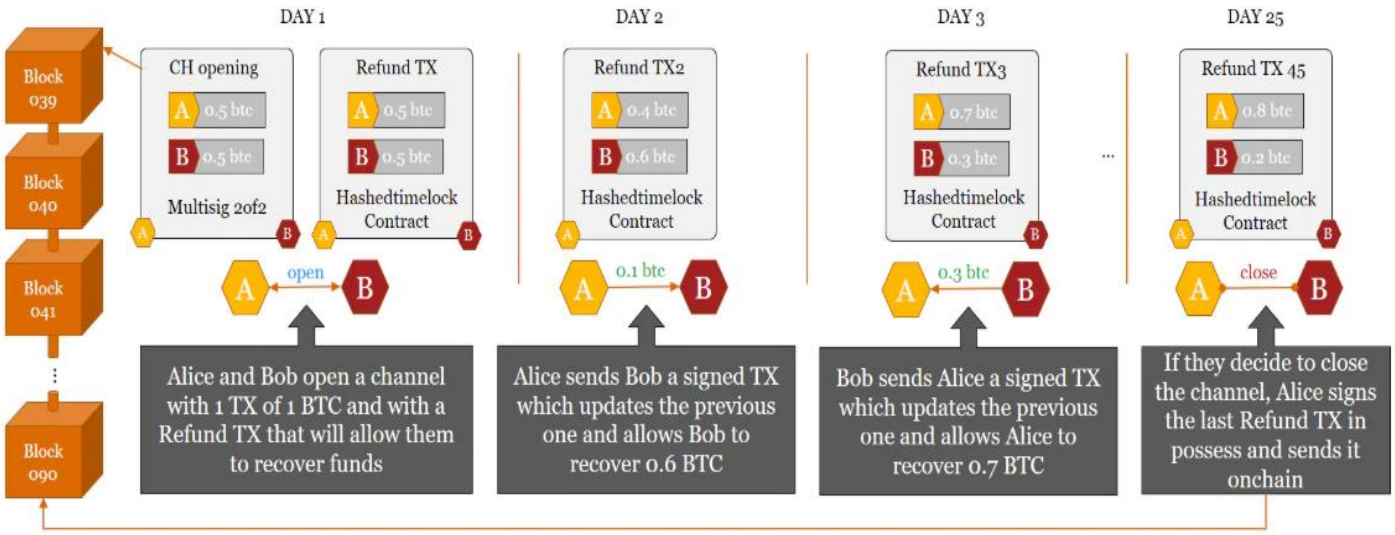


Figura 47 – Funzionamento dei Canali di Pagamento – Bitcoin: from digital gold to electronic cash with Lightning Network - PwC

Anche se i Canali di Pagamento funzionano correttamente, non sono in grado di risolvere il problema della scalabilità di Bitcoin, poiché gli utenti dovrebbero creare molti canali tra di loro per poter effettuare transazioni, bloccando troppi bitcoin (e quindi la liquidità).

Un'estensione dei Canali di Pagamento potrebbe essere in grado di risolvere il problema di scalabilità. Ciò è reso possibile da Lightning Network, che permette di "estendere" i Canali grazie alla crittografia, formando una "rete" di collegamenti.

Lightning Network funziona attraverso una struttura ramificata di canali di pagamento. Ciò significa che non è necessario avere un canale aperto con ciascun utente per effettuare una transazione: è sufficiente "trovare un modo" per raggiungere il “bersaglio” finale, passando attraverso i diversi percorsi nei canali. Nella Figura 48, Alice desidera inviare una transazione a Tim senza avere un canale diretto con lui. Alice è in grado di effettuare una transazione con Tim sfruttando il suo canale con Bob, che possiede un canale diretto con Tim.

Per il suo lavoro di "intermediario", Bob può chiedere una commissione. Dopotutto, Alice sarà in grado di inviare una transazione a Tim senza includerla nella blockchain, istantaneamente, senza fiducia e quasi liberamente grazie a Lightning Network. La transazione viene inviata in quanto il protocollo è progettato per non permettere a Bob di prendere i bitcoin di Alice, non pagando Tim.

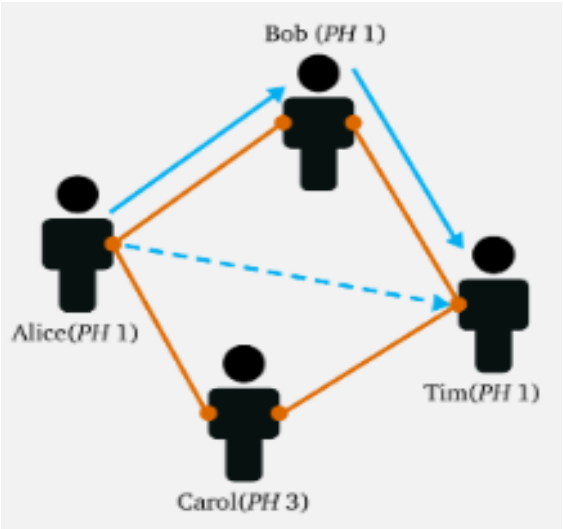


Figura 48 – Lightning Network - Bitcoin: from digital gold to electronic cash with Lightning Network - PwC

Lightning Network, per progettazione, include un algoritmo che permette di pesare tutti i salti (intermediari), al fine di ridurre al minimo il costo totale. Per esempio, Alice avrebbe potuto pagare Tim passando per Carol, ma quest'ultima ha una commissione sul salto più grande di quella di Bob.

Vale la pena notare che Alice potrebbe passare da Bob se quest'ultimo avesse abbastanza bitcoin per pagare Tim. Se Bob non ha abbastanza bitcoin nel suo canale, il portafoglio di Alice lo rileverà e troverà un'altra strada per pagare Tim.

Questo significa che, più bitcoin sono bloccati nei canali, più traffico i partecipanti possono gestire su Lightning Network, potendo così riscuotere più tasse.

Lightning Network introduce quindi diversi vantaggi: il più importante e visibile è una drastica diminuzione delle commissioni di pagamento; attualmente, la commissione per ogni salto è di circa 1 satoshi, ovvero meno di 0,001 euro. Inoltre, le transazioni sono istantanee e la privacy degli utenti aumenta potenzialmente in quanto le transazioni non vengono registrate su un registro pubblico condiviso. Di conseguenza, se Lightning Network aumenterà di partecipanti, ci saranno meno transazioni sulla blockchain del Bitcoin e quindi le commissioni potrebbero essere indirettamente abbassate anche sulla stessa blockchain.

Oltre ai molteplici vantaggi introdotti dalla soluzione off-chain offerta da Lightning Network, vi sono però diversi potenziali punti deboli sia dal punto di vista del routing, che della capacità e della sicurezza.

Come si accennava in precedenza, per condurre una transazione tramite Lightning Network è necessaria una serie di canali di pagamento collegati (dove non è possibile avere un canale di pagamento diretto con un utente specifico).

Pertanto, per far sì che il pagamento avvenga, è necessario disporre di un percorso. Se Alice vuole pagare Tim, deve conoscere l'intero "stato" di ogni canale dell'intera rete e questo stato deve essere aggiornato "in tempo reale" man mano che la transazione avviene sul network, per consentire ad Alice di trovare un "percorso" funzionante. Con poche migliaia di nodi questo può essere gestito da una rete grafica, ma non appena la rete sarà costituita da milioni di nodi diventerà un problema; oggi non c'è ancora una soluzione appropriata al problema del routing.

Quando si invia una transazione, ogni salto da cui la transazione sta passando (il che significa che si sta instradando), deve avere fondi disponibili per "gestirla". Se Alice sta inviando 1 BTC a Tim, passando attraverso Carol, che funge da salto, significa che Carol dovrà avere almeno 1 BTC nel suo canale con Tim. Ne deriva che, per far funzionare Lightning Network, è necessario che i canali abbiano sempre capacità economica e quindi bitcoin bloccati nel canale.

Un altro punto debole che si può notare in Lightning Network è ancora legato all'"essere online". Infatti, per mantenere aperti i canali all'interno del protocollo, i fondi bitcoin devono rimanere online, ma, secondo le migliori pratiche, il modo più sicuro per conservarli è quello di tenere i bitcoin in modalità cold storage, quindi "offline", ad esempio nel Ledger Nano S. Con Lightning Network, questo non può accadere e c'è la necessità di soluzioni avanzate di sicurezza informatica, che siano in grado di proteggere i fondi dai black hackers. Tali soluzioni verranno anche dallo sviluppo tecnologico, ma soprattutto dalle definizioni dei processi interni.

Soluzioni di sicurezza informatica sono necessarie anche per prevenire bug nel software, dato che, se presenti, la controparte potrebbe approfittarne, rubando fondi dal canale. [29] [30] [31]

Figura 49 – Numero transazioni confermate al giorno in blockchain attualmente – Blockchain.info

[29] Bitcoin: from digital gold to electronic cash with Lightning Network – PwC -2018

[30] Blockchain Technology Explained – Alan T. Norman – 2017

[31] The Business Blockchain: promise, practise and application of the next Internet technology – William Mougayar – 2016

3. Smart Contract

L'inizio del XXI secolo è stato caratterizzato da molteplici tecnologie innovative che hanno avuto un impatto sostanziale sulla nuova economia basata sui dati; tra queste le più importanti sono: Cloud Computing, Big Data, Internet of Things e blockchain.

Quest'ultima tecnologia, inizialmente introdotta come spina dorsale tecnologica del Bitcoin, ha iniziato ad avere un significato a sé stante. I governi e le aziende di tutto il mondo sono però per lo più perplessi sulla possibile implementazione della blockchain in molte aree della vita, non associate all'uso della criptovaluta.

Una delle aree più promettenti di applicazione della blockchain è la creazione di contratti completamente automatizzati cioè accordi che vengono eseguiti senza il coinvolgimento umano. Questi, nell'ambiente IT, sono chiamati “contratti intelligenti" o “smart contracts”.

Non esiste una definizione universalmente condivisa di contratto "smart"; ciò è dovuto sia alla sua natura innovativa, sia alla sua complessa base tecnologica.

Secondo la definizione più semplice, il contratto smart è un contratto la cui esecuzione è automatizzata.

Per Nick Szabo, uno dei pionieri nell'analisi degli accordi automatizzati auto-esecutivi, lo smart contract è un algoritmo di transazione computerizzato, che esegue i termini del contratto.

Tuttavia, questa definizione può difficilmente cogliere la differenza dei contratti smart rispetto ad alcuni già noti costrutti contrattuali che implementano l'esecuzione automatizzata, ad esempio i distributori automatici. Quest’ultimi sono definiti come macchine automatiche autonome che erogano beni o forniscono servizi quando si inseriscono monete o si effettua il pagamento in altre forme (e-cash, carta di credito). I distributori automatici sono programmati con determinate regole che potrebbero essere definite in un contratto e funzionano sulla base di queste.

Il grado di novità e la presenza di alcune caratteristiche speciali nei contratti intelligenti diventa particolarmente rilevante negli exchange market, dove i sistemi di trading automatizzati sono ampiamente utilizzati. Ad esempio, negli exchange markets, le operazioni di trading vengono spesso eseguite non dal trader, ma da un sistema informatico basato su una strategia di trading calcolata grazie ad un algoritmo.

A partire dal 2014, oltre il 75% delle azioni scambiate nelle borse degli Stati Uniti proviene da ordini di sistemi di trading automatizzati, quindi, i contratti automatizzati, di per sé non sono una novità.

E’ opportuno fare riferimento ad un'altra definizione di smart contract fornita da Gideon Greenspan: "Un contratto smart è un pezzo di codice che viene memorizzato su una blockchain, attivato da transazioni su blockchain e che legge e scrive i dati nel database di quella blockchain". Questa definizione è più concreta, poiché pone l'accento sulla blockchain come una delle caratteristiche fondamentali del contratto intelligente.

La blockchain può essere considerata un "cambio di paradigma" in ambito contrattuale perché permette di automatizzare il processo di esecuzione del contratto di entrambe le parti. I distributori automatici automatizzano le prestazioni di una sola parte, richiedendo almeno un certo coinvolgimento personale dall'altra parte (ad es. l'inserimento di monete o di una carta di credito).

Quando la prestazione di entrambe le parti può essere completamente automatizzata, sorge il dubbio se esista ancora un contratto in senso giuridico o non invece un altro tipo di fenomeno.

Il teorico organizzativo Arthur Stinchcombe una volta scrisse che i contratti sono solo organizzazioni in miniatura e, per estensione, tutte le organizzazioni sono solo complessi di contratti. Le aziende sono create utilizzando una serie di accordi contrattuali, che vanno dai contratti di lavoro per i dipendenti, ai rapporti con i fornitori, agli obblighi verso i propri clienti, ai contratti di locazione, alla vendita e all'acquisto di attrezzature. Tradizionalmente, questi obblighi contrattuali sono piuttosto costosi perché devono essere fatti rispettare grazie un sistema legale affidabile e attraverso l'applicazione della legge.

Con uno smart contract basato su una blockchain, tuttavia, gran parte di questi costi sono notevolmente ridotti o eliminati. Questo permette di rendere le organizzazioni basate su blockchain più efficienti, convenienti e competitive rispetto alle aziende tradizionali sul mercato. Tutto ciò dimostra che gli smart contracts vanno ben oltre i modelli esistenti di contrattazione e rappresentano un nuovo paradigma di interazione in un cyberspazio. Gli smart contracts consentono di creare pool di risorse e di ripartirle secondo criteri concordati, cosa che può essere particolarmente rilevante per le attività di crowdfunding o per i contratti di carattere assicurativo.

Nel primo caso, gli smart contracts possono tenere traccia dell'importo dei fondi appartenenti ad un progetto di crowdfunding e, una volta che viene raggiunto l'importo necessario, lo stesso viene trasferito al beneficiario. In caso contrario, i fondi vengono restituiti ai donatori.

Nel secondo caso, un gruppo di agricoltori potrebbe mettere insieme un pool di risorse come assicurazione contro la siccità, le inondazioni o altri disastri naturali. Una volta che si verificasse un tale disastro, il contratto lo accerterebbe secondo la procedura specificata nello stesso (ad esempio controllando il meteo o le notizie in fonti predesignate) e assegnerebbe le risorse. Inutile dire che lo smart contract fornisce il massimo grado di trasparenza e di verificabilità, attenuando i rischi associati al processo decisionale dell'intermediario e al "fattore umano", nonché ai ritardi temporali.

Sebbene l'esecuzione dello smart contract sia automatizzata, essa richiede comunque la presenza della volontà della parte contraente per diventare efficace. La persona esprime il suo consenso ai termini del contratto e alle modalità della sua esecuzione al momento della conclusione dello stesso. Tenendo conto del fatto che tale persona non sarà in grado di influenzare l'esecuzione del contratto una volta stipulato, dovrebbe esserci una certa fiducia, che dà luogo ad una sorta di rapporto "fiduciario" nel contratto intelligente. Ma a differenza del contratto classico, in cui la fiducia è riposta nella controparte, negli smart contracts tale fiducia è riposta nell'algoritmo informatico che sta alla base del contratto ("fiducia senza fiducia").

Sulla base dell'attuale comprensione degli smart contracts è possibile delineare le seguenti loro caratteristiche: 1) natura esclusivamente elettronica 2) implementazione di software 3) maggiore certezza 4) natura condizionale 5) autoapplicazione 6) autosufficienza.

1. Natura esclusivamente elettronica: i contratti classici possono esistere in varie forme, per esempio in forma orale o scritta. Naturalmente, lo sviluppo del e-commerce ha aumentato notevolmente la quantità di accordi conclusi in forma elettronica, i cui esempi più evidenti sono i vari accordi "clickwrap" (Un accordo di clickwrap è un prompt digitale che offre agli individui l'opportunità di accettare o rifiutare

una politica digitally-mediated). Tuttavia, anche nel caso di contratti di e-commerce, possono essere ancora necessari alcuni classici documenti cartacei, come ad esempio fatture, ricevute o certificati di consegna, soprattutto quando tali contratti elettronici coprono l'acquisto di beni o servizi offline. Talvolta, questi documenti sono l'unica prova o manifestazione del contratto esistente in forma elettronica.

Al contrario, gli smart contracts possono esistere solo in forma elettronica e non è possibile utilizzare qualsiasi altra forma di contratto. Esso è inoltre caratterizzato da uno specifico oggetto che può essere un bene digitale (ad esempio una criptovaluta) o una manifestazione digitale di un bene offline registrato su blockchain. Questo diversifica lo smart contract dalla maggior parte dei contratti "clickwrap", che esistono anche in forma elettronica, ma impongono solo alcuni obblighi negativi all'utente (ad esempio, non eseguire determinate attività durante l'utilizzo del servizio o non opporsi a determinate attività svolte dal fornitore dello stesso). Inoltre, lo smart contract per sua natura richiede l'utilizzo di firme digitali elettroniche, basate sulla crittografia.

2. Implementazione di software: lo smart contract ha una duplice natura giuridica; funge da "documento" che disciplina i rapporti contrattuali tra le parti ed è anche oggetto dei diritti di proprietà intellettuale. Pertanto, la programmazione di alcuni contratti intelligenti in base alle esigenze del cliente può essere trattata come un processo di sviluppo software, mentre la distribuzione dei diritti successivi allo smart contract deve essere effettuata nell'ambito della licenza/assegnazione dei diritti di proprietà intellettuale.
3. Maggiore certezza: poiché il codice del software rappresenta l’essenza del contratto intelligente, i termini di quest’ultimo sono espressi in uno dei linguaggi informatici caratterizzati da una semantica e una sintassi rigorosamente definite. Il linguaggio informatico non consente la discrezione nella sua interpretazione. I termini contrattuali intelligenti sono interpretati da un computer basato sulla logica booleana, a differenza del contratto classico, dove l'interpretazione dei termini è effettuata dal cervello umano sulla base di criteri soggettivi. Pertanto, la precisione dei linguaggi di programmazione è in grado di mitigare le possibili problematiche associate all'interpretazione imprevedibile dei termini contrattuali da parte del soggetto contraente. Di conseguenza le regole esistenti sull'interpretazione del contratto non si applicano allo smart contract. I contratti intelligenti sono da intendersi come accordi a sé stanti, non soggetti a interpretazione da parte di entità o giurisdizioni esterne. Il codice stesso è inteso come l'arbitro finale dell'"accordo" che rappresenta.

   Considerate le complessità tecniche e la necessità di possedere competenze di programmazione avanzate, in molti casi gli smart contracts saranno creati da aziende specializzate sulla base della richiesta del cliente. A causa della separazione tra chi programma il codice e chi intende utilizzarlo nelle sue attività commerciali, c'è il rischio di fraintendimenti sui termini del futuro accordo.

   Inoltre, poiché è solo il codice informatico che regola il contratto smart, quest'ultimo diventa automaticamente soggetto a vari difetti e bug. Il recente attacco hacker a uno dei contratti intelligenti della piattaforma Ethereum ne è un ottimo esempio: nel giugno 2016 gli aggressori hanno sfruttato una

vulnerabilità del software e prosciugato milioni di ether. In una lettera aperta alla comunità Ethereum, l'aggressore ha affermato di non aver fatto nulla di illegale, ma di "utilizzare questa funzione esplicitamente codificata secondo i termini del contratto intelligente".

Pertanto, è possibile affermare che i contratti intelligenti sono ancora vulnerabili agli errori di codifica, cosa che, probabilmente, deve essere affrontata dalle nuove regole di interpretazione di tali contratti.

4. Natura condizionale: in precedenza si è sostenuto che il contratto intelligente è redatto in uno dei linguaggi di programmazione. Le affermazioni condizionali sono fondamentali per l'informatica: il codice del computer si basa su affermazioni come “se x allora y" che permettono l’esecuzione del contratto.
5. Autoapplicazione: una volta concluso lo smart contract, la sua esecuzione non dipende più dalla volontà delle parti o di terzi, né richiede ulteriori approvazioni o azioni da parte loro. Il computer verifica tutte le condizioni, trasferisce i beni e inserisce in blockchain i dati relativi a tali trasferimenti. Pertanto, il contratto intelligente è tecnicamente vincolante per tutte le parti, non dipendendo più dall'intermediario umano, che è soggetto a errori e discrezione soggettiva.
6. Autosufficienza: lo smart contract non ha bisogno di istituzioni giuridiche per esistere, né del corpus di norme giuridiche, come avviene per i contratti classici. L'autosufficienza è particolarmente importante nelle transazioni internazionali, in quanto rende irrilevanti le differenze linguistiche, le leggi nazionali e la loro interpretazione: le stesse regole sono applicabili in tutto il mondo. Sulla base di queste caratteristiche, è possibile definire il contratto intelligente come un codice software, implementato su piattaforma blockchain, che garantisce l'autosufficienza e l'autonomia dei suoi termini basati su condizioni definite in anticipo e applicate ai beni che stanno su blockchain. Tra i benefici dei contratti intelligenti è possibile annoverare la loro capacità di ridurre molti dei costi di transazione che accompagnano i normali contratti.

   Anche i costi associati al coinvolgimento dell'intermediario nell'esecuzione del contratto (ad es. banca o organizzazione assicurativa), sono esclusi negli smart contracts a causa della loro natura disintermediante. Tuttavia, non sarebbe corretto concludere che i contratti intelligenti siano più economici di quelli regolari: l'infrastruttura necessaria per la loro attuazione e i costi associati allo sviluppo ("redazione") dei loro termini sono ancora piuttosto elevati.

   Sono già emerse piattaforme di contratti intelligenti che hanno guadagnato popolarità e riconoscimento. L'esempio più evidente è Ethereum, che è una piattaforma informatica open source basata su blockchain, con funzionalità di creazione di smart contracts. A differenza dell'ecosistema Bitcoin, che non permette lo scambio di nessun altro oggetto che non sia la criptovaluta stessa, Ethereum permette lo scambio di qualsiasi classe di asset che si possa trasferire su Internet. [32]

[32] Contract Law 2.0: “Smart” Contracts as the beginning of the end of the classic contract law – Alexander Savelyev – 2016

3.1. Dove si scrivono gli smart contracts: Ethereum

Ethereum, piattaforma informatica usata per creare smart contracts, viene spesso soprannominata "il computer del mondo".

Dal punto di vista informatico, Ethereum è una state-machine (cioè permette di descrivere con precisione e in maniera formale il comportamento di molti sistemi) con due funzioni base: la prima è uno stato singleton accessibile a livello globale e la seconda è una macchina virtuale che applica cambiamenti a quello stato.

Ethereum è stato creato nel novembre 2013 come piattaforma generale per la creazione di blockchain, combinando insieme la nozione di consenso economico attraverso la proof of work (o eventualmente la proof of stake) con un linguaggio Turing completo; in questo modo si voleva sia consentire agli sviluppatori di implementare molto più facilmente applicazioni decentralizzate, sia evitare di creare una nuova blockchain per ogni nuova applicazione.

Mentre i precedenti protocolli potevano essere visti come strumenti che risolvevano una singola funzione o al massimo come strumenti multifunzione, Ethereum è lo "smartphone" delle blockchain: una piattaforma universale dove, qualsiasi cosa si voglia costruire, si può semplicemente progettare come "app" decentralizzata.

Lo scopo di Ethereum non è quello di creare un nuovo metodo di pagamento come Bitcoin. La sua criptocurrency, l'ether, è necessaria principalmente per il funzionamento della piattaforma.

A differenza di Bitcoin, che ha un linguaggio di scripting molto limitato, Ethereum è progettato per essere una blockchain programmabile di uso generale che esegue una macchina virtuale in grado di "runnare" codici di varia complessità senza limiti. [33]

3.1.1. La nascita di Ethereum

Tutte le grandi innovazioni risolvono problemi reali ed Ethereum non fa eccezione. Esso è stato concepito in un'epoca in cui le persone pur riconoscendo la potenza di Bitcoin, cercavano di andare oltre per ottenere qualcosa di ancora più disruptive, sfruttando le potenzialità della blockchain.

Gli sviluppatori si trovarono di fronte ad un bivio: o costruivano sopra il Bitcoin o creavano una nuova blockchain. Costruire sopra Bitcoin significava convivere con i vincoli della rete e con le dimensioni ridotte dello storage dei dati, il che limitava la creazione di applicazioni con Bitcoin.

Per i progetti che necessitavano di maggiore libertà e flessibilità, l'unica opzione era quella di avviare una nuova blockchain.

Verso la fine del 2013, Vitalik Buterin, un giovane programmatore, appassionato di Bitcoin, ha iniziato a studiare il modo per estendere ulteriormente le capacità di programmazione con Bitcoin e Mastercoin (un

[33] Ethereum White Paper: a next generation smart contract and decentralized application platform – Vitalik Buterin – 2013

protocollo di overlay che ha esteso Bitcoin per offrire contratti intelligenti rudimentali). Nell'ottobre del 2013, Buterin ha proposto al team creatore di Mastercoin un update della piattaforma, che permetteva di creare contratti flessibili e scriptabili (ma non Turing complete) che avrebbero sostituito il linguaggio dei contratti di Mastercoin. Nonostante il team fosse rimasto impressionato da tale proposta, la stessa rappresentava un cambiamento troppo radicale per la loro tabella di marcia.

Nel dicembre 2013, Buterin ha iniziato a condividere un whitepaper che delineava l'idea alla base di Ethereum: una blockchain programmabile e general purpose Turing completa. Poche decine di persone hanno visto questa prima bozza e hanno dato un feedback positivo a Buterin, aiutandolo a sviluppare gradualmente il progetto.

A partire da dicembre 2013, Buterin e Gavin Wood (programmatore di fama mondiale) hanno affinato e perfezionato l'idea, costruendo insieme il progetto iniziale che è diventato in seguito la piattaforma Ethereum. I fondatori pensavano ad una blockchain che non mirasse ad uno scopo specifico, ma che invece potesse supportare un'ampia varietà di applicazioni. L'idea era che, utilizzando una blockchain generica come Ethereum, uno sviluppatore potesse programmare la sua applicazione senza dover avviare i meccanismi sottostanti delle reti peer-to-peer, blockchain, algoritmi di consenso ecc. Ethereum è stato progettato per astrarre questi dettagli e fornire un ambiente di programmazione deterministico e sicuro per le applicazioni decentralizzate su blockchain.

Proprio come Nakamoto, Buterin e Wood non si sono limitati a inventare un nuovo protocollo, ma hanno combinato le nuove invenzioni con le tecnologie già esistenti e hanno lanciato il codice sorgente per dimostrare al mondo le loro idee.

I fondatori hanno lavorato per anni, costruendo e perfezionando il progetto.

Il 30 luglio 2015 è stato estratto il primo blocco di Ethereum. Il computer del mondo ha iniziato a servire l’intero globo terrestre.

3.1.2. Un linguaggio Turing completo

Non appena si inizia a leggere di Ethereum, si sente subito parlare di linguaggio "Turing Complete" (differentemente da Bitcoin). Il termine "Turing Complete" prende il nome dal matematico inglese Alan Turing, considerato il padre dell'informatica. Nel 1936 egli creò un modello matematico applicabile su macchina costituito da un programma che manipola i simboli, leggendoli e scrivendoli su memoria sequenziale (simile a un nastro magnetico a lunghezza infinita). Con questa costruzione, Alan Turing ha fornito una base matematica per rispondere (in negativo) alle domande sulla computabilità universale, cioè se tutti i problemi siano risolvibili. Egli ha dimostrato che ci sono classi di problemi che sono irrisolvibili anche con i computer; in particolare ha dimostrato che l’Halting Problem (problema che consiste nel valutare se un programma prima o poi smetterà di funzionare) non è risolvibile.

Alan Turing ha definito un sistema Turing Complete, se può essere usato per simulare qualsiasi macchina di Turing. Tale sistema è chiamato Universal Turing Machine (UTM).

Ethereum ha la capacità sia di eseguire un programma memorizzato in una state-machine chiamata Ethereum Virtual Machine, sia di leggere e scrivere i dati in memoria, il che ne fa un sistema Turing Completo e quindi una UTM.

L'innovazione rivoluzionaria di Ethereum è quella di combinare l'architettura di calcolo generale di una macchina in un programma memorizzato su blockchain, creando così un computer mondiale distribuito a stato singolo (singleton). I programmi di Ethereum funzionano "ovunque", ma producono uno stato comune che è garantito dalle regole del consenso.

Il fatto che Ethereum sia Turing Complete significa che qualsiasi programma di qualsiasi complessità può essere calcolato in Ethereum, ma questa flessibilità comporta alcuni spinosi problemi di sicurezza e di gestione delle risorse.

Turing ha dimostrato che non è possibile prevedere se un programma terminerà, simulandolo su un computer. In parole povere, non possiamo prevedere il percorso di un programma senza eseguirlo. I sistemi Turing completi possono essere eseguiti in "cicli infiniti", un termine usato per descrivere un programma che non termina mai. Ma i loop non voluti e senza fine possono sorgere senza preavviso, a causa di complesse interazioni tra le condizioni di partenza e il codice. In Ethereum questo pone una sfida: il nodo (client), sempre partecipante, deve convalidare ogni transazione, eseguendo ogni contratto intelligente che viene inserito in blockchain. Ma, come ha dimostrato Turing, Ethereum non può prevedere se un contratto smart terminerà, o per quanto tempo durerà, senza che venga effettivamente eseguito.

Naturalmente, tra un programma che richiede un millisecondo per essere validato e uno che funziona per sempre, c'è un'infinita gamma di programmi cattivi, che monopolizzano le risorse, che fanno fluttuare la memoria, che surriscaldano la CPU e che quindi sprecano risorse.

Come fa Ethereum a limitare le risorse utilizzate da un contratto intelligente se non è in grado di prevedere in anticipo la loro quantità necessaria?

Per rispondere a questa sfida, Ethereum introduce un meccanismo di misurazione chiamato gas. Poiché l'EVM esegue un contratto intelligente, tiene conto accuratamente di ogni istruzione (calcolo, accesso ai dati, ecc.). Ogni istruzione ha un costo predeterminato in unità di gas. Quando una transazione fa scattare l'esecuzione di un contratto smart, deve includere una quantità di gas che stabilisce il limite superiore di calcolo che può essere consumato in esecuzione del contratto smart. L'EVM terminerà l'esecuzione se la quantità di gas consumata dal calcolo supera il gas disponibile nella transazione.

Il gas è il meccanismo che Ethereum utilizza per consentire il calcolo, limitando al contempo le risorse che qualsiasi programma può consumare. [34]

[34] Making Smart Contracts Smarter – Loi Luu, Duc Hiep Chu, Hrishi Olickel, Prateek Saxena, Aquinas Hobor - 2016

3.1.3. Blockchain di Ethereum

La blockchain di Etereum è molto simile a quella di Bitcoin, ma se ne distingue per l'uso più esteso degli alberi di Merkle. Questa novità permette l'esistenza di "client leggeri" che scaricano e verificano solo le intestazioni dei blocchi e che possono comunque determinare e verificare in modo sicuro qualsiasi parte specifica dello stato della blockchain. Mentre in Bitcoin un'intestazione di blocco contiene solo l'hash del Merkle root di un albero di Merkle, in Ethereum ogni intestazione del blocco contiene una "radice di stato", cioè essenzialmente un hash root di un albero di hash crittografico contenente l'intero stato corrente, compresi i saldi dei conti, i numeri di sequenza, il codice e la memorizzazione.

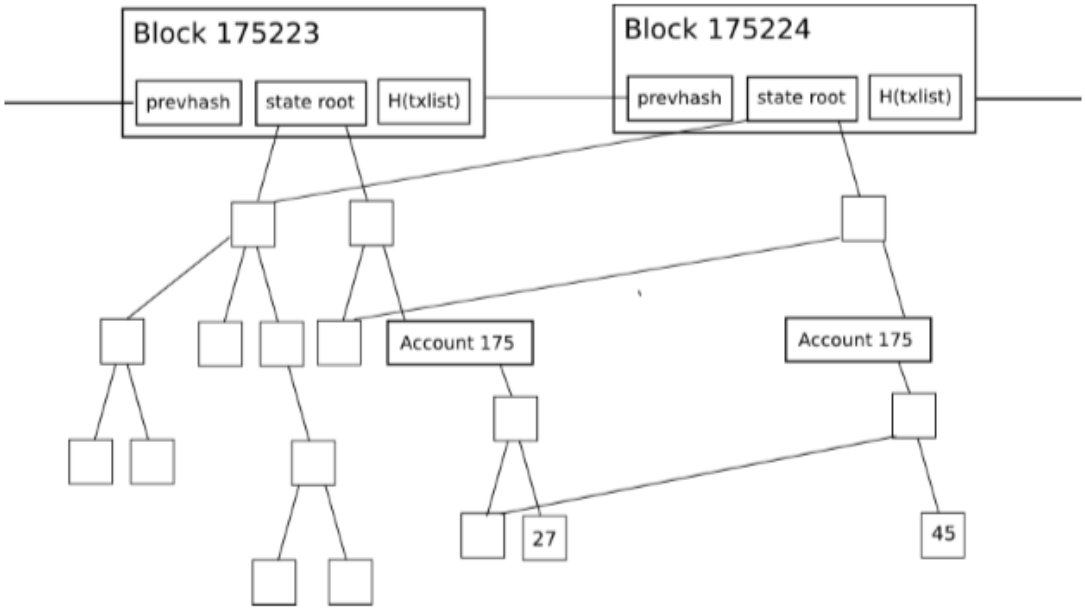


Figura 50 – La blockchain di Ethereum – Ethereum Platform Review. Opportunities and challenges for private and consortium blockchains

Un client light scaricherà quindi normalmente solo le intestazioni dei blocchi e ove voglia imparare qualche valore specifico nello stato (ad es. "qual è il saldo del conto 0x124b6f72?" o "qual è la memorizzazione del conto 0x38c9f1e5 alla chiave 178233? ") può chiedere a qualsiasi "nodo completo" della rete di fornirgli un "ramo", cioè una serie di hash dal blocco di dati che specifica quella particolare informazione; così il light client può da solo verificare l'integrità del ramo. Se il ramo è corretto, accetta la risposta. Il modello "light client" è utile in particolare per gli utenti di smartphone e IoT/embedded di Ethereum, così come per gli utenti con computer di bassa qualità o con connessioni internet lente o scadenti.

Affinché un nodo faccia parte della rete, deve connettersi ad altri nodi della stessa in modo da poter trasmettere transazioni/blocchi.

Un nodo non ha bisogno di connettersi ad ogni nodo della rete; al contrario, esso si connette a pochi altri nodi. Ma come fa un nodo a trovare altri nodi nella rete, dato che non c'è un server centrale al quale tutti possano connettersi? Ethereum ha un proprio protocollo di scoperta dei nodi, chiamato Kadelima. In esso si ha un tipo speciale di nodi chiamati nodi Bootstrap. I Bootstrap mantengono una lista di tutti i nodi che sono collegati ad essi per un certo periodo di tempo. Quando i peers si collegano alla rete di Ethereum, si collegano prima ai

nodi Bootstrap, che condividono le liste dei peers che si sono collegati ad essi nell'ultimo periodo e gli permettono la connessione ad altri nodi. [35]

3.1.4. Ethereum Virtual Machine

La Macchina Virtuale dell'Etereum (Ethereum Virtual Machine - EVM) è un computer mondiale che chiunque può utilizzare, a fronte di una piccola tassa, pagabile in ether.
L'EVM è un unico "computer" globale a 256 bit in cui tutte le transazioni sono locali su ogni nodo della rete ed eseguite in relativa sincronia. Questo gigantesco computer, al quale può accedere chiunque abbia un nodo o un'applicazione, rende semplice spostare grandi quantità di denaro quasi istantaneamente. Anche se chiunque può usare questa macchina virtuale globale, nessuno può creare denaro falso al suo interno o spostare fondi senza autorizzazione.
L'EVM può eseguire programmi scritti nel linguaggio di programmazione Solidity. Questi programmi, dato un particolare input, produrranno sempre quel dato output, con gli stessi cambiamenti di stato sottostanti. Questo rende i programmi Solidity completamente deterministici e garantiti, a condizione che si sia pagato abbastanza gas per la transazione. I programmi Solidity sono in grado di esprimere tutti i compiti realizzabili dai computer, rendendoli teoricamente Turing completi. Quando un utente carica un contratto intelligente attraverso il suo nodo Ethereum, questo viene incluso nell'ultimo blocco e propagato in rete, dove viene memorizzato in ogni altro nodo della rete.
L'EVM è totalmente sandboxed, libero da interferenze e isolato anche dalle altre reti; ciò rende impossibile ad una parte rescindere un contratto intelligente. In termini pratici, questo è dovuto al fatto che i contratti smart hanno il potere di tenere i beni (ether o altri gettoni) in deposito a garanzia e di spostarli solo quando i termini del contratto sono soddisfatti.
L'EVM fa run continuamente ad un loop che tenta di eseguire qualsiasi istruzione ci sia nel program counter. Il program counter funziona come una coda in un negozio: ogni programma prende un numero e aspetta il suo turno. Questo loop deve compiere alcune attività: calcola il costo del gas per ogni istruzione e usa la memoria, se necessario, per eseguire la transazione. Questo ciclo si ripete fino a quando la VM non finisce di eseguire tutto il codice, oppure si trova un errore e quella transazione viene riportata indietro.
Le state-machines possono essere considerate come meccanismi che non dormono mai. Come state-machine, l'EVM ha una storia costante di tutte le transazioni all'interno dei propri banchi di memoria.
A differenza delle persone, che hanno una memoria imperfetta, la state-machine è il risultato specifico di ogni singolo cambio di stato che ha avuto luogo all'interno di quella macchina da quando è stata accesa per la prima volta. L'ultima versione dello stato della macchina si può dire che è la "verità" di quest'ultima. In Ethereum, questa verità riguarda i saldi dei conti e la serie di transazioni che hanno portato ad essi.

[35] Ethereum: Platform Review. Opportunities and challenges for private and consortium blockchains – Vitalik Buterin – 2016

In Ethereum, il tempo di blocco non è una funzione del programma di emissione dell'ether come in Bitcoin. Al contrario, il tempo di blocco è una variabile che viene mantenuta il più basso possibile, per una rapida conferma della transazione.
Ogni nodo esegue le transazioni e memorizza lo stato finale per vari motivi. Ad esempio, se esiste un contratto intelligente che memorizza i nomi e i dettagli di tutti i partecipanti ad un progetto, ogni volta che viene aggiunta una nuova persona, una nuova transazione viene trasmessa alla rete. Affinché un qualsiasi nodo possa visualizzare i dettagli di tutti i partecipanti al progetto, è sufficiente che legga lo stato finale del contratto. Ogni transazione richiede un po' di calcolo e di memorizzazione nella rete. Pertanto, ci deve essere un costo di transazione, altrimenti l'intero network sarebbe inondato di transazioni spam e i minatori non avrebbero motivo di includere transazioni ma inizierebbero a estrarre blocchi vuoti. Ogni transazione richiede una quantità diversa di calcolo e di memorizzazione; pertanto, ogni transazione ha costi diversi. [36]

3.1.5. Gas

Il gas è un'unità di lavoro utilizzata per misurare il costo computazionale di un'operazione su Ethereum. I costi del gas sono pagati con piccole quantità di ether.
Lo scopo del gas è duplice. In primo luogo, garantisce una ricompensa predefinita per i minatori che eseguono il codice e mettono in sicurezza la rete. In secondo luogo, aggira il problema dell'interruzione e assicura che l'esecuzione non possa durare più a lungo del tempo che è stato pagato in anticipo. Il gas è un'unità di lavoro che non si può tenere o accumulare. Semplicemente misura quanto sforzo comporterà ogni fase di una transazione, in termini computazionali. Per poter pagare i costi del gas, è sufficiente aggiungere ether al proprio conto e non è necessario acquistarlo separatamente. Ogni operazione possibile sull'EVM ha un costo di gas associato.
I costi del gas assicurano che il tempo di calcolo sulla rete abbia un prezzo adeguato. Questo funziona in modo diverso in Bitcoin, dove la tariffa è basata sulla dimensione della transazione in kilobyte. Poiché il codice di Solidity può essere arbitrariamente complesso, un breve frammento di istruzioni potrebbe generare molto lavoro di calcolo, mentre un lungo frammento potrebbe generarne meno. Questo è il motivo per cui le tariffe nell'EVM si basano sulla quantità di lavoro svolto e non sulla dimensione della transazione.
Il gas non è prezzato in ether perchè quest'ultimo, essendo scambiato pubblicamente sugli exchange di criptovalute, è soggetto a momenti speculativi. L'utilizzo dell'unità di conto del gas per il lavoro di calcolo è utile perché separa il prezzo del calcolo dal prezzo altamente volatile dell'ether.
Se si invia all'EVM un insieme di istruzioni difficili da calcolare, il lavoro spenderà l'ether assegnato alla transazione e si fermerà quando quest'ultimo si sarà esaurito. Non avrà alcun effetto sulle transazioni altrui. La scalabilità viene gestita di fatto attraverso il sistema delle tariffe del gas. I minatori sono liberi di scegliere

[36] Building Blockchain Projects – Narayan Prusty – 2017

le transazioni che pagano le tariffe più alte e possono anche scegliere il limite di blocco del gas collettivamente. Il limite del gas determina la quantità di calcolo che può avvenire per blocco.
I costi di transazione influenzano la quantità di ether che un conto può trasferire su un altro conto. Ad esempio, se un conto ha un saldo di cinque ether, non può trasferire tutti e cinque su un altro conto perché così facendo non rimarrebbe un saldo da cui dedurre le spese di transazione.

3.1.6. Account

Esistono due tipi di account in Ethereum: Externally owned accounts (EOA) e Contract accounts.
Un conto di proprietà esterna (EOA) è controllato da un paio di chiavi private, che possono essere detenute da una persona o da un server esterno. Infatti, gli EOA sono controllati dagli utenti, spesso tramite un software esterno alla piattaforma Ethereum. Questi conti non possono contenere il codice EVM. Un EOA è caratterizzato dalle seguenti proprietà: - contiene un saldo in ether – è capace di inviare transazioni – è controllato dalle chiavi private del conto - non ha alcun codice associato ad esso – contiene un database chiavi/valori contenuto in ogni conto, dove le chiavi e i valori sono entrambi stringhe di 32 byte.
I Contract accounts non sono controllati dagli esseri umani, ma dagli smart contracts che vengono eseguiti dalla EVM. Essi hanno le seguenti caratteristiche: - hanno un saldo in ether - tengono in memoria qualche codice di contratto - possono essere attivati da esseri umani (invio di una transazione) o da altri contratti che inviano un messaggio - quando vengono attivati, possono eseguire operazioni complesse - hanno un proprio stato persistente e possono chiamare altri contratti - non hanno un proprietario dopo essere stati rilasciati sull'EVM – contengono un database di chiavi/valori contenuto in ogni conto, dove le chiavi e i valori sono entrambi stringhe di 32 byte.
In breve, gli EOA sono conti semplici senza alcun codice associato o memorizzazione dei dati, mentre i Contract accounts hanno sia il codice associato che la memorizzazione dei dati. Gli EOA sono controllati da transazioni create e firmate crittograficamente con una chiave privata nel "mondo reale" esterno e indipendente dal protocollo, mentre i Contract accounts non hanno chiavi private e quindi "si controllano" nel modo predeterminato prescritto dal loro contratto intelligente. Entrambi i tipi di conti sono identificati da un indirizzo Ethereum.

3.1.7. Solidity

Solidity è un nuovo linguaggio di programmazione utilizzato per scrivere i contratti intelligenti, che possono essere eseguiti dall'EVM. Esso è un insieme di convenzioni di rete, linguaggio di assemblaggio e sviluppo web. Solidity è orientato verso i contratti, con somiglianze con JavaScript e C. Permette di sviluppare contratti e di compilare il bytecode EVM e attualmente è il linguaggio di punta di Ethereum. Ci sono quattro linguaggi

nel protocollo Ethereum allo stesso livello, ma la comunità ha preferito Solidity, che ha messo fuori gioco Serpent (simile a Python), Lisp-Like Language (LLL), e Mutan. [37]

3.1.8. Mining Ethereum

Il mining è il processo attraverso il quale la rete di Ethereum raggiunge il consenso sull'ordine delle transazioni in un determinato periodo di tempo.
Anche Bitcoin usa il mining per raggiungere il consenso, ma il modo in cui le cose funzionano su Ethereum è un po' diverso, grazie alla sua capacità di eseguire contratti intelligenti.
L'ether viene creato dal nulla durante il processo di estrazione, come pagamento per il lavoro minerario svolto dai computer. Poiché il mining è computazionalmente impegnativo, può generare grandi costi di elettricità.
Ethereum sta passando da una proof of work ad una proof of stake. L'algoritmo di proof of work per il protocollo Ethereum era Ethash, una funzione creata dagli sviluppatori per affrontare il problema della centralizzazione mineraria presente in Bitcoin.
Mentre la data della proof of stake di Ethereum era stata originariamente fissata per gennaio 2020, questa scadenza non è stata rispettata e non è chiaro quando il PoS di Ethereum verrà lanciato.
La formula della PoW per la difficoltà di blocco utilizzava una soglia di 10 secondi differentemente da Bitcoin in cui in media venivano impiegati 10 minuti.
La PoS, d'altra parte, è un altro modo di convalidare le transazioni che funziona in modo diverso dalla PoW. A differenza dei minatori, i validatori delle transazioni bloccano o mettono in gioco la loro quantità di criptovaluta come garanzia per il diritto di verificare le transazioni.
A seconda della rete, alcuni fattori, come ad esempio il numero di monete e la durata della puntata, determinano se un validatore può verificare o meno un nuovo blocco di transazioni (in contrapposizione all'hashing della PoW). Come nella PoW, se si convalida un nuovo blocco, si viene ricompensati con dei nuovi ether.
Uno dei principali limiti del modello PoW è la quantità di energia necessaria per alimentare tutto l'hardware che viene utilizzato in tutto il mondo per estrarre le più famose criptovalute come Bitcoin ed Ethereum.
Dal momento che la Proof of Stake non richiede ai minatori di usare molta energia per convalidare le transazioni, molti pensano che un sistema di questo tipo sarebbe migliore per l'ambiente, dato che le reti crittografiche probabilmente diventeranno sempre più grandi e consumeranno sempre più energia.
Casper è il nome dell'implementazione di Ethereum che lo trasformerà in una blockchain PoS (Ethereum 2.0). Il passaggio di Ethereum da 1.0 a 2.0 (noto anche come aggiornamento "Serenity") avverrà in 3 fasi separate. Attualmente ci sono 2 implementazioni Casper che sono già state introdotte nella comunità di Ethereum: Casper Correct-by-Construction (CBC) e Casper Friendly Finality Gadget (FFG). Vitalik Buterin sta

[37] Introducing Ethereum and Solidity – Chris Dannen – 2017

conducendo una ricerca su Casper FFG, che è l'implementazione che alimenterà la prima fase di Ethereum 2.0.

Dato che Ethereum sta passando alla PoS e si sta liberando dell'attività mineraria, i minatori, che hanno accumulato attrezzature minerarie nel corso degli anni, probabilmente non smetteranno di estrarre ma porteranno il loro mining power in una blockchain diversa, provocando l'aumento dell'hash rate di altre reti.

Oppure, se vorranno rimanere parte dell'ecosistema di Ethereum, venderanno le loro ASIC per il mining per accumulare ether e partecipare alla Proof of Stake.

In entrambi i casi, i minatori avranno il tempo di decidere la migliore linea d'azione, poiché il passaggio alla Proof of Stake non avverrà da un giorno all'altro. Infatti, l'attuale versione di Casper propone di utilizzare la PoS su ogni 100° blocco che viene convalidato, il che significa che l'aggiornamento di Ethereum 2.0 sarà probabilmente un blocco ibrido PoW/PoS fino a quando tutti i nodi del PoS non saranno stati pronti. [38]

3.2. Oltre gli smart contract: altri usi di Ethereum

Oltre alla realizzazione di smart contract, Ethereum consente la creazione di blockchain, dApps e monete digitali.

3.2.1. DApp

Quasi tutte le applicazioni basate su Internet sono centralizzate, cioè i loro server sono di proprietà di una particolare azienda o persona. Esse tuttavia presentano alcuni limiti come ad esempio il fatto che sono meno trasparenti, o che hanno un unico punto di fallimento, o che non riescono a prevenire la censura della rete. Per cercare di risolvere questi problemi, è emersa una nuova tecnologia per la costruzione di applicazioni basate su Internet chiamate applicazioni decentralizzate (dApp).

Ethereum è nata come piattaforma atta a creare diverse blockchain e tale da poter essere programmata per una varietà di usi. Ma molto rapidamente, la visione di Ethereum si sviluppò fino a diventare una piattaforma per la programmazione di applicazioni decentralizzate (dApp). Le dApp rappresentano una prospettiva più ampia rispetto ai "contratti intelligenti". Più in generale, una dApp è un'applicazione web che si basa su servizi infrastrutturali aperti, decentralizzati e peer-to-peer.

Una dApp è composta almeno da:

- Contratti intelligenti su una blockchain.
- Un'interfaccia utente web front-end.

Inoltre, molte dApp includono altre componenti decentralizzate, come ad esempio:

[38] Ethereum Proof of Stake Date: date + what you need to know – Daniel Won – 2020

- Un protocollo e una piattaforma di storage decentralizzato (P2P).
- Un protocollo e una piattaforma di messaggistica decentralizzata (P2P).

Una dApp è una sorta di applicazione Internet il cui backend gira su una rete peer-to-peer decentralizzata e il cui codice sorgente è open source. Nessun singolo nodo della rete ha il controllo completo della dApp. A seconda della funzionalità di quest'ultima, per memorizzare i dati dell'applicazione vengono utilizzate diverse strutture di dati. Questi peers possono essere computer collegati ad Internet; pertanto, è difficile rilevare ed evitare che i peers apportino modifiche non valide ai dati dell'applicazione e condividano informazioni errate con altri. Non esiste un server centrale in una dApp per coordinare i peers e decidere cosa sia giusto e cosa sia sbagliato. Ci sono alcuni protocolli (chiamati specificamente protocolli di consenso) per supplire a tali limiti.
Le applicazioni decentralizzate hanno alcuni vantaggi:

- Non essendoci un unico punto di guasto perché sono distribuite per default, le dApp funzionanti possono continuare a lavorare mentre viene risolto il problema
- Prevengono la violazione della censura in rete in quanto non esiste un'autorità centrale a cui il governo possa fare pressione per rimuovere alcuni contenuti. I governi non possono nemmeno bloccare il dominio o l'indirizzo IP dell'applicazione, poiché le dApp non sono accessibili tramite un particolare indirizzo IP o dominio. Ovviamente il governo può rintracciare i singoli nodi della rete in base al loro indirizzo IP e rimuoverli, ma se la rete è enorme, diventa quasi impossibile spegnere l'app, soprattutto se i nodi sono distribuiti in paesi diversi.
- È facile per gli utenti fidarsi dell'applicazione in quanto la stessa non è controllata da una singola autorità che li potrebbe eventualmente truffare a scopo di lucro.

Ovviamente, ogni sistema ha alcuni vantaggi e svantaggi. Alcuni svantaggi delle applicazioni decentralizzate sono:

- La correzione di bug o l'aggiornamento delle dApp è difficile, in quanto ogni peer della rete deve aggiornare il proprio software del nodo.
- Alcune applicazioni richiedono la verifica dell'identità dell'utente (cioè KYC) e, poiché non esiste un'autorità centrale per verificarla, diventa un problema durante lo sviluppo di tali applicazioni.
- Sono difficili da costruire perché usano protocolli molto complessi per raggiungere il consenso e devono essere costruiti per essere scalabili fin dall'inizio.

Uno dei principali vantaggi delle dApp è che in genere garantiscono l'anonimato degli utenti, ma molte applicazioni richiedono la verifica dell'identità dell'utente per poter utilizzare l'app.
Nelle applicazioni centralizzate, si verifica l'identità dell'utente chiedendo alle persone di inviare determinati documenti scansionati, la verifica OTP e così via. Questo processo si chiama "conosci il tuo cliente" (KYC). Poiché nelle dApp non si verifica ciò, le stesse devono verificare l'identità dell'utente in modo diverso e in maniera digitale.
Esistono varie forme di identità digitale, che potrebbero risolvere questo problema delle dApp, ma attualmente la forma più raccomandata e popolare è il certificato digitale. Esso (chiamato anche certificato a chiave pubblica o certificato d'identità) è un documento elettronico utilizzato per dimostrare la proprietà di una chiave

pubblica. Fondamentalmente, un utente possiede una chiave privata, una chiave pubblica e un certificato digitale. La chiave privata è segreta e l'utente non dovrebbe condividerla con nessuno. La chiave pubblica può essere condivisa con chiunque. Il certificato digitale contiene la chiave pubblica e le informazioni su chi la possiede. Ovviamente, non è difficile produrre questo tipo di certificato; pertanto esso viene sempre rilasciato da un ente autorizzato di cui ci si può fidare.

Il certificato digitale ha un campo criptato dalla chiave privata dell'autorità di certificazione. Per verificare l'autenticità dello stesso, è sufficiente decifrare il campo utilizzando la chiave pubblica dell'autorità di certificazione, e, se la decifrazione avviene con successo, allora il certificato è valido. Anche se gli utenti ottengono le identità digitali e queste ultime vengono verificate dalla dApp, c'è ancora un problema importante; ci sono infatti diverse autorità di emissione di certificati digitali, ma per accertarli c'è bisogno della chiave pubblica dell'autorità di emissione. A causa di questo problema, l'unica opzione che rimane è la verifica manuale dell'identità dell'utente da parte di una persona autorizzata dalla società. Il fine dell'accertamento dell'identità degli utenti nelle applicazioni è quello di rendere difficile la fuga degli stessi dopo aver eseguito una qualche attività fraudolenta.

Una dApp non dovrebbe dipendere da app centralizzate a causa di un singolo punto di guasto, ma in alcuni casi non ci sono altre opzioni. Ad esempio, se una dApp vuole leggere un punteggio di calcio, ha bisogno di recuperare i dati da un'applicazione centralizzata. Il problema principale è come la dApp sa che i dati recuperati da un dominio non vengano manomessi da un servizio intermedio/uomo e rappresentino la risposta corretta.

Ci sono vari modi per risolvere questo problema a seconda dell'architettura della dApp. Ad esempio, in Ethereum, per accedere alle API centralizzate, i contratti smart possono utilizzare il servizio Oraclize come intermediario, poiché gli stessi non possono effettuare richieste HTTP dirette. Oraclize fornisce una prova TLSNotary per i dati che recupera per il contratto smart da servizi centralizzati.

Affinché un'applicazione centralizzata possa durare a lungo, il proprietario della stessa deve realizzare un profitto. Le dApp non hanno un proprietario; tuttavia, come qualsiasi altra app centralizzata, i nodi di una dApp hanno bisogno di hardware e risorse di rete per mantenerla in funzione. Quindi i nodi necessitano di qualcosa di profittevole in cambio per mantenere la dApp in esecuzione. Qui entrano in gioco i token interni e infatti la maggior parte delle dApp ha una valuta incorporata. Sulla base del protocollo di consenso si decide quanta valuta riceve ogni nodo.

3.2.2. Token

Ethereum è una piattaforma adatta anche al lancio di nuove monete digitali. Infatti, quasi tutti i progetti di Ethereum oggi si lanciano con una sorta di token.

Per chiarire il ruolo di un token in un nuovo progetto, bisogna considerare che la maggior parte dei progetti utilizza i gettoni in uno dei due modi: o come "gettoni di utilità" o come "gettoni di equità". Molto spesso, questi due ruoli sono conflittuali e difficili da distinguere.

I token di utilità sono quelli il cui uso è richiesto per pagare un servizio, un'applicazione o una risorsa. Ne sono esempi i token che rappresentano risorse come lo stoccaggio condiviso, l'accesso a servizi come i social network o l'ether stesso come gas per la piattaforma Ethereum. A titolo di confronto, i token di utilità sono quelli che rappresentano le azioni di una startup. I token azionari possono essere come le azioni senza diritto di voto ma che distribuiscono dividendi o come le azioni con diritto di voto in un'organizzazione dove la gestione della piattaforma avviene attraverso la maggioranza dei voti dei detentori dei token.

3.2.3. Token ERC20 Standard

Il token più comune in circolazione emesso sulla blockchain di Ethereum è il token ERC20 Standard. La prima versione del token è stata introdotta nel novembre 2015 da Fabian Vogelsteller, come Ethereum Request for Comments (ERC). Ad essa è stato assegnato automaticamente il numero 20 di GitHub, dando origine al nome "ERC20 token". La stragrande maggioranza dei token si basa attualmente su ERC20.

ERC20 consente due diversi flussi di lavoro. Il primo è un flusso di lavoro a singola transazione, semplice e diretto, che utilizza la funzione di trasferimento. Questo flusso di lavoro è quello utilizzato dai portafogli per inviare token ad altri portafogli.

L'esecuzione del contratto di trasferimento è molto semplice. Se Alice vuole inviare 10 gettoni a Bob, il suo portafoglio invia una transazione all'indirizzo del contratto, chiamando la funzione di trasferimento con l'indirizzo di Bob e "10" come argomento. Il contratto token regola il saldo di Alice (-10) e il saldo di Bob (10) ed emette un evento chiamato +Trasferimento.

Il secondo flusso di lavoro è un flusso a due transazioni che utilizza la funzione approve, seguita da transferFrom. Questo flusso di lavoro permette al proprietario di un token di delegare il suo controllo ad un altro indirizzo. Ad esempio, se un'azienda vende gettoni per un ICO, può indicare un indirizzo di un contratto di crowdsale per distribuire una certa quantità di gettoni. Il contratto di crowdsale può quindi usare la funzione transferFrom dal saldo del proprietario del contratto del gettone ad ogni nuovo acquirente.

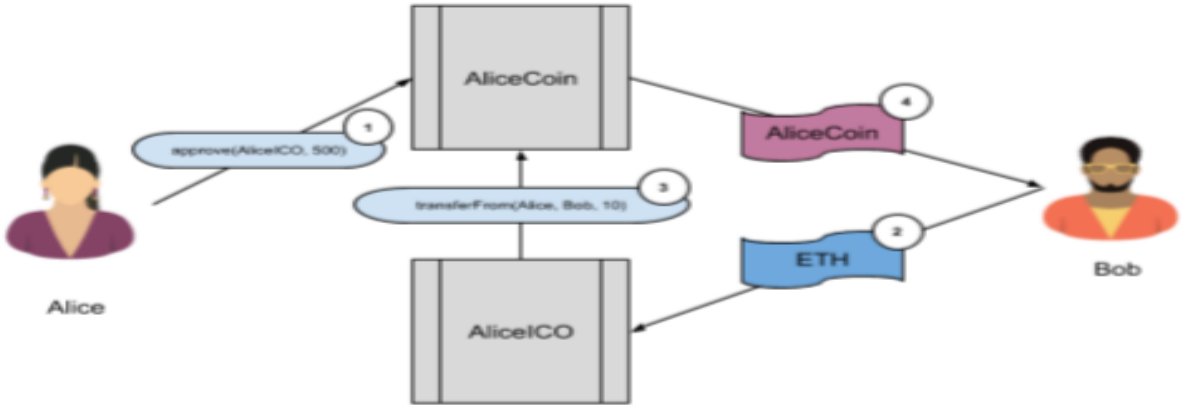


Figura 51 – Transazione con token ERC20 – Mastering Ethereum

Per approve e transferFrom sono necessarie due transazioni. Per esempio, Alice vuole permettere al contratto AliceICO di vendere il 50% di tutti i gettoni AliceCoin ad acquirenti come Bob e Charlie. In primo luogo, Alice lancia il contratto AliceCoin ERC20, rilasciando tutti i gettoni AliceCoin nel proprio indirizzo. Poi, Alice lancia il contratto AliceICO che può vendere gettoni in cambio di ether. Successivamente, Alice utilizza la funzione approve e transferFrom e invia una transazione ad AliceCoin, chiamando approve, con l'indirizzo di AliceICO e il 50% del finanziamento totale. Questo farà scattare l'evento di approvazione. Ora, il contratto AliceICO può vendere AliceCoin.

Quando AliceICO riceve l'ether da Bob, deve inviare i token AliceCoin previsti sulla base del tasso di cambio a Bob. Il tasso di cambio che Alice ha fissato quando ha creato l'AliceICO determina quanti gettoni Bob riceverà per la quantità di ether inviata ad AliceICO. Quando AliceICO chiama la funzione transferFrom di AliceCoin, imposta l'indirizzo di Alice come mittente, l'indirizzo di Bob come destinatario e usa il tasso di cambio per determinare quanti gettoni AliceCoin saranno trasferiti a Bob. Il contratto AliceCoin trasferisce il saldo dall'indirizzo di Alice all'indirizzo di Bob e attiva un evento di trasferimento. [39] [40] [41]

[39] Mastering Ethereum – Andreas M. Antonopoulos and Gavin Wood – 2018

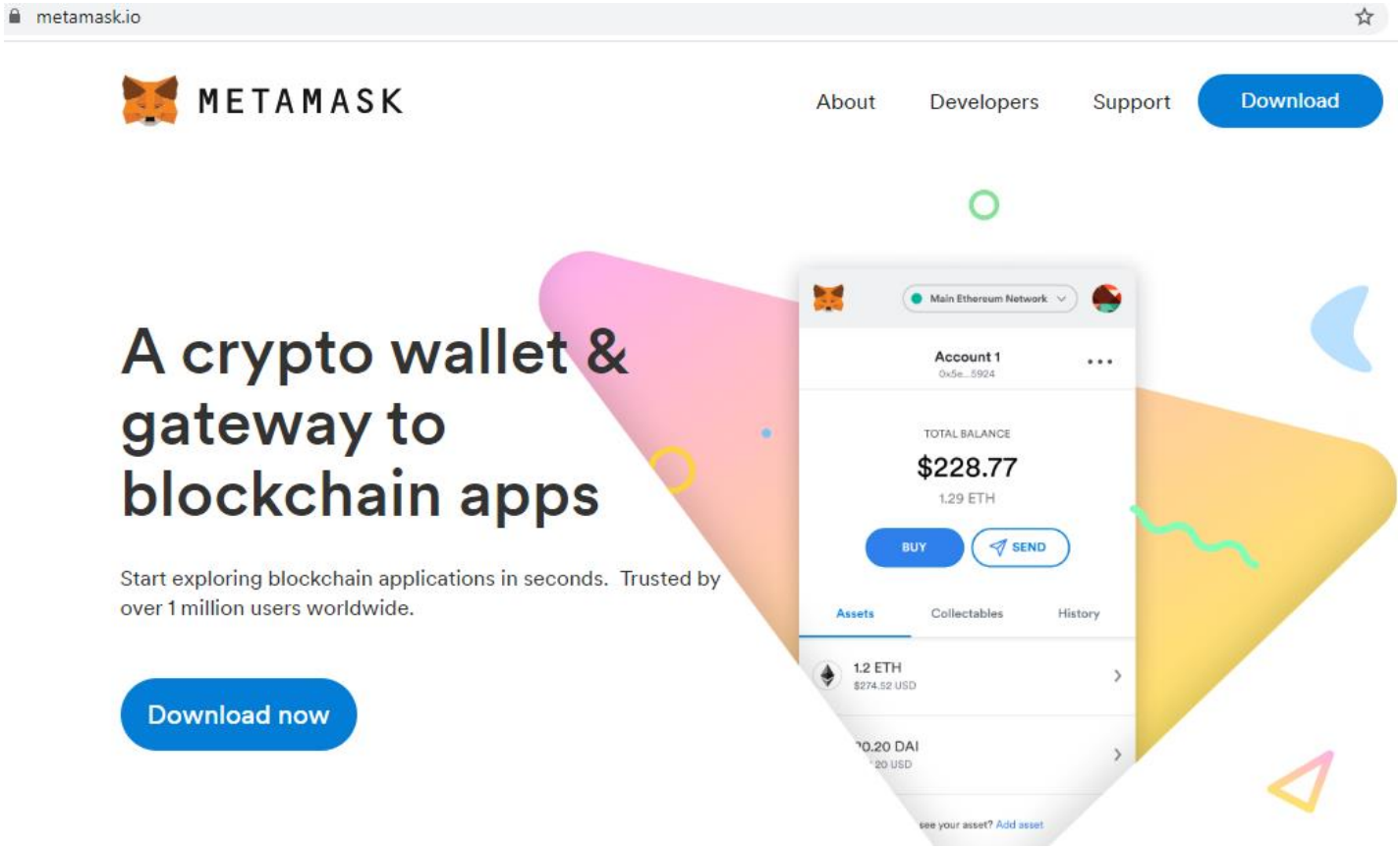
[40] Blockchain Disruption and Smart Contracts – Lin William Cong, Zhiguo He - 2018

[41] Blockchain based smart contracts: a systematic mapping study – Maher Alharby, Aad van Moorsel - 2017

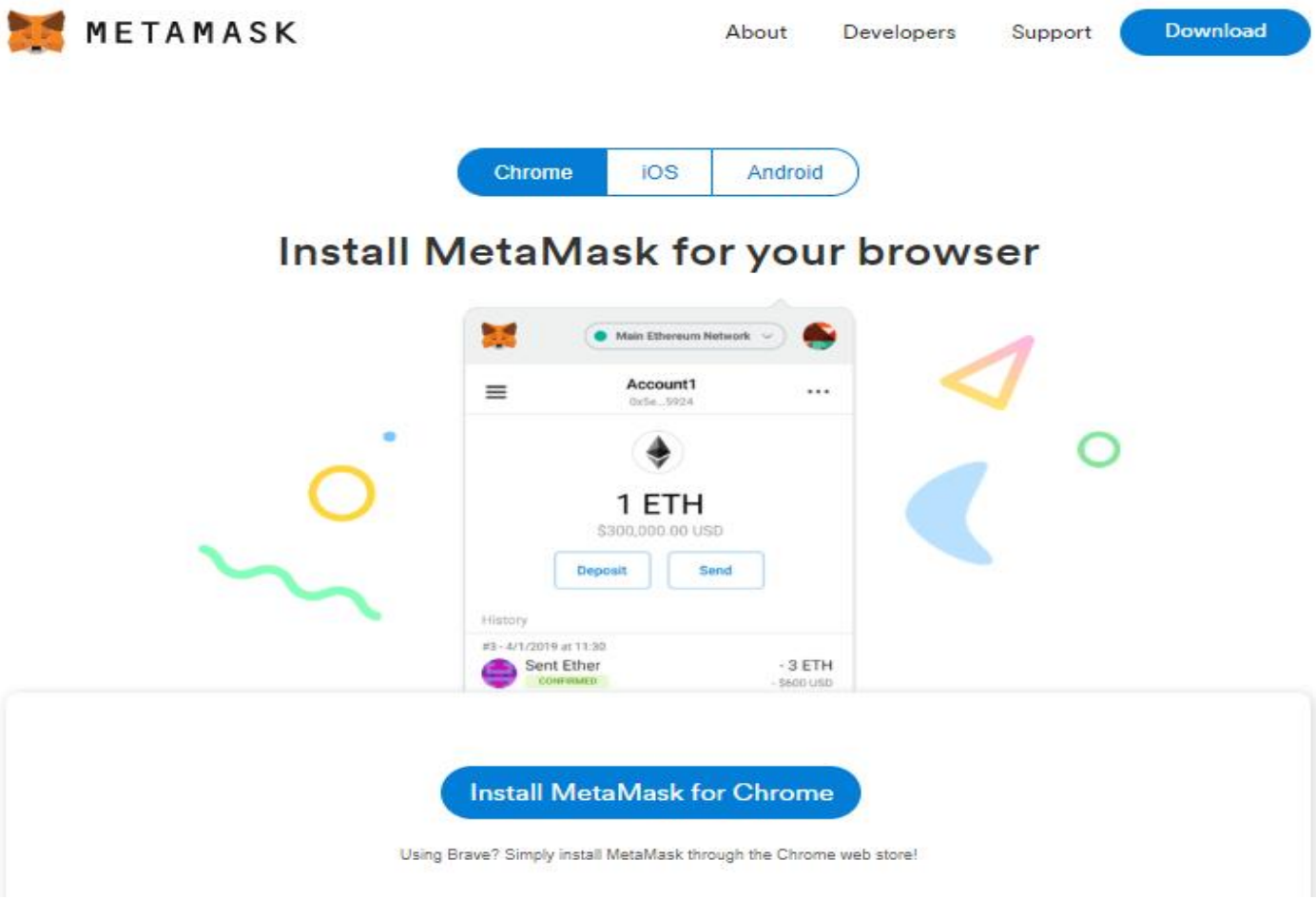
4. Metamask, Solidity e Remix

4.1. Metamask

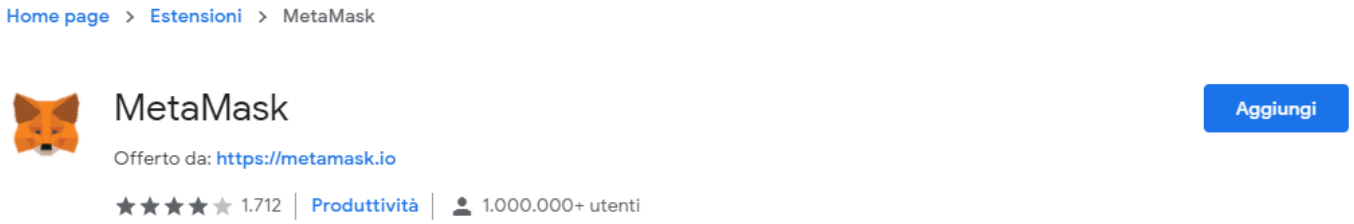
La prima cosa da fare nel caso in cui si voglia scrivere uno smart contract all'interno della blockchain di Ethereum è procurarsi del gas che deve essere pagato in ether. L'applicazione usata per questo scopo è chiamata Metamask ed è un'integrazione del browser, che funziona da wallet Ethereum.
Oltre lo storage di ether, Metamask permette di interagire con tutta la piattaforma Ethereum perché è un portafoglio di criptovaluta che ti consente di usare applicazioni decentralizzate basate su Ethereum direttamente dal tuo browser. Quindi Metamask è un plugin che consente agli utenti di interfacciarsi con Ethereum, partecipare alle ICO, interagire con le dApps ecc.. attraverso tutti i siti web che operano con criptovalute. Essa pertanto ha il pregio di connettere i front end Ethereum con il normale web che chiunque è in grado di utilizzare; infatti comunemente si dice che Metamask è un ponte che ti consente di visitare il web distribuito di domani nel tuo browser di oggi.
Iniziamo con il download di Metamask.
Cerchiamo su Google Chrome metamask.io che mi rimanda al download dell'estensione (Metamask è una estensione di Google Chrome per cui è consigliato usare quest'ultimo come motore di ricerca).



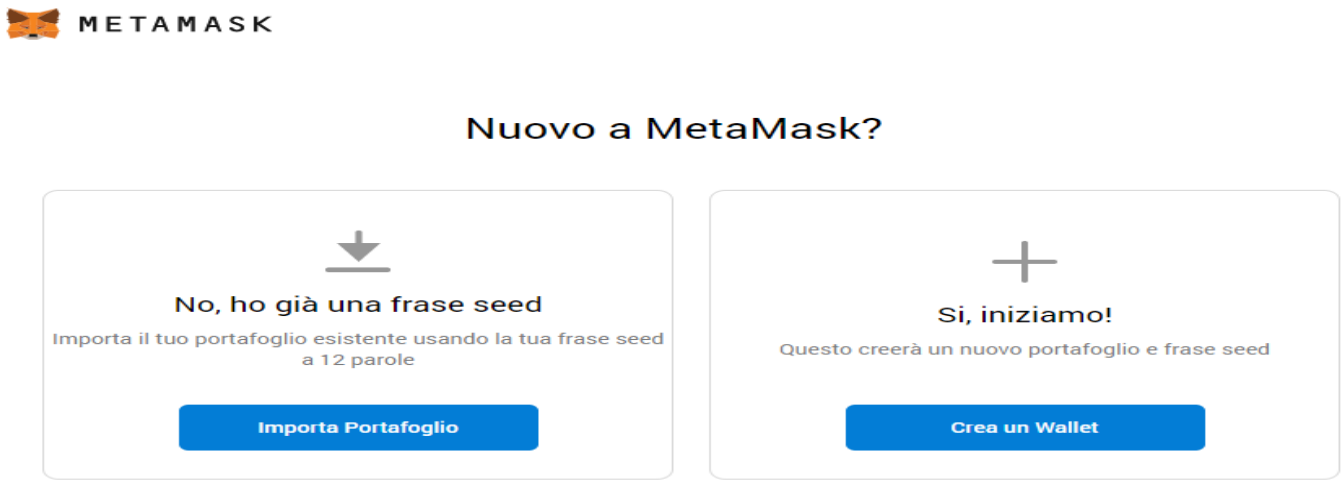
Facciamo download now e arriviamo su questa pagina: come detto Metamask è stato progettato per essere un'estensione di Chrome per cui lo scarichiamo su di esso.



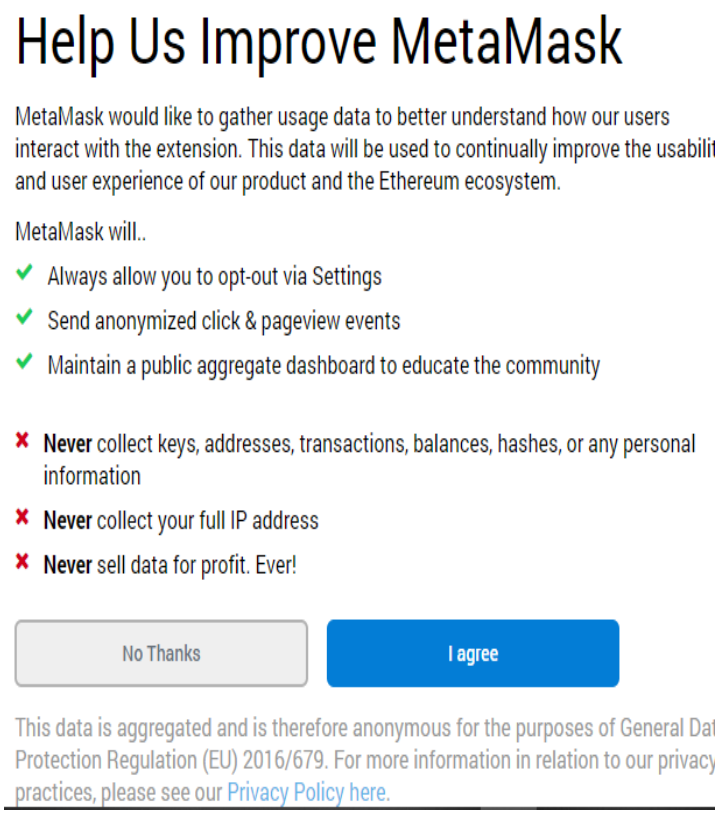
Sotto possiamo vedere anche un video introduttivo, ma quello che interessa è scaricare l'estensione per aver la possibilità di lanciare gli smart contracts sulla blockchain di Ethereum per cui clicchiamo su Aggiungi.

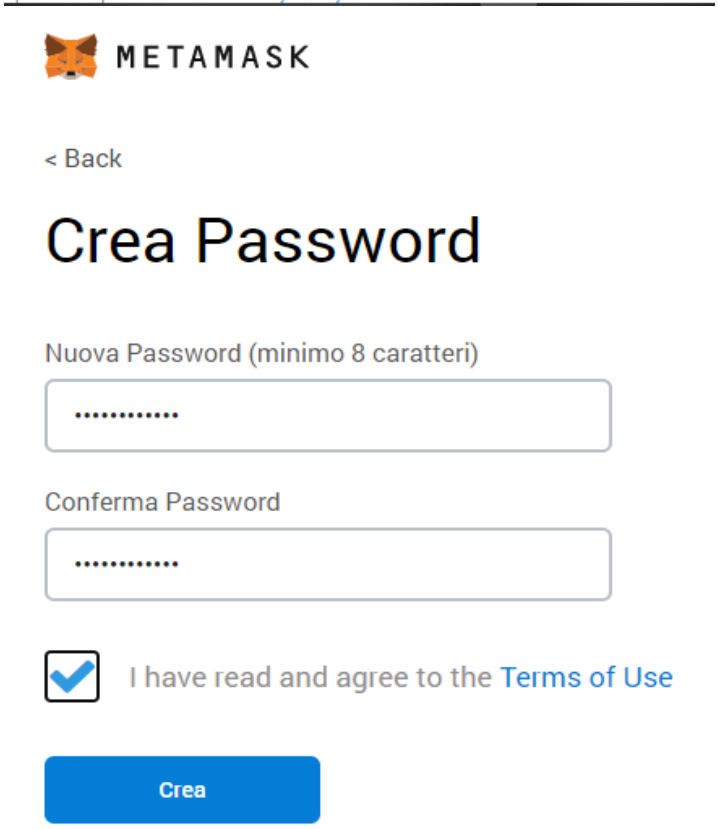


Ora, una volta scaricata l'estensione su Chrome, possiamo andare a scaricare il nostro portafoglio Ethereum su Metamask. La schermata che appare è la seguente.

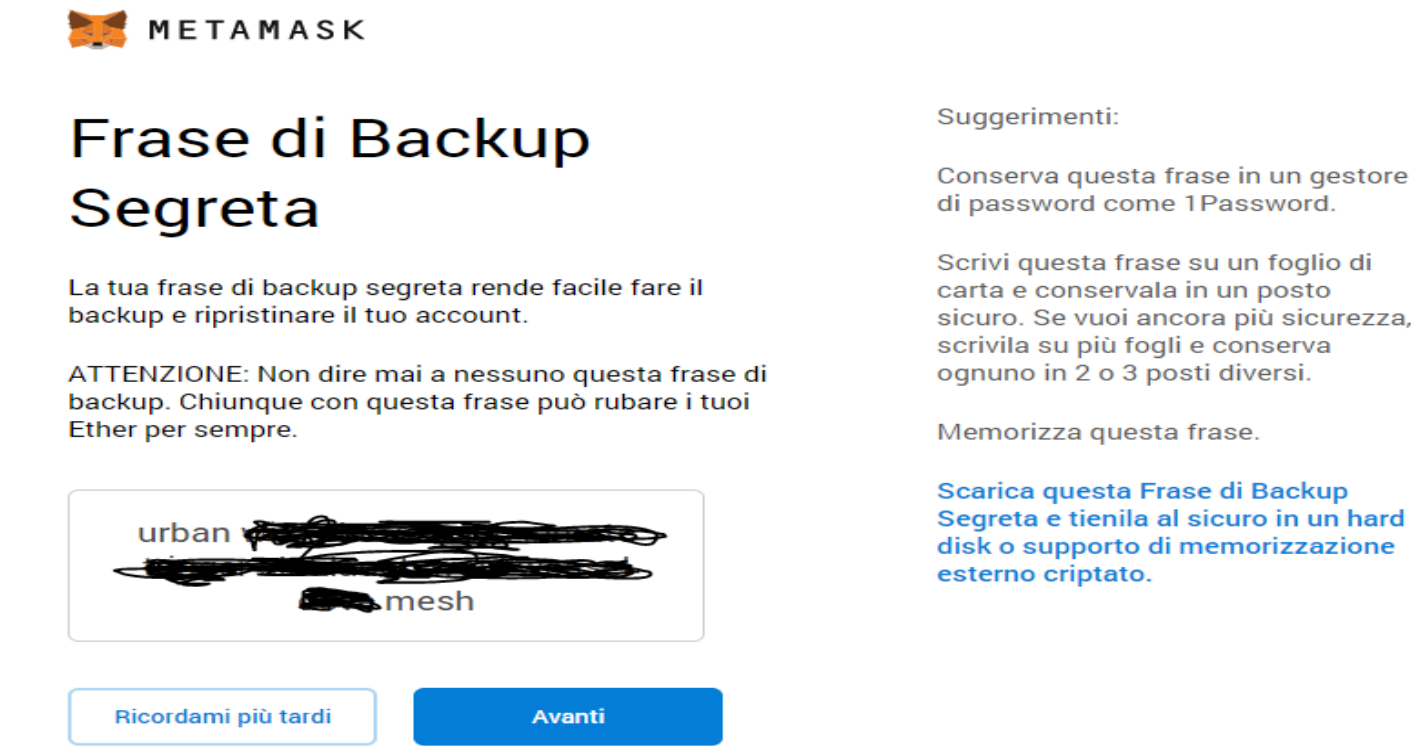


Clicchiamo su Crea un Wallet

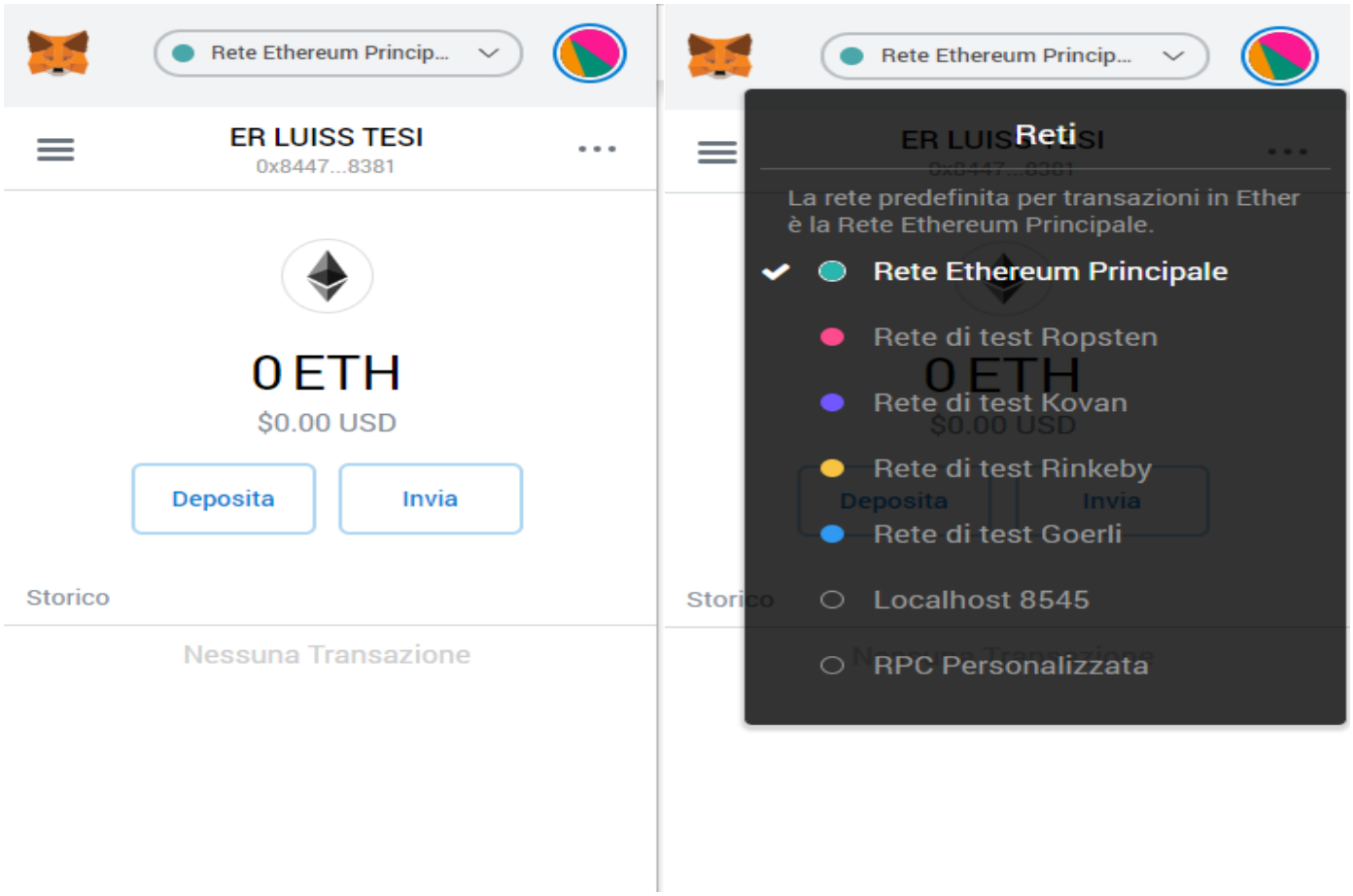




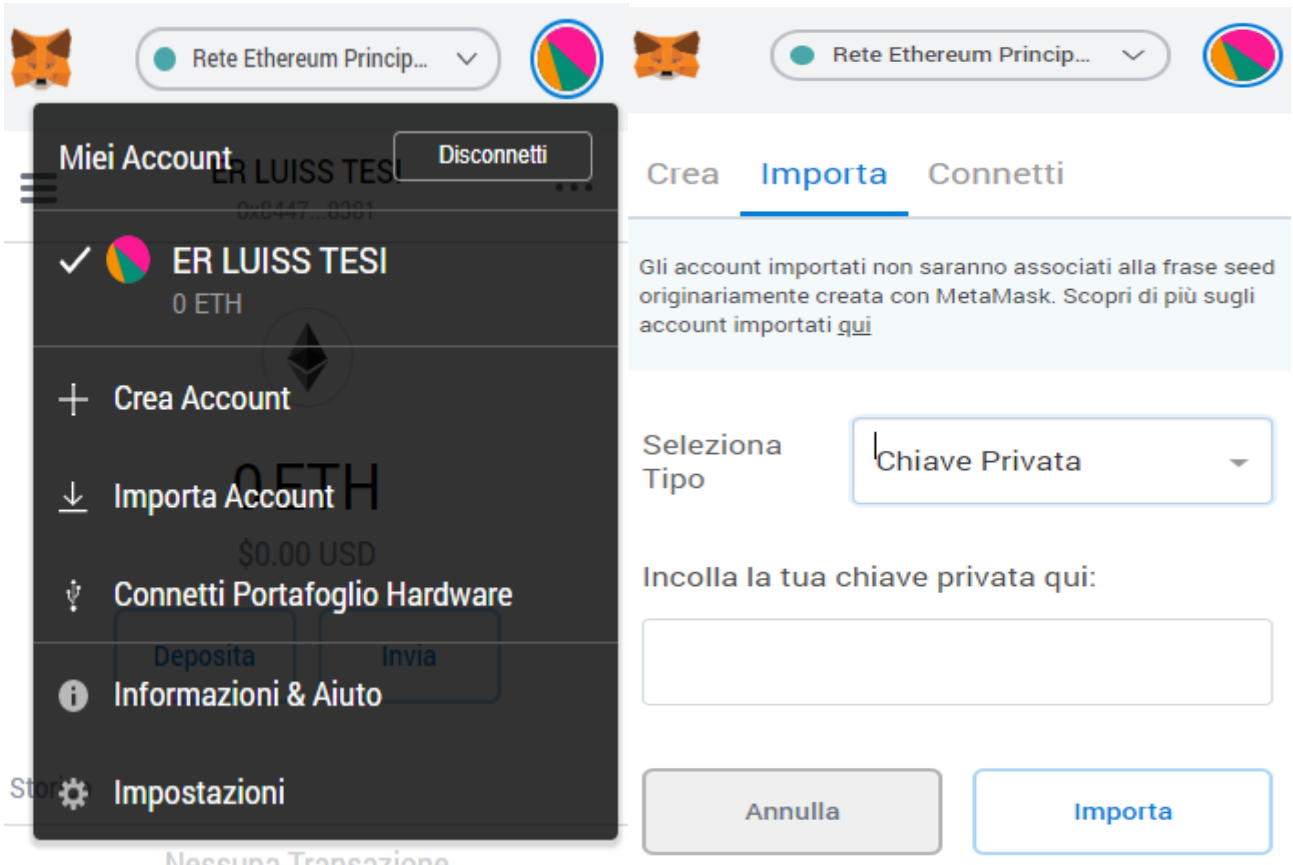
Accettiamo le condizioni di utilizzo e creiamo una nuova password. La fase successiva è una frase senza senso composta da 12 parole (il "Seed" di Metamask) che dobbiamo sempre ricordare e tenere al sicuro perché, nel caso in cui cadesse in mani sbagliate, ci sarebbe la possibilità di entrare nel nostro wallet e rubare gli ether (per questioni di privacy ho cancellato le 10 parole in mezzo, lasciando solo la prima e l'ultima). Le 12 parole sono l'unico strumento utilizzabile ove si perdesse la password creata al punto precedente. Se si smarrissero le 12 parole, i fondi resterebbero bloccati per sempre.



Dopo un'altra prova di verifica password, che per privacy non posso inserire, l'account Metamask è finalmente completato e, come previsto, si presenta come una estensione di Chrome. La homepage del nostro account appare in questa maniera.

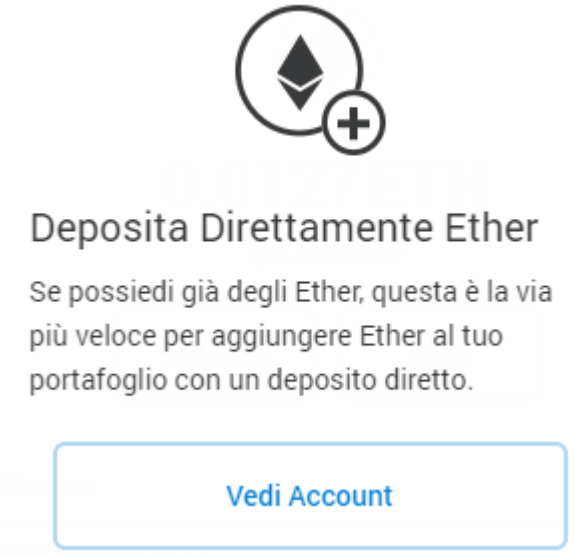


Ho chiamato l'account "ER LUISS TESI" in quanto costituisce un account di prova che verrà usato per pagare il gas del nostro smart contract; ER sono le iniziali del mio nome e cognome. Nel caso in cui si clicchi sulla freccetta in alto si possono vedere le reti sulle quali effettuare i test. Noi siamo interessati solo alla rete Ethereum, mentre le altre reti vengono utilizzate principalmente dagli sviluppatori. Ad esempio, la rete Ropsten può essere utilizzata come test per sperimentare la piattaforma e non per spendere soldi reali come invece accade nell'Ethereum principale. Possiamo infatti richiedere su "faucet.metamask.io" un ether gratuito che possiamo usare come prova che verrà caricato nel nostro account Metamask.
Vediamo alcune funzionalità di quest'ultimo.
La prima cosa che si può fare su Metamask è cliccare sul cerchio colorato in alto a destra che ci dà la possibilità di creare un nuovo account o di importare un nuovo account. Per esempio, possiamo collegare Metamask a MyEtherWallet che è un altro famoso portafoglio Ethereum per avere una condivisione immediata dei nostri ether. Per attuare la condivisione, possiamo collegare la nostra chiave privata oppure possiamo utilizzare un file JSON da inserire. Quest'ultimo ci viene fornito nel caso avessimo creato un account con MyEtherWallet.

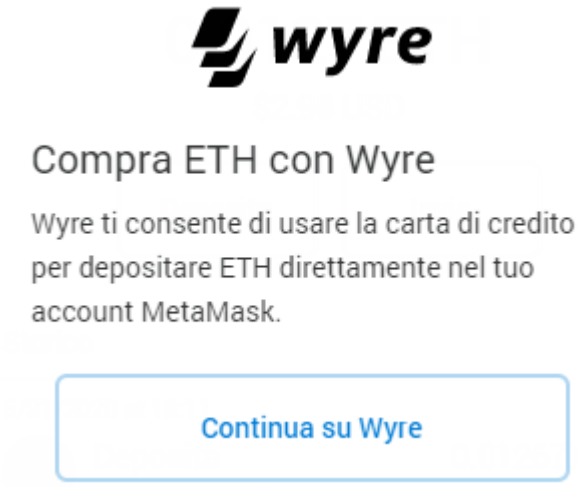


Ora vediamo la parte più importante che ci permette di far "girare" gli smart contract sulla blockchain Ethereum. Attraverso il nuovo aggiornamento di Metamask, mentre prima era possibile spostare gli ether da un portafoglio Coinbase, ora è più difficile fare ciò perché bisognerebbe sincronizzare i due account. Per questo motivo le procedure che offre l'applicazione per il deposito di ether sono:

- "Deposita direttamente ether" che ci consente di avere accesso alla chiave pubblica da inserire su qualche altro wallet in cui sono già presenti degli ether e inviarli su un portafoglio di Metamask



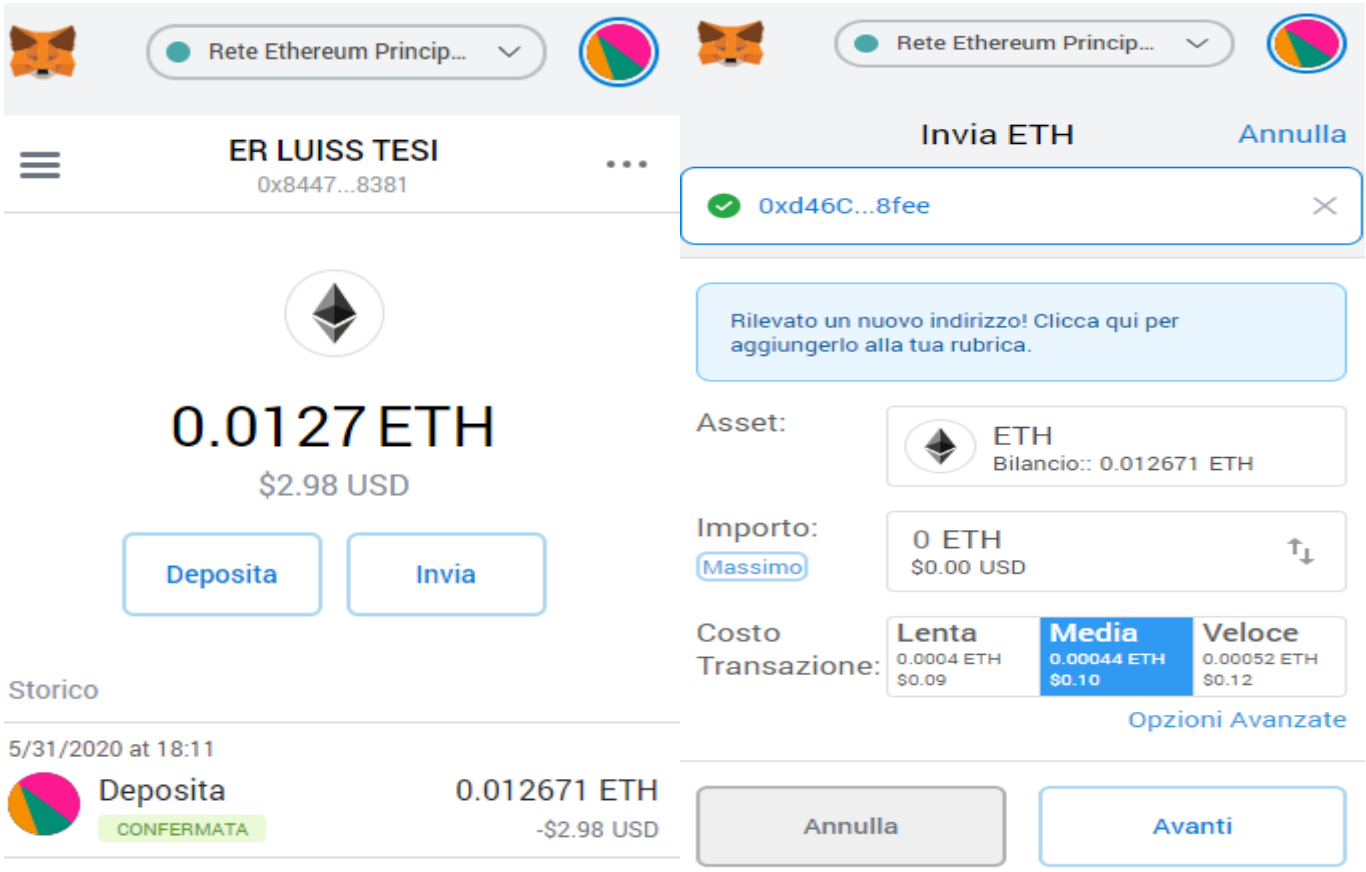
- "Wyre" che è il procedimento più semplice per chi non possiede già criptovalute o non vuole trasferirle da altri wallet. Attraverso Wyre è sufficiente avere una carta di credito per convertire euro in ether in pochi minuti.



- "CoinSwitch" che è un sistema che consente di convertire criptovalute presenti in differenti portafogli in ether con commissioni molto convenienti e depositarli nel proprio wallet di Metamask.

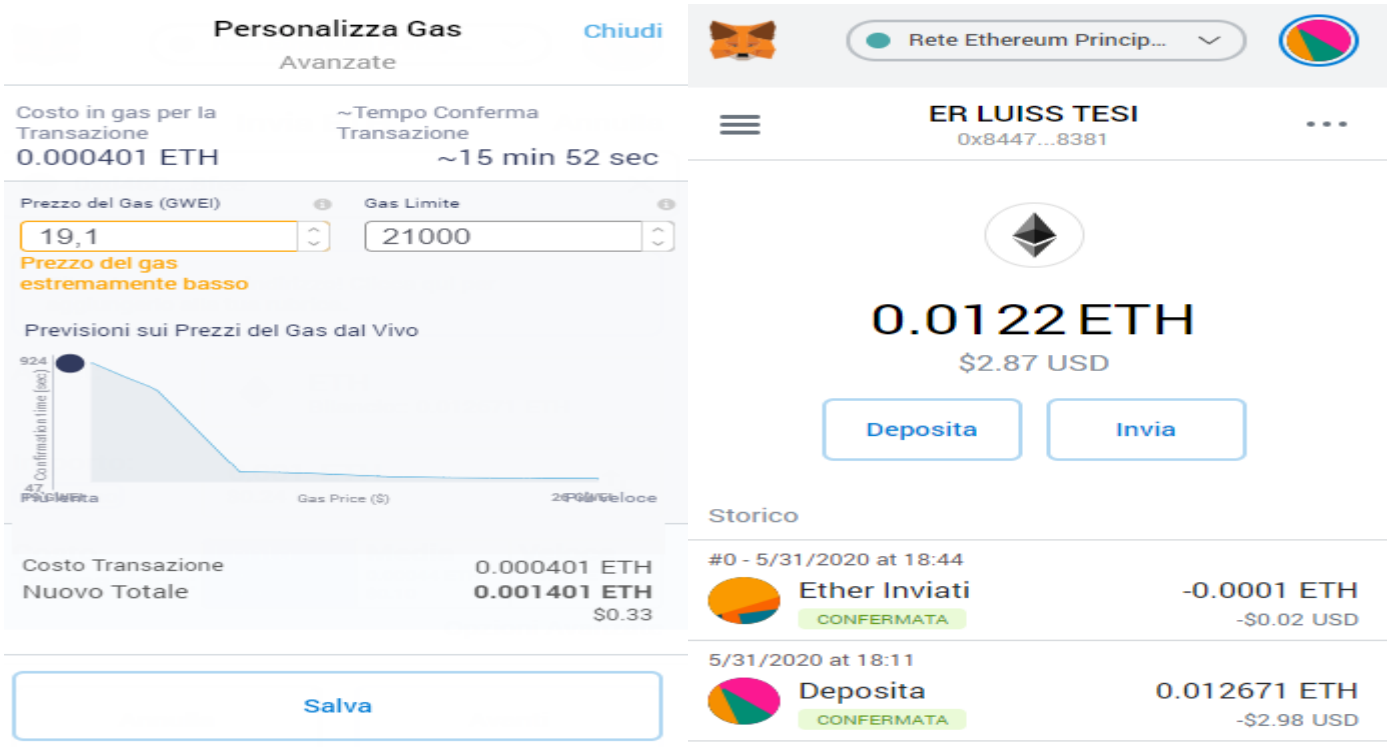
A questo punto possediamo ether nel nostro portafoglio “ER LUISS TESI” per un valore di 3 dollari che ho deciso di inviare mediante un trasferimento con Wyre. Differentemente da prima ci si apre la possibilità di scambiare ether mediante il tasto invia.

Possiamo vedere che, quando vogliamo inviare ether, Metamask ci chiede una chiave (l’indirizzo a cui vogliamo inviare i nostri ether) che ovviamente è una chiave pubblica e non privata. Nel caso in cui si inserisca una chiave errata o invalida, Metamask la riconosce e ci nega la transazione. Al contrario, quando si inserisce una chiave pubblica valida, che riconduce ad un portafoglio di ether, come nel passaggio sotto a destra, Metamask riconosce la transazione (abbiamo inserito la chiave pubblica di un portafoglio di Bitcoin e Ethereum che abbiamo sull’exchange Coinbase. Attenzione: la chiave pubblica deve essere quella del portafoglio di Ethereum, non quella di Bitcoin sennò la transazione sarà riconosciuta come invalida). Possiamo scegliere anche il costo della transazione, che sono le fees che verranno pagate ai miners. Più sarà alta la commissione che decidiamo di impostare, più velocemente avverrà il trasferimento dei nostri ether.



Cliccando sulle opzioni avanzate possiamo scegliere anche la quantità di gas da utilizzare, verificare entro quanto tempo avverrà la transazione e leggere alcune previsioni sul prezzo del gas. A questo punto confermiamo la transazione e inviamo gli ether all’altro portafoglio su Coinbase. Teniamo in considerazione

che la stessa cosa che è stata fatta con una transazione, in seguito verrà fatta con uno smart contract.

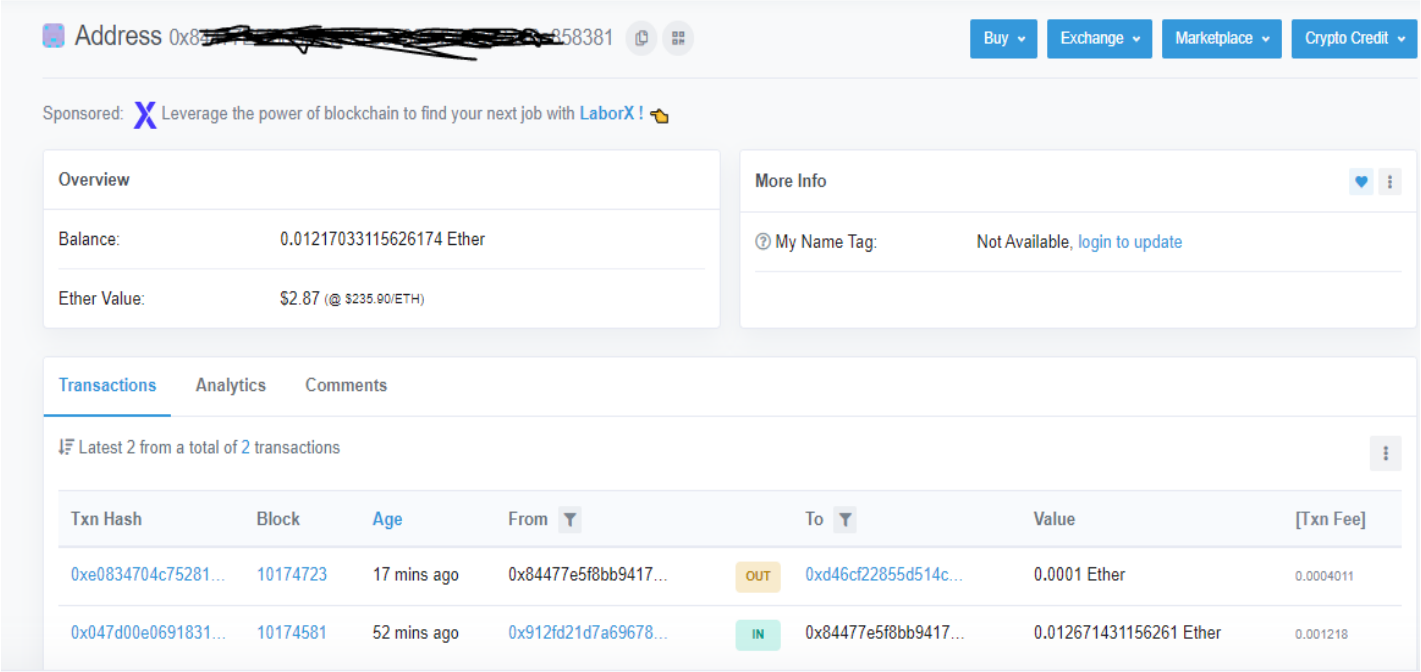


La transazione dopo circa 5 minuti è stata completata come mostra il portafoglio aggiornato su Metamask, ma possiamo verificare la stessa cosa anche sul portafoglio di Coinbase (per privacy ho coperto il saldo del mio portafoglio).

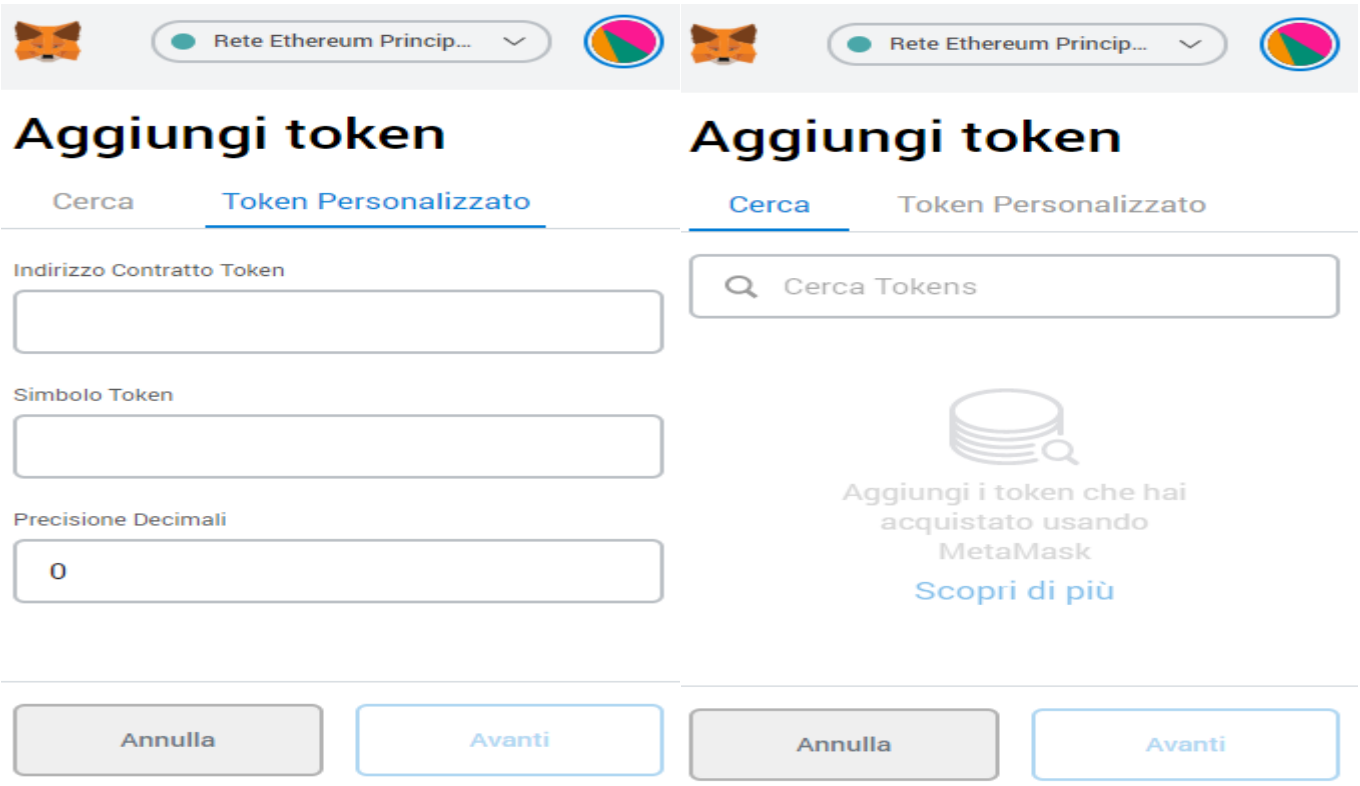
Possiamo anche verificare l'avvenuta transazione sulla blockchain di Ethereum attraverso il sito etherscan.io che permette di esplorare e cercare le transazioni, gli indirizzi, i token e i prezzi che hanno luogo su di esso. Etherscan è conosciuta come il principale "block explorer" di Ethereum. È essenzialmente un motore di ricerca che permette agli utenti di cercare, confermare e convalidare le transazioni sulla piattaforma decentralizzata di contratti intelligenti. Inserendo un indirizzo nella casella di ricerca, è possibile visualizzare il saldo, il valore e tutte le transazioni effettuate attraverso quell'indirizzo.

Il team dietro Etherscan ha creato la piattaforma come entità indipendente la cui missione è quella di facilitare la trasparenza della blockchain. Etherscan non è un fornitore di servizi di portafoglio, non memorizza le chiavi

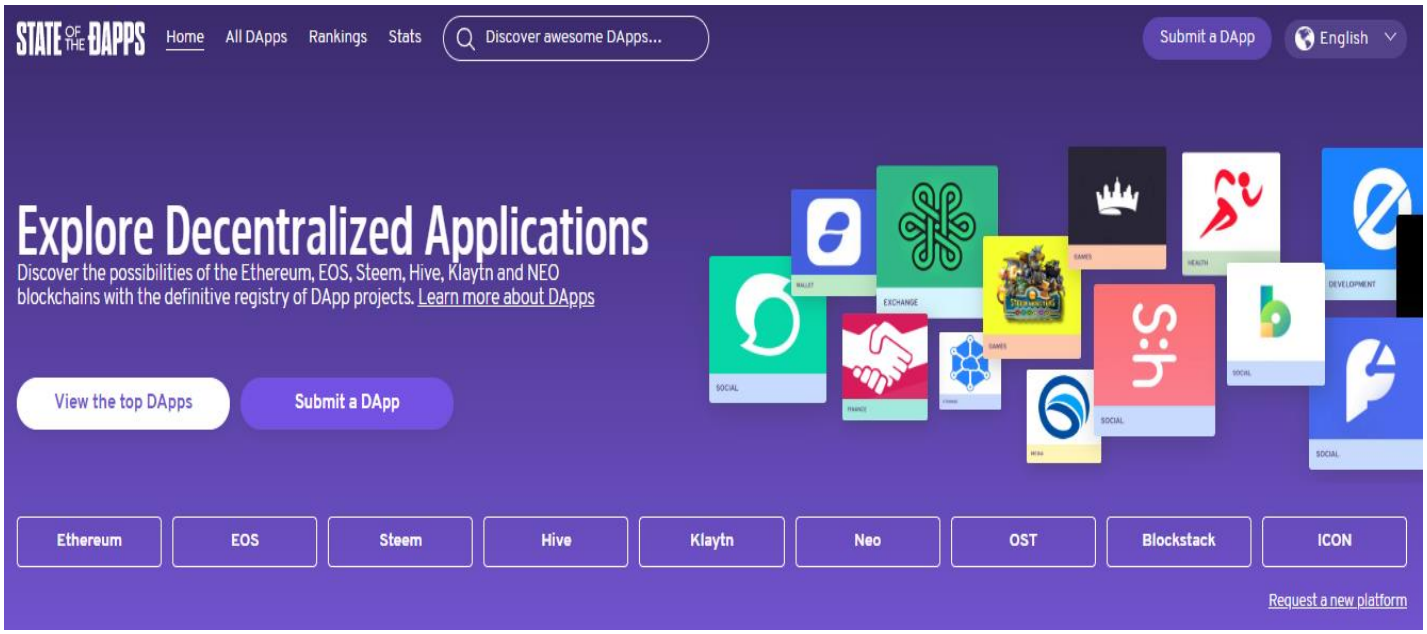
private delle persone e non ha alcun controllo sulle transazioni che avvengono sulla rete di Ethereum. Non è inoltre in grado di risolvere i problemi relativi alle transazioni.



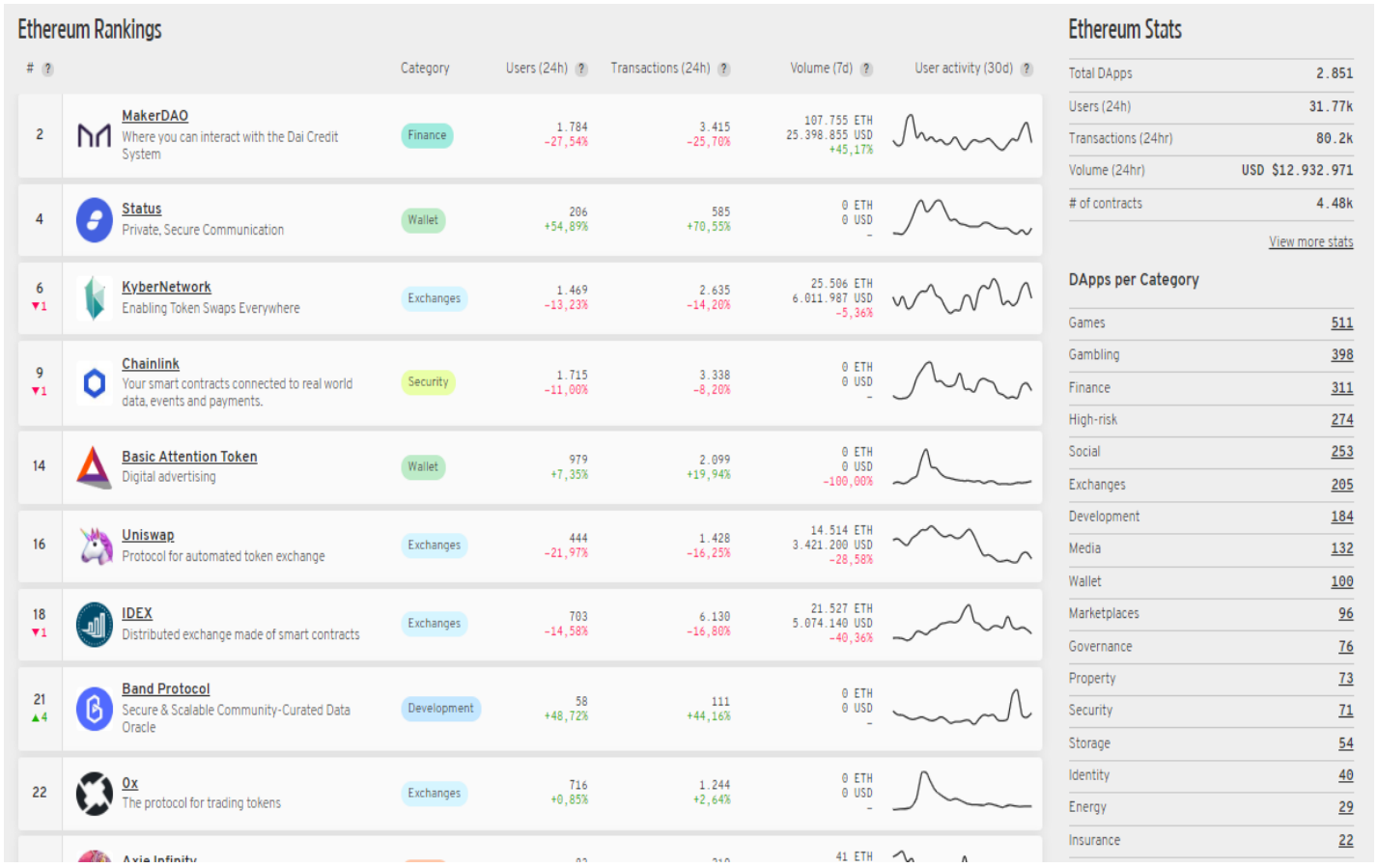
Metamask consente inoltre altre due implementazioni che sono comunemente lanciate su Ethereum. Permette di partecipare alle ICO: infatti quando viene lanciata una Initial Coin Offering, possiamo andare nella sezione di Metamask illustrata sotto, scrivere il nome del token, il simbolo e il gas price che ci verrà consigliato dai lanciatori della ICO. In caso di mancanza di gas sufficiente, non ci verrà consentito di partecipare alla ICO e di provare ad acquistare i nuovi token in cambio di ether.



Un'ultima funzione di Metamask è quella di interagire con le applicazioni decentralizzate (dApps) che è ciò per cui è stato creato Ethereum. Infatti, a titolo esemplificativo, se scriviamo su Google "stateofthedapps" ci troviamo in un sito in cui sono presenti centinaia di dApps che possiamo liberamente aprire.

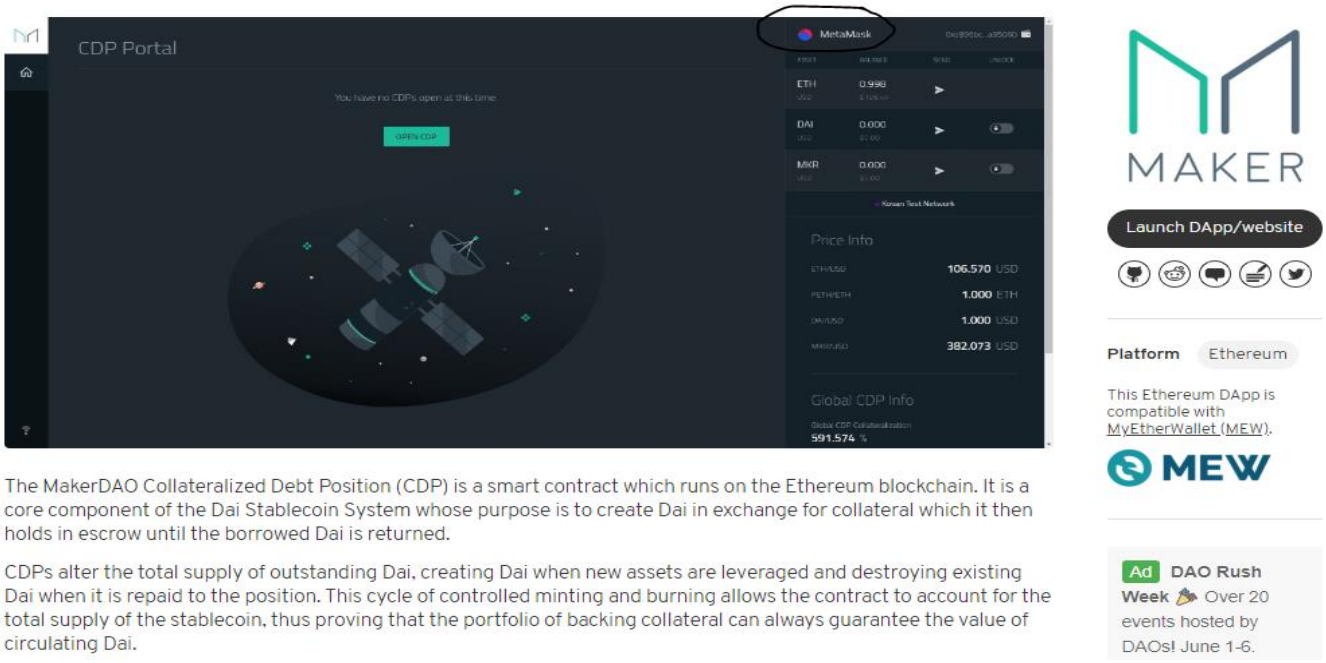


Selezionando le dApps presenti su Ethereum, si apre una lista delle più comuni dApps sulla blockchain di quest'ultimo.



Qui vengono esposte e classificate tutte le dApps a cui possiamo accedere che vivono sulla blockchain di Ethereum. Sulla destra possiamo vedere la quantità totale di dApps presenti al momento, il numero di persone che stanno utilizzando una dApp, la capitalizzazione di mercato e il numero di smart contracts usati per creare tutte queste applicazioni. Sotto invece possiamo vedere la suddivisione delle dApps per categoria e scegliere quelle a cui noi siamo maggiormente interessati.

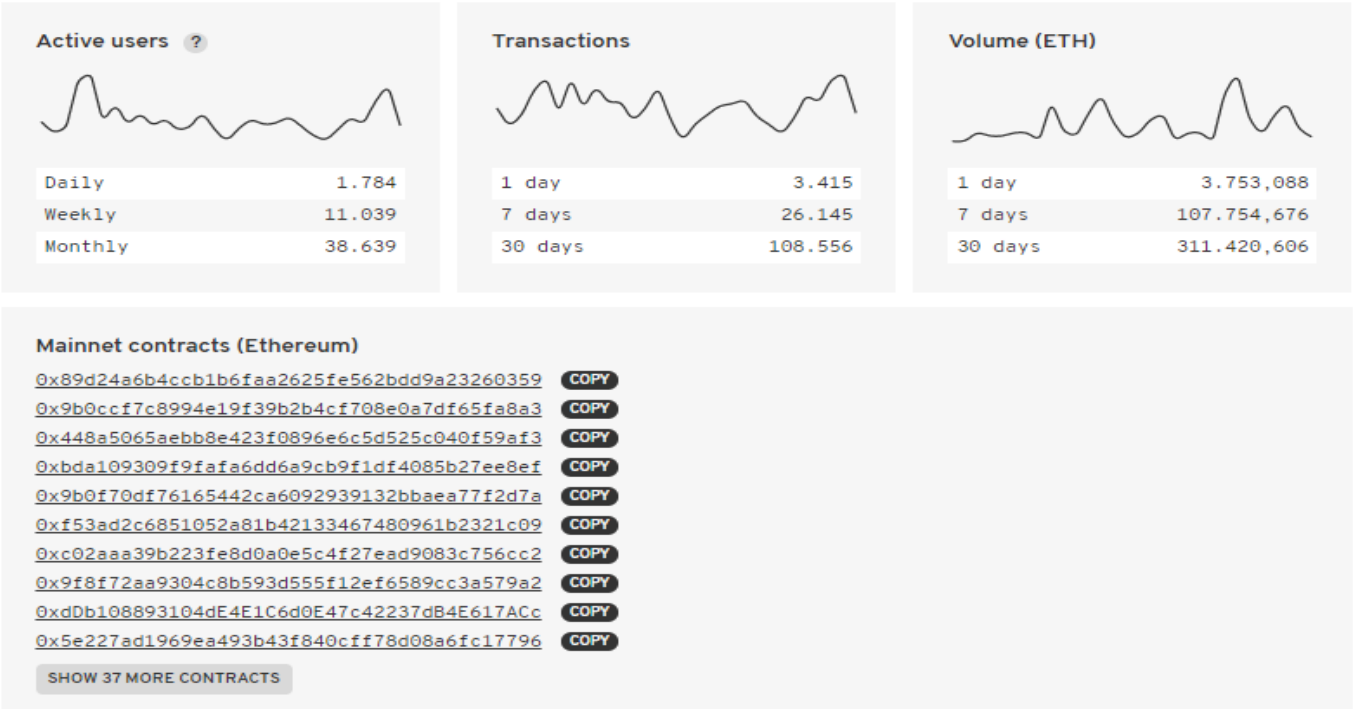
Ad esempio, possiamo vedere MakerDAO, che è un'applicazione per la finanza. Ci viene spiegata sotto l'utilità dell'applicazione, lo scopo per cui è stata creata, il link che ci rimanda al sito e la possibilità di interagirci con Metamask.



The MakerDAO Collateralized Debt Position (CDP) is a smart contract which runs on the Ethereum blockchain. It is a core component of the Dai Stablecoin System whose purpose is to create Dai in exchange for collateral which it then holds in escrow until the borrowed Dai is returned.

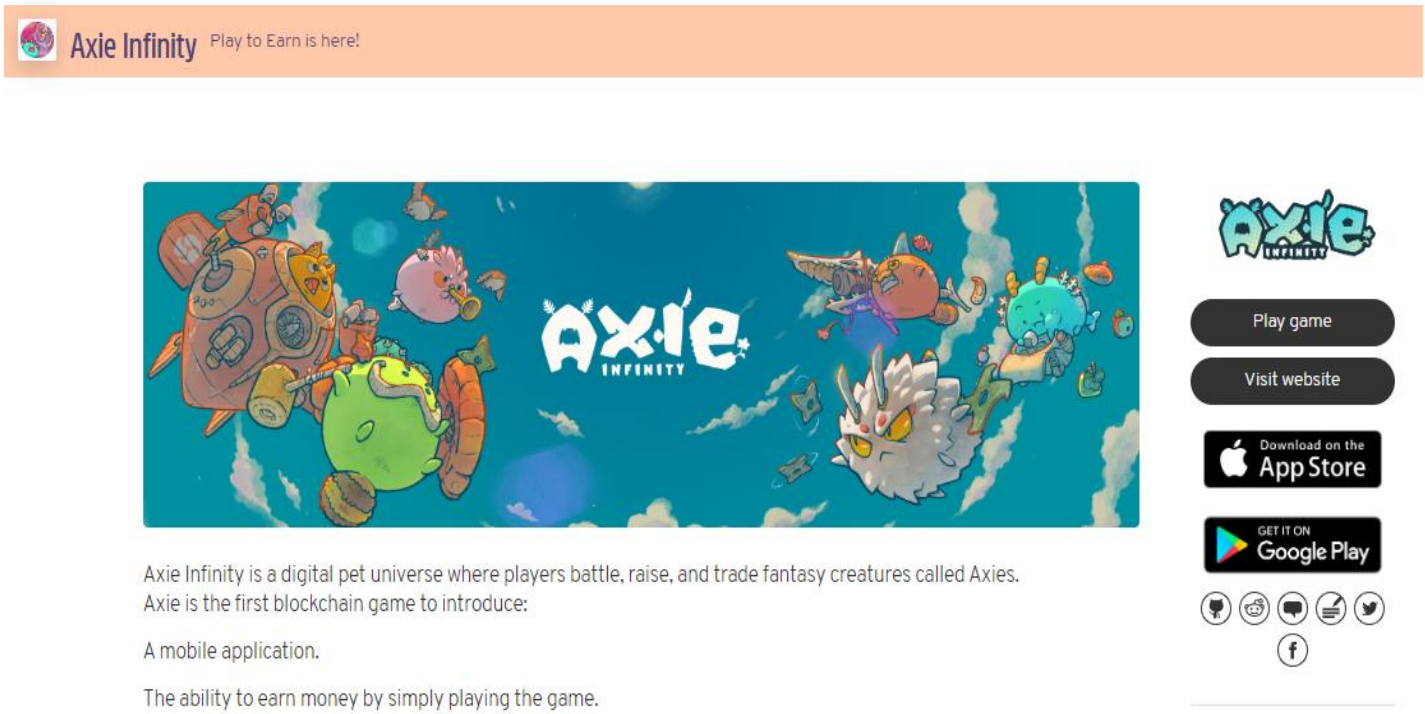
CDPs alter the total supply of outstanding Dai, creating Dai when new assets are leveraged and destroying existing Dai when it is repaid to the position. This cycle of controlled minting and burning allows the contract to account for the total supply of the stablecoin, thus proving that the portfolio of backing collateral can always guarantee the value of circulating Dai.

La cosa interessante delle dApps è che sono totalmente decentralizzate, per cui si possono liberamente vedere gli smart contracts che vengono usati per l'applicazione. Scendendo nella pagina per esempio possiamo vedere il volume e i dati sulle transazioni che coinvolgono MakerDAO, ma anche i principali smart contracts usati.

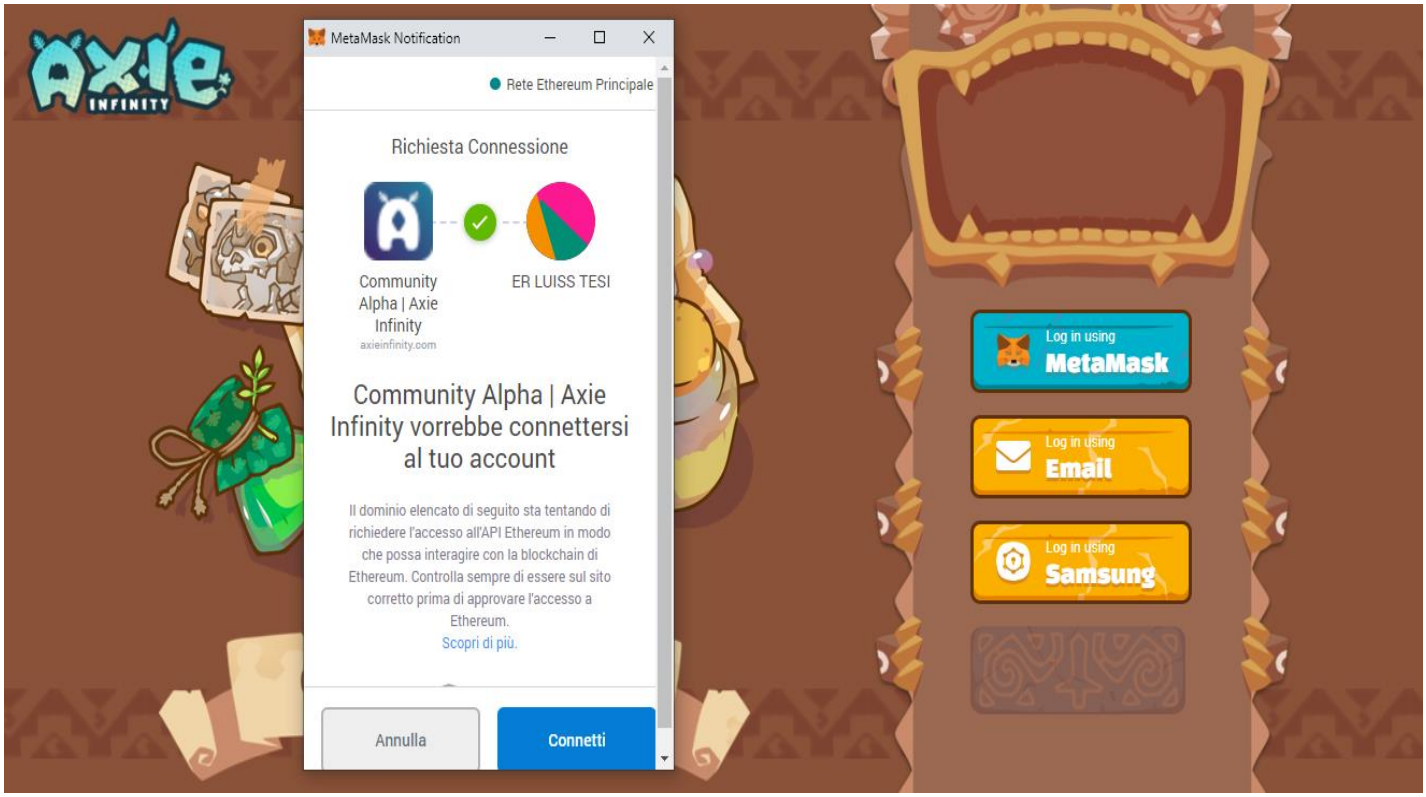


Per capire maggiormente l'importanza di Metamask nell'interazione con una dApp, possiamo vederne una che riguarda il gaming.

Ad esempio, selezioniamo questa applicazione chiamata Axie Infinity, la cui licenza del software appartiene all'MIT, che ci permette di giocare e guadagnare degli ether nel caso in cui riuscissimo a vincere. Questa applicazione funziona totalmente su blockchain ed è regolata interamente tramite smart contracts.



Per esempio, facendo il log in del gioco ci si apre una connessione con Metamask. Infatti, nel caso in cui si voglia iniziare a giocare, dobbiamo per forza investire degli ether per cercare di guadagnarne di più a seconda che si riesca a vincere o meno. Una volta che clicchiamo su "Log in using Metamask" si apre in automatico il nostro wallet Metamask "ER LUISS TESI" che ci chiede di impostare una connessione ed accedere ai nostri ether per iniziare a giocare. [42]



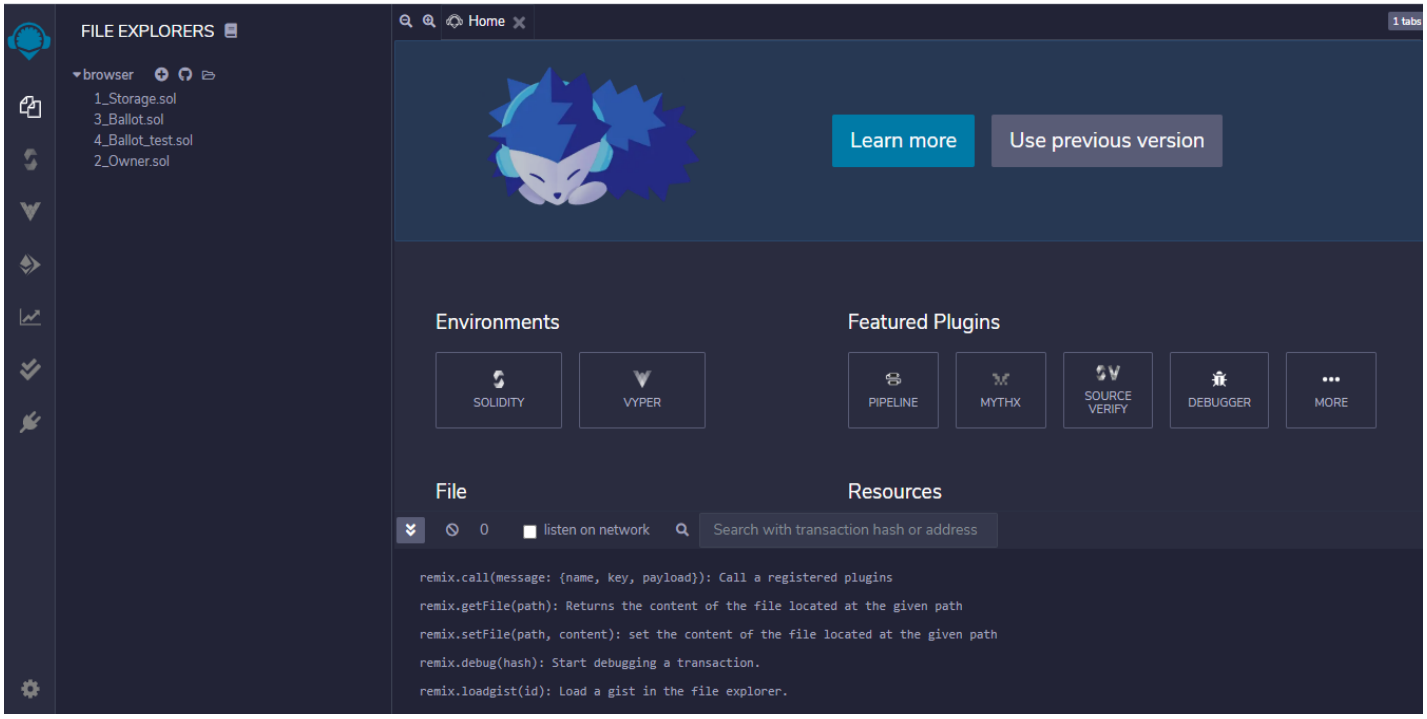
[42] Beginning Ethereum Smart Contracts Programming – Wei-Meng Lee – 2019

4.2. Come scrivere uno Smart Contract

Negli anni Novanta, il crittografo Nick Szabo ha inventato il termine "smart contract" definendo quest'ultimo come "un insieme di promesse, specificate in forma digitale, che comprendono i protocolli all'interno dei quali le parti adempiono alle altre promesse". Da allora, il concetto di contratto intelligente si è evoluto, soprattutto dopo l'introduzione delle blockchain. Si usa anche il termine "smart contracts" per riferirsi a programmi informatici immutabili che funzionano in modo deterministico nel contesto dell'Ethereum Virtual Machine.
Ogni contratto è identificato da un indirizzo Ethereum, che viene definito alla creazione del contratto stesso.
È importante notare che i contratti vengono eseguiti solo se vengono richiamati da una transazione. Tutti gli smart contracts vengono lanciati a causa di una transazione avviata da un EOA (Externally owned accounts). Un contratto può chiamare un altro contratto che può chiamare un altro contratto e così via, ma il primo della lista sarà sempre lanciato inizialmente da una transazione partita da un EOA. I contratti non vengono mai eseguiti "da soli" o "in background".
Le transazioni vengono portate a termine nella loro interezza, con eventuali cambiamenti nello stato globale, solo se tutte le esecuzioni terminano con successo. Affinché ciò accada, il programma deve essere completato senza errori e raggiungere la fine dell'esecuzione. Se questa fallisce, tutti i suoi effetti (cambiamenti di stato) vengono "retrocessi" come se la transazione non fosse mai stata eseguita. Una transazione fallita viene comunque registrata e, l'ether speso per il gas, viene detratto dal conto di origine.
E' importante ricordare che il codice di uno smart contract non può essere modificato. Tuttavia, un contratto può essere "cancellato", eliminando il codice e il contenuto dal suo indirizzo. Qualsiasi transazione inviata a quell'indirizzo dopo la cancellazione del contratto non comporta l'esecuzione di alcun codice da eseguire, perché esso non esiste più. Per cancellare un contratto, si esegue un opcode EVM chiamato SELFDESTRUCT (precedentemente chiamato SUICIDE). Tale operazione costa "gas negativo", cioè un rimborso di gas. La cancellazione di un contratto in questo modo non elimina lo storico delle transazioni, in quanto la blockchain è immutabile. È inoltre importante notare che SELFDESTRUCT sarà disponibile solo se l'autore ha programmato il contratto intelligente per avere tale funzionalità. Se il codice non ha un opcode di SELFDESTRUCT o esso è inaccessibile, lo smart contract non può essere cancellato.
Solidity è stato creato da Gavin Wood come linguaggio per la scrittura di contratti intelligenti per supportarne direttamente l'esecuzione nell'ambiente decentralizzato di Ethereum.
Il "prodotto" principale del progetto Solidity è il compilatore Solidity, "solc", che converte i programmi scritti nel linguaggio Solidity in bytecode EVM. Il progetto gestisce anche l'importante standard ABI (Application Binary Interface) per gli smart contracts Ethereum.
Se Solidity è il linguaggio di programmazione, si adopererà Remix come mezzo per scrivere smart contracts.
Remix è un potente strumento open source che aiuta a creare contratti su Solidity direttamente dal browser. Scritto in JavaScript, Remix supporta anche operazioni di testing, debugging, implementazioni di contratti intelligenti e molto altro ancora. Infatti, Remix è un IDE (ambiente di sviluppo integrato) cioè un software che

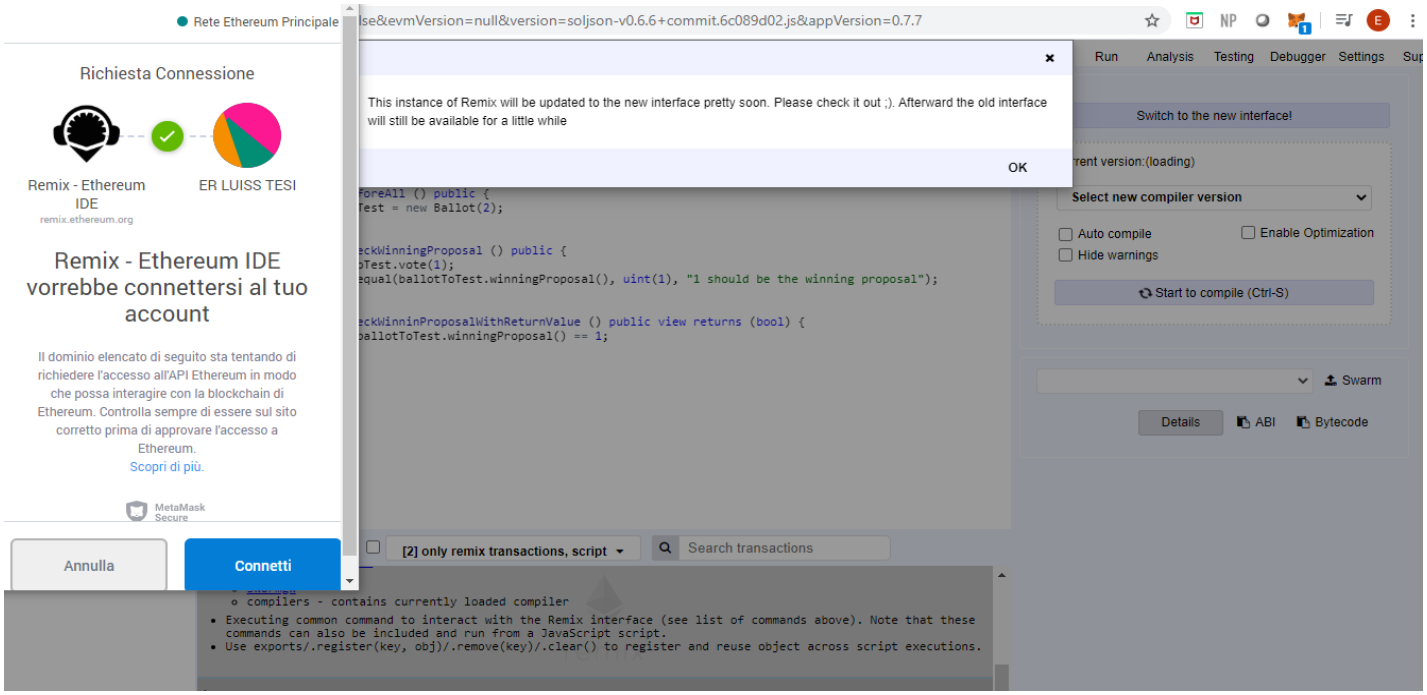
permette ai programmatori di sviluppare il codice sorgente del loro programma e comunica direttamente loro eventuali errori nella compilazione, spiegando dove e in cosa consiste lo sbaglio.

Remix è estremamente comodo perché è un IDE online, quindi non c'è la necessità di scaricare nulla nel computer ma basta accedere a Remix dal sito remix.ethereum.org. La schermata che appare è la seguente



Analizziamo l'ambiente di sviluppo Remix.

Partendo da destra verso sinistra possiamo vedere che è possibile usare la vecchia versione di Remix nel caso in cui si abbia maggiore dimestichezza con quella. Cliccando su "Use previous version" veniamo spostati sulla vecchia versione, che presenta un'interfaccia grafica completamente differente.

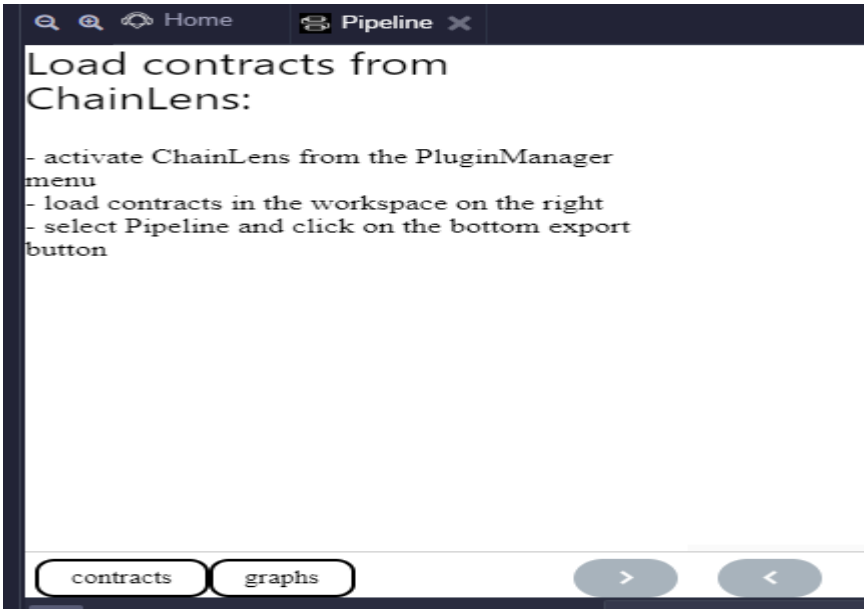


Fatto questo passaggio, il nostro account Metamask "ER LUISS TESI" viene subito riconosciuto e ci viene chiesto di instaurare una connessione al fine di avere gli ether necessari per lanciare smart contracts, dApps e token nella blockchain Ethereum.

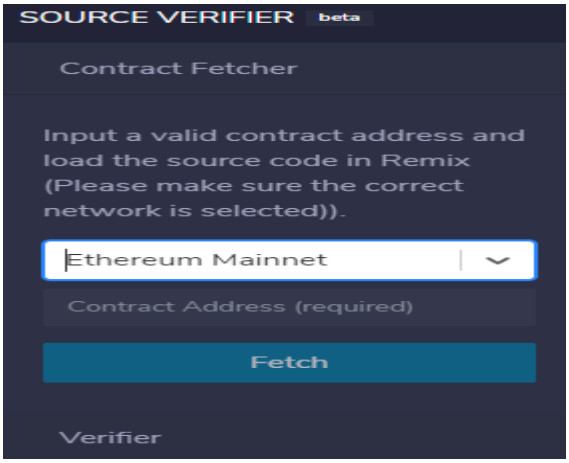
Torniamo all'ultima versione di Remix e, cliccando su "Learn more", come in molti siti, ci viene fornita una panoramica di utilizzo di Remix.

Vediamo le Featured Plugins:

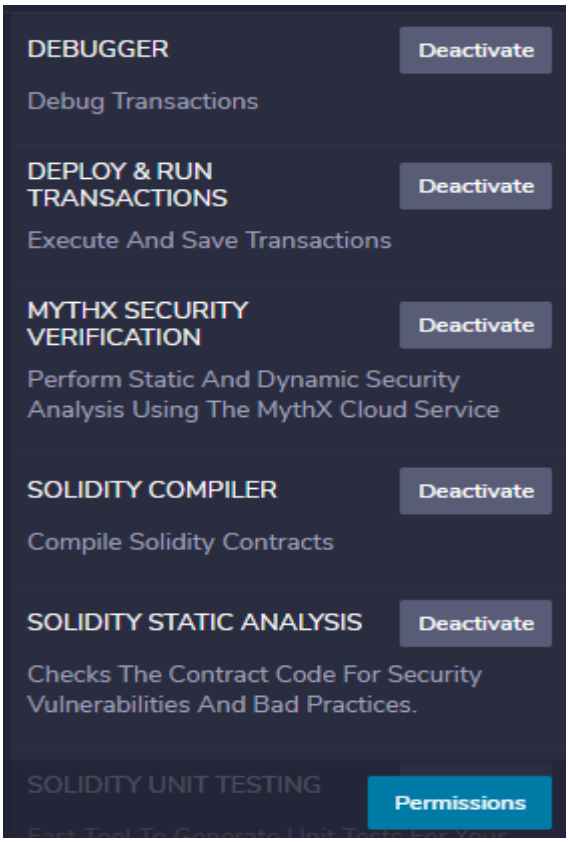
- Pipeline ti consente di caricare contratti da ChainLens e richiede, anche in questo caso, una connessione con Metamask



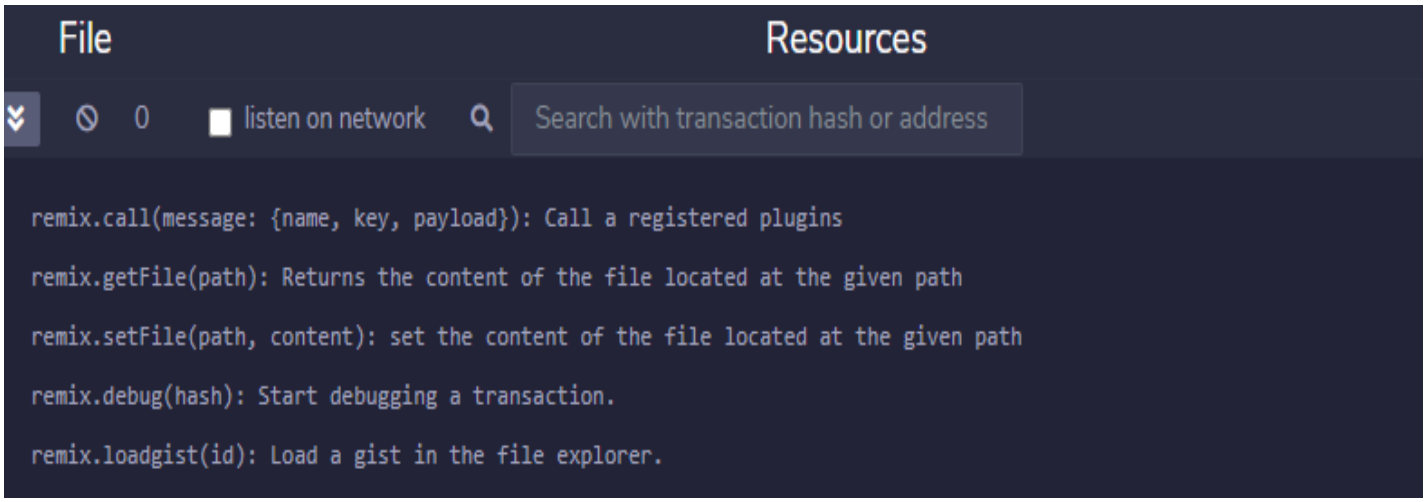
- Mythx è un servizio importante per i programmatori che scrivono smart contracts di elevata difficoltà. Come sappiamo, i contratti intelligenti, una volta che vengono lanciati, non possono più essere ritirati dalla blockchain e diventano immutabili, per cui Mythx è un servizio che offre controllo e verifica sui codici del contratto
- Source verify permette di cercare il tuo smart contract, sia che esso sia stato lanciato sull'Ethereum mainnet, quindi con soldi veri, sia negli ambienti di prova come Ropsten e Rinkeby come abbiamo già visto su Metamask



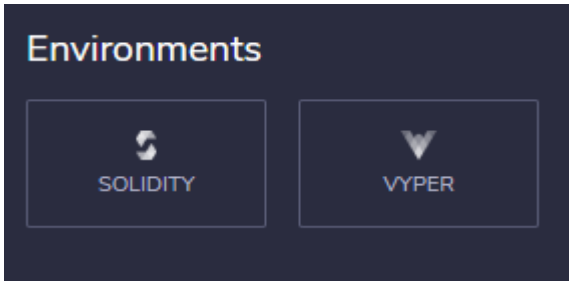
- Debugger è un normale debugger che ti dice se sono presenti bug all'interno dello smart contracts (è una caratteristica comune agli IDE quella di possedere un debugger)
- More permette di personalizzare Remix con le caratteristiche che più si adattano all'applicazione/smart contract che si vuole scrivere



La parte inferiore di Remix è il terminale (Resources) e ti permette di vedere i risultati delle interazioni con l'interfaccia grafica (GUI). Nel terminale è possibile anche eseguire gli script



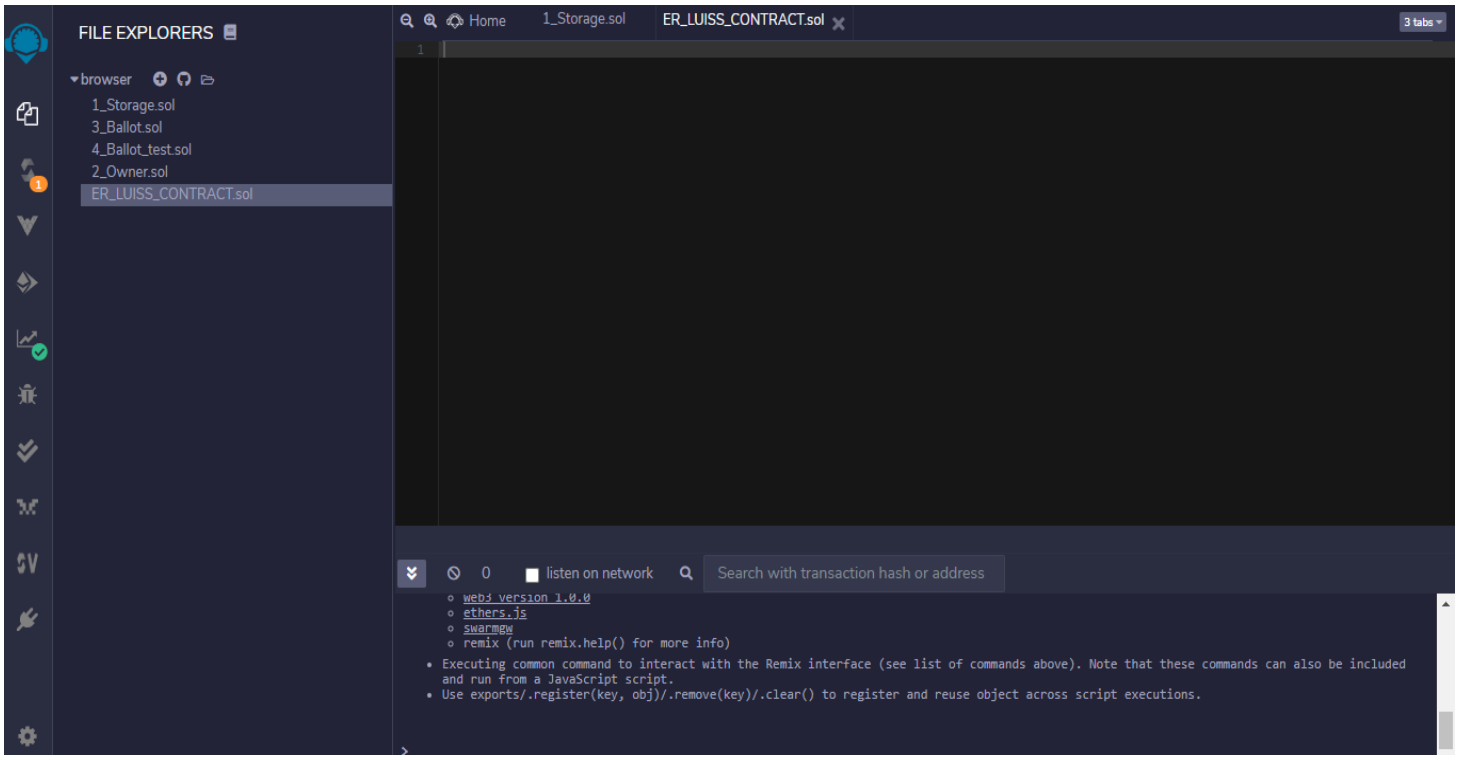
Salendo verso l'alto della pagina viene data la possibilità di selezionare quale linguaggio di programmazione utilizzare: Solidity o Vyper. Di Solidity abbiamo già descritto le principali caratteristiche mentre Vyper è un linguaggio sviluppato più recentemente, simile a Serpent e con una sintassi simile a Python.
Selezioniamo Solidity che è il linguaggio che solitamente si utilizza per la scrittura degli smart contracts.



Ora possiamo analizzare la parte sinistra di Remix che è quella adatta alla stesura vera e propria degli smart contracts, quindi il nostro ambiente di sviluppo.

Creo un nuovo file chiamato “ER_LUISS_CONTRACT” dove con ER intendo sempre le iniziali del mio nome e cognome.

L’ambiente di sviluppo che ci si presenta dopo la creazione del contratto “ER_LUISS_CONTRACT” è il seguente.

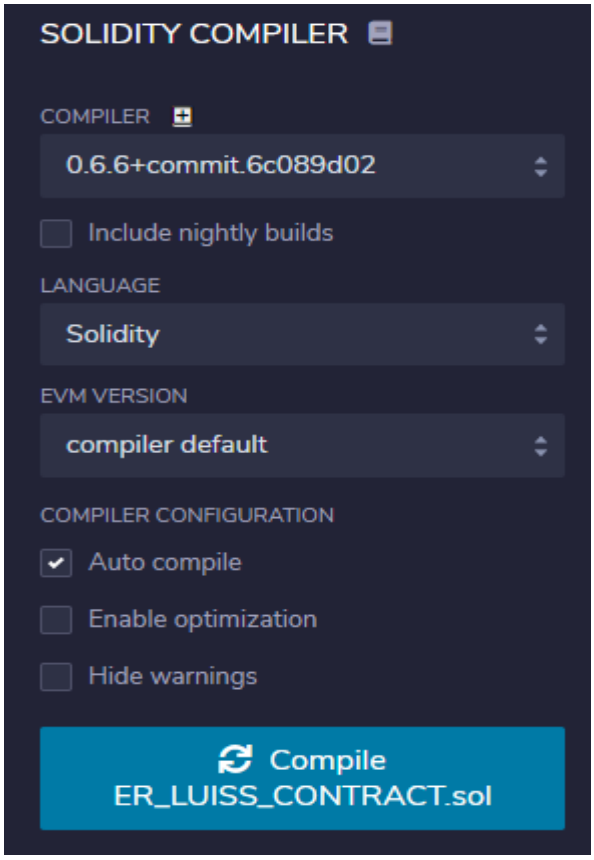


Sotto abbiamo sempre il terminale, al centro sulla superficie nera viene scritto lo smart contract mentre sulla sinistra, sotto ad alcuni esempi che di default mi creano alcuni contratti intelligenti preimpostati, abbiamo il contratto “ER_LUISS_CONTRACT”

Vediamo sulla sinistra alcune delle funzionalità più importanti di Remix.

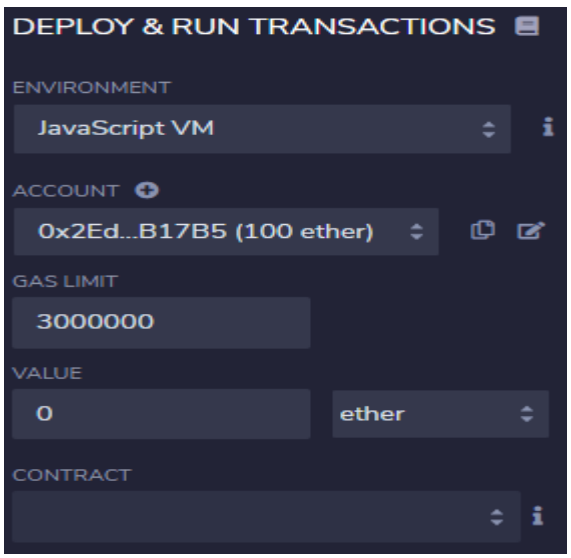
Attraverso “File explorers” ci viene presentata la schermata dei contratti intelligenti che automaticamente Remix fornisce come base di partenza. Si consideri che su Solidity ogni nuovo smart contract deve essere salvato con l’estensione .sol per essere riconosciuto.

Il Solidity complier serve per compilare il nostro contratto una volta che è stato creato. Senza doverlo fare ogni volta che viene apposta una modifica a qualche codice, è consigliato impostare l'Auto compile, che lo fa in automatico.



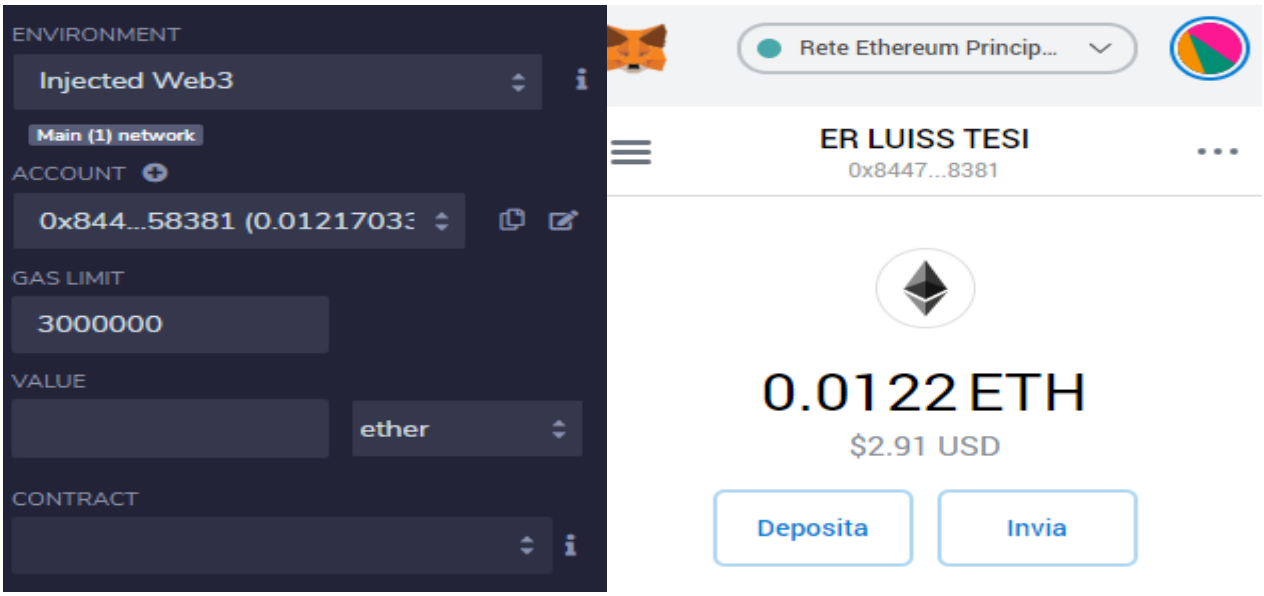
Un altro aspetto importante del compiler è selezionare la versione di Solidity che si vuole usare perché gli aggiornamenti sono molto frequenti e vengono lanciate spesso nuove versioni che modificano alcuni comandi. Per la scrittura dell'ER_LUISS_CONTRACT.sol useremo una versione leggermente meno aggiornata cioè la versione 0.5.11. Si tenga in considerazione che per scrivere contratti intelligenti gli aggiornamenti interessano principalmente programmatori esperti, mentre per contratti semplici un update non fa grande differenza. Sotto la voce "Linguaggio" lasciamo impostato Solidity mentre per l'"EVM version" rimaniamo su quella di default che meglio si presta alla stesura di uno smart contract.

Un'altra parte fondamentale è "Deploy and run transactions"



Per quanto riguarda l'environment si può scegliere tra tre opzioni: non considerando Web3 Provider, che è un sistema che funziona attraverso il collegamento ad un nodo di localhost (per cui è anche abbastanza pericoloso in quanto viene cambiata la connessione), a noi interessano i primi due.

- JavaScript VM (mostrato in schermata sopra) è un ambiente di sviluppo totalmente interno al nostro computer che si basa su ether non reali. JavaScript VM viene utilizzato nel caso in cui si voglia fare delle prove e non si voglia spendere soldi veri. Di default infatti ci vengono creati 15 account finti (nella precedente versione erano 10) che possiedono 100 ether ciascuno. Questo risulta essere particolarmente utile quando per esempio si crea uno smart contract contenente una transazione fake che sposta ether da un account di prova all'altro. Quindi JavaScript VM è un ambiente di prova, in cui le transazioni non vanno a finire in blockchain ma rimane tutto internamente al nostro computer. E' possibile regolare la quantità di gas da usare per far funzionare il contratto intelligente. Solitamente, siccome l'utilizzatore medio non conosce la quantità di gas da usare nel caso in cui volesse registrare una transazione in blockchain, questa informazione viene scritta nel terminale, una volta che il contratto è stato compilato.
- Injected Web3 è invece la funzione che deve essere utilizzata nel caso in cui si volesse lanciare un vero smart contract sulla rete Ethereum.

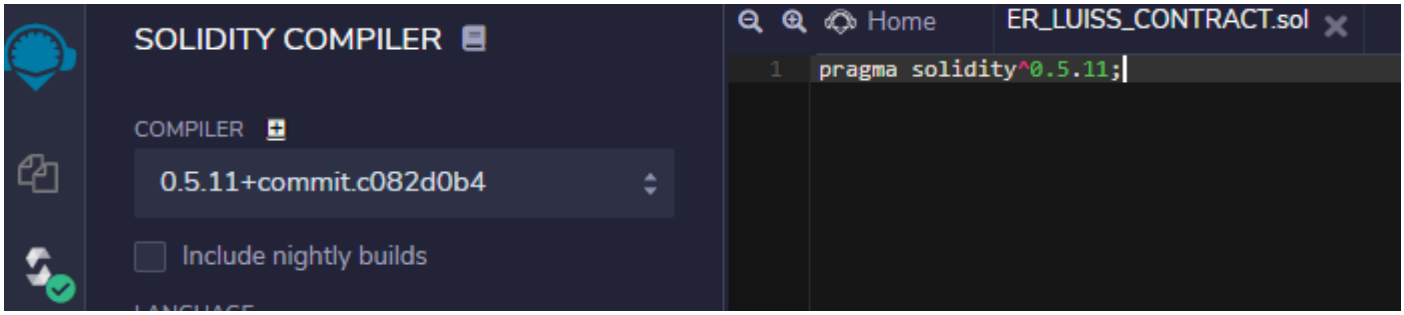


Possiamo vedere che le caratteristiche sono le stesse di JavaScript VM ma in questo caso ho sincronizzato il nostro account di Metamask "ER LUISS TESI" a Remix. Sotto la voce account infatti non abbiamo più i 15 account fake, ma abbiamo un solo account (il nostro) la cui chiave pubblica corrisponde a quella di Metamask (0x844…58381) e tra parentesi ci viene detto anche il numero di ether reali, che sono quelli presenti nel nostro portafoglio su Metamask (0,0122 ETH). Per questo motivo è bene prima studiare lo smart contract su JavaScript VM e poi, una volta sicuri, passare su Injected Web3 e lanciarlo sulla vera blockchain, spendendo soldi reali.

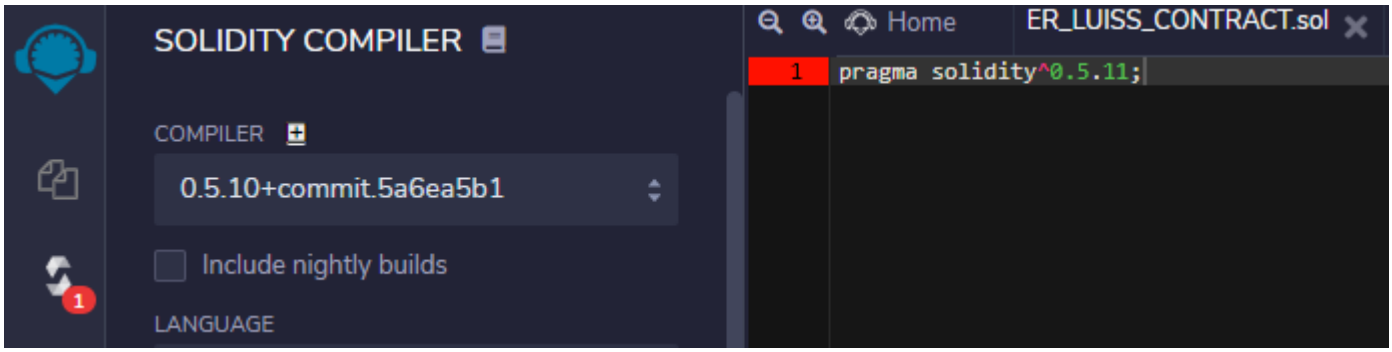
Dopo aver spiegato alcune funzioni di Remix, passiamo alla scrittura vera e propria dello smart contract.

La notazione da usare per l'inizio di ogni contratto intelligente è "pragma" che specifica che il contratto deve essere scritto con una certa versione di Solidity. Questo serve per prevenire degli errori perché, se faccio compile ad un contratto che non è compatibile con quella versione di Solidity, ci viene riconosciuto e segnalato un problema di errata versione in uso.

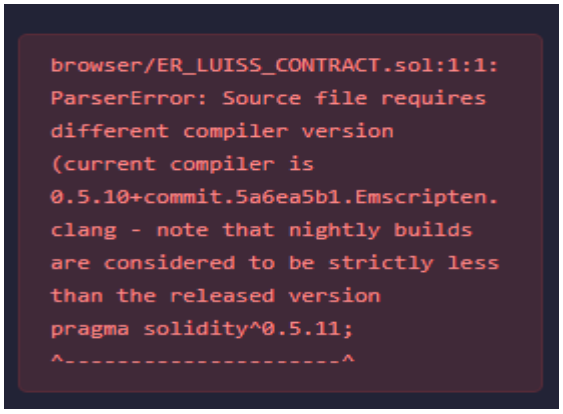
Per il nostro smart contract ER_LUISS_CONTRACT.sol usiamo la versione di Solidity 0.5.11. Questo significa che oltre la versione di 0.5.11, Solidity riconosce la versione 0.5.12, 0.5.13 ecc fino ad arrivare alla versione 0.5.17. Dopo Solidity cambia e passa alle versioni con 0.6.0, 0.6.1, 0.6.2 ecc.. che il nostro smart contract non riconosce. Quindi, le versioni che Solidity riconosce sono quelle maggiori o uguali allo 0.5.11 ma fino ad arrivare alle versioni 0.6. che invece non vengono riconosciute. Inoltre, tutte le versioni precedenti alla 0.5.11 non sono compatibili, anche per esempio la versione 0.5.10 che è quella appena precedente. Concludiamo la notazione con punto e virgola come in ogni linea di codice in Solidity.



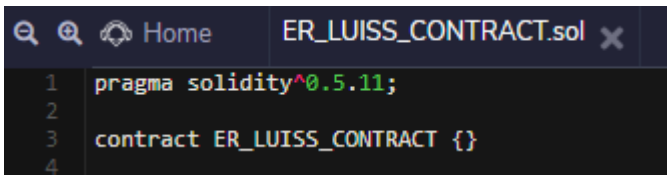
E' importante quindi ricordarsi di aggiornare il compliler con la versione che si sta usando. Laddove questo non succedesse, Solidity è molto funzionale perché ci segnala l'errore. Per esempio, imposto una versione non compatibile con la 0.5.11 ovvero la 0.5.10 e mi appare questa schermata.



Avendo impostato la versione precedente (0.5.10), Solidity non la riconosce e mi segna rosso sia a fianco al pragma, sia nel simbolo del compiler. Per una maggiore comprensione, mi viene specificato in fondo al compiler il motivo dell'errore.



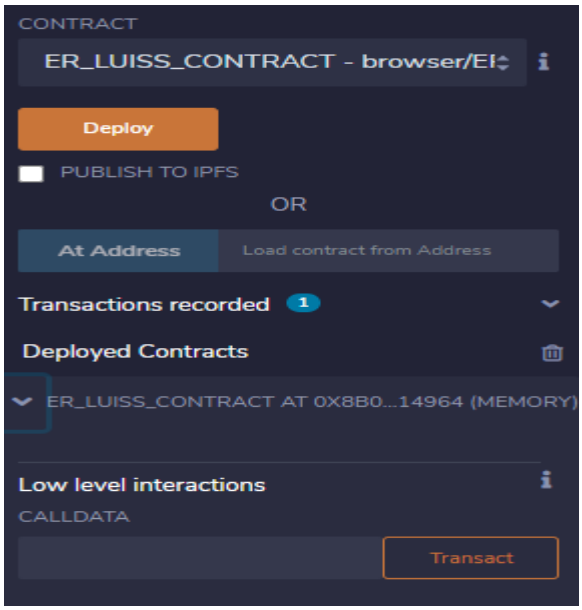
Vediamo alcuni smart contracts base, insieme con alcune funzioni che vengono comunemente usate su Solidity per la stesura di contratti intelligenti.



Con la funzione “contract” ho definito il contratto. Questo risulta essere particolarmente utile nel caso in cui avessi molti smart contracts che voglio “chiamare” e dovessi scegliere quale usare in quel momento. Nel nostro caso ne ho solo uno, per cui chiamo ER_LUISS_CONTRACT ma, se volessi usarne altri, posso farlo con la funzione contract. In qualsiasi situazione le linee di codice di uno smart contract devono essere scritte tra le parentesi graffe, che indicano l’inizio e la fine di un contratto intelligente su Solidity. In questo caso il contratto è vuoto ma rappresenta comunque un contratto!

Infatti, appena scritto, posso vedere che sotto mi compaiono alcune possibilità con cui mi sarebbe possibile lanciarlo sulla rete.

Inoltre, posso passare alla sezione “deploy and run transactions” dove, completato lo smart contract, lo faccio funzionare (in questo caso non è particolarmente utile perché il contratto è vuoto). Cliccando su deploy eseguo il contratto che mi appare sotto nella sezione deployed contracts.



I contratti, una volta fatto il deploy, vengono chiamati “smart contract instances”. Mi viene anche data la possibilità, cliccando su deploy, di eseguire molte instances dello stesso contratto, dove tutte avranno un codice diverso. Questo risulta essere utile perché per esempio posso lanciare due smart contracts che hanno le stesse funzionalità (cliccando due volte su deploy). Considerando che all’interno degli smart contracts posso trascrivere alcune informazioni, nel caso in cui in futuro avessi bisogno di aggiornare alcuni dati presenti solamente in uno dei due contratti, l’altro rimarrebbe indipendente dal cambiamento fatto. Pertanto, avendo due codici diversi, posso cercarli in maniera separata in blockchain. Ora non ci sono interazioni, ma nei prossimi smart contracts posso interagire con le funzioni e i codici che ho creato nel contratto.

Vediamo come possono essere divise le funzioni su Solidity.

Home ER_LUISS_CONTRACT.sol

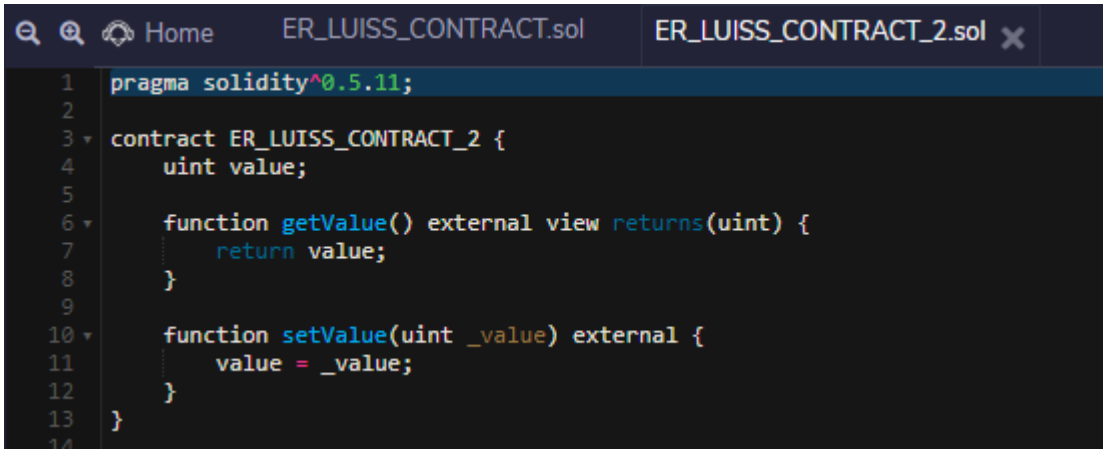
```
pragma solidity^0.5.11;

contract ER_LUISS_CONTRACT {
    //1. fixed-size types
    bool isReady;
    uint a;
    address indirizzo;
    bytes32 informazione;

    //2. variable-size types
    string nome;
    bytes _informazione;
    uint[] quantita;
    mapping(uint => string) users;

    //3. used-defined data
    struct user {
        uint id;
        string Emanuele;
        uint[] idamici;
    }
    enum LUISS{
        scienze_politiche,
        giurisprudenza,
        economia
    }
}
```

Ho diviso le variabili in 3 categorie:

1. Fixed-size types sono delle variabili che vengono registrate come un numero fisso in memoria. Alcuni esempi sono:
    - Variabili booleane ci dicono se un valore è vero o falso. Ho scritto “isReady” ma bool poteva essere usato per qualsiasi situazione in cui serviva un vero/falso
    - Uint ovvero unsigned integer è una funzione davvero molto usata perché ogni volta che viene fatta una transazione (come per esempio invio di ether) utilizzo uint. Lo posso adoperare solo con numeri interi e positivi
    - Address indica l’indirizzo con cui si può interagire. Può indicare un indirizzo verso cui è rivolta una transazione o un indirizzo di uno smart contract a cui si vuole inviare un messaggio; gli indirizzi Ethereum sono di 20-bytes
    - Bytes32 serve per rappresentare qualsiasi informazione binaria casuale in 32-bytes. Spesso se i programmatori sanno che la dimensione non eccede i 32-bytes, può essere utile usare la notazione bytes32 anzi che string

2. Variable-size types sono invece variabili che registrano dei valori di cui non si sa la lunghezza esatta
    - String serve per rappresentare una stringa di qualsiasi dimensione
    - Bytes è una generalizzazione di bytes32 che viene usata quando non si conosce esattamente la dimensione dei bytes
    - Uint [] serve per rappresentare un vettore o una matrice. Si possono avere matrici formate dallo stesso tipo di variabile, quindi per esempio non può succedere che il primo valore sia un intero mentre il secondo sia un booleano
    - Mapping serve per combinare le matrici, per esempio quando si vuole combinare una matrice composta da numeri interi con una composta da stringhe
3. User-defined data si usano quando ci serve dare una rappresentazione personalizzata delle informazioni
    - Struct serve per raccogliere i dati definiti dall'utente per il raggruppamento delle variabili: per esempio io ho raggruppato una variabile che mi fornisce gli id, una stringa con il mio nome, una matrice con gli id dei miei amici ecc..
    - Enum è utile per raggruppare le variabili discrete definite dall'utente. Per esempio, come macrocategoria ho messo la LUISS mentre poi ho selezionato alcune sottocategorie che corrispondono alle lauree triennali presenti in LUISS. La funzione enum può essere usata per delle classificazioni/etichette.

Per riuscire ad interagire con i dati che vengono scritti all'interno dello smart contract, bisogna creare delle funzioni. Mi creo un nuovo contratto ER_LUISS_CONTRACT_2.sol e lo chiamo con la funzione contract.



Successivamente vado a scrivere un uint che chiamo genericamente value.

Ora vado a creare la prima funzione, che in Solidity si fa con "function"; ipotizziamo una funzione che per esempio mi permette di ottenere un dato valore, che posso chiamare getValue. In seguito, devo aggiungere external che indica una funzione che può essere chiamata o da altri contratti o da un EOA, ma non internamente al contratto (diversamente da public che rappresenta la funzione usata di default)

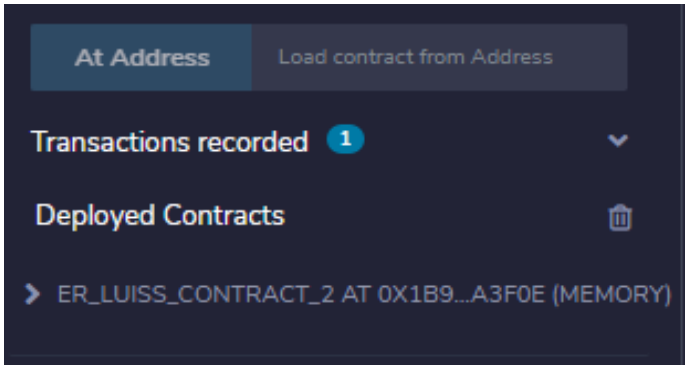
Dopo aggiungo view che mi indica che non vado a modificare alcuno stato della blockchain e per finire returns con il valore che voglio avere (nel nostro caso era uint chiamato value).

Ora creo una seconda funzione che mi dà la possibilità di poter cambiare questo valore; la chiamo function setValue e tra parentesi voglio poter settare il valore della variabile value. Tuttavia, nel caso in cui la chiamassi semplicemente value creo confusione con la variabile già definita sopra, per cui metto un underscore davanti (_value).

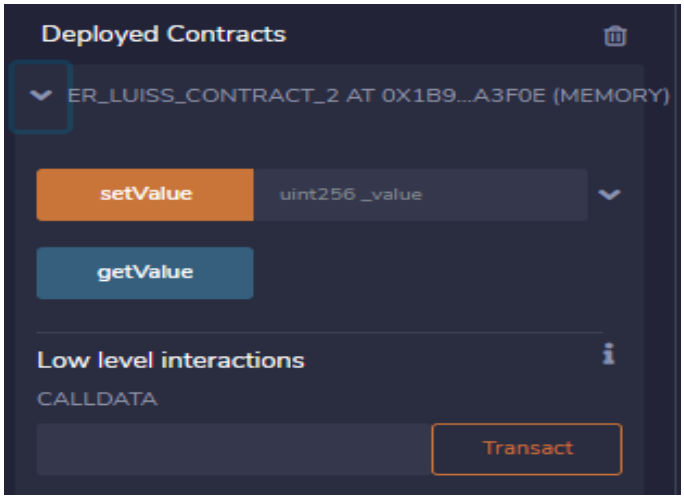
Notiamo che, dato che vogliamo poter cambiare la variabile (per questo stiamo facendo setValue) non dobbiamo scrivere la funzione view poichè ora vogliamo modificare la blockchain. Similmente per return, perchè non vogliamo che la funzione ci dia indietro nulla (differentemente da prima con getValue che volevamo un valore indietro).

Infine, value = _value perché la variabile a cui ci stiamo riferendo è la stessa (non sono due funzioni che lavorano separate)

Lo smart contract è stato creato e per renderlo funzionante clicco su deploy.

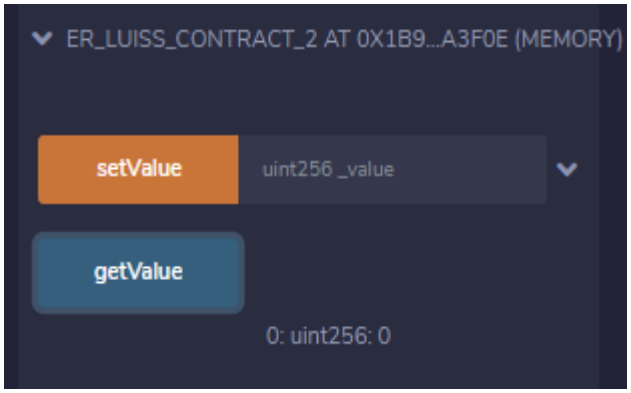


Ci compare il nostro contratto e cliccando sulla freccetta, ci appare la seguente schermata.

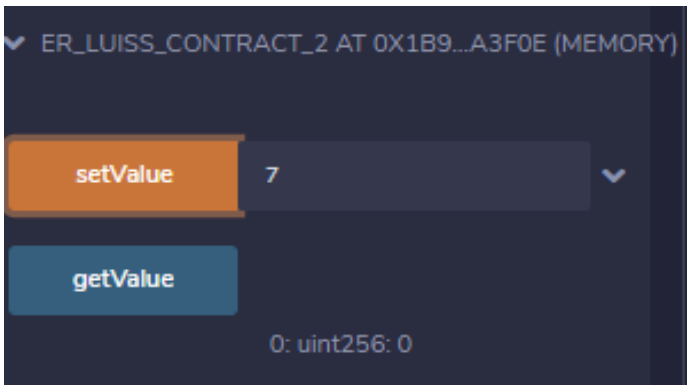


Ecco che il contratto sta funzionando. Mi si aprono due possibilità in cui posso cliccare: getValue e setValue.

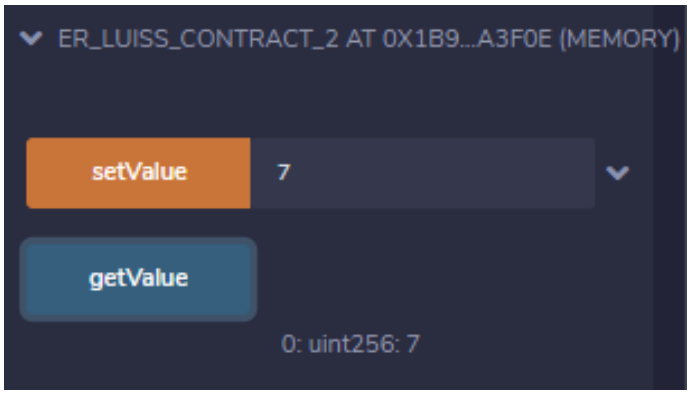
Clicco su getValue



Possiamo vedere come, non avendo impostato alcun tipo di valore, di default mi appare 0. Mi viene fornito un uint256 perché, avendo scritto nella funzione solo uint, in automatico mi si imposta su un uint a 256 bytes.
Ora inserisco invece un valore su setValue nello spazio in cui si può scrivere; metto un numero casuale intero positivo, per esempio 7, e clicco su setValue.



Verifichiamo se lo smart contract funziona e ci restituisce il valore che abbiamo settato cliccando su getValue.



Dopo aver cliccato su getValue possiamo vedere che funziona: infatti ci viene restituito il numero 7 che era lo scopo per cui era stato scritto questo contratto. Per ora abbiamo lavorato su un circuito interno di prova (Javascript VM), ma se si volesse lanciare sulla blockchain reale bisognerebbe passare a Injected Web3.

E’ interessante notare come il terminale sotto segua e registri le operazioni che stiamo svolgendo, fornendoci i dettagli.
Rappresenta una casualità che getValue sia di colore blu, mentre setValue sia arancione? No, perché getValue è una funzione read-only, che quindi ci può dare solo un output, mentre setValue, di colore arancione, è una funzione che ci permette di modificare la blockchain impostando un numero a nostro piacimento. Questa capacità viene sancita dalla funzione view: se in una funzione viene scritto view, significa che questa è solo una funzione read-only non in grado di modificare la blockchain (nelle vecchie versioni questa funzione era

chiamata constant). Nel caso in cui avessimo deciso di fare return su un'operazione matematica anziché su un numero, avremmo dovuto usare la funzione "pure".
Nel nostro smart contract abbiamo usato la funzione external vedendone già il significato. Un'altra funzione che permette di chiamare le funzioni del contratto solamente dall'interno è "private"; laddove avessi usato questa funzione avrei avuto solamente la funzione setValue e quindi non avrei potuto avere il risultato finale. Un'altra possibilità, che rende settabile il valore anche da colui che usa lo smart contract e non solo da chi lo scrive, è la funzione public che però significa che posso chiamare lo smart contract anche da dentro. Posso infatti impostare una funzione di getValue sotto a quella di setValue e scegliere un numero anche da dentro i codici del contratto. Nel nostro caso invece, prima clicco su deploy e dopo imposto il numero.

Vediamo le variabili "built-in" che si usano molto spesso negli smart contracts. Creo un altro contratto, il terzo, chiamato ER_LUISS_CONTRACT_3.sol

```
pragma solidity^0.5.11;

contract ER_LUISS_CONTRACT_3 {

    //transazioni (tx)
    tx.origin
    tx.gasprice

    //messaggi (msg)
    msg.value
    msg.sender

    Emanuele => Smart contract A        =>     Smart Contract B
                tx.origin = Emanuele           tx.origin = Emanuele
                msg.sender = Emanuele          msg.sender = Smart contract A

    //blocchi (block)
    block.timestamp
    block.number

}
```

Dividiamo le variabili Built-in in 3 tipologie:

- Transazioni (tx) sono quelle variabili che forniscono un mezzo per accedere alle informazioni relative alle transazioni. Per esempio tx.origin dice l'indirizzo dell'EOA per quella transazione, mentre tx.gasprice enuncia il prezzo del gas nella transazione chiamata
- Messaggi (msg) sono le transazioni o i messaggi che hanno lanciato l'esecuzione del contratto. Per esempio msg.value indica il valore dell'ether inviato (misurato in wei) oppure msg.sender che rappresenta l'indirizzo che ha dato inizio a questo contratto, non necessariamente l'EOA di origine che ha inviato la transazione. Se il nostro contratto è stato chiamato direttamente da un EOA, allora questo è l'indirizzo che ha firmato la transazione, altrimenti sarà un indirizzo che proviene da un contratto. Per capire meglio la differenza tra tx.origin e msg.sender si veda l'esempio sopra. Ipotizziamo che Emanuele chiami un contratto e mandi una transazione allo smart contract A. In questo caso sia il tx.origin che msg.sender coincidono con Emanuele. Assumiamo ora che sia lo smart contract A che

chiama lo smart contract B. Il tx.origin rimane Emanuele, che è colui che ha firmato la transazione iniziale, ma l'msg.sender in questo caso diventa lo smart contract A.

Con la stessa logica potremmo proseguire nell'esempio ipotizzando lo smart contract C chiamato da B con il tx.origin che rimarrebbe Emanuele mentre il msg.sender sarebbe smart contract B e così via.

- Blocchi (block) contengono informazioni inerenti ai blocchi correnti. Per esempio possiamo vedere block.timestamp che ci dice il timestamp apposto dal miner mentre block.number ci dice il numero del blocco corrente. [43] [44] [45]

[43] Smart Contracts: How to understand Smart Contracts and Be Ahead of Competition. Learn About the Future of Blockchain Technology – Ruth Chandler – 2017

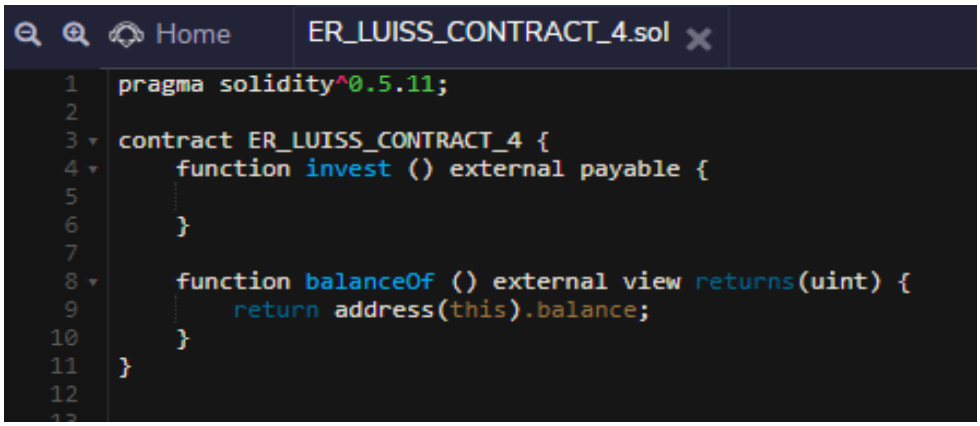
[44] How to write a simple Smart Contract – Morgan Fogarty – 2018

[45] Ethereum for Architect and Developers – Debajani Mohanty - 2018

5. Esempi di smart contracts

5.1. Inviare fondi ad uno smart contract

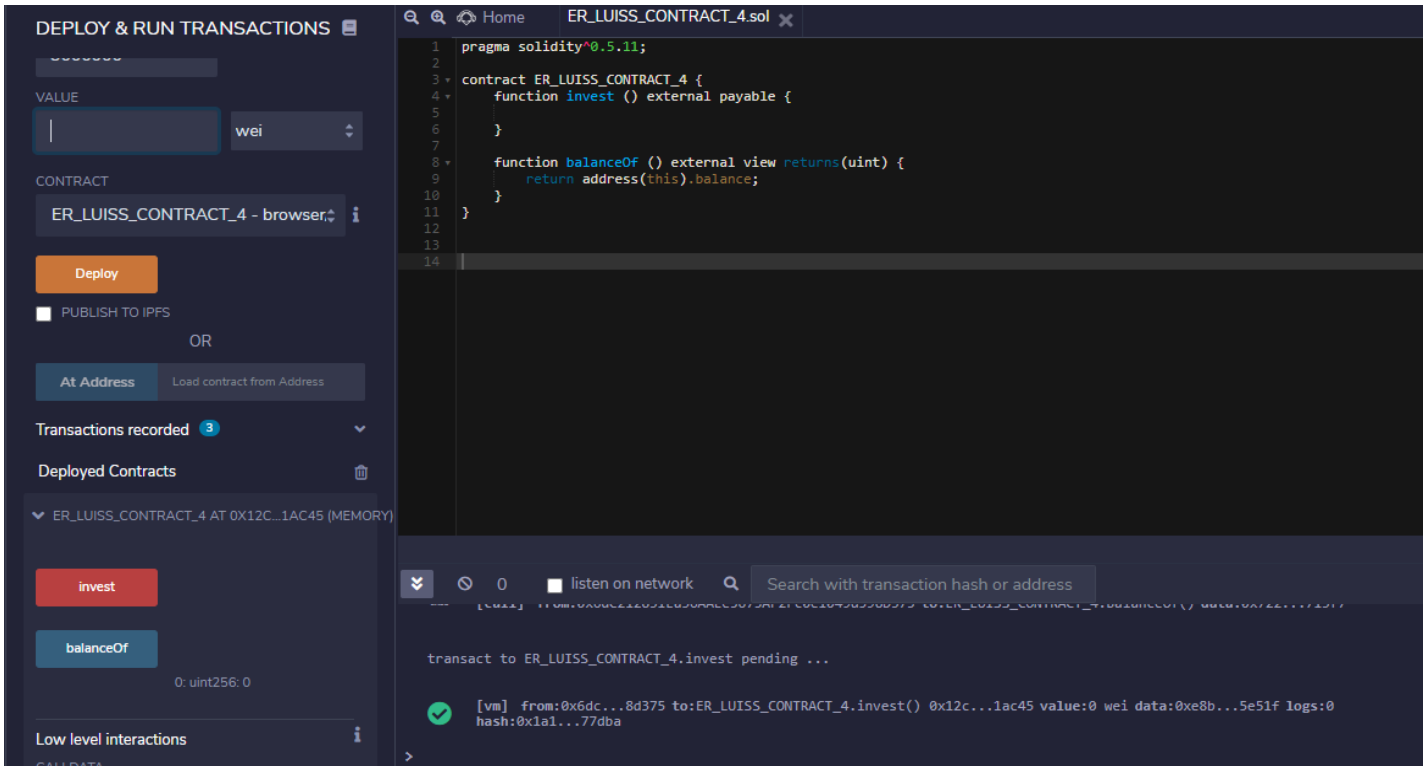
Vediamo un contratto che permette di mandare ether da un portafoglio ad uno smart contract. Creo l'ER_LUISS_CONTRACT_4.sol e il contratto sarà:



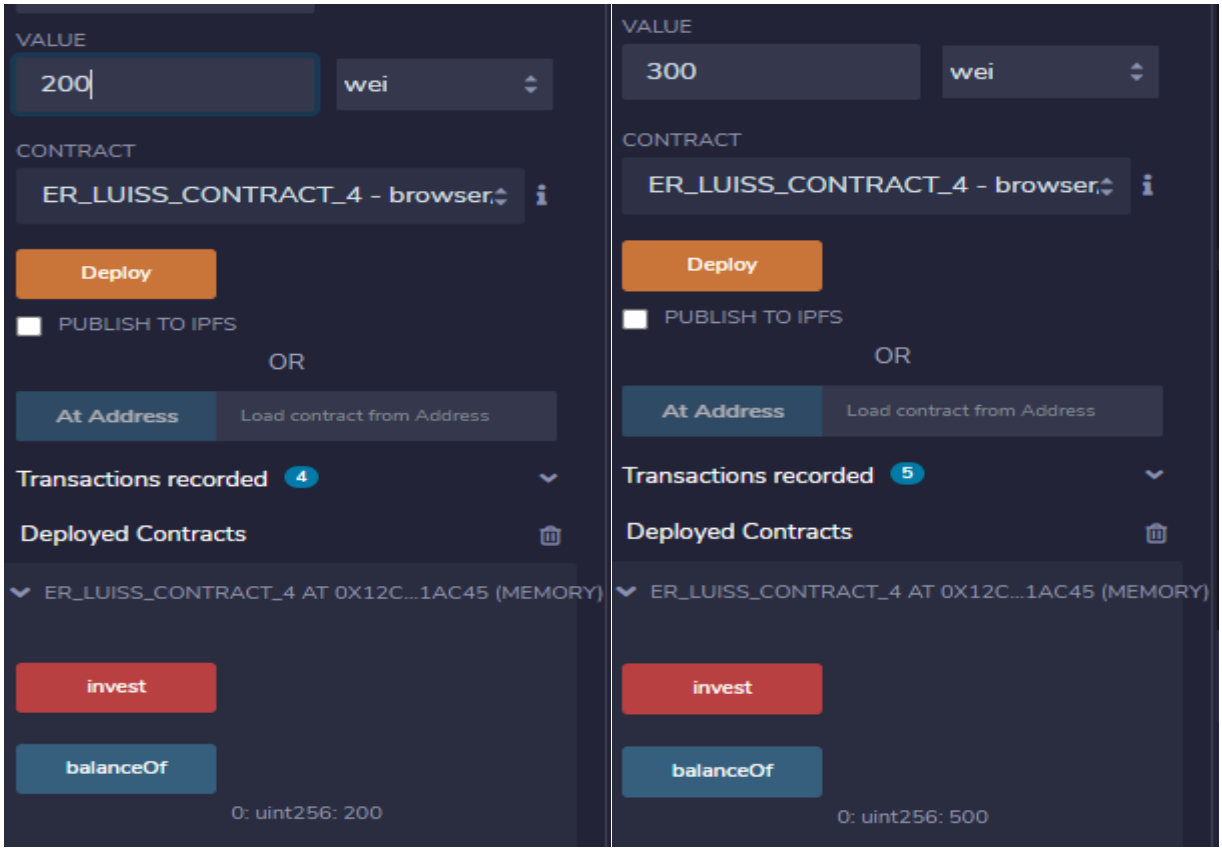
Vediamone le caratteristiche fondamentali. Mi creo la funzione invest con cui posso decidere di inviare wei o ether all'interno del contratto, external perché li invio da fuori rispetto al contratto e payable che mi permette di inviare fondi.

Poi aggiungo una funzione che mi aggiorna il bilancio ogni volta che invio fondi (si chiama balanceOf), aggiungo view perché è una funzione di sola lettura e returns(uint) perché voglio che mi dia come output un ammontare di fondi costituiti da un numero intero positivo. Concludo con lo specificare che i fondi devono essere inviati allo smart contract mediante la funzione address(this).balance.

Vediamo ora se funziona.



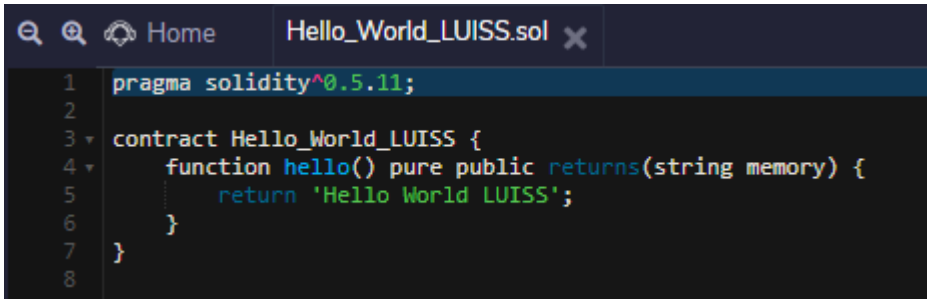
Lanciamo il contratto con deploy e clicchiamo la freccetta per accedere alle sue funzionalità. Vado su invest e vediamo a fianco di balanceOf che per ora il contratto è vuoto e non presenta fondi. Decido di inviare 200 wei al contratto.



Il contratto funziona! Nella schermata di sinistra carico 200 wei, clicco su invest e poi su balanceOf e posso vedere che il bilancio del conto è stato aggiornato. Provo ad aggiungere altri 300 wei con lo stesso procedimento fatto ora. Il conto, come si vede nella schermata di destra, è stato aggiornato e infatti ora presenta un ammontare di 500 wei totali.

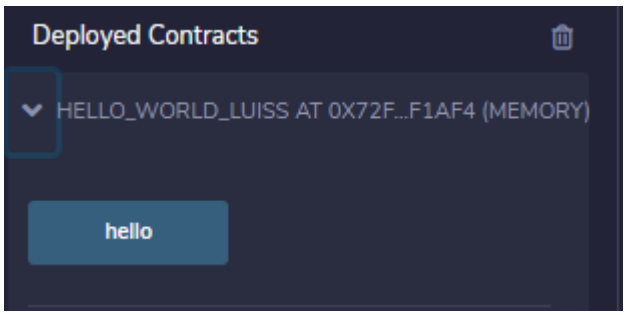
5.2. Contratto Hello World

Creo un nuovo contratto che chiamo Hello_World_LUISS.sol. L'obiettivo è quello di ideare un contratto che mi restituisca la stringa "Hello World LUISS" che mi sono inventato come prova.

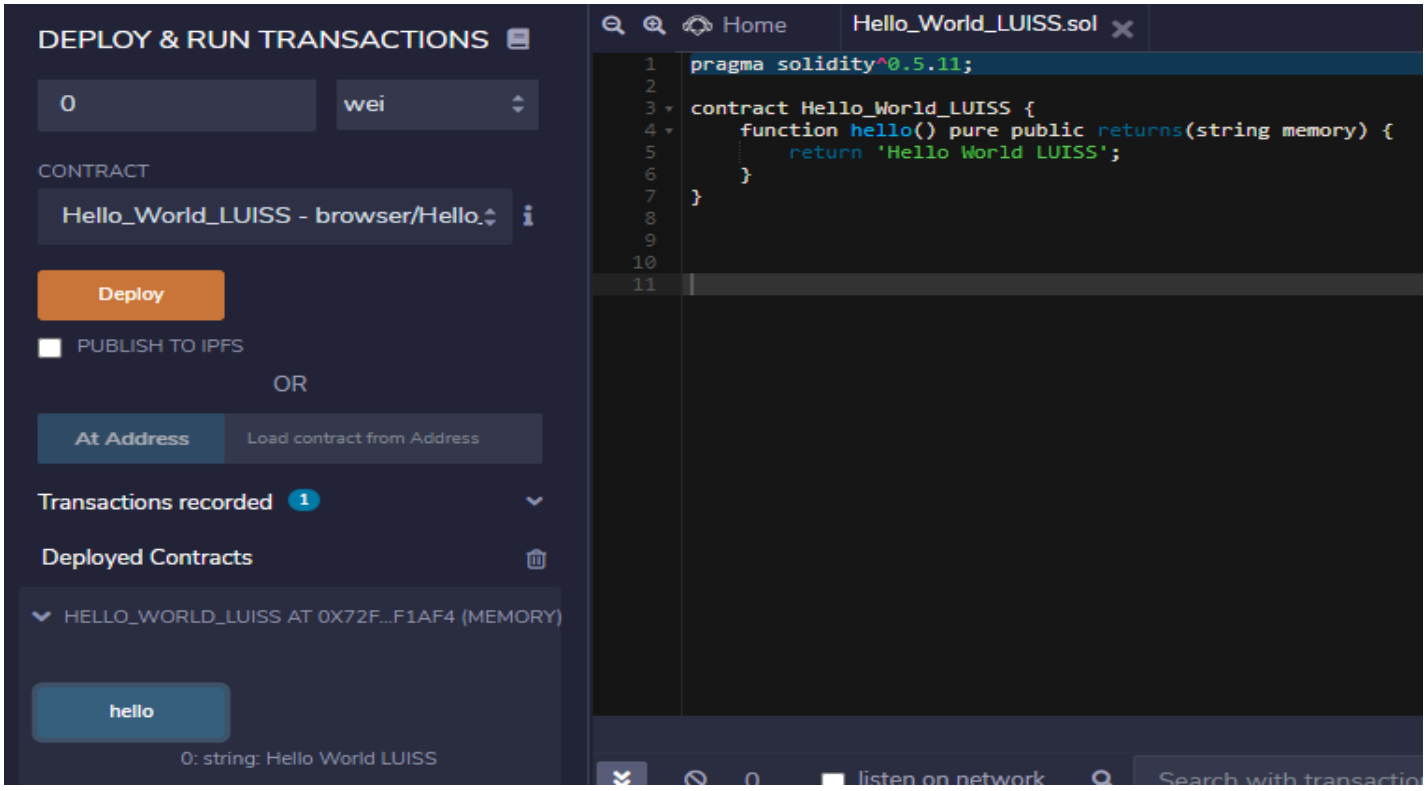


Come al solito chiamo il contratto nello stesso modo in cui ho creato l'estensione.sol e mi scrivo una funzione che chiamo casualmente hello senza oggetto (apro e chiudo parentesi). La funzione è public perché la voglio poter chiamare da fuori lo smart contract e non voglio modificare la blockchain ma solo avere indietro la stringa, quindi uso la funzione pure. Concludo la riga di codice con returns (con la s finale) e voglio che la

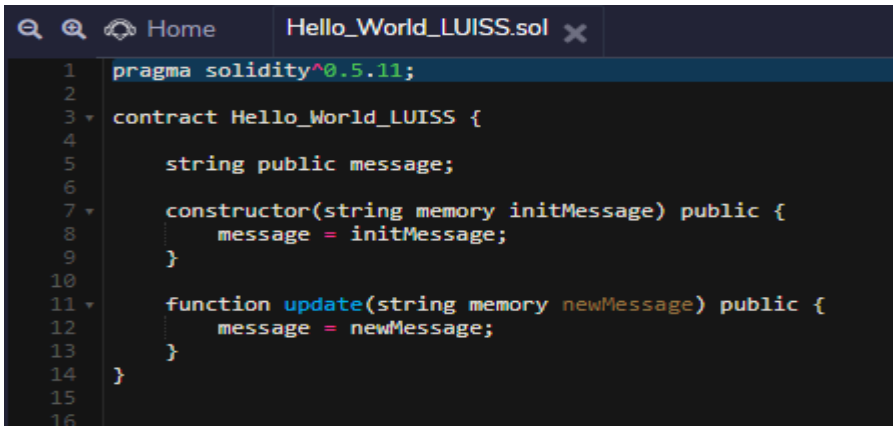
stringa mi rimanga in memoria (string memory). Come return desidero la frase ‘Hello World LUISS’; finito il codice faccio deploy.



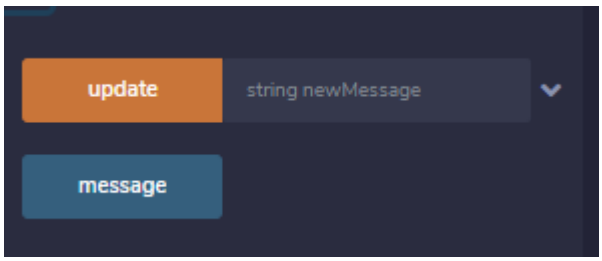
Come mi aspettavo, avendo chiamato la funzione hello, mi si apre un pulsante di colore blu denominato in questo modo e cliccandoci appare ciò che stavo cercando ovvero la stringa ‘Hello World LUISS’



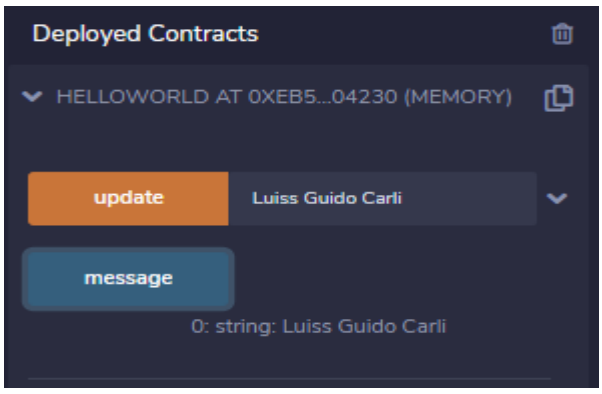
Vediamo un altro tipo di contratto leggermente più complicato. Cancello il codice di prima e sempre nel file Hello_World_LUISS.sol scrivo il seguente contratto. L’obiettivo è quello di avere indietro una frase non scritta dentro al codice, come nell’esempio precedente (‘Hello World LUISS’), ma scritta fuori dal codice e che posso cambiare liberamente ogni volta.



La prima riga di codice dice che stiamo creando un messaggio sotto forma di stringa pubblica. Constructor è una funzione eseguita durante la creazione del contratto che può essere richiamata in seguito e memory mi permette di memorizzare il messaggio. Ci creiamo poi la funzione update che ci permette di aggiornare il messaggio quando vogliamo.



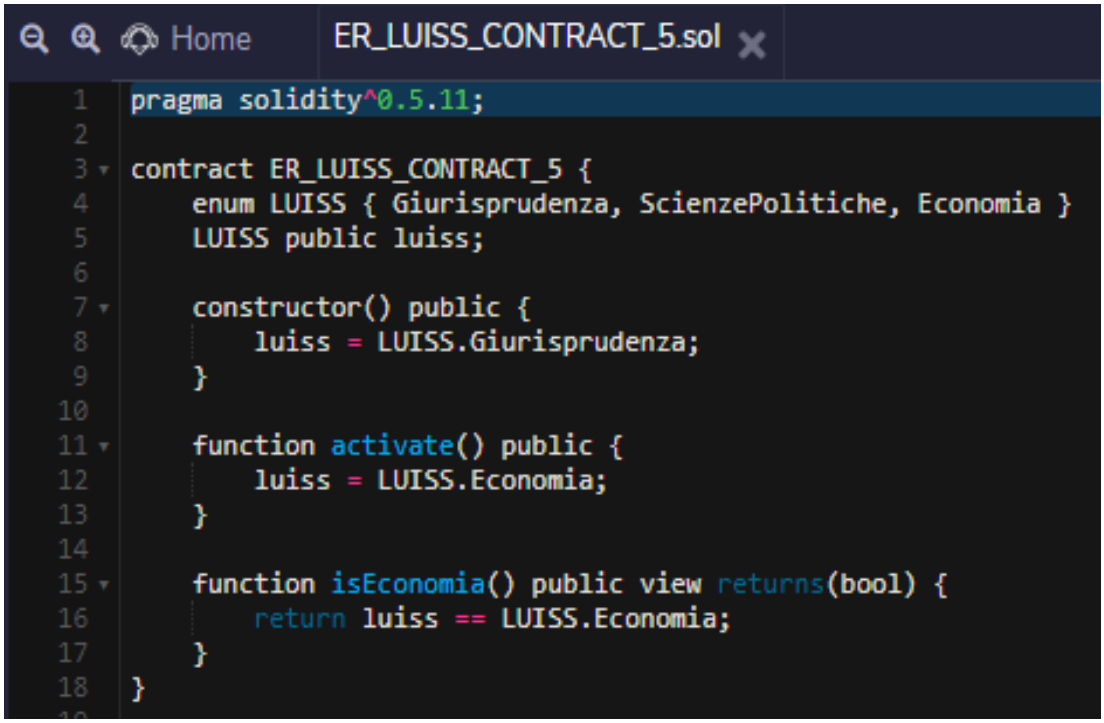
Come abbiamo già visto, update permette di aggiornare e digitare ciò che vogliamo mentre message ci dà solo un output. Inserisco come prova la stringa “Luiss Guido Carli” e faccio update.



Il contratto funziona e il messaggio è proprio quello che ho inserito.

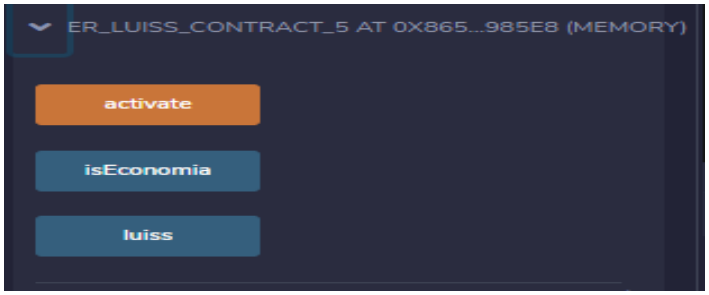
5.3. Smart contract con la funzione enum e bool

Una funzione enum è una funzione che ti permette di creare una lista numerata e tenere traccia di un set di informazioni presenti nelle liste del nostro smart contract. Vediamo come si usa.



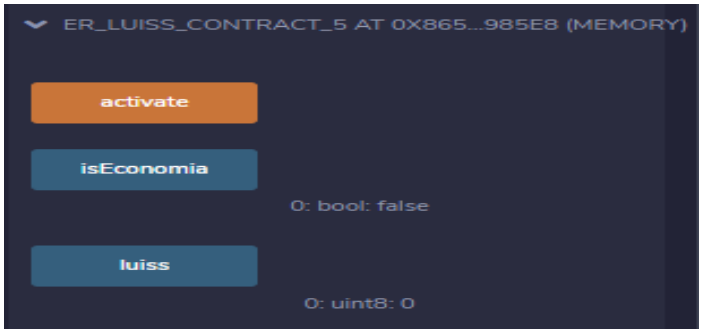
Utilizzo subito la funzione enum per dividere le lauree triennali della LUISS in Giurisprudenza, Scienze Politiche ed Economia, in modo che poi possiamo usare separatamente le informazioni contenute in ogni rispettiva lista. In seguito, inserisco una funzione public che mi permette di vedere questa funzione anche fuori dal contratto.

Creata la lista, il passaggio dopo è scegliere lo stato iniziale di partenza con cui voglio fare delle operazioni. Dopo decido di cambiare set per cui mi creo una funzione che risponde al comando activate che mi sposta verso Economia. Infine creo la funzione isEconomia per vedere se il contratto è effettivamente traslato dal set di Giurisprudenza a quello di Economia e per capirlo uso un indicatore booleano; come ultima cosa voglio che il mio output sia LUISS:Economia per cui lo metto con il doppio = ovvero che il Vero del booleano sia uguale al set di Economia. Lancio il contratto con deploy e mi appare questa schermata.

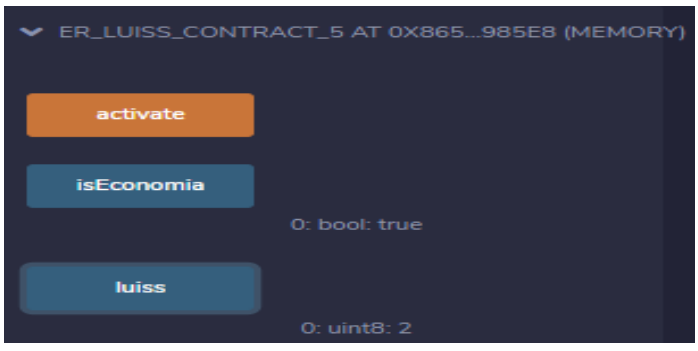


Se clicco su isEconomia succede che l'operatore booleano mi dà falso perché nella prima riga avevo impostato come constructor iniziale Giurisprudenza, mentre a fianco al pulsante luiss ho un uint a 8 bytes (impostato in maniera automatica) e lo 0. Lo zero mi indica che è stato preso il valore 0 della lista a disposizione della categoria LUISS composta da 3 set: in Solidity, le liste iniziano con 0 per cui il primo valore che è Giurisprudenza = 0, ScienzePolitiche = 1, Economia = 2.

Clicchiamo su activate per cambiare verso Economia.



Come ci aspettavamo l'operatore booleano in questo caso mi dà true perché come return avevamo impostato che luiss == LUISS.Economia per cui il set finale dove il booleano restituisce vero deve essere Economia e anche a fianco a uint mi compare il 2, perché è stato scelto il secondo valore della lista.



5.4. Contratto per comprare token e mandare ether ad un wallet

Vediamo come creare un contratto che ci permetta di comprare dei token simulando quello che succede in una ICO e inviare fondi ad un wallet. In questo caso non vado a scrivere il codice di creazione della moneta ma lo faccio in maniera generica con la funzione buyToken.

Home ER_LUISS_CONTRACT_6.sol

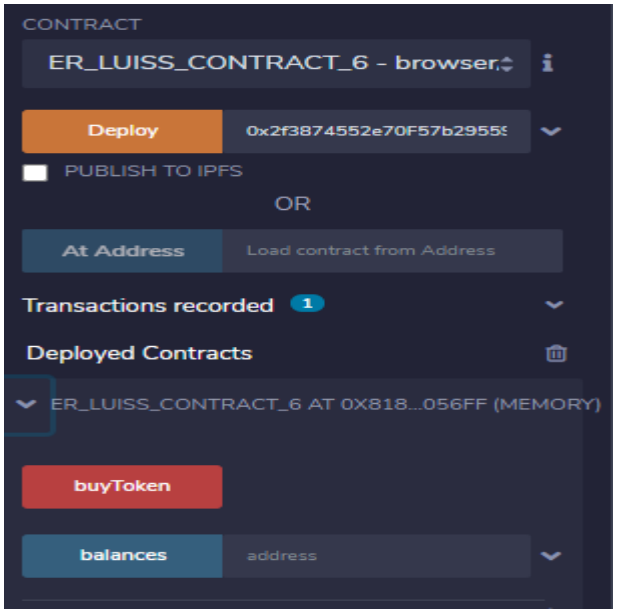
```
pragma solidity^0.5.11;

contract ER_LUISS_CONTRACT_6 {
    mapping(address => uint256) public balances;
    address payable wallet;

    constructor(address payable _wallet) public {
        wallet = _wallet;
    }

    function buyToken() public payable {
        balances[msg.sender] += 1;
        wallet.transfer(msg.value);
    }
}
```

La prima parte del codice è indirizzata dalla funzione mapping che ci permette di tracciare l'indirizzo della persona che emette token e con address payable wallet li vado a depositare all'interno di un portafoglio.

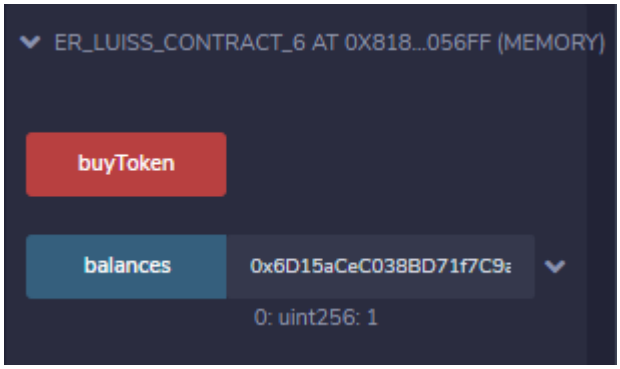
Dopo uso la funzione constructor che mi permette di fissare un wallet, a cui metto davanti l'underscore, per differenziarlo rispetto a prima. Siccome però stiamo parlando dello stesso wallet inserisco che wallet = _wallet.

Infine, creiamo la funzione per comprare questi token generici (buyToken) e con balances[msg.sender] incrementiamo di 1 ogni volta che facciamo una transazione. Dato che vogliamo aggiornare il valore monetario del nostro wallet, usiamo anche la funzione (msg.value).

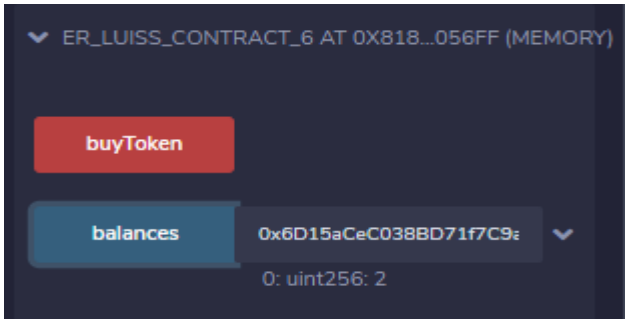
Completato lo smart contract, lo andiamo a lanciare: prima però, dobbiamo settare gli account da cui comprare e depositare i fondi. Lavorando in un ambiente di prova che rimane tutto interno a Remix, uso gli account di Javascript VM, per cui imposto il secondo account come il conto che riceve i fondi, mentre il primo come quello che li spedisce.



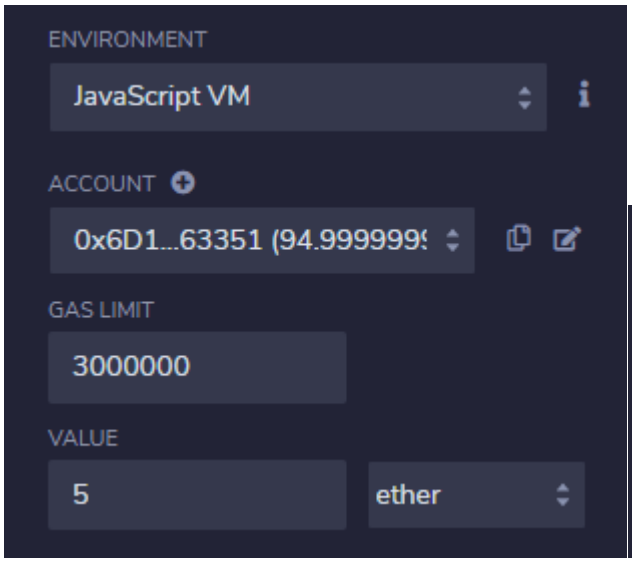
Per fare ciò, essendo un contratto che coinvolge due account a causa di un trasferimento di denaro, copio a fianco di deploy il codice della chiave pubblica del secondo account e lancio il contratto. Mi appaiono due possibilità: nella prima posso comprare i token, mentre nella seconda posso controllare le transazioni che coinvolgono il contratto che invia token. Compro i token e vediamo che il bilancio dell'account che li ha spediti cambia e registra una transazione (a fianco a uint256)

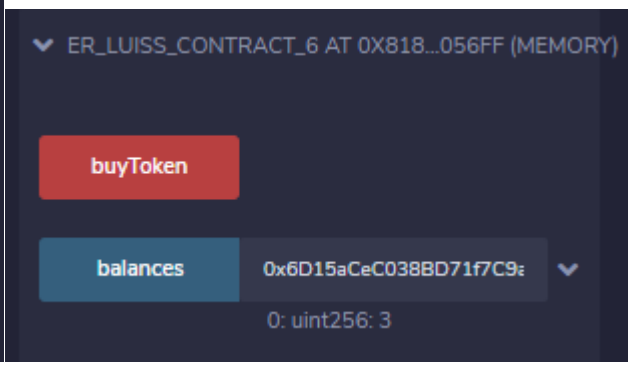


Posso anche continuare a inviare transazioni sempre dall'account numero 1, la cui chiave pubblica è quella a fianco a balances, verso l'account 2 e vediamo che il saldo delle transazioni compiute si aggiorna.

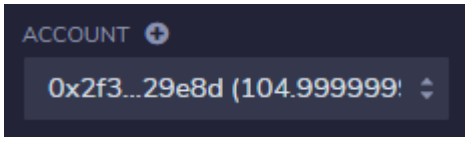


Vediamo invece se gli account aggiornano il loro saldo monetario. Ipotizziamo un trasferimento di 5 ether sempre dall'account 1 verso l'account 2. Cambio la valuta (da wei a ether) e clicco su buyToken.





Anche in questo caso lo storico delle transazioni è stato aggiornato (3), ma cerchiamo anche di capire se gli ether siano stati trasferiti. A parte una leggera commissione per il gas e per i trasferimenti precedenti che erano stati fatti in wei (impatto solo in maniera superficiale sul valore monetario degli account), possiamo notare che ora i due conti sono stati aggiornati. Il primo account infatti possiede circa 95 ether, mentre il secondo 105, motivo per cui il trasferimento da un conto all'altro è avvenuto con successo.



5.5. Contratto per depositare e ritirare soldi

Uno degli scopi principali per una banca è quello di permettere al cliente di depositare e ritirare liberamente i propri soldi dal conto corrente. Questo potrebbe essere gestito con degli smart contracts che automaticamente permettono il deposito dei fondi e aggiornano il saldo del conto ogni qualvolta il cliente ritiri il proprio denaro. Tale processo diventerebbe immutabile e incontestabile a causa della trascrizione in blockchain e mediante il timestamp si potrebbe conoscere esattamente tutti i dettagli di ogni transazione. Attraverso la blockchain non sarebbe necessario fidarsi di un'autorità centrale che mantiene i registri (in questo caso la banca) e non sarebbero possibili alterazioni esterne. Vediamo come funzionerebbe un contratto del genere.

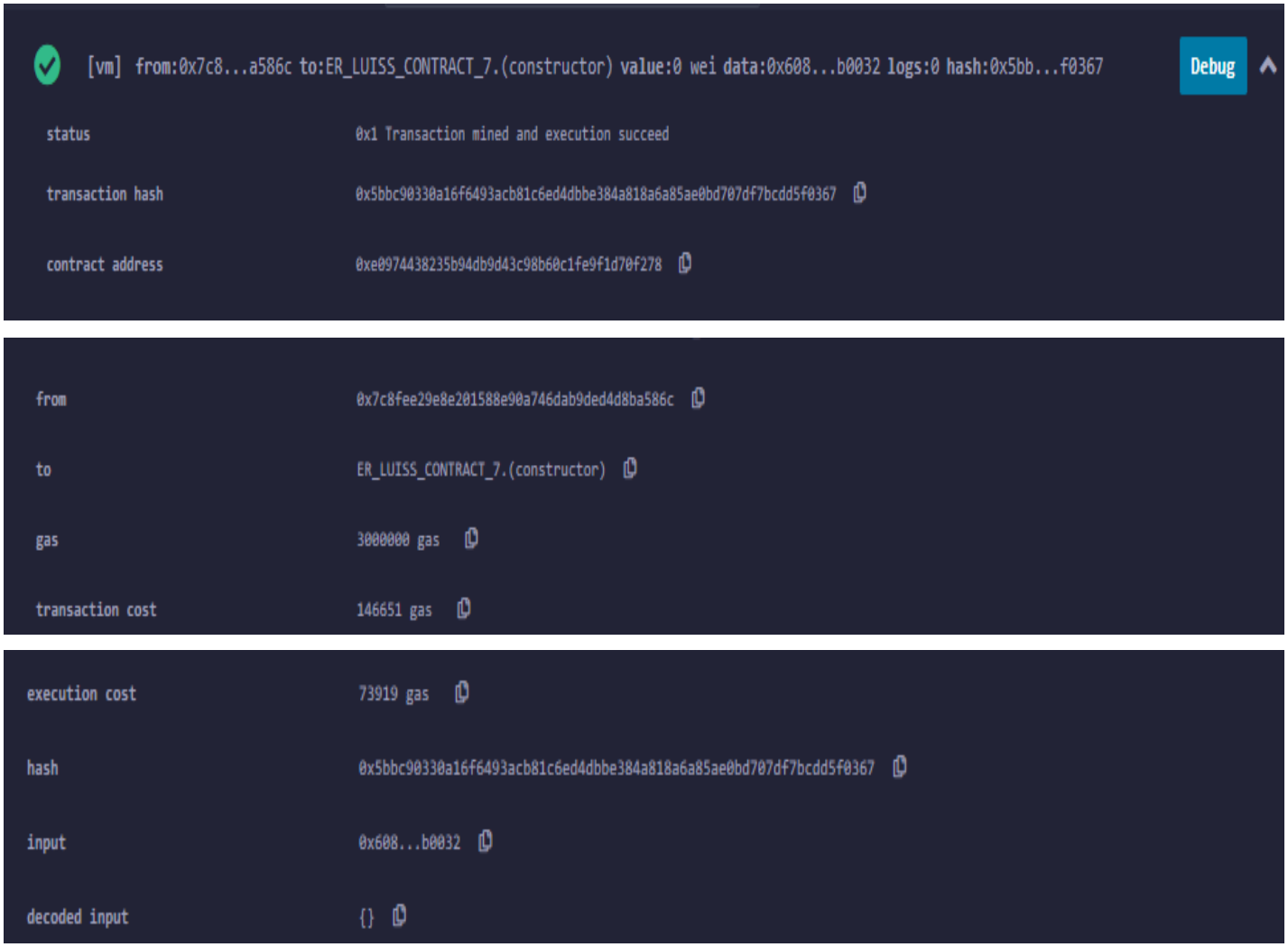
Home ER_LUISS_CONTRACT_7.sol

```
1  pragma solidity^0.5.11;
2
3  contract ER_LUISS_CONTRACT_7 {
4      int balance;
5      constructor() public {
6          balance = 100;
7      }
8      function getBalance() view public returns (int) {
9          return balance;
10     }
11     function withdraw(int amount) public{
12         balance = balance - amount;
13     }
14     function deposit(int amount) public {
15         balance = balance + amount;
16     }
17 }
```

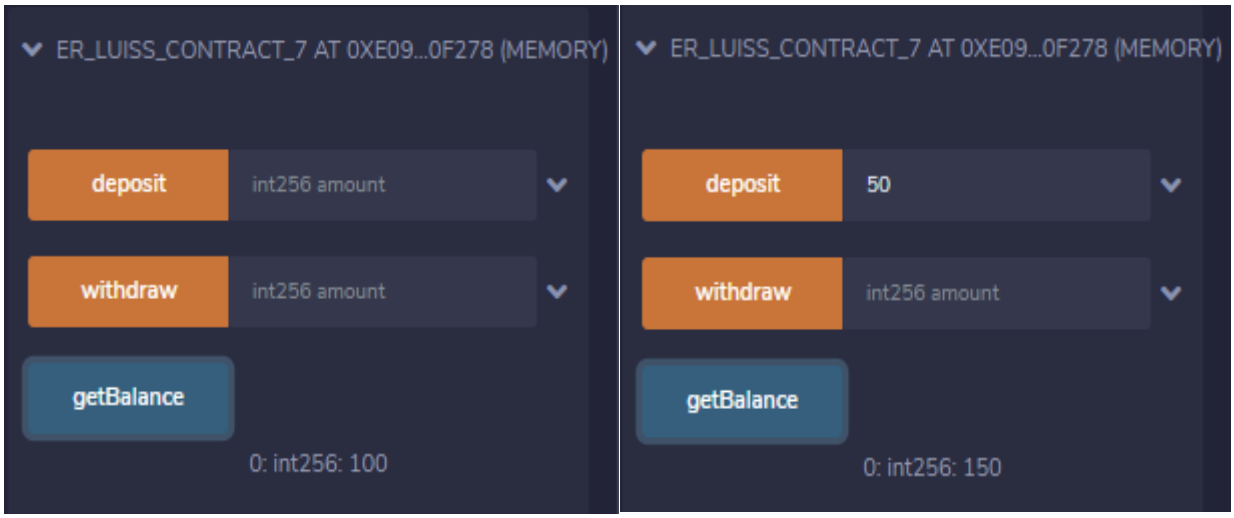
Nella prima riga di codice, dichiaro che voglio che la funzione balance sia un numero intero e come al solito parto da constructor che mi serve per dichiarare lo stato iniziale del contratto. Nel mio caso ipotizzo un balance di 100, che tradotto nel mondo reale vuol dire che nel mio conto corrente sono presenti 100 di una valuta. Devo creare tre funzioni: una funzione che mi permetta di depositare i soldi, un'altra che mi permetta di ritirarli e una terza con cui possa controllare il bilancio.

Parto da quest'ultima: chiamo getBalance la funzione che mi serve per ottenere un bilancio aggiornato, inserisco view perché è una funzione che non modifica la blockchain, ma mi dà solo un output sotto forma di intero. Dichiaro che in uscita devo avere il bilancio con return balance.

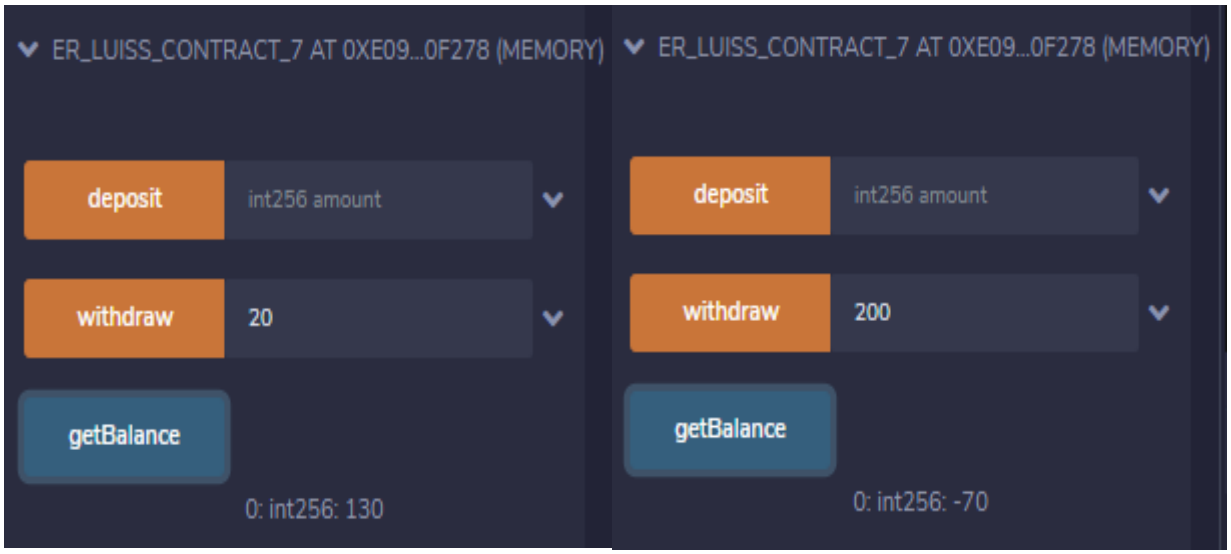
La seconda funzione, che chiamo withdraw, è quella che mi consente di ritirare una somma di denaro dal mio conto corrente. Posso ritirare solo un numero intero che chiamo amount; per calcolare il saldo successivo al ritiro di fondi, si deve sottrarre alla somma di partenza (balance) l'ammontare di fondi ritirati (amount), per cui il risultato finale sarà balance – amount.
Infine, creo la funzione che mi permette di depositare fondi (che chiamo deposit), con la stessa struttura di withdraw. Tuttavia, con un deposito, andrò ad aggiungere fondi al conto corrente (contrariamente da quanto fatto prima) per cui il nuovo saldo sarà balance + amount. Terminato il contratto, clicco su deploy.



Prima di vedere come funziona il contratto, analizziamo le informazioni principali che ci fornisce il terminal: al lancio del contratto, viene fornito subito un feedback (positivo o negativo) circa l'effettiva riuscita dello stesso insieme con l'indirizzo e il codice hash. Si vede da chi è stato scritto (chiave pubblica dell'account) e soprattutto il gas. Quest'ultima informazione è fondamentale perché se fossimo in un ambiente reale e volessimo davvero lanciare lo smart contract sulla blockchain, bisognerebbe assicurarsi di avere la quantità di fondi necessaria su Metamask per fare funzionare per sempre il contratto.
Cliccando su getBalance possiamo vedere il saldo iniziale del conto che, come avevamo stabilito in partenza, è di 100. Ipotizzo un deposito di 50, clicco su deposit e dopo su getBalance: il contratto funziona e il saldo si aggiorna a 150.



Proviamo a ritirare fondi per 20 ed infatti il saldo passa a 130. Dato che la mia variabile è int (e non uint come abbiamo usato finora), il saldo può essere anche negativo per cui ipotizzo di ritirare 200 e infatti il saldo mi si aggiorna e diventa negativo (-70).



Abbiamo verificato come, anche attraverso gli smart contracts, si possano gestire con una certa facilità e certezza i saldi dei conti correnti di una banca. Questo può essere raggiunto combinando la precisione e l'automatizzazione di uno smart contract con l'immutabilità e la decentralizzazione della blockchain.

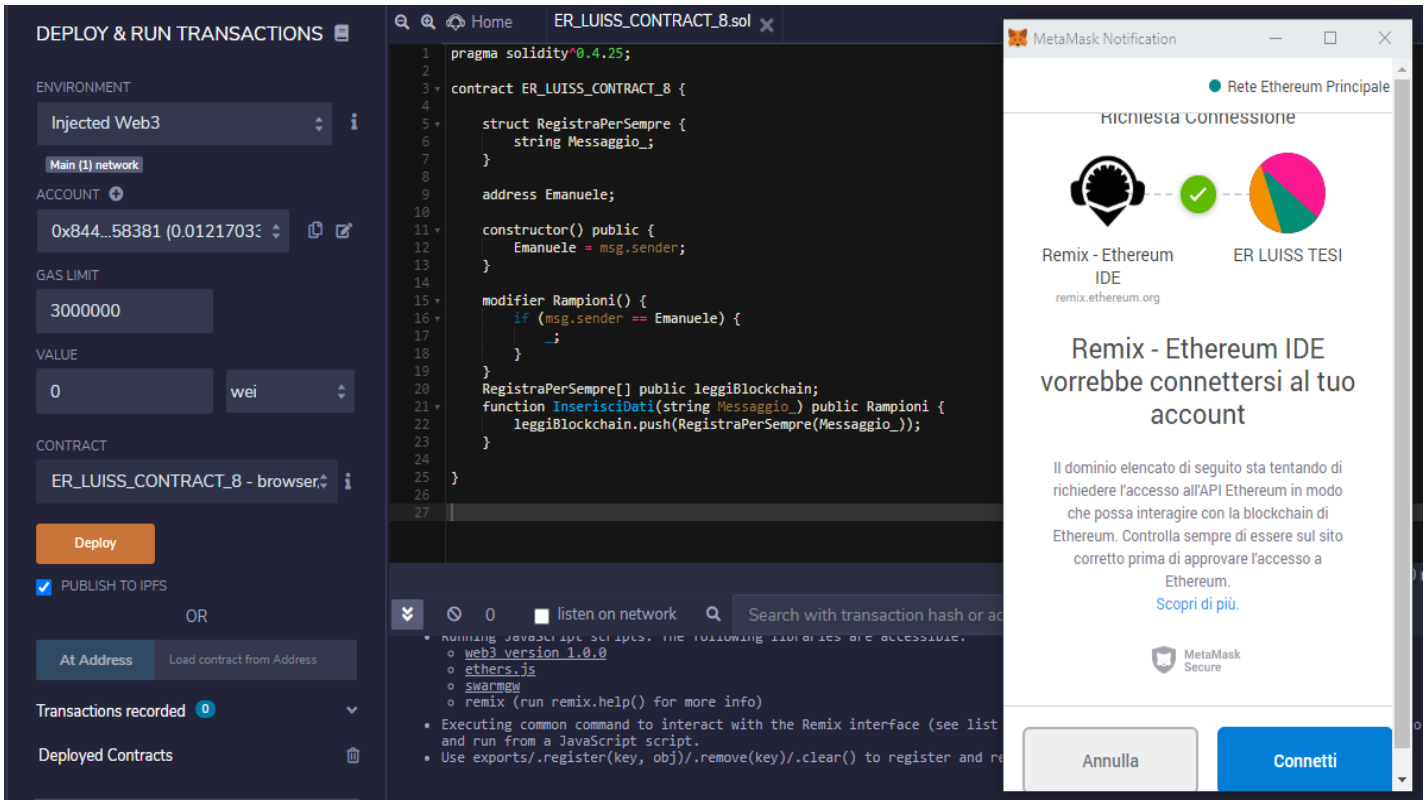
5.6. Smart contract per scrivere messaggi sulla blockchain reale

Fino ad ora ho lavorato in un ambiente interno al computer che non era reale e non aveva interazioni con il mondo esterno. Ora vediamo come lanciare contratti intelligenti sulla mainnet Ethereum, quindi la blockchain Ethereum reale, consultabile da tutti e immutabile. Per farlo, avremo bisogno dell'account Metamask, con alcuni ether all'interno che serviranno per il gas e per le commissioni da pagare ai miners. Lo smart contract che propongo serve per scrivere un messaggio, una stringa di testo di dimensione arbitraria, all'interno della blockchain.

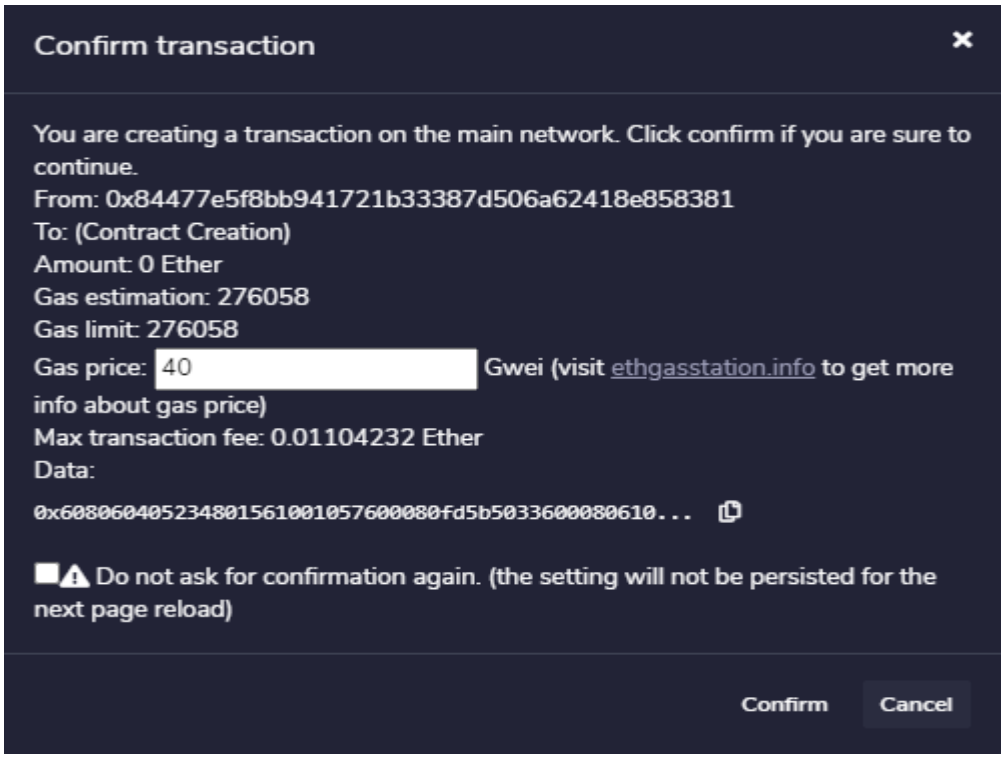
Per scrivere il codice, mi affido alla versione 0.4.25 di Solidity che si adatta alle mie necessità. Creo un nuovo contratto (l'ottavo) e inizio con la funzione struct, che mi permette di combinare alcuni dati, mentre poi dichiaro che voglio avere una stringa di testo che chiamo Messaggio_. Con address indico l'indirizzo del contratto (ho messo il mio nome ma potevo metterne uno qualsiasi) che mi servirà da mettere = alla funzione msg.sender, dato che sono io che sto lanciando il contratto. Aggiungo una funzione modifier che serve per dare alcune caratteristiche aggiuntive o applicare alcune restrizioni ad una funzione; bisogna ricordarsi che modifier ha la particolarità di concludersi con una linea di codice in cui è presente solo l'underscore e, come al solito, il punto e virgola. In base a dove metti l'underscore decidi dove eseguire la funzione originale (infatti esso in Solidity viene anche definito "placeholder" a causa di questa sua caratteristica). Nel caso in cui avessi un contratto lungo, in cui l'address Emanuele interagisse con moltre altre funzioni, bisognerebbe inserire la riga di codice msg.sender == Emanuele in ognuna di queste. Invece, con la funzione modifier, lo posso tralasciare, ma solo se (ecco perché uso una funzione if) l'msg.sender == Emanuele.

Infine mi richiamo la funzione RegistraPerSempre e ne creo una nuova che mi dice di leggere la blockchain. A questo punto abbiamo quasi terminato, manca una funzione in cui posso scrivere la stringa che voglio (InserisciDati) insieme ad un leggiBlockchain.push che sarà una funzione che mi dà indietro un output (funzione read-only che nei precedenti esempi chiamavamo con view) del messaggio registrato.

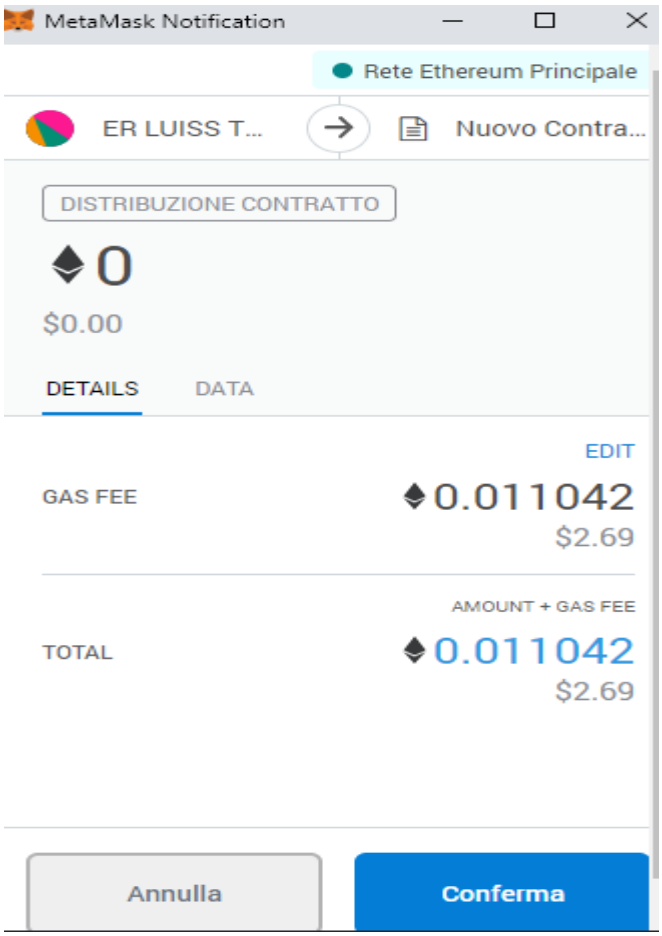
Finito il contratto, notiamo che l'auto compiler non presenta errori, per cui lo lancio. Come detto, questa volta voglio deployarlo davvero sulla blockchain reale, per cui cambio l'environment da Javascript VM a Injected Web3. Differentemente da quanto fatto fino ad ora con i 15 account finti di Javascript VM, mi viene richiesta una connessione con il nostro account Metamask ER LUISS TESI. Connetto l'account e sincronizzo Remix con Metamask.



Dopo che ho connesso il nostro account e cliccato su deploy, mi viene fuori un messaggio di conferma; prima questo non succedeva perché lavoravamo in un ambiente di prova. Mi viene anche specificato l'ammontare di gas necessario, per cui mi devo assicurare di avere gli ether necessari su Metamask. Confermo il contratto.

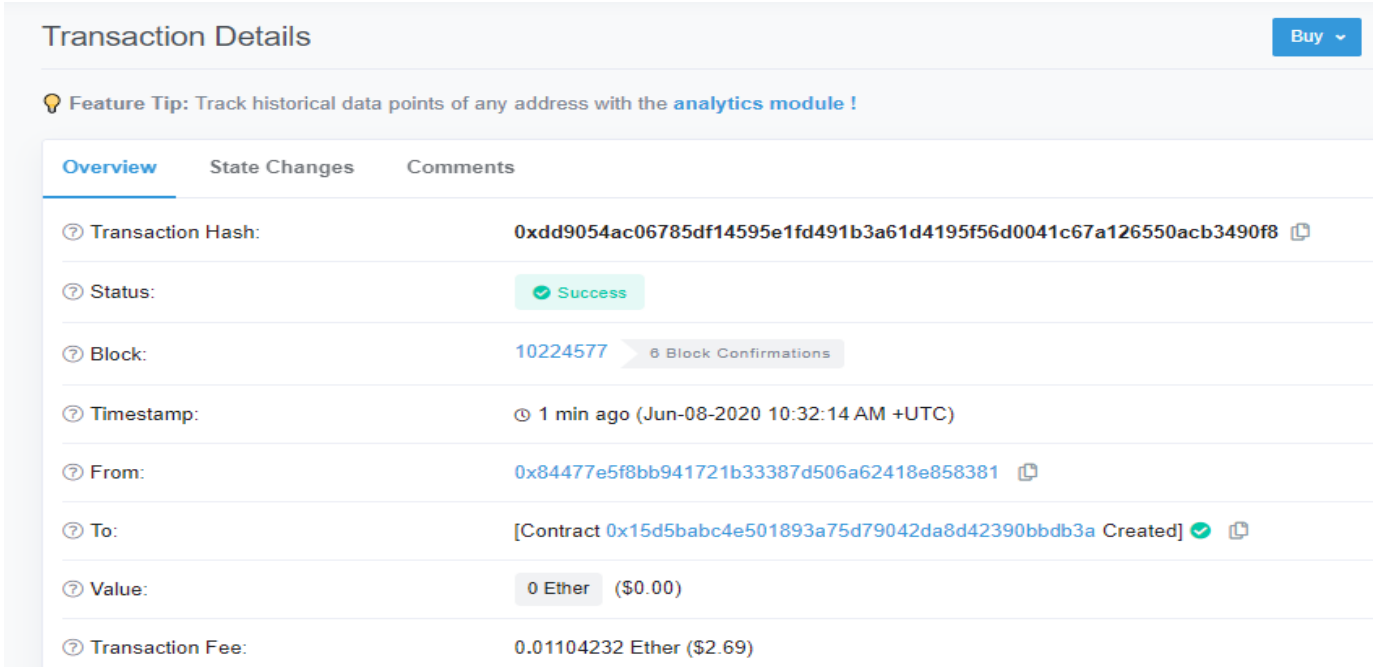


Una volta confermato, automaticamente Metamask mi specifica l'ammontare di fee per il gas necessario a lanciare il contratto in maniera permanente sulla rete Ethereum; anche in questo caso, confermo e mi vengono detratte le spese del gas e del mining dall'account.



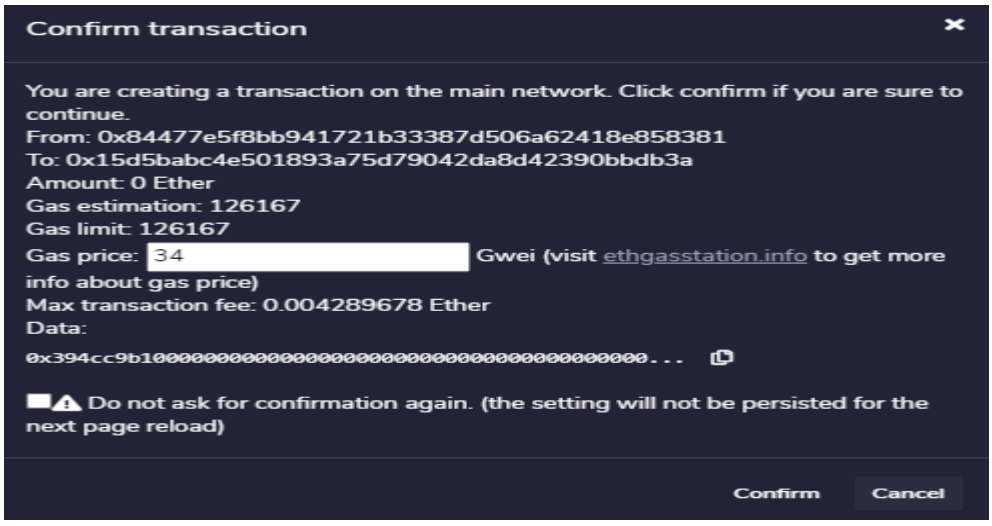
Ora, differentemente dall'ambiente di prova interno al nostro computer, stiamo lavorando sulla rete reale per cui si deve aspettare poco meno di un minuto affinché i miners convalidino e scrivano in un blocco della blockchain il nostro contratto ER_LUISS_CONTRACT_8. Una volta confermato, ci arriva questo messaggio.

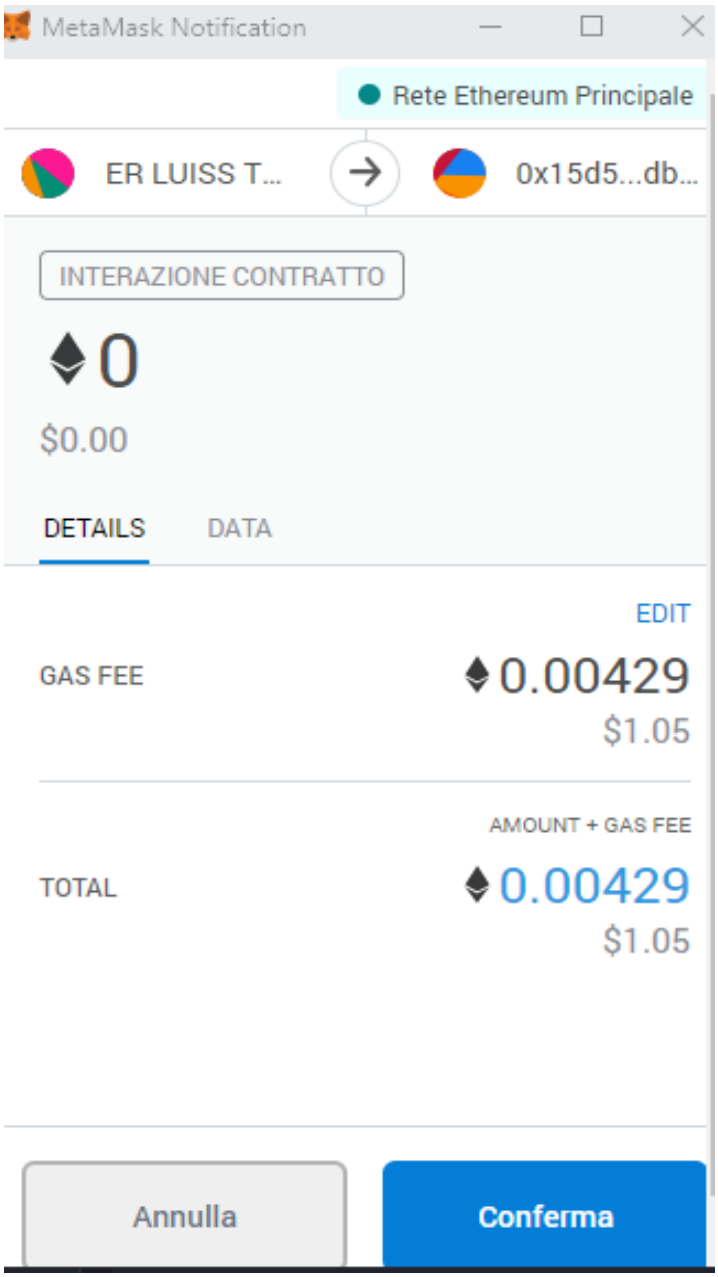
Come ulteriore verifica possiamo andare sul sito etherscan.io, utilizzato per monitorare la blockchain di Ethereum, dato che fornisce la possibilità di cercare indirizzi, transazioni, token, prezzi e altre attività disponibili su Ethereum. Per trovare il contratto lanciato, mi copio l'hash dello smart contract presente nel terminale e lo incollo nella barra di ricerca di Etherscan. In questo modo lo trovo facilmente, insieme ad altre informazioni di cui posso aver bisogno (il blocco in cui è stato minato, le commissioni e soprattutto il timestamp).



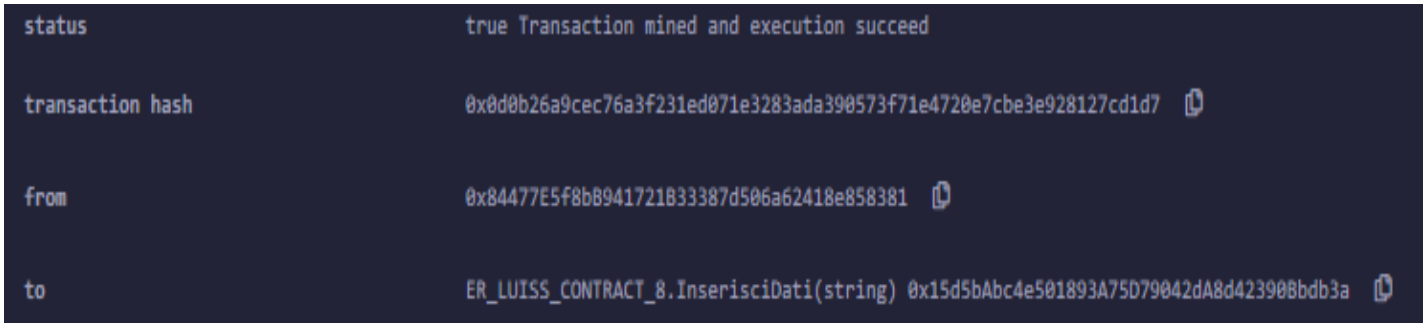
Il contratto che è stato creato tuttavia non è stato ancora reso operativo, perché una volta lanciato in blockchain, non ho ancora scritto il messaggio che volevo salvare per sempre. Clicco sulla freccetta, come facevo anche su Javascript VM, e posso inserire una frase a mio piacimento in InserisciDati (colore arancione), mentre posso leggere quanto ho scritto cliccando su leggiBlockchain (colore blu). Nel mio caso, ho scritto la frase "La LUISS è una delle migliori università in Europa. Codice: 710581". Oltre alla frase, ho aggiunto anche un codice (vedremo dopo perché può essere utile) che corrisponde al numero della mia matricola magistrale LUISS.

Ora finalmente posso dire di aver usato il contratto e lo lancio. Come prima, mi viene chiesta una conferma per la transazione (insieme alla quantità di gas richiesta) e la confermo sia su Remix che su Metamask

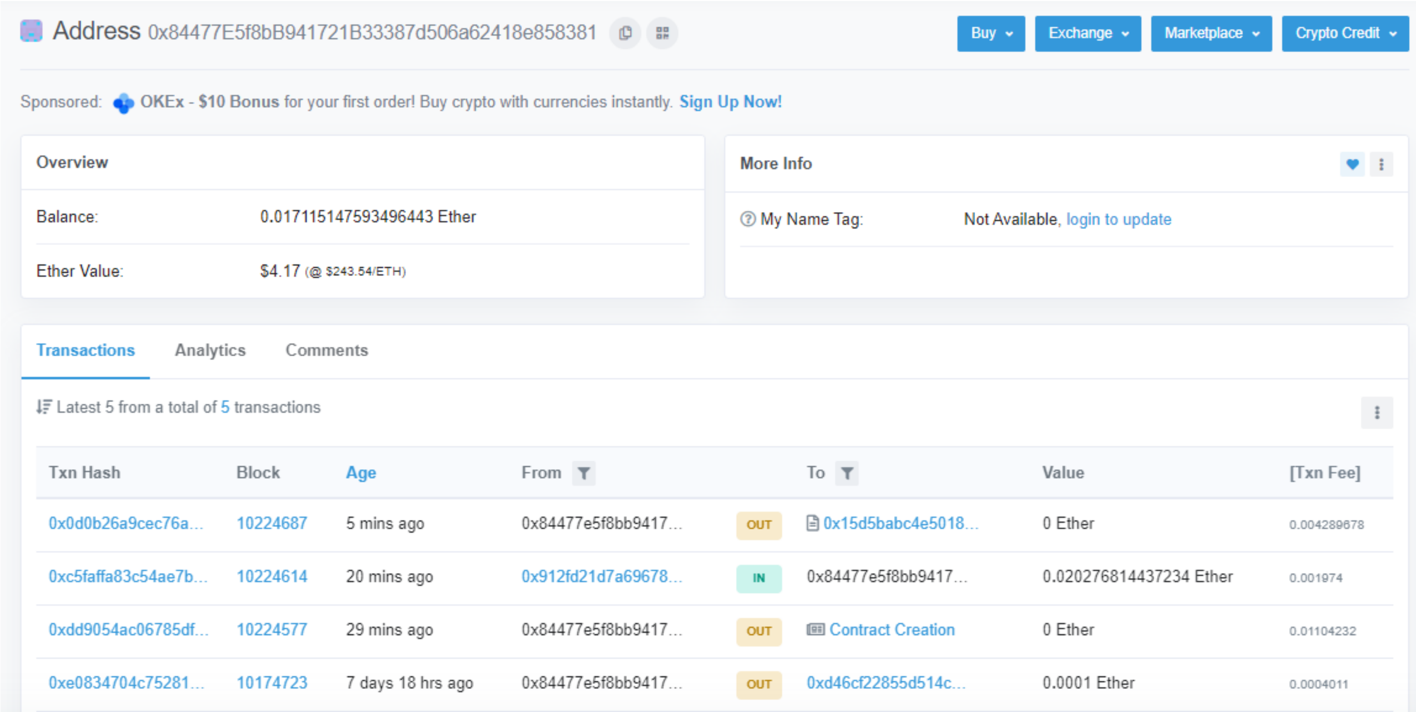




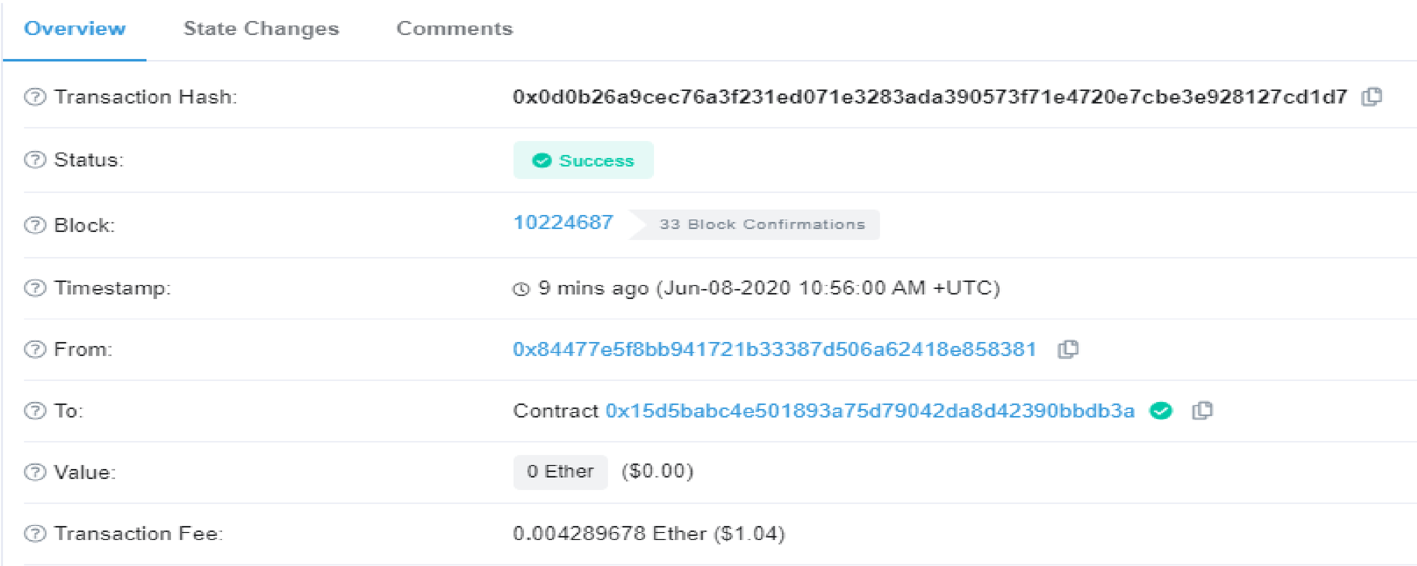
Dopo aver aspettato qualche secondo, la transazione viene convalidata e sul compiler mi appaiono i dati della stessa. In questo modo, posso vedere sia l'hash della transazione, sia l'account da cui è stata inviata. Mi copio l'account e come verifica vado su etherscan.io



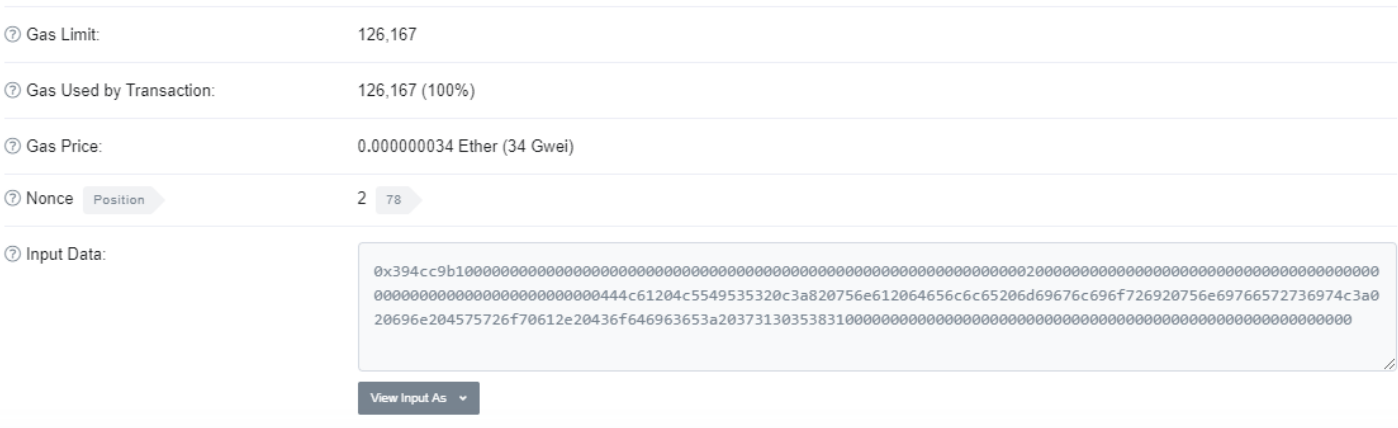
Posso vedere le transazioni che coinvolgono il mio account, ma in particolare il "Contract Creation" e una transazione. Clicco sulla transazione per vedere se la mia stringa è stata aggiunta alla blockchain.



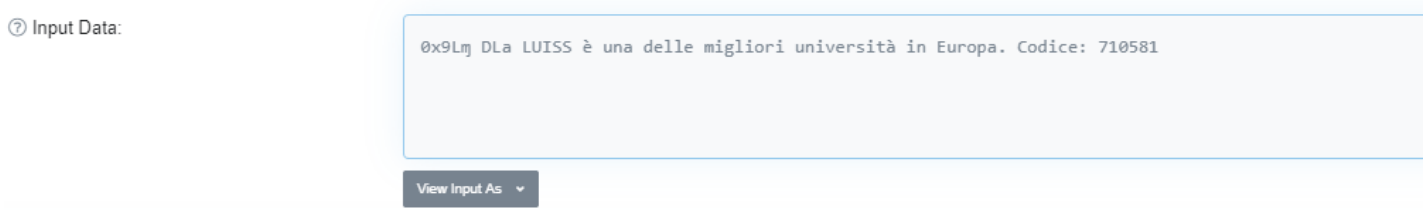
Nella prima parte della schermata mi compare l'hash, lo status che mi dice che la transazione è avvenuta con successo, il timestamp, che mi dà una prova inconfutabile della data e l'ora in cui è stata registrata, ed altre informazioni generali di cui posso aver bisogno.



Se però scendo nella pagina e vado su "click to see more" oltre alle informazioni sul gas utilizzato, mi compare la parte criptata con l'input data.



Cambio la visuale dell'input attraverso il metodo di lettura UTF-8, disponibile cliccando sulla freccetta a fianco di View Input As, e posso vedere la frase che avevo scritto sullo smart contract ovvero "La LUISS è una delle migliori università in Europa. Codice: 710581". Il nostro contratto intelligente ha funzionato, questo messaggio proveniente dal mio account rimarrà per sempre registrato sulla blockchain e potrà essere per sempre consultato da coloro che possiedono l'hash della transazione utilizzando il sito etherscan.io.



Attraverso questo smart contract ho cercato di dimostrare perché i contratti intelligenti potrebbero essere usati in futuro. Per esempio, ipotizziamo una situazione in cui un consumatore ordini alcuni prodotti dall'altra parte del mondo; nel caso in cui volesse avere una prova della autenticità della merce ordinata, il produttore potrebbe servirsi di un contratto come quello che ho appena ideato come certificazione di garanzia. Infatti, all'interno della stringa di testo, si potrebbero scrivere delle informazioni circa la provenienza, la denominazione di origine protetta e un codice; il produttore poi potrebbe apporre il codice (come ho fatto io con la mia matricola) sul prodotto e spedirlo al consumatore. Quest'ultimo, una volta ricevuto il prodotto, potrebbe, mediante il codice hash della transazione, andare sulla blockchain di Ethereum e vedere la stringa di testo (e il corrispondente codice) scritta nello smart contract. Laddove il codice sulla blockchain corrispondesse con il codice scritto sul prodotto, si avrebbe la certezza che la merce arrivata, sia il prodotto veritiero e non uno contraffatto.

Potrebbero essere aggiunte nella stringa di testo altre informazioni sensibili che non si vuole divulgare pubblicamente o sottoporre ad un canale informativo non sicuro. Anche in questo caso la blockchain e gli smart contracts ci vengono in aiuto perché il messaggio con i dati sensibili sono pubblici, ma non possono essere criptati se non si è a conoscenza dell'hash della transazione; per questo motivo possiamo essere sicuri che le informazioni non cadranno mai in mani sbagliate e che la controparte leggerà i dati inviati, il tutto senza l'aiuto di un intermediario che potrebbe appropriarsi o leggere quanto scritto dal produttore.

Un'ultima caratteristica fondamentale è l'immutabilità del registro garantito dal timestamp. Infatti, quando il contratto viene lanciato, automaticamente viene registrata la data e l'ora dello stesso garantendone quindi l'immutabilità. Anche questo sarebbe importante in una catena di produzione perché si conoscerebbe con certezza il momento esatto in cui le merci sono state spedite senza possibilità di alterazioni. Per esempio, possiamo ipotizzare una supply chain composta da molti fornitori in un processo just in time: nel caso in cui si utilizzassero gli smart contracts, si potrebbe facilmente capire da dove arrivano eventuali ritardi o mancanze nella catena. E' sufficiente scrivere sugli smart contracts un codice che certifica le merci, un messaggio con le informazioni rilevanti e lanciare il contratto; in questo modo automaticamente si sapranno data e ora della spedizione e dell'arrivo; diventerebbe quindi facile gestire gli eventuali errori o processi da migliorare nella supply chain.

Un altro mondo che già sta iniziando a beneficiare di smart contracts simili a quello creato ora, è quello assicurativo. Senza doversi affidare ad un intermediario che verifica e registra il sinistro, per esempio nel caso di ritardi nei viaggi aerei, potrebbe essere fatto un contratto intelligente che legge dal sito gli orari di partenza, li confronta con gli orari di "effettiva" partenza e se c'è stato un ritardo rimborsa automaticamente i clienti con un trasferimento di denaro sul loro conto corrente.
Anche la LUISS per esempio potrebbe beneficiare degli smart contracts. Nel caso in cui l'Università volesse avere un maggiore livello di sicurezza inerentemente ad alcune questioni, non volesse affidarsi alla mail per comunicare i voti agli studenti oppure volesse garantire l'immutabilità delle votazioni di laurea, potrebbe creare uno smart contract che soddisfi queste necessità e lanciarlo in blockchain. [46] [47]

5.7. Token ER LUISS ERC-20

Il token ERC-20 è la più famosa tipologia di moneta digitale che possa essere creata mediante Ethereum.
Il token ERC-20 è diventato popolare tra le società di crowdfunding che lavorano sui casi di Initial Coin Offering (ICO) grazie alla semplicità di implementazione insieme con l'interoperabilità con le altre criptomonete.
Affinché un token possa essere definito ERC-20 deve avere alcune caratteristiche principali:

- Total supply dice l'ammontare delle monete totali
- BalanceOf espone il saldo del conto degli account
- Transfer invia un valore monetario ad un certo indirizzo
- Approve che permette di spendere fondi dal tuo account
- Allowance[address_owner, address_sender] restituisce l'importo che il sender può ancora prelevare
- Transfer from invia una quantità di token da un indirizzo

Vediamo ora l'applicazione di queste regole all'interno di un vero contratto per la creazione di un token ERC-20.

[46] Ethereum: the ultimate guide to the world of Ethereum – Ikuya Takashima – 2017

[47] Hands-on Smart Contracts Development with Ethereum and Solidity. From fundamentals to deployment – Kevin Solorio, Randall Kanna, David H. Hoover – 2019

Home ER_LUISS_TOKEN_ERC20.sol

```
pragma solidity^0.5.11;

contract ER_LUISS_TOKEN_ERC20 {

    string public name;
    string public symbol;
    uint8 public decimals;

    mapping (address => uint256) public balanceOf;

    constructor(uint256 initialSupply, string memory tokenName, string memory tokenSymbol, uint8 decimalUnits) public {
        balanceOf[msg.sender] = initialSupply;
        name = tokenName;
        symbol = tokenSymbol;
        decimals = decimalUnits;
    }

    function transfer(address _to, uint256 _value) public returns (bool success) {
        require(balanceOf[msg.sender] >= _value);
        require(balanceOf[_to] + _value >= balanceOf[_to]);
        balanceOf[msg.sender] -= _value;
        balanceOf[_to] += _value;

        return true;
    }
}
```

Chiamo il contratto ER_LUISS_TOKEN_ERC20 e uso un Solidity versione 0.5.11. Mi creo una lista di funzioni che specifico pubbliche, poiché devono essere utilizzate sia da chi distribuisce il token che da chi lo riceve. Nella prima stringa metto il nome del token, mentre nella seconda il simbolo. La terza funzione invece mi dice i decimali con cui si può suddividere il token e, siccome devo inserire un numero, questa non è una stringa ma un uint. Con mapping (serve per tenere una mappa delle transazioni che sta facendo lo smart contract) rendo le funzionalità del token descritte sopra operative.

La seconda parte dello smart contract prevede l'inizializzazione del contratto con scritto all'interno quanti token creare per il suo lancio. Durante questa fase si prepara il contratto con tutte le cose che servono per farlo funzionare: come spesso abbiamo fatto, uso la funzione constructor che permette di legare assieme tutte le variabili e farle funzionare poi all'interno del programma. Sotto distribuisco l'ammontare totale di fondi al creatore del token (msg.sender che è colui che chiama la funzione); infatti con initialSupply scelgo un numero di token che voglio generare, il quale verrà letto dalla funzione balanceOf (indirizzo mittente del token). In seguito, creo delle variabili affinché possa decidere, una volta terminato il contratto, il nome, il simbolo e i decimali dello stesso (Bitcoin per esempio ne ha 18). Laddove avessi voluto, avrei potuto definirli anche internamente al contratto, sostituendo l'ammontare di token iniziali, i nomi e i decimali al posto di initialSupply, tokenName, tokenSymbol e decimalUnits.

Nell'ultima parte dobbiamo costruire la funzione di invio del token. Mediante transfer lo posso spedire da un indirizzo (address) verso un altro (_to), e tale trasferimento può essere deciso mediante un booleano nel caso in cui le condizioni che andremo a definire sotto verranno rispettate. Se esse saranno true, il booleano darà 1 (1 = true, 0 = false) e farà partire il trasferimento.

Le condizioni sono require(balanceOf [msg.sender] che impone che la persona che sta inviando i fondi li possegga nel suo portafoglio (infatti >= value). Pertanto, il secondo require mi controlla che nel sistema non ci siano degli overflows: se per esempio ho creato 1000 token e ne voglio spedire 800 ad un indirizzo, ma

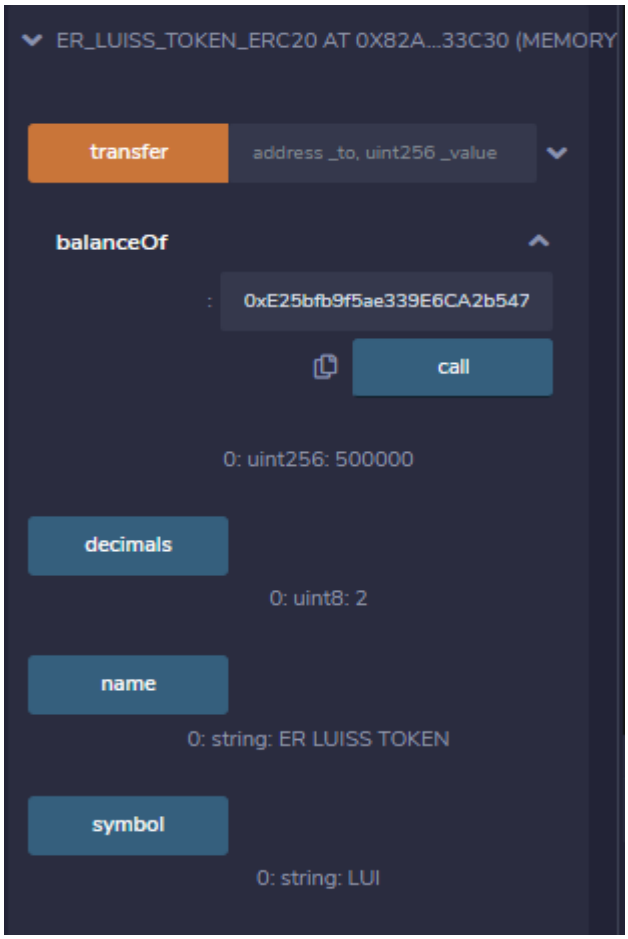
quest'ultimo ne possiede già 500 nel suo portafoglio, lo smart contract si chiede come sia possibile che ci siano 1300 token in circolazione se ne sono stati creati 1000. Quindi, con questo require si prova a capire se qualcuno riesce a creare token falsi o ci sono episodi di double spending.

balanceOf [msg.sender] -= value toglie i token dal mio indirizzo e aggiorna il bilancio mentre balanceOf [_to] += _value fa la cosa opposta, ovvero preleva i token da me e li accredita sul conto del ricevente.

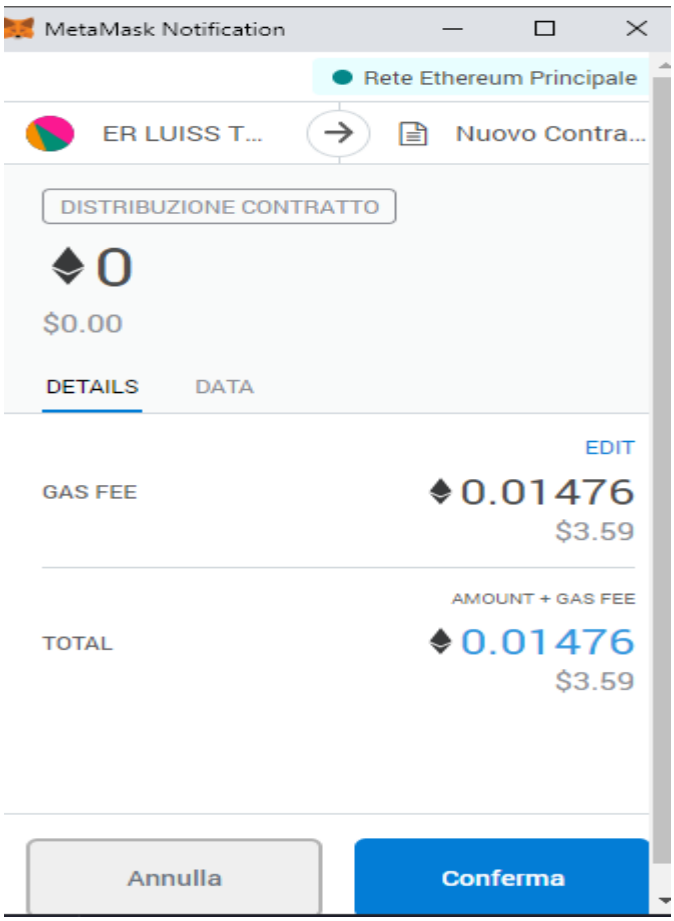
Infine, return true perché, come detto prima, si usa una funzione booleana per completare il trasferimento, per cui tutte queste condizioni devono essere true.

Completato il contratto, prima di fare deploy, devo decidere le caratteristiche del token.

Nello stesso ordine con cui ho suddiviso le variabili, vado a settare i parametri del token che ho creato. Per esempio, in maniera casuale, ho deciso di lanciare 500000 nuovi token, chiamarlo "ER LUISS TOKEN", con acronimo "LUI" e con due cifre decimali. Faccio deploy e il mio token è stato lanciato con le caratteristiche da me richieste. In questo caso ho creato anche la funzione transfer; tuttavia nell'ambiente Javascript VM, essendo una nuova criptovaluta inventata da me ora, non posso effettuare trasferimenti poiché Remix supporta solo scambi in ether o wei. Laddove volessi provare a trasferire veramente l'ER LUISS TOKEN dovrei cambiare a Injected Web3 e aggiungere la criptocurrency al mio wallet su Metamask.



Infatti, cambio la versione su Injected Web3 per lanciare l'ER LUISS TOKEN e mi vengono calcolate le commissioni su Metamask.



Una volta lanciato, andrò a copiare sulla sezione "token" il codice della nuova moneta digitale ER LUISS TOKEN e mi verrà registrato l'ammontare di cryptocurrencies che ho creato. Da questo momento le potrò muovere e scambiare a mio piacimento.

I token ERC-20 sono valute digitali con caratteristiche semplici ma funzionanti, motivo per cui sono alla base di molti progetti e molte ICO. In una ICO ci sono tre situazioni che devono essere rispettate: prima i clienti pagano lo smart contract, poi esso trasferisce i soldi al proprietario il quale infine consegna i token ERC-20 al cliente.

Prima che venisse creato lo standard ERC-20, si erano manifestati diversi problemi di compatibilità tra le varie tipologie di token su Ethereum e ognuno di questi possedeva un proprio smart contract. Per questo motivo, supportare la crescente varietà di token stava diventando problematico e dispendioso in termini di tempo. Come soluzione, l'ambiente Ethereum ha creato un protocollo standard che tutti i token devono seguire, l'ERC-20.

I token ERC-20 possono essere usati in vari modi: quote di un progetto, certificati di proprietà di asset e persino criptovalute vere e proprie. Possono anche ricoprire simultaneamente diversi di questi ruoli.

Un esempio famoso di token ERC-20 è Binance, la criptovaluta nata nel 2017, legata ad uno dei migliori exchange di valute digitali online: Binance.com.

Il BNB, ottava criptovaluta per capitalizzazione (al momento della scrittura), serve per avere sconti sulle commissioni e migliorare le prestazioni interne della piattaforma. Quindi, chi possiede BNB riduce le commissioni sul trading, può utilizzarlo per pagare interessi sul margin trading e può pagarci direttamente le ICO. In cambio Binance utilizza i fondi ricavati dagli acquirenti per aggiornamenti della piattaforma (nuove assunzioni, formazione del personale e budget di sviluppo), attività di branding, marketing e promozione; infine una parte viene adibita a riserva in caso di situazioni di emergenza.
Il prezzo della criptovauta viene dettato dall'interazione tra domanda e offerta ma a ciò va aggiunto che, ogni 4 mesi, Binance utilizza i suoi profitti per acquistare il 20% dei Binance coin in circolazione sul mercato e distruggerli (in gergo "burn"). Aumentando la scarsità della moneta, Binance contribuisce ad incrementarne il valore.
Con l'esempio di Binance, ho cercato di spiegare come potrebbe essere utilizzato un token ERC-20 per ottenere dei finanziamenti in maniera rapida. Bisogna tenere in considerazione che, affinché gli investitori possano comprare il nuovo token emesso, devono avere fiducia nel progetto sottostante. Per esempio, se Binance iniziasse a perdere i fondi nei conti o a non eseguire correttamente le transazioni, i clienti cambierebbero exchange e nessuno sarebbe più disposto a comprare dei BNB. Come si può notare, la facilità di reperibilità di fondi e l'ancora scarsa legislazione a riguardo, ha reso estremamente popolare le ICO mediante token ERC-20 come metodo di crowdfunding; per questo motivo, gli investitori sono chiamati ad un'attenta analisi riguardo i progetti su cui investire, soprattutto perché le garanzie legislative in materia non sono ancora complete.

Possiamo pensare che anche la LUISS, per finanziare qualche progetto o iniziativa, possa lanciare una propria criptovaluta. Sono molte le società anche fuori dal mondo cripto che stanno lanciando token di proprietà con lo scopo di aumentare il legame con i propri clienti o sostenitori (vedi Juventus). Lanciando un token ERC-20 acquistabile dagli studenti, la LUISS potrebbe garantire ai possessori della valuta sconti sui servizi all'interno dell'Ateneo o il possibile coinvolgimento in alcune decisioni sul futuro dell'Università. In questo modo sia si aumenterebbe il coinvolgimento da parte degli studenti nelle decisioni interne all'Ateneo, sia si capirebbe quali siano le volontà reali degli studenti LUISS, i quali, pagando il token, avrebbero il diritto di partecipare alle scelte inerenti all'Università in cui studiano.
Per esempio, ipotizziamo che la LUISS lanci il proprio token ad un prezzo iniziale che poi si aggiusterà sulla base della domanda degli studenti. In questo modo mentre l'Università riceverà dei soldi da poter spendere, gli studenti compratori del token potranno concorrere a decidere se l'Ateneo necessiti di un'aula studio più grande o di nuovi volumi per la biblioteca, se debba organizzare o meno degli incontri su determinate tematiche o se ci sia la necessità di potenziare il servizio navette piuttosto che la mensa.
Inoltre, attraverso questo meccanismo sarebbero chiamati a prendere delle decisioni gli studenti che vivono in maniera più "attiva" la LUISS, essendo quelli che sono stati disposti a pagare per incidere sulla vita universitaria. In qualsiasi momento inoltre si dovrebbe dare la possibilità agli studenti di scambiare i propri

token, come avviene normalmente nei mercati finanziari, e obbligarli alla vendita di tutte le loro monete digitali una volta concluso il percorso universitario.
Un'idea potrebbe essere quella di lanciare un ammontare di token pari al numero di studenti immatricolati, in modo che se tutti comprassero un token, tutti avrebbero le stesse possibilità di decidere le sorti universitarie. Tuttavia, non sarà mai così e il mercato si regolerà automaticamente verso l'equilibrio.
Gli studenti compratori si sentirebbero parte di un'Università che li coinvolge attivamente nelle proprie scelte, che li avvantaggia in alcune cose (sconti a mensa per esempio) e che gli dà la possibilità di liquidare il loro mini investimento laddove non si sentissero sufficientemente coinvolti o volessero chiudere la loro posizione per avere un piccolo profitto perché il valore del LUISS token è aumentato.
Un progetto più ambizioso potrebbe essere quello di lanciare la criptovaluta LUISS a livello globale, quindi come un normale token le cui fluttuazioni dipendono dall'andamento dell'Università e che possa essere comprato da qualsiasi investitore che crede nella LUISS. [48] [49] [50] [51]

Uno dei temi su cui si sta dibattendo negli ultimi mesi è quello del possibile lancio di una digital currency da parte delle Banche Centrali ("Central Bank Digital Currency" ovvero CBDC). Infatti, stimolate dal lancio di token effettuati da società private (vedi per esempio Libra, la criptocurrency appartenente a Facebook), molte Banche Centrali hanno iniziato a lavorare su progetti riguardanti monete digitali personali rappresentanti una valida alternativa al contante e ai depositi bancari.
Il progetto Ubin con la Monetary Authority di Singapore ha insegnato che una CBDC abilitata sulla blockchain supporta flussi di pagamento complessi più efficienti di quelli attuali, compreso un meccanismo decentralizzato di risparmio di liquidità.
La Bank of Lithuania ha in programma l'emissione di "Digital Collector Coin" per testare la blockchain su scala ridotta e su un ambiente reale. Sarà collegata alle monete da collezione fisiche custodite nei caveau della Bank of Lithuania.
La People's Bank of China ha avviato un'iniziativa di "yuan digitale" e la Banca Centrale svedese, Sveriges Riksbank, ha annunciato un progetto per una versione digitale della sua valuta per uso retail, denominata e-krona.
Recentemente la Bank of England ha svolto ricerche su quello che chiama il "modello di piattaforma", in cui la banca è l'unica entità autorizzata a creare o distruggere un token, lasciando che i fornitori di interfacce di pagamento interagiscano con gli utenti finali. Inoltre, la Federal Reserve, la Banca del Giappone e la Banca Centrale Europea stanno attualmente esplorando tale tecnologia.

[48] How to write and deploy your first smart contract? – Mayank Sahu – 2020

[49] Security Tips on Writing Smart Contracts – Nenad Palinkasevic – 2019

[50] Ethereum Smart Contract Development – Mayukh Mukhopadhyay – 2018

[51] Blockchain Technology Smart Contract – Bashar Almunyyer, Abdallah Daodiah - 2019

La nazione che è più avanti nel progetto di lancio di una CBDC è il Canada che sta per creare una propria valuta digitale come soluzione di emergenza in caso di necessità. Proprio come il contante, la CBDC non genera alcun interesse e sarà universalmente accessibile; un aspetto fondamentale della currency virtuale canadese è l'adattamento al mondo digitale dinamico, per esempio con i sistemi IoT, l'interoperabilità con i sistemi di pagamento esistenti, la sua facilità di scambio e l'accessibilità in modo da essere estensibile con strumenti come smartphone, smartwatch e computer. [52] [53] [54]

Abbiamo visto che la BCE sta pensando di implementare una propria valuta digitale; come facilmente immaginabile, molti sono gli aspetti da definire in modo da massimizzarne il risultato.

Da un punto di vista strettamente informatico, sarà una moneta completamente diversa da Bitcoin, basata su una blockchain di tipo permissioned con il processo di mining non basato sulla proof of work. Tuttavia, si parla di un progetto che potrebbe essere supportato dalla piattaforma Ethereum, la stessa che ho adottato per la stesura degli smart contracts.

Ethereum in particolare è la blockchain più pronta a supportare i requisiti di una CBDC in termini di scalabilità e privacy. Infatti, i sistemi distribuiti, come le blockchain, garantiscono la disponibilità e la resilienza dei dati, oltre alla fiducia e alla trasparenza sulla storia delle transazioni. Ethereum ha dimostrato la sua capacità di supportare reti molto grandi con più di 10000 nodi e centinaia di migliaia di utenti.

Infine, una CBDC basata su blockchain si avvantaggia dei prodotti e dei servizi innovativi che si stanno costruendo nell'ambiente delle blockchain open source, compresi i portafogli online, la crittografia e la finanza decentralizzata. Ethereum è il più grande ecosistema di blockchain del mondo con oltre 350.000 sviluppatori.[55]

Da un punto di vista concettuale e ideologico, sono stati pubblicati moltissimi papers riguardanti la necessità o la convenienza dell'introduzione di un token da parte della BCE, ma quello che reputo più completo nell'esposizione del problema è stato scritto da Fabio Panetta, membro del comitato esecutivo della BCE.

Panetta parte dalla semplice domanda se la Banca Centrale debba o meno lanciare una propria digital currency e risponde al quesito analizzando le funzioni della moneta come mezzo di pagamento e riserva di valore (Queste sono due delle tre funzioni del denaro. La terza, il denaro come unità di conto, non è rilevante in questo caso, poiché una moneta digitale emessa dalle Banche Centrali sarebbe denominata nella stessa unità delle banconote esistenti).

Come mezzo di pagamento, Panetta ritiene che una CBDC si aggiungerebbe ai servizi di pagamento digitali disponibili, aumentando così il grado di concorrenza nel settore. Tuttavia, l'insieme di strumenti che permettono pagamenti istantanei è già ampio: al giorno d'oggi possiamo effettuare un pagamento digitale

[52] Central Banks and Distributed Ledger Technology: How are Central Banks Exploring Blockchain Today? – World Economic Forum – 2019

[53] Central Bank Digital Currencies: Changing the Architecture of Money – George Calle - 2020

[54] Crypto Boom 2.0? Il Canada accelera i piani per l'emissione di una CBDC – Kevin Wilson - 2020

[55] Blockchain for Central Bank Digital Currency (CBDC) – Consensys - 2020

tramite bonifico bancario (attraverso l'online banking), carte di credito o di debito, utilizzando Paypal o Apple pay; possiamo farlo tramite computer, smartphone o smartwatch, semplicemente mettendo il polso vicino ad una macchinetta. Da questo punto di vista i vantaggi di una CBDC sono poco chiari: i suoi potenziali benefici in termini di miglioramento della facilità delle transazioni sono probabilmente insufficienti per giustificare il coinvolgimento delle Banche Centrali.

Invece l'introduzione di un mezzo di pagamento digitale potrebbe essere giustificata dall'obiettivo di ridurre il costo del contante, ossia le spese per la sua produzione, il trasporto, lo smaltimento ecc. Secondo stime recenti, questi costi ammontano a circa mezzo punto percentuale del PIL nell'Unione Europea, ovvero a circa 76 miliardi di euro.

I costi di gestione del contante sono dovuti alla sua natura fisica e in un mondo digitale scomparirebbero, mentre i costi di hardware e software aumenterebbero. Tuttavia, la tecnologia digitale svolge già un ruolo cruciale nel settore finanziario poiché viene utilizzata per il trasferimento di denaro tra le banche commerciali, per l'acquisto e la vendita di titoli e per l'elaborazione di informazioni. La tecnologia necessaria per il trasferimento di denaro digitale avrebbe probabilmente forti complementarità con le reti e le infrastrutture digitali esistenti e ciò suggerisce che i costi complessivi della fornitura di un mezzo di pagamento potrebbero diminuire con l'introduzione di una moneta virtuale.

Come riserva di valore, poiché sarebbe completamente dematerializzato, una CBDC avrebbe pochissimi costi di stoccaggio e rappresenterebbe un metodo conveniente di conservazione di liquidità per le famiglie e le imprese. Oltre ad essere superiore al contante come riserva di valore, una moneta digitale sarebbe un'attività con caratteristiche uniche, priva di rischio di credito e di liquidità. Potrebbe essere preferita ad altri strumenti comunemente usati per conservare la ricchezza, come i depositi bancari.

Laddove questo dovesse succedere, potrebbero sorgere alcuni problemi: il passaggio dai depositi bancari a una CBDC potrebbe portare a una carenza di fondi nel sistema bancario, con potenziali effetti negativi sull'offerta e sul costo del credito nell'economia reale. In condizioni estreme, la disponibilità di una CBDC potrebbe addirittura aumentare il rischio di una bank run digitale. Tuttavia, non è detto che le conseguenze possano essere di così ampia portata e che le banche subiscano necessariamente ingenti perdite. In primo luogo, solo alcune categorie di depositi potrebbero migrare verso la Banca Centrale (molto probabilmente depositi a vista, che pagano poco o nessun interesse). In secondo luogo, le banche possono competere offrendo servizi che le CBDC non possono offrire, come l'accesso ai servizi di credito e di pagamento. In terzo luogo, le banche potrebbero aumentare il ricorso alla raccolta all'ingrosso.

E’ innegabile però che il modello di business delle banche ne risentirebbe. La diminuzione delle passività callable porterebbe verso un sistema bancario "ristretto ", che è un quadro operativo in cui le banche hanno uno scarso disallineamento di scadenze tra attività e passività.

Vediamo alcune caratteristiche tecniche: una CBDC sarebbe una passività della Banca Centrale e sarebbe sostenuta dal suo attivo. Non fluttuerebbe liberamente spostata dalle variazioni del mercato, ma sarebbe certificata dalla credibilità della Banca Centrale e, in ultima analisi, dal sistema legislativo sottostante. Le

criptovalute come Bitcoin, invece, sono una passività che non appartiene a nessuno: non c'è un attivo che li sostenga e non c'è una chiara struttura di governance che possa garantire la fiducia.

I rischi e i benefici delle CBDC, insieme al loro impatto sul sistema finanziario, sull'economia reale e sulla società, dipendono strettamente dalle loro caratteristiche. Probabilmente la questione più importante è se la moneta digitale debba essere rintracciabile o se debba essere concepita in modo da garantire, per quanto possibile, l'anonimato. Il denaro contante permette l'anonimato di terzi nelle transazioni e non lascia traccia. Se da un lato ciò implica che si tratta di un mezzo di pagamento efficace per attività illecite come il riciclaggio di denaro sporco, il finanziamento al terrorismo o l'evasione fiscale, dall'altro garantisce la privacy dei suoi utenti. La possibilità di rintracciare le nostre transazioni digitali può avere importanti implicazioni economiche ed etiche. Chi dovrebbe decidere il grado di anonimato associato all'uso di una CBDC? E' chiaro che si tratta di qualcosa di più di una questione tecnica e come tale la scelta non appartiene solo alle Banche Centrali, ma anche alla sfera politica.

Un'altra questione fondamentale è se una CBDC debba essere remunerata o, come nel caso del contante, non debba pagare interessi. Questa scelta avrebbe conseguenze di vasta portata per le attività fondamentali della Banca Centrale, dalla stabilità finanziaria alla politica monetaria.

Ad esempio, il pagamento degli interessi renderebbe la CBDC un sostituto più stretto dei depositi bancari e ne aumenterebbe la volatilità: in tempi difficili, i depositanti potrebbero passare rapidamente e a costo zero dal conto bancario alla CBDC. La Banca Centrale potrebbe limitare tali rischi - ad esempio fissando un massimale per l'importo di criptovalute che ogni singolo investitore può detenere o portando a zero la remunerazione per il possesso di token al di sopra di una certa soglia - ma ciò solleverebbe una serie di questioni tecniche.

Allo stesso tempo, una CBDC che paga interessi rafforzerebbe la trasmissione degli impulsi monetari alle banche, alle famiglie e alle imprese. In periodi di crisi, abbassando la remunerazione della moneta digitale, la Banca Centrale potrebbe spingere le banche a ridurre i tassi di deposito, migliorando la loro capacità di stimolare l'economia senza necessariamente ricorrere a misure non convenzionali. Un meccanismo simmetrico si verificherebbe in periodi di ripresa economica, quando un aumento della remunerazione della moneta virtuale (che rappresenterebbe il minimo dei tassi di mercato) costringerebbe le banche ad agire rapidamente per aumentare anche la remunerazione dei loro depositi.

Per quanto riguarda la tipologia di moneta digitale, le Banche Centrali dovrebbero decidere se le CBDC debbano essere token-based o account-based. [56]

In un sistema token-based, gli scambi avverrebbero in maniera diretta tra due controparti, senza l'intermediazione di nessuna terza parte, come avverrebbe nel caso in cui Alice volesse comprare 8 dollari di pane da Bob (Figura 52). In questo caso, la CBDC sostituirebbe il contante nella transazione.

[56] 21st century cash: Central banking, technological innovation and digital currencies – Fabio Panetta - 2018

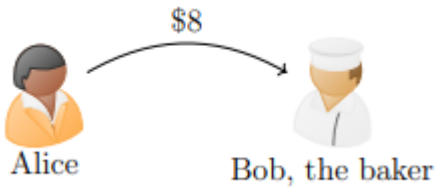


Figura 52 – Modello Token-based – Digital Currencies and Fast Payment System: Disruption is coming – Darrell Duffie

La Figura 53 mostra l'analogo pagamento da Alice a Bob in un sistema account-based. I moderni metodi di pagamento account-based implicano molte differenti tecnologie tra cui bonifici bancari, assegni cartacei, carte di credito, carte di debito e molti altri tipi di applicazioni di pagamento mobile. Un sistema account-based comporta una riduzione dei fondi bancari di Alice, un trasferimento e aumento dei fondi di Bob. Un pagamento token-based viene invece effettuato tramite un trasferimento diretto tra le parti. [57]

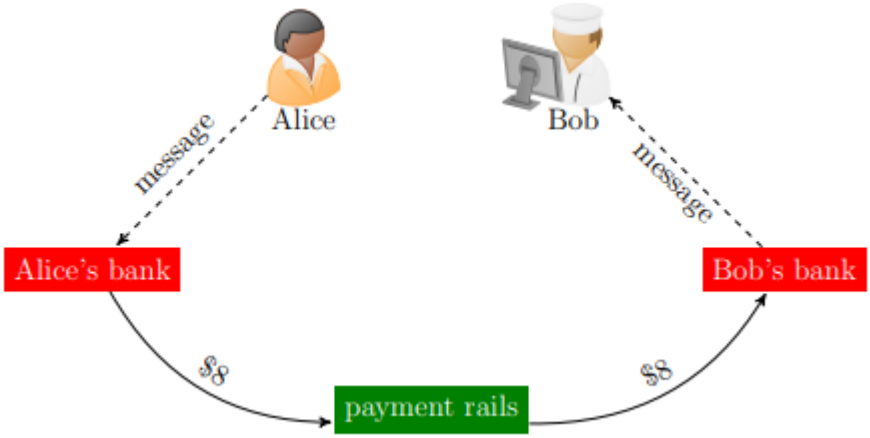


Figura 53 – Modello Account-based – Digital Currencies and Fast Payment System: Disruption is coming – Darrell Duffie

Una CBDC token-based implicherebbe anonimato e una migliore protezione della privacy; sarebbe inoltre più facile e potrebbe essere delegata ad un privato.

Un sistema account-based invece necessiterebbe di una gestione di milioni di conti di criptovaluta, ognuno dei quali potrebbe cambiare di valore ogni giorno e richiederebbe quindi uno sforzo incredibile da parte della Banca Centrale.

[57] Digital Currencies and Fast Payment System: Disruption is coming – Darrell Duffie – 2019

Conclusioni

Lo scopo principale che mi sono preposto all'inizio della stesura dell'elaborato era quello di spiegare come funzionasse nel dettaglio la blockchain, in particolare ad un lettore che ne avesse sempre sentito parlare in maniera generica, associata al Bitcoin o a vari settori senza capirne il reale vantaggio. In particolare, ho adottato un approccio informatico, sempre però con un occhio di riguardo al mondo finanziario, poiché sono convinto dell'assoluta importanza che riveste la comprensione del meccanismo di funzionamento della blockchain.

Nella seconda parte della tesi, mosso sempre dall'obiettivo di avvicinare il più possibile il lettore ad applicazioni reali, ho definito e applicato gli smart contracts ad alcune situazioni collegate al mondo finanziario. Infatti, prima ho definito quale fosse la piattaforma decentralizzata comunemente usata per la loro stesura (Ethereum), poi ho introdotto il linguaggio di programmazione Solidity che è quello in cui la maggior parte dei programmatori scrive i contratti intelligenti ed infine ho reso quest'ultimi operativi nell'ambiente di sviluppo di Remix.

In un primo esempio ho dimostrato come un contratto intelligente fosse sufficiente per gestire un conto corrente bancario; anche con un codice piuttosto semplice, ho dimostrato che lo smart contract ben eseguiva i comandi da me richiesti e aggiornava perfettamente il bilancio del conto corrente.

In un secondo contratto ho lanciato un messaggio in blockchain: mentre fino a prima avevo fatto delle prove su un ambiente di sviluppo interno al mio computer, con gas e commissioni finte, in questo caso ho prima scritto un messaggio da me inventato e poi ho deciso di modificare la blockchain reale di Ethereum per lasciare scritto per sempre un mio pensiero sulla LUISS. In questo modo, ho illustrato come si potrebbero scrivere informazioni immutabili, criptate e sicure in una blockchain mediante uno smart contract. Queste caratteristiche sono quelle per cui la blockchain viene spesso associata a settori quali le catene di produzione, i sistemi di voto e gli ambienti assicurativi. Con basse commissioni, lanciare messaggi in blockchain diventa possibile; anche la LUISS potrebbe essere interessata a farlo laddove per esempio volesse comunicare alcune informazioni sensibili agli studenti o a partners ed avesse paura dei normali sistemi centralizzati.

Infine, ho dimostrato come si possa scrivere un token ERC-20 standard sulla blockchain di Ethereum. La diffusione dei token ERC-20 ha avuto una incredibile spinta negli ultimi anni a causa della possibilità di lanciare le ICO (Initial Coin Offering), che sono progetti di crowdfunding in cui principalmente le start-up, ma non solo, decidono di finanziarsi lanciando un'emissione di token privati. In questo modo, le nuove aziende che vogliono superare le difficoltà burocratiche di una IPO oppure non sono nelle condizioni di poterla fare, possono sfruttare questo nuovo metodo di finanziamento, creando fin da subito un certo legame con i compratori della loro moneta digitale.

Per concludere, ho ipotizzato che anche la LUISS potesse farsi carico di un progetto simile: creando inizialmente un ammontare di token pari al numero di studenti immatricolati e vendendo gli stessi ad un prezzo iniziale, l'Università potrebbe da un lato finanziarsi e dall'altro creare una maggiore partecipazione degli studenti alle decisioni universitarie.

Questo creerebbe anche un piccolo mercato finanziario interno all'Università poiché si darebbe la possibilità agli studenti di scambiare tra loro il token in base alle oscillazioni del suo valore.

Bibliografia

- Beginning Ethereum Smart Contracts Programming – *Wei-Meng Lee – 2019*
- Big Data: a revolution that will transform how we live, work and think – *Kenneth Cukier, Viktor Mayer Schonberger – 2013*
- Bitcoin – *Kerry Gan – 2019*
- Bitcoin Essentials – *Albert Szmigielski – 2016*
- Bitcoin: from digital gold to electronic cash with Lightning Network – *PwC -2018*
- Bitcoin's Block Timestamp Protection Rules – *Bitmex.com – 2019*
- Blockchain based smart contracts: a systematic mapping study – *Maher Alharby, Aad van Moorsel - 2017*
- Blockchain basics – *Daniel Drescher – 2017*
- Blockchain Disruption and Smart Contracts – *Lin William Cong, Zhiguo He – 2018*
- Blockchain Education Network (BEN) – *2019*
- Blockchain for Central Bank Digital Currency (CBDC) – *Consensys - 2020*
- Blockchain Technology Explained – *Alan T. Norman – 2017*
- Blockchain Technology Smart Contract – *Bashar Almunyyer, Abdallah Daodiah - 2019*
- Blockchain: Blueprint for a new economy - *Melany Swan - 2015*
- Building Blockchain Projects – *Narayan Prusty – 2017*
- Central Banks and Distributed Ledger Technology: How are Central Banks Exploring Blockchain Today? – *World Economic Forum – 2019*
- Central Bank Digital Currencies: Changing the Architecture of Money – *George Calle - 2020*
- Contract Law 2.0: "Smart" Contracts as the beginning of the end of the classic contract law – *Alexander Savelyev – 2016*
- Crypto Boom 2.0? Il Canada accelera i piani per l'emissione di una CBDC – *Kevin Wilson - 2020*
- Cryptocurrencies simply explained – *Julian Hosp – 2017*
- Cryptocurrency Mining – *Devan Hansel - 2018*
- Cryptographic hash function – *Tim Fisher – 2020*
- Cryptography hash functions – *Tutorialspoint – 2018*
- Digital Currencies and Fast Payment System: Disruption is coming – *Darrell Duffie - 2019*
- Distributed Ledger Technology (DLT) and Blockchain - *World Bank Group - 2017*
- Ethereum for Architect and Developers – *Debajani Mohanty - 2018*
- Ethereum Proof of Stake Date: date + what you need to know – *Daniel Won – 2020*
- Ethereum Smart Contract Development – *Mayukh Mukhopadhyay – 2018*
- Ethereum White Paper: a next generation smart contract and decentralized application platform – *Vitalik Buterin – 2013*
- Ethereum: Platform Review. Opportunities and challenges for private and consortium blockchains – *Vitalik Buterin – 2016*
- Ethereum: the ultimate guide to the world of Ethereum – *Ikuya Takashima – 2017*

- Giorgio Angiolini - *Blockchain Week Rome 2019 - 2019*
- Hands-on Smart Contracts Development with Ethereum and Solidity. From fundamentals to deployment – *Kevin Solorio, Randall Kanna, David H. Hoover – 2019*
- How to write a simple Smart Contract – *Morgan Fogarty – 2018*
- How to write and deploy your first smart contract? – *Mayank Sahu – 2020*
- Intro IoT - *Cisco Networking Academy - 2020*
- Introducing Ethereum and Solidity – *Chris Dannen – 2017*
- La sicurezza dietro Bitcoin: Crittografia Asimmetrica – *Danilo Giudice – 2018*
- Learning Bitcoin – *Richard Caetano - 2015*
- Learning with Big Data: the future of education – *Viktor Mayer Schonberher - 2014*
- Making Smart Contracts Smarter – *Loi Luu, Duc Hiep Chu, Hrishi Olickel, Prateek Saxena, Aquinas Hobor - 2016*
- Mastering Bitcoin – *Andreas M. Antonopoulos – 2014*
- Mastering Ethereum – *Andreas M. Antonopoulos and Gavin Wood – 2018*
- Merkle Root (Cryptocurrency) – *Investopedia.com - 2020*
- Proof of Work system – *Gerardus Blokdyk - 2018*
- Public versus Private Blockchain - *BitFury Group -2015*
- Security Tips on Writing Smart Contracts – *Nenad Palinkasevic – 2019*
- Smart Contracts: How to understand Smart Contracts and Be Ahead of Competition. Learn About the Future of Blockchain Technology – *Ruth Chandler – 2017*
- The Bitcoin Big Bang - *Brian Kelly – 2014*
- The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments – *Joseph Poon, Thaddeus Dryja - 2016*
- The Business Blockchain: promise, practise and application of the next Internet technology – *William Mougayar – 2016*
- Understanding Bitcoin – *Pedro Franco – 2014*
- What is Bitcoin Timestamp? Can it be changed? – *Dotwallet.com – 2019*
- What is Proof of Work – *Ledger.com – 2019*
- When do you need blockchain? Decision model - *Sebastien Meunier - 2019*
- Who can you trust - *Rachel Botsman – 2017*
- 21st century cash: Central banking, technological innovation and digital currencies – *Fabio Panetta - 2018*

Sitografia

- *www.binance.com*
- *www.bitcoin.com*
- *www.bitcoin.org*
- *www.blockchain.info.com*
- *www.coinbase.com*
- *www.coinmarketcap.com*
- *www.ethereum.org*
- *www.etherscan.io*
- *www.github.com*
- *www.metamask.io*
- *www.remix.ethereum.org*
- *www.solidity.readthedocs.io*
- *www.stateofthedapps.com*

*Elenco delle figure**

**L'elenco delle figure arriva fino a "Metamask, Solidity, Remix e Smart Contracts". Dopo non si tratta più di figure o grafici ma di parti del mio computer trasferite sull'elaborato*

Riassunto

La blockchain è una nuova tecnologia di cui spesso si sente parlare in ambiti diversi, completamente eterogenei tra loro, che finora ha trovato prevalente applicazione nel mondo Bitcoin. Tuttavia, associarla unicamente alla criptovaluta più famosa al mondo è erroneo e riduttivo.

La blockchain infatti a livello mondiale si sta affermando in diversi settori. Il primo tra questi è quello finanziario, a causa della sua capacità di trasferimento di denaro oltre i confini internazionali senza passare per un intermediario, con basse commissioni ed in maniera quasi istantanea. Un altro settore è quello della cybersecurity poiché la blockchain utilizza chiavi crittografiche che nascondono e non alterano il messaggio fino a quando quest'ultimo non arrivi all'effettivo destinatario. Infine, anche il voto elettorale beneficia molto dell'introduzione della blockchain perché permette l'autenticazione degli elettori, la conservazione dei registri di voto in maniera sicura e un'attività di spoglio precisa e trasparente.

Gli smart contracts invece vengono definiti "protocolli di transazione informatizzati, che eseguono i termini di un contratto", dato che, come dice Nick Szabo (colui a cui si fa risalire la creazione degli smart contracts) "traducendo le clausole contrattuali in codici e incorporandoli in hardware o software in grado di auto applicarle, è possibile ridurre al minimo la necessità di intermediari tra le parti ed eccezioni dannose o accidentali"

Blockchain e Smart Contract si adattano perfettamente all'IoT, AI, Machine Learning, Big Data e CryptoCurrencies e possono essere considerati un solido punto di partenza per lo sviluppo della rivoluzione tecnologica attuale.

La blockchain, ritenuta da molti la vera innovazione portata da Bitcoin, da un punto di vista ideologico risolve uno dei più grandi problemi presenti in un rapporto di scambio: la fiducia. Infatti, all'interno di un rapporto compratore-venditore si è sempre inserita una terza parte (il cosiddetto "Middle Man"), un'entità che si mette di mezzo, che garantisce che una transazione su Internet venga effettuata correttamente, senza il problema del double-spending. Questo concetto è abbastanza radicato nella nostra storia; la banca ne rappresenta un classico esempio. Tuttavia, la società si sta accorgendo che la terza parte spesso non è un valore aggiunto per la transazione, quanto piuttosto un costo sia in termini monetari, sia in termini di fiducia. Si fa strada infatti il concetto di disintermediazione, intesa come il fenomeno di riduzione dei flussi intermediati, ovvero la perdita di importanza del Middle Man, con l'annullamento della sua capacità di intercedere presso i due attori sociali per il raggiungimento dell'accordo. Questo concetto di disintermediazione nel rapporto di scambio si è verificato anche nel settore dei servizi: si pensi ad esempio ad Uber. Uber è un'azienda che fondamentalmente offre un servizio di trasporto senza possedere mezzi di trasporto. Lo stesso avviene per quanto riguarda Airbnb: offre servizi di locazione senza avere luoghi fisici di proprietà. Eppure, Uber e Airbnb sono due aziende che hanno "spaccato" il mercato facendo, qualcuno dice anche in maniera non propriamente legale a causa dei vantaggi fiscali, profitti enormi. Potrebbero esistere Uber e Airbnb senza che il cliente avesse pienamente fiducia nel funzionamento di queste due applicazioni?

In Airbnb, ad esempio, il cliente non è chiamato a fidarsi di una società (nel caso di Airbnb possiamo prendere come riferimento una catena di hotel) come è sempre stato finora, bensì di un'altra persona, un qualsiasi privato, che sta mettendo a disposizione la sua casa. Il cliente, a sua volta, potrebbe addirittura passare dall'altra parte del "rapporto", essendo lui in futuro a mettere a disposizione la propria casa (logicamente la stessa cosa non è possibile con un hotel). Un simile sistema senza la fiducia di ambedue le parti non potrebbe esistere.

Esistono due tipi di blockchain: blockchain permissioned e blockchain permissionless.

Una blockchain permissioned, che solitamente coincide con una blockchain privata, è quel tipo di blockchain in cui l'autorizzazione a "scrivere" i blocchi viene mantenuta centralizzata presso un'unica organizzazione. E' quindi una blockchain di tipo verticale, dove uno o più soggetti sono in grado di validare le transazioni che vengono registrate sulla catena. Le autorizzazioni per la consultazione della blockchain possono essere pubbliche o limitate discrezionalmente. Quindi questo implica che essa sia ristretta ad una cerchia conosciuta di entità: il termine "permissioned" riflette il fatto che l'autorità centrale possa introdurre delle policies per censurare le transazioni e quindi potenzialmente restringere l'uso della blockchain a sviluppatori e utilizzatori finali, i quali possono solamente fare affidamento su un'interfaccia fornita dagli operatori della blockchain per leggere e inviare le transazioni.

Le blockchain permissioned sono più attrattive per le istituzioni: queste blockchain potrebbero creare un ambiente più controllato e prevedibile rispetto a quelle permissionless. A differenza delle criptovalute, le blockchain permissioned generalmente non hanno token nativi. Infatti, i token nativi sono necessari nelle criptocurrencies per fornire incentivi a validare le transazioni; nelle blockchain permissioned invece i metodi per confermare le transazioni sono altri e tendenzialmente meno complicati. Nel più semplice dei casi, creare i blocchi su una blockchain permissioned non coinvolge calcoli associati con la proof of work.

Una blockchain permissionless è liberamente accessibile a chiunque. Non vi sono restrizioni circa la disponibilità di consultazione degli scambi, l'effettuazione degli stessi e la possibilità di partecipazione al meccanismo di consenso. Si tratta del modello su cui è stata realizzata la blockchain Bitcoin, pensata per disintermediare i meccanismi di fiducia reciproca tra i partecipanti. Essa non pone requisiti specifici per la partecipazione al network, incentivando ed auspicando anzi la sua espansione tramite meccanismi che remunerano coloro che mettono a disposizione la potenza computazionale dei loro hardware. Tipicamente i partecipanti a questo tipo di blockchain vengono denominati "miners" in quanto il loro compito è quello di impiegare delle risorse per consentire la creazione dei blocchi.

Il termine "permissionless" riflette proprio il fatto che non ci sono ampie policies restrittive riguardo l'uso della blockchain perché viene data liberamente la possibilità a utenti finali e sviluppatori di entrare e uscire dal protocollo.

Mentre le blockchain permissioned meglio si adeguano alle strutture legislative in corso e sono molto più attrattive per un'introduzione del sistema blockchain all'interno di un ledger già esistente nel breve-medio termine, tale tipologia limita uno degli aspetti fondamentali di questa tecnologia: la necessità di non doversi fidare della terza parte. Difatti uno degli obiettivi della blockchain è quello di rimuovere il fattore umano

(middle man) dalle transazioni processate, rimpiazzandolo con un protocollo rigoroso e pubblicamente disponibile, che viene reso funzionante da un network di computer indipendenti.

Se la blockchain non è consultabile e poco trasparente per l'utilizzatore finale, l'aspetto della fiducia viene ampiamente ridimensionato. Addirittura, l'utilizzatore finale potrebbe arrivare a dubitare dell'effettivo utilizzo di un sistema blockchain, dato che non c'è possibilità di consultarla né c'è libertà di accesso alle informazioni scritte. Il fattore umano rimarrà una vulnerabilità nelle blockchain private fino a quando non si riuscirà a ridurlo al minimo possibile. Inoltre, questa condizione di difficile consultazione, le fa diventare scarsamente accessibili a persone esterne al progetto; invece i codici open source e standardizzati delle blockchain pubbliche le rendono un ambiente di sviluppo per ricercatori informatici.

La blockchain è un libro mastro sicuro, trasparente e decentralizzato.

Sicuro non significa che sia possibile nascondere le informazioni, ma significa semplicemente che nessuno può manomettere la blockchain senza provocare dei "campanelli d'allarme" che verrebbero rapidamente notati dagli sviluppatori. Anche se qualcuno tenta di alterare fraudolentemente la blockchain, i dati originali resterebbero comunque validi e si scoprirebbe il tentativo di manomissione confrontando le informazioni sui vari ledger distribuiti tra i nodi.

La blockchain è progettata per utilizzare un hash crittografico e il timestamp (marca temporale) per rendere il registro inalterabile. Per gli esperti di blockchain, questo permette di ispezionare i dati per determinare le caratteristiche di una transazione o per rilevare tentativi di manomissione. La blockchain di Bitcoin assicura che nessuna parte in uno scambio dovrà fidarsi di una singola terza parte che tiene per loro il denaro.

La tecnologia dietro la blockchain può essere utilizzata per creare un efficace sistema di deposito a garanzia automatizzato in cui il sistema non scomparirà con il denaro o negherà mai l'accesso ad ogni conto personale. Questo dà a Bitcoin la capacità di essere una valuta che può attraversare i confini internazionali, creare wallets Bitcoin ed evitare di essere controllata da qualsiasi autorità centralizzata. Il mantenimento della blockchain si basa sul funzionamento di più nodi in grado di memorizzare i dati delle transazioni e su processori capaci di convalidare gli scambi. Si pensi ai nodi come a dei server uguali che si aggiornano regolarmente l'un l'altro e che permettono ai clients autenticati di connettersi ad essi. In caso di malfunzionamento di un nodo, il personale IT può lavorare con un esperto di blockchain per isolare il server e risolvere i problemi determinando cosa sia andato storto. I malfunzionamenti di un nodo di solito comportano la mancata trasmissione di informazioni valide, come ad esempio dati che non hanno senso, oppure il rifiuto di trasmettere blocchi in rete. Il nodo potrebbe "andare fuori strada", creare la propria versione della blockchain e inventare informazioni che gli altri nodi non possono utilizzare in alcun modo significativo. In casi come questo, un esperto di blockchain avvertirà il personale IT di isolare il server dal resto della rete fino a quando non sarà in grado di individuare e risolvere il problema.

Se un nodo funziona male e deve essere isolato a causa di una manutenzione all'interno di un'azienda che si basa sull'accesso ai dati in tempo reale, non ci sarà una perdita di tempo e di ricavi perché i nodi rimanenti prenderanno il suo posto e i clienti verranno reindirizzati e potranno ricollegarsi rapidamente ad un nodo

funzionante. Questa è un'ottima soluzione per un imprenditore che non vuole spiegare ai clienti perché il server di una rete centralizzata sia andato fuori uso.
La blockchain è stata originariamente progettata per essere un sistema decentralizzato che tiene traccia degli addebiti e degli accrediti: l'esistenza di migliaia di nodi Bitcoin in sei continenti diversi dimostra la sua capacità di poter in futuro diventare effettivamente il "World Wide Web" della finanza.
Bitcoin combina le idee alla base del timestamp e della proof of work in modalità Hashcash per arrivare al risultato di rendere sicura la blockchain: questa è la principale innovazione introdotta da Bitcoin. Ogni blocco contiene un gruppo di nuove transazioni e un collegamento al blocco precedente della catena. Si noti che le vecchie transazioni sono ancora scritte in blockchain: i vecchi blocchi non vengono mai rimossi, quindi la blockchain può solo aumentare in lunghezza.
In primo luogo, ogni blocco include un gruppo di transazioni (valide), l'hash del blocco precedente e un nonce. Il nonce in un blocco risolve il problema dell'inversione parziale dell'hash, cioè è un numero tale per cui l'hash dell'intero blocco (incluso il nonce) inizi con un certo numero di zeri. È facile regolare la difficoltà del blocco aumentando il numero di zeri di partenza. Il protocollo Bitcoin regola questa difficoltà per raggiungere la durata media di 10 minuti tra un blocco e l'altro. Questa regolazione della difficoltà fa parte delle regole di Bitcoin ed è codificata per ogni client. La difficoltà del blocco viene regolata ogni 2.016 blocchi minati o approssimativamente ogni 2 settimane. La regolazione tiene conto della variazione della potenza di calcolo dell'intera rete (hashing power) dall'ultima regolazione, confrontando i timestamp di due blocchi distanti 2.016 posizioni l'uno dall'altro. Quando la potenza di calcolo cresce (quindi più miners in circolazione o sviluppi hardware maggiori), i blocchi saranno estratti più velocemente rispetto ai 10 minuti previsti. La difficoltà quindi verrà regolata più in alto, seguendo tuttavia il livello dell'hashing power raggiunto. La ricompensa per ogni blocco è la remunerazione che viene pagata ogni volta che un minatore risolve il problema dell'inversione parziale dell'hash. Così i nuovi bitcoin emessi vengono assegnati ai minatori che contribuiscono con la loro potenza di calcolo a certificare la blockchain.
Il processo di risoluzione dei blocchi è chiamato mining (estrazione mineraria) in analogia con l’estrazione dei metalli preziosi. Questa analogia, sebbene utile, può portare ad un fraintendimento: la ricompensa del blocco è fissata dal protocollo e non è influenzata dal numero di miners in circolazione. Contrariamente all'estrazione dell'oro, un aumento dei minatori non aumenta il numero di bitcoin in circolazione. Un ingresso di nuovi miners invece incrementa l'hashing power, riducendo il numero di minatori originali, ma mantenendo costante la ricompensa totale per ogni blocco.
Un fork si verifica quando due miners provano a minare un nuovo blocco più o meno nello stesso momento. Si immagini ad esempio di aggiungere degli anelli a una catena; se qualcuno arriva e sceglie un anello al centro della catena, iniziando ad aggiungere i propri anelli incurante degli altri, la catena sta subendo un fork. Entrambi i blocchi risolvono il problema dell'inversione parziale dell'hash, ma solo uno di essi può far parte della blockchain.
Sono possibili all’interno del protocollo blockchain due tipi differenti di fork: l’hard fork e il soft fork.

L'hard fork è uno scenario in cui la rete cambia le regole di consenso: si chiama hard fork, perché dopo il fork, la rete non si ricongiunge su una singola catena, ma al contrario, le due catene evolveranno in modo indipendente. Gli hard fork si verificano quando una parte della rete opera secondo una serie di regole di consenso diverse rispetto al resto della rete: può avvenire a causa di un bug o di un cambiamento deliberato nell'implementazione delle regole di partecipazione.

Gli hard fork possono essere usati per cambiare le regole del consenso, ma richiedono un coordinamento tra tutti i partecipanti al sistema. I nodi che non passano alle nuove regole non possono partecipare al meccanismo di consenso e sono spinti in un'altra blockchain al momento dell'hard fork.

Non tutti i cambiamenti delle regole del consenso, ma solo i cambiamenti incompatibili con il sistema precedente, provocano un hard fork.

Se la modifica viene attuata in modo tale che un nodo che non aggiorna il sistema, veda ancora la transazione o il blocco come valido secondo le regole precedenti, la modifica è stata attuata senza un fork.

Il termine soft fork è stato introdotto per distinguere questo metodo di aggiornamento da un "hard fork".

In pratica, un soft fork non è affatto un fork. Un soft fork è un cambiamento compatibile con le regole di consenso che permette ai clienti non aggiornati di continuare ad operare in accordo con le nuove regole.

Un aspetto del soft fork è che gli aggiornamenti possono essere usati solo per limitare le regole del consenso, non per espanderle. Per essere compatibile, le transazioni e i blocchi creati secondo le nuove regole devono essere validi anche secondo le vecchie regole, ma non viceversa. Le nuove regole possono solo limitare ciò che è valido; altrimenti, se respinte secondo le vecchie regole, daranno luogo ad un hard fork.

I soft fork possono essere implementati in diversi modi, infatti il soft fork non definisce un singolo metodo, ma piuttosto un insieme di metodi che hanno tutti in comune il non richiedere l'aggiornamento di tutti i nodi o l’espulsione dei nodi non aggiornati al di fuori dal network.

Una delle critiche spesso sollevate contro la blockchain di Bitcoin è che non sia abbastanza scalabile per gestire un tasso di transazioni paragonabile a quello delle normali reti di elaborazione di pagamenti. Al momento della scrittura, i blocchi includono una media di 2100 transazioni.

Infatti, la crescente adozione delle criptovalute ha sollevato molte preoccupazioni circa la loro capacità di scalare. Poiché Bitcoin è un sistema che si autoregola e che funziona estraendo i blocchi ad intervalli regolari, il suo crescente volume di transazioni si scontra con il limite massimo di dimensione di ciascun blocco rispetto all'intervallo in cui viene minato.

La tendenza crescente riguardante le dimensioni dei blocchi della blockchain di Bitcoin, fa presagire un potenziale problema che si concretizzerebbe ove il sistema addirittura arrivasse a cancellare le transazioni a causa sia della scarsa velocità nel minare i blocchi sia della dimensione del blocco rispetto al numero di transazioni.

Di conseguenza, la comunità sta discutendo da tempo le tecniche per migliorare la scalabilità delle blockchain, in particolare di quella di Bitcoin. Questi dibattiti sono molto accesi e hanno portato a spaccature interne, senza aver delineato un chiaro percorso per affrontare il problema della scalabilità.

I sistemi più conosciuti che cercano di risolvere il problema della scalabilità nella blockchain di Bitcoin sono i Payment Channels e Lightning Network.

Il Payment Channels consistono nella creazione di un canale bidirezionale tra due utenti, che sono in grado di effettuare transazioni tra loro istantaneamente, liberamente e senza fiducia. Il canale tra gli utenti viene aperto con una transazione di finanziamento, che è inclusa nella blockchain, proprio come una normale transazione Bitcoin, richiedendo così una commissione per il mining.

Gli stessi utenti generano, firmano e scambiano anche una transazione di rimborso (refund transaction) che permetterà alle controparti di recuperare i loro fondi bloccati. Una volta aperto il canale descritto, gli utenti saranno in grado di effettuare transazioni tra di loro, entro i limiti della loro capacità di canale (l'importo bloccato) semplicemente aggiornando la transazione di rimborso, che sostituisce ogni volta la precedente. Nessuna transazione che si verifichi tra i due utenti finirà sulla blockchain, non necessitando quindi di tasse minerarie o di tempo di conferma.

Sarà infatti trasmessa una sola transazione, quella finale, che chiuderà il canale ogni volta che una delle controparti lo deciderà. In questo caso, l'utente dovrà solo inviare l'ultima transazione di rimborso e attendere che venga minata, per riprendere il controllo dei suoi bitcoin.

Un'estensione dei Payment Channels potrebbe essere in grado di risolvere il problema di scalabilità. Ciò è reso possibile da Lightning Network, che permette di "estendere" i canali grazie alla crittografia, formando una "rete" di collegamenti.

Lightning Network funziona attraverso una struttura ramificata di canali di pagamento. Ciò significa che non è necessario avere un canale aperto con ciascun utente per effettuare una transazione: è sufficiente "trovare un modo" per raggiungere il “bersaglio” finale, passando attraverso i diversi percorsi nei canali.

Una delle aree più promettenti di applicazione della blockchain è la creazione di contratti completamente automatizzati cioè accordi che vengono eseguiti senza il coinvolgimento umano. Questi, nell'ambiente IT, sono chiamati “contratti intelligenti" o “smart contracts”.

Non esiste una definizione universalmente condivisa di contratto "smart"; ciò è dovuto sia alla sua natura innovativa, sia alla sua complessa base tecnologica.

Secondo la definizione più semplice, il contratto smart è un contratto la cui esecuzione è automatizzata. E’ opportuno fare riferimento ad un'altra definizione di smart contract fornita da Gideon Greenspan: "Un contratto smart è un pezzo di codice che viene memorizzato su una blockchain, attivato da transazioni su blockchain e che legge e scrive i dati nel database di quella blockchain". Questa definizione è più concreta, poiché pone l'accento sulla blockchain come una delle caratteristiche fondamentali del contratto intelligente.

La blockchain può essere considerata un "cambio di paradigma" in ambito contrattuale perché permette di automatizzare il processo di esecuzione del contratto di entrambe le parti.

Il teorico organizzativo Arthur Stinchcombe una volta scrisse che i contratti sono solo organizzazioni in miniatura e, per estensione, tutte le organizzazioni sono solo complessi di contratti. Le aziende sono create utilizzando una serie di accordi contrattuali, che vanno dai contratti di lavoro per i dipendenti, ai rapporti con i fornitori, agli obblighi verso i propri clienti, ai contratti di locazione, alla vendita e all'acquisto di attrezzature.

Tradizionalmente, questi obblighi contrattuali sono piuttosto costosi perché devono essere fatti rispettare grazie un sistema legale affidabile e attraverso l'applicazione della legge.
Con uno smart contract basato su una blockchain, tuttavia, gran parte di questi costi sono notevolmente ridotti o eliminati. Questo permette di rendere le organizzazioni basate su blockchain più efficienti, convenienti e competitive rispetto alle aziende tradizionali sul mercato. Tutto ciò dimostra che gli smart contracts vanno ben oltre i modelli esistenti di contrattazione e rappresentano un nuovo paradigma di interazione in un cyberspazio.
Gli smart contracts consentono di creare pool di risorse e di ripartirle secondo criteri concordati, cosa che può essere particolarmente rilevante per le attività di crowdfunding o per i contratti di carattere assicurativo.
Nel primo caso, gli smart contracts possono tenere traccia dell'importo dei fondi appartenenti ad un progetto di crowdfunding e, una volta che viene raggiunto l'importo necessario, lo stesso viene trasferito al beneficiario. In caso contrario, i fondi vengono restituiti ai donatori.
Nel secondo caso, un gruppo di agricoltori potrebbe mettere insieme un pool di risorse come assicurazione contro la siccità, le inondazioni o altri disastri naturali. Una volta che si verificasse un tale disastro, il contratto lo accerterebbe secondo la procedura specificata nello stesso (ad esempio controllando il meteo o le notizie in fonti predesignate) e assegnerebbe le risorse. Inutile dire che lo smart contract fornisce il massimo grado di trasparenza e di verificabilità, attenuando i rischi associati al processo decisionale dell'intermediario e al "fattore umano", nonché ai ritardi temporali.
Sebbene l'esecuzione dello smart contract sia automatizzata, essa richiede comunque la presenza della volontà della parte contraente per diventare efficace. La persona esprime il suo consenso ai termini del contratto e alle modalità della sua esecuzione al momento della conclusione dello stesso. Tenendo conto del fatto che tale persona non sarà in grado di influenzare l'esecuzione del contratto una volta stipulato, dovrebbe esserci una certa fiducia, che dà luogo ad una sorta di rapporto "fiduciario" nel contratto intelligente. Ma a differenza del contratto classico, in cui la fiducia è riposta nella controparte, negli smart contracts tale fiducia è riposta nell'algoritmo informatico che sta alla base del contratto ("fiducia senza fiducia").
Ethereum, piattaforma informatica usata per creare smart contracts, viene spesso soprannominata "il computer del mondo".
Dal punto di vista informatico, Ethereum è una state-machine (cioè permette di descrivere con precisione e in maniera formale il comportamento di molti sistemi) con due funzioni base: la prima è uno stato singleton accessibile a livello globale e la seconda è una macchina virtuale che applica cambiamenti a quello stato.
Ethereum è stato creato nel novembre 2013 come piattaforma generale per la creazione di blockchain, combinando insieme la nozione di consenso economico attraverso la proof of work (o eventualmente la proof of stake) con un linguaggio Turing completo; in questo modo si voleva sia consentire agli sviluppatori di implementare molto più facilmente applicazioni decentralizzate, sia evitare di creare una nuova blockchain per ogni nuova applicazione.
Mentre i precedenti protocolli potevano essere visti come strumenti che risolvevano una singola funzione o al massimo come strumenti multifunzione, Ethereum è lo "smartphone" delle blockchain: una piattaforma

universale dove, qualsiasi cosa si voglia costruire, si può semplicemente progettare come "app" decentralizzata.

Lo scopo di Ethereum non è quello di creare un nuovo metodo di pagamento come Bitcoin. La sua criptocurrency, l'ether, è necessaria principalmente per il funzionamento della piattaforma.

A differenza di Bitcoin, che ha un linguaggio di scripting molto limitato, Ethereum è progettato per essere una blockchain programmabile di uso generale che esegue una macchina virtuale in grado di "runnare" codici di varia complessità senza limiti.

La Macchina Virtuale dell'Etereum (Ethereum Virtual Machine - EVM) è una macchina mondiale che chiunque può utilizzare, a fronte di una piccola tassa, pagabile in ether.

L'EVM è un unico "computer" globale a 256 bit in cui tutte le transazioni sono locali su ogni nodo della rete ed eseguite in relativa sincronia. Questo gigantesco computer, al quale può accedere chiunque abbia un nodo o un'applicazione, rende semplice spostare grandi quantità di denaro quasi istantaneamente. Anche se chiunque può usare questa macchina virtuale globale, nessuno può creare denaro falso al suo interno o spostare fondi senza autorizzazione.

Il gas è un'unità di lavoro utilizzata per misurare il costo computazionale di un'operazione su Ethereum. I costi del gas sono pagati con piccole quantità di ether.

Lo scopo del gas è duplice. In primo luogo, garantisce una ricompensa predefinita per i minatori che eseguono il codice e mettono in sicurezza la rete. In secondo luogo, aggira il problema dell'interruzione e assicura che l'esecuzione non possa durare più a lungo del tempo che è stato pagato in anticipo. Il gas è un'unità di lavoro che non si può tenere o accumulare. Semplicemente misura quanto sforzo comporterà ogni fase di una transazione, in termini computazionali. Per poter pagare i costi del gas, è sufficiente aggiungere ether al proprio conto e non è necessario acquistarlo separatamente. Ogni operazione possibile sull'EVM ha un costo di gas associato.

L'EVM può eseguire programmi scritti nel linguaggio di programmazione Solidity.

Solidity è un nuovo linguaggio di programmazione utilizzato per scrivere i contratti intelligenti, che possono essere eseguiti dall'EVM. Esso è un insieme di convenzioni di rete, linguaggio di assemblaggio e sviluppo web. Solidity è orientato verso i contratti, con somiglianze con JavaScript e C. Permette di sviluppare contratti e di compilare il bytecode EVM e attualmente è il linguaggio di punta di Ethereum. Ci sono quattro linguaggi nel protocollo Ethereum allo stesso livello, ma la comunità ha preferito Solidity, che ha messo fuori gioco Serpent (simile a Python), Lisp-Like Language (LLL), e Mutan.

La prima cosa da fare nel caso in cui si volesse scrivere uno smart contract all'interno della blockchain di Ethereum è procurarsi del gas che deve essere pagato in ether. L'applicazione usata per questo scopo è chiamata Metamask ed è un'integrazione del browser, che funziona da wallet Ethereum.

Oltre lo storage di ether, Metamask permette di interagire con tutta la piattaforma Ethereum perché è un portafoglio di criptovaluta che ti consente di usare applicazioni decentralizzate basate su Ethereum direttamente dal tuo browser. Quindi Metamask è un plugin che consente agli utenti di interfacciarsi con Ethereum, partecipare alle ICO, interagire con le dApps ecc.. attraverso tutti i siti web che operano con

criptovalute. Essa pertanto ha il pregio di connettere i front end Ethereum con il normale web che chiunque è in grado di utilizzare; infatti comunemente si dice che Metamask è un ponte che ti consente di visitare il web distribuito di domani nel tuo browser di oggi.

Se Solidity è il linguaggio di programmazione, si adopererà Remix come mezzo per scrivere smart contracts. Remix è un potente strumento open source che aiuta a creare contratti su Solidity direttamente dal browser. Scritto in JavaScript, Remix supporta anche operazioni di testing, debugging, implementazioni di contratti intelligenti e molto altro ancora. Infatti, Remix è un IDE (ambiente di sviluppo integrato) cioè un software che permette ai programmatori di sviluppare il codice sorgente del loro programma e comunica direttamente loro eventuali errori nella compilazione, spiegando dove e in cosa consiste lo sbaglio.

Remix è estremamente comodo perché è un IDE online, quindi non c'è la necessità di scaricare nulla nel computer ma basta accedere a Remix dal sito remix.ethereum.org.

Combinando il linguaggio di programmazione Solidity, l'ambiente di sviluppo Remix e l'estensione/portafoglio Metamask è possibile scrivere degli smart contracts e io in particolare ho cercato di dimostrare la loro possibile applicazione nel mondo della finanza.

In un primo esempio ho dimostrato come un contratto intelligente fosse sufficiente per gestire un conto corrente bancario; anche con un codice piuttosto semplice, ho dimostrato che lo smart contract ben eseguiva i comandi da me richiesti e aggiornava perfettamente il bilancio del conto corrente.

In un secondo contratto ho lanciato un messaggio in blockchain: mentre fino a prima avevo fatto delle prove su un ambiente di sviluppo interno al mio computer, con gas e commissioni finte, in questo caso ho prima scritto un messaggio da me inventato e poi ho deciso di modificare la blockchain reale di Ethereum per lasciare scritto per sempre un mio pensiero sulla LUISS. In questo modo, ho illustrato come si potrebbero scrivere informazioni immutabili, criptate e sicure in una blockchain mediante uno smart contract.

Per esempio, ipotizziamo una situazione in cui un consumatore ordini alcuni prodotti dall'altra parte del mondo; nel caso in cui volesse avere una prova della autenticità della merce ordinata, il produttore potrebbe servirsi di un contratto come quello che ho appena ideato come certificazione di garanzia. Infatti, all'interno della stringa di testo, si potrebbero scrivere delle informazioni circa la provenienza, la denominazione di origine protetta e un codice; il produttore poi potrebbe apporre il codice (come ho fatto io con la mia matricola) sul prodotto e spedirlo al consumatore. Quest'ultimo, una volta ricevuto il prodotto, potrebbe, mediante il codice hash della transazione, andare sulla blockchain di Ethereum e vedere la stringa di testo (e il corrispondente codice) scritta nello smart contract. Laddove il codice sulla blockchain corrispondesse con il codice scritto sul prodotto, si avrebbe la certezza che la merce arrivata, sia il prodotto veritiero e non uno contraffatto.

Potrebbero essere aggiunte nella stringa di testo altre informazioni sensibili che non si vuole divulgare pubblicamente o sottoporre ad un canale informativo non sicuro. Anche in questo caso la blockchain e gli smart contracts ci vengono in aiuto perché il messaggio con i dati sensibili sono pubblici, ma non possono essere criptati se non si è a conoscenza dell'hash della transazione; per questo motivo possiamo essere sicuri

che le informazioni non cadranno mai in mani sbagliate e che la controparte leggerà i dati inviati, il tutto senza l'aiuto di un intermediario che potrebbe appropriarsi o leggere quanto scritto dal produttore.
Un'ultima caratteristica fondamentale è l'immutabilità del registro garantito dal timestamp. Infatti, quando il contratto viene lanciato, automaticamente viene registrata la data e l'ora dello stesso garantendone quindi l'immutabilità. Anche questo sarebbe importante in una catena di produzione perché si conoscerebbe con certezza il momento esatto in cui le merci sono state spedite senza possibilità di alterazioni. Per esempio, possiamo ipotizzare una supply chain composta da molti fornitori in un processo just in time: nel caso in cui si utilizzassero gli smart contracts, si potrebbe facilmente capire da dove arrivano eventuali ritardi o mancanze nella catena. E' sufficiente scrivere sugli smart contracts un codice che certifica le merci, un messaggio con le informazioni rilevanti e lanciare il contratto; in questo modo automaticamente si sapranno data e ora della spedizione e dell'arrivo; diventerebbe quindi facile gestire gli eventuali errori o processi da migliorare nella supply chain.
Un altro mondo che già sta iniziando a beneficiare di smart contracts simili a quello creato ora, è quello assicurativo. Senza doversi affidare ad un intermediario che verifica e registra il sinistro, per esempio nel caso di ritardi nei viaggi aerei, potrebbe essere fatto un contratto intelligente che legge dal sito gli orari di partenza, li confronta con gli orari di "effettiva" partenza e se c'è stato un ritardo rimborsa automaticamente i clienti con un trasferimento di denaro sul loro conto corrente.
Anche la LUISS per esempio potrebbe beneficiare degli smart contracts. Nel caso in cui l'Università volesse avere un maggiore livello di sicurezza inerentemente ad alcune questioni, non volesse affidarsi alla mail per comunicare i voti agli studenti oppure volesse garantire l'immutabilità delle votazioni di laurea, potrebbe creare uno smart contract che soddisfi queste necessità e lanciarlo in blockchain.
Infine, ho dimostrato come si possa scrivere un token ERC-20 standard sulla blockchain di Ethereum.
I token ERC-20 sono valute digitali con caratteristiche semplici ma funzionanti, motivo per cui sono alla base di molti progetti e molte ICO. In una ICO ci sono tre situazioni che devono essere rispettate: prima i clienti pagano lo smart contract, poi esso trasferisce i soldi al proprietario il quale infine consegna i token ERC-20 al cliente.
Prima che venisse creato lo standard ERC-20, si erano manifestati diversi problemi di compatibilità tra le varie tipologie di token su Ethereum e ognuno di questi possedeva un proprio smart contract. Per questo motivo, supportare la crescente varietà di token stava diventando problematico e dispendioso in termini di tempo. Come soluzione, l'ambiente Ethereum ha creato un protocollo standard che tutti i token devono seguire, l'ERC-20.
I token ERC-20 possono essere usati in vari modi: quote di un progetto, certificati di proprietà di asset e persino criptovalute vere e proprie. Possono anche ricoprire simultaneamente diversi di questi ruoli.
Un esempio famoso di token ERC-20 è Binance, la criptovaluta nata nel 2017, legata ad uno dei migliori exchange di valute digitali online: Binance.com.
Il BNB, ottava criptovaluta per capitalizzazione (al momento della scrittura), serve per avere sconti sulle commissioni e migliorare le prestazioni interne della piattaforma. Quindi, chi possiede BNB riduce le

commissioni sul trading, può utilizzarlo per pagare interessi sul margin trading e può pagarci direttamente le ICO. In cambio Binance utilizza i fondi ricavati dagli acquirenti per aggiornamenti della piattaforma (nuove assunzioni, formazione del personale e budget di sviluppo), attività di branding, marketing e promozione; infine una parte viene adibita a riserva in caso di situazioni di emergenza.

Il prezzo della criptovauta viene dettato dall'interazione tra domanda e offerta ma a ciò va aggiunto che, ogni 4 mesi, Binance utilizza i suoi profitti per acquistare il 20% dei Binance coin in circolazione sul mercato e distruggerli (in gergo "burn"). Aumentando la scarsità della moneta, Binance contribuisce ad incrementarne il valore.

Con l'esempio di Binance, si è cercato di spiegare come potrebbe essere utilizzato un token ERC-20 per ottenere dei finanziamenti in maniera rapida. Bisogna tenere in considerazione che, affinché gli investitori possano comprare il nuovo token emesso, devono avere fiducia nel progetto sottostante. Per esempio, se Binance iniziasse a perdere i fondi nei conti o a non eseguire correttamente le transazioni, i clienti cambierebbero exchange e nessuno sarebbe più disposto a comprare dei BNB. Come si può notare, la facilità di reperibilità di fondi e l'ancora scarsa legislazione a riguardo, ha reso estremamente popolare le ICO mediante token ERC-20 come metodo di crowdfunding; per questo motivo, gli investitori sono chiamati ad un'attenta analisi riguardo i progetti su cui investire, soprattutto perché le garanzie legislative in materia non sono ancora complete.

Si può pensare che anche la LUISS, per finanziare qualche progetto o iniziativa, possa lanciare una propria criptovaluta. Sono molte le società anche fuori dal mondo cripto che stanno lanciando token di proprietà con lo scopo di aumentare il legame con i propri clienti o sostenitori (vedi Juventus). Lanciando un token ERC-20 acquistabile dagli studenti, la LUISS potrebbe garantire ai possessori della valuta sconti sui servizi all'interno dell'Ateneo o il possibile coinvolgimento in alcune decisioni sul futuro dell'Università. In questo modo sia si aumenterebbe il coinvolgimento da parte degli studenti nelle decisioni interne all'Ateneo, sia si capirebbe quali siano le volontà reali degli studenti LUISS, i quali, pagando il token, avrebbero il diritto di partecipare alle scelte inerenti all'Università in cui studiano.

Per esempio, ipotizziamo che la LUISS lanci il proprio token ad un prezzo iniziale che poi si aggiusterà sulla base della domanda degli studenti. In questo modo mentre l'Università riceverà dei soldi da poter spendere, gli studenti compratori del token potranno concorrere a decidere se l'Ateneo necessiti di un'aula studio più grande o di nuovi volumi per la biblioteca, se debba organizzare o meno degli incontri su determinate tematiche o se ci sia la necessità di potenziare il servizio navette piuttosto che la mensa.

Inoltre, attraverso questo meccanismo sarebbero chiamati a prendere delle decisioni gli studenti che vivono in maniera più "attiva" la LUISS, essendo quelli che sono stati disposti a pagare per incidere sulla vita universitaria. In qualsiasi momento inoltre si dovrebbe dare la possibilità agli studenti di scambiare i propri token, come avviene normalmente nei mercati finanziari, e obbligarli alla vendita di tutte le loro monete digitali una volta concluso il percorso universitario.

Un’idea potrebbe essere quella di lanciare un ammontare di token pari al numero di studenti immatricolati, in modo che se tutti comprassero un token, tutti avrebbero le stesse possibilità di decidere le sorti universitarie. Tuttavia, non sarà mai così e il mercato si regolerà automaticamente verso l’equilibrio.
Gli studenti compratori si sentirebbero parte di un’Università che li coinvolge attivamente nelle proprie scelte, che li avvantaggia in alcune cose (sconti a mensa per esempio) e che gli dà la possibilità di liquidare il loro mini investimento laddove non si sentissero sufficientemente coinvolti o volessero chiudere la loro posizione per avere un piccolo profitto perché il valore del LUISS token è aumentato.
Un progetto più ambizioso potrebbe essere quello di lanciare la criptovaluta LUISS a livello globale, quindi come un normale token le cui fluttuazioni dipendono dall’andamento dell’Università e che possa essere comprato da qualsiasi investitore che crede nella LUISS.
Uno dei temi su cui si sta dibattendo negli ultimi mesi è quello del possibile lancio di una digital currency da parte delle Banche Centrali (“Central Bank Digital Currency” ovvero CBDC). Infatti, stimolate dal lancio di token effettuati da società private (vedi per esempio Libra, la criptocurrency appartenente a Facebook), molte Banche Centrali hanno iniziato a lavorare su progetti riguardanti monete digitali personali rappresentanti una valida alternativa al contante e ai depositi bancari.
Il progetto Ubin con la Monetary Authority di Singapore ha insegnato che una CBDC abilitata sulla blockchain supporta flussi di pagamento complessi più efficienti di quelli attuali, compreso un meccanismo decentralizzato di risparmio di liquidità.
La Bank of Lithuania ha in programma l'emissione di "Digital Collector Coin" per testare la blockchain su scala ridotta e su un ambiente reale. Sarà collegata alle monete da collezione fisiche custodite nei caveau della Bank of Lithuania.
La People's Bank of China ha avviato un'iniziativa di "yuan digitale" e la Banca Centrale svedese, Sveriges Riksbank, ha annunciato un progetto per una versione digitale della sua valuta per uso retail, denominata e-krona.
Recentemente la Bank of England ha svolto ricerche su quello che chiama il "modello di piattaforma", in cui la banca è l'unica entità autorizzata a creare o distruggere un token, lasciando che i fornitori di interfacce di pagamento interagiscano con gli utenti finali. Inoltre, la Federal Reserve, la Banca del Giappone e la Banca Centrale Europea stanno attualmente esplorando tale tecnologia.
La nazione che è più avanti nel progetto di lancio di una CBDC è il Canada che sta per creare una propria valuta digitale come soluzione di emergenza in caso di necessità. Proprio come il contante, la CBDC non genera alcun interesse e sarà universalmente accessibile; un aspetto fondamentale della currency virtuale canadese è l’adattamento al mondo digitale dinamico, per esempio con i sistemi IoT, l’interoperabilità con i sistemi di pagamento esistenti, la sua facilità di scambio e l’accessibilità in modo da essere estensibile con strumenti come smartphone, smartwatch e computer.
Abbiamo visto che la BCE sta pensando di implementare una propria valuta digitale; come facilmente immaginabile, molti sono gli aspetti da definire in modo da massimizzarne il risultato.

Da un punto di vista strettamente informatico, sarà una moneta completamente diversa da Bitcoin, basata su una blockchain di tipo permissioned con il processo di mining non basato sulla proof of work. Tuttavia, si parla di un progetto che potrebbe essere supportato dalla piattaforma Ethereum, la stessa che ho adottato per la stesura degli smart contracts.

Ethereum in particolare è la blockchain più pronta a supportare i requisiti di una CBDC in termini di scalabilità e privacy. Infatti, i sistemi distribuiti, come le blockchain, garantiscono la disponibilità e la resilienza dei dati, oltre alla fiducia e alla trasparenza sulla storia delle transazioni. Ethereum ha dimostrato la sua capacità di supportare reti molto grandi con più di 10000 nodi e centinaia di migliaia di utenti.

Infine, una CBDC basata su blockchain si avvantaggia dei prodotti e dei servizi innovativi che si stanno costruendo nell'ambiente delle blockchain open source, compresi i portafogli online, la crittografia e la finanza decentralizzata. Ethereum è il più grande ecosistema di blockchain del mondo con oltre 350.000 sviluppatori.

Da un punto di vista concettuale e ideologico, sono stati pubblicati moltissimi papers riguardanti la necessità o la convenienza dell'introduzione di un token da parte della BCE, ma quello che reputo più completo nell'esposizione del problema è stato scritto da Fabio Panetta, membro del comitato esecutivo della BCE.

Panetta parte dalla semplice domanda se la Banca Centrale debba o meno lanciare una propria digital currency e risponde al quesito analizzando le funzioni della moneta come mezzo di pagamento e riserva di valore (Queste sono due delle tre funzioni del denaro. La terza, il denaro come unità di conto, non è rilevante in questo caso, poiché una moneta digitale emessa dalle Banche Centrali sarebbe denominata nella stessa unità delle banconote esistenti).

Come mezzo di pagamento, Panetta ritiene che una CBDC si aggiungerebbe ai servizi di pagamento digitali disponibili, aumentando così il grado di concorrenza nel settore. Tuttavia, l'insieme di strumenti che permettono pagamenti istantanei è già ampio: al giorno d'oggi possiamo effettuare un pagamento digitale tramite bonifico bancario (attraverso l'online banking), carte di credito o di debito, utilizzando Paypal o Apple pay; possiamo farlo tramite computer, smartphone o smartwatch, semplicemente mettendo il polso vicino ad una macchinetta. Da questo punto di vista i vantaggi di una CBDC sono poco chiari: i suoi potenziali benefici in termini di miglioramento della facilità delle transazioni sono probabilmente insufficienti per giustificare il coinvolgimento delle Banche Centrali.

Invece l'introduzione di un mezzo di pagamento digitale potrebbe essere giustificata dall'obiettivo di ridurre il costo del contante, ossia le spese per la sua produzione, il trasporto, lo smaltimento ecc. Secondo stime recenti, questi costi ammontano a circa mezzo punto percentuale del PIL nell'Unione Europea, ovvero a circa 76 miliardi di euro.

I costi di gestione del contante sono dovuti alla sua natura fisica e in un mondo digitale scomparirebbero, mentre i costi di hardware e software aumenterebbero. Tuttavia, la tecnologia digitale svolge già un ruolo cruciale nel settore finanziario poiché viene utilizzata per il trasferimento di denaro tra le banche commerciali, per l'acquisto e la vendita di titoli e per l'elaborazione di informazioni. La tecnologia necessaria per il trasferimento di denaro digitale avrebbe probabilmente forti complementarità con le reti e le infrastrutture

digitali esistenti e ciò suggerisce che i costi complessivi della fornitura di un mezzo di pagamento potrebbero diminuire con l'introduzione di una moneta virtuale.

Come riserva di valore, poiché sarebbe completamente dematerializzato, una CBDC avrebbe pochissimi costi di stoccaggio e rappresenterebbe un metodo conveniente di conservazione di liquidità per le famiglie e le imprese. Oltre ad essere superiore al contante come riserva di valore, una moneta digitale sarebbe un'attività con caratteristiche uniche, priva di rischio di credito e di liquidità. Potrebbe essere preferita ad altri strumenti comunemente usati per conservare la ricchezza, come i depositi bancari.

Laddove questo dovesse succedere, potrebbero sorgere alcuni problemi: il passaggio dai depositi bancari a una CBDC potrebbe portare a una carenza di fondi nel sistema bancario, con potenziali effetti negativi sull'offerta e sul costo del credito nell'economia reale. In condizioni estreme, la disponibilità di una CBDC potrebbe addirittura aumentare il rischio di una bank run digitale. Tuttavia, non è detto che le conseguenze possano essere di così ampia portata e che le banche subiscano necessariamente ingenti perdite. In primo luogo, solo alcune categorie di depositi potrebbero migrare verso la Banca Centrale (molto probabilmente depositi a vista, che pagano poco o nessun interesse). In secondo luogo, le banche possono competere offrendo servizi che le CBDC non possono offrire, come l'accesso ai servizi di credito e di pagamento. In terzo luogo, le banche potrebbero aumentare il ricorso alla raccolta all'ingrosso.

E’ innegabile però che il modello di business delle banche ne risentirebbe. La diminuzione delle passività callable porterebbe verso un sistema bancario "ristretto ", che è un quadro operativo in cui le banche hanno uno scarso disallineamento di scadenze tra attività e passività.